Nested, but Separate: Isolating Unrelated Critical Sections in Real-Time Nested Locking

James Robb
Reykjavik University (RU)
me@jamesrobb.ca

Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS)
bbb@mpi-sws.org

Abstract

Prior work has produced multiprocessor real-time locking protocols that ensure asymptotically optimal bounds on priority inversion, that support fine-grained nesting of critical sections, *or* that are independence-preserving under clustered scheduling. However, while several protocols manage to come with two out of these three desirable features, no protocol to date accomplishes all three. Motivated by this gap in capabilities, this paper introduces the *Group Independence-Preserving Protocol* (GIPP), the first protocol to support fine-grained nested locking, guarantee a notion of independence preservation for fine-grained nested locking, *and* ensure asymptotically optimal priority-inversion bounds. As a stepping stone, this paper further presents the *Clustered k-Exclusion Independence-Preserving Protocol* (CKIP), the first asymptotically optimal independence-preserving k-exclusion lock for clustered scheduling. The GIPP and the CKIP rely on allocation inheritance (a.k.a. migratory priority inheritance) as a key mechanism to accomplish independence preservation.

2012 ACM Subject Classification Computer systems organization → Real-time systems

Keywords and phrases multiprocessor real-time locking, nested locking, independence preservation, suspension-oblivious analysis, priority inversion, asymptotically optimal blocking, RNLP, OMIP

Related Version The conference version of this report appears in *Proceedings of the 32nd Euromicro Conference on Real-Time Systems* (ECRTS 2020).

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803111).

1 Introduction

From a practical point of view, any effective multiprocessor real-time locking protocol should avoid some obvious pitfalls:

1. Non-conflicting accesses to different resources should *not* be needlessly serialized.
2. Tasks should *not* be delayed due to contention for resources they do not even access.
3. A real-time locking protocol should *not* make it impossible to provision latency-sensitive tasks carefully designed to not require any shared resources (such as critical interrupt handlers with stringent sub-millisecond deadlines).
4. Worst-case blocking should *not* be exponential.

It is not difficult to see how a protocol that fails to meet these requirements would result in costly and inefficient over-provisioning. It may thus come as a surprise that *no multiprocessor real-time locking protocol in the published literature satisfies all four properties!*

The reason, however, is all the more understandable: these innocuous-looking requirements translate to well-known real-time locking protocol properties that are difficult to ensure by themselves, let alone *jointly* in a single protocol. In particular, Requirement 3 rules out any locking protocol that relies on the non-preemptive execution of critical sections, a trait of virtually all spin-lock protocols [8]. Requirement 1 implies that a protocol must support *fine-grained nested locking* [3, 30, 31]—that is, tasks must be able to incrementally lock additional resources while already holding some other shared resources—because the alternative, namely coarse-grained *group locking* [4], serializes even trivially non-conflicting requests for resources in the same group. Fine-grained nested real-time locking, however, is a notoriously difficult problem [3, 8, 30], and easily gives rise to blocking bounds that are exponential in the number of simultaneously acquired resources [8, 19, 30]. In fact, it is a fundamental algorithmic challenge to ensure both Requirements 1 and Requirements 4 in a single protocol. The only known protocol to surmount this challenge is Ward and Anderson's *Real-Time Nested Locking Protocol* (RNLP) [31, 32], and actually does so with *asymptotically optimal* bounds on *priority inversion blocking* [10, 31].

The RNLP, in turn, does not satisfy Requirement 2. As we discuss in more detail in Section 2, the RNLP relies on a *token lock* that regulates contention for shared resources, an ingenious element of the RNLP's design that ensures its asymptotic optimality. However, in its configuration for suspension-based locking (under "suspension-oblivious analysis," see Section 2), this token lock becomes a global bottleneck that causes tasks to delay each other even if they do not share any resources.

To satisfy Requirements 2 and 3, a locking protocol must temporally isolate tasks from each other when they do not access the same resources, which is known as *independence preservation* [6], a concept we discuss in detail in Section 2. The only protocol to date to realize independence preservation for clustered scheduling is the $\mathcal{O}(m)$ *Independence-Preserving Protocol* (OMIP) [6]. However, the OMIP as originally proposed fails to satisfy Requirement 1 since it can realize nested locking only through group locks—and if the OMIP is extended to permit fine-grained locking, it fails to satisfy Requirement 4 due to its FIFO queuing structure, which gives rise to exponential worst-case blocking [30].

Seemingly, the satisfaction of one of the four requirements comes at the cost of another. Is this a fundamental limitation? Is it perhaps *impossible* to satisfy all four requirements at once? As we show in this paper, the answer to both questions is *no*—it is in fact possible to combine fine-grained nesting, independence preservation, and asymptotically optimal pi-blocking in a single protocol, which we demonstrate by constructing the first such protocol.

In **related work**, the *Priority Inheritance Protocol* (PIP) [16, 28, 29] provides independence preservation, but only on uniprocessors or globally-scheduled systems, and the multiprocessor variant [16, 36] does not support nested critical sections. The *Flexible Multiprocessor Locking Protocol* (FMLP) [4] likewise is independence-preserving only under global scheduling, and only supports group locks [4, 36]. The *Multiprocessor Bandwidth Inheritance Protocol* (MBWI) [18, 19] and the *Multiprocessor Resource Sharing Protocol* (MrsP) [14] both allow for fine-grained nested locking. Unfortunately, they are subject to the exponential blow-up in blocking times described by Takada and Sakamura [30]. Several variants of the RNLP [31, 32] have been introduced in recent years to enable reader-writer synchronization [33], to provide contention-sensitive pi-blocking bounds [23], and to reduce implementation overheads in the locking protocol itself by means of a fast path [26] and lock servers [25]. However, none of these variants removes the *algorithmic* bottleneck of a single, shared token lock. For further discussion of the larger area of multiprocessor real-time locking protocols, we refer the interested reader to a recent comprehensive survey [8].

The **contributions** of this paper are as follows. First, we examine what it means to be independence-preserving in the presence of nested locking (Section 3), and the ensuing implications on asymptotic pi-blocking bounds (Section 3.1). Our main contribution is the *Group Independence-Preserving Protocol* (GIPP), the first asymptotically optimal, independence-preserving, real-time fine-grained nested locking protocol for clustered scheduling under suspension-oblivious analysis (Section 4). In other words, the GIPP is the first multiprocessor real-time locking protocol that meets all of the desirable Requirements 1–4. To realize the GIPP, we develop and analyze a novel *Clustered k-Exclusion Independence-Preserving Protocol* (CKIP), an asymptotically optimal independence-preserving k-exclusion lock for clustered scheduling (Section 4.1). Lastly, we provide a fine-grained pi-blocking analysis of the GIPP using a state-of-the-art blocking analysis method based on linear programming (Section 5), and present an empirical evaluation that shows the GIPP to perform favorably in comparison to both the OMIP and the RNLP across a wide range of workloads (Section 6).

2 Background and Definitions

We assume the sporadic task model with n tasks $\tau = \{T_1, \ldots, T_n\}$ scheduled on m identical processors. Tasks are executed as a series of *jobs*, and we use J_i to denote a job of T_i. Each T_i is characterized by a *worst-case execution time* (WCET) e_i, a *period* p_i (*i.e.*, the minimum arrival separation between jobs), and a *relative deadline* d_i. We assume implicit deadlines in this work, *i.e.*, $d_i = p_i$, but the derived results do not depend on this constraint.

A job is said to be *pending* from the time it arrives until the time it completes. While a job is pending, it can be in one of two states: a *ready* job can be scheduled on a processor, whereas a *suspended* job cannot be scheduled. We assume that jobs do not self-suspend, and that all suspensions are due to interactions with the locking protocol(s) of the system.

Shared Resources Tasks compete for a set of q serially-reusable shared resources $\Gamma = \{\ell_1, \ldots, \ell_q\}$. Each task T_i accesses a possibly empty subset $\gamma_i \subseteq \Gamma$ of the shared resources in the system. A *locking protocol* arbitrates requests from tasks for the shared resources in Γ such that no shared resource is held at the same time by two different tasks.

We say J_i *requires* a shared resource ℓ_a at the first instant access to ℓ_a is required for the continued execution of J_i. Once J_i requires ℓ_a, it *issues* a request $\mathcal{R}$ to the locking protocol to *acquire* ℓ_a. $\mathcal{R}$ is said to be *unsatisfied* from the time it is issued until J_i acquires ℓ_a, at which point $\mathcal{R}$ is said to be *satisfied*. $\mathcal{R}$ becomes *complete* when J_i *releases* ℓ_a, which is also when J_i no longer requires ℓ_a. Conversely, we say that $\mathcal{R}$ is *incomplete* from the time it is issued until it is completed. While J_i waits to acquire ℓ_a, it is said to make *progress* if (one of) the job(s) that prevent(s) J_i from acquiring ℓ_a is scheduled. Any method employed by a locking protocol to ensure that a job makes progress is called a *progress mechanism*.

If a J_i issues a request $\mathcal{R}$ for a shared resource ℓ_a while holding no other shared resources, then $\mathcal{R}$ is said to be an *outermost request*. Conversely, if J_i issues $\mathcal{R}$ for a shared resource ℓ_b while holding ℓ_a, then $\mathcal{R}$ is said to be a *nested request*. We do not require that requests are properly nested; it is possible for J_i to acquire ℓ_a, and then ℓ_b via a nested request, but release ℓ_a before releasing ℓ_b, as seen in Figure 1.

We construct a strict (irreflexive) partial ordering $\succ$ on Γ from the behavior of the tasks in τ: let $\ell_a \succ \ell_b$ iff there exists a task that requests ℓ_b while holding ℓ_a. It follows that we do not permit workloads where $\ell_a \succ \ell_b$ and $\ell_b \succ \ell_a$ both hold (which precludes deadlock).

Consider Figure 1. Let t_1 be the time that J_i issues an outermost request for a shared resource ℓ_a, and t_4 be the next point in time where both the outermost request is complete, and J_i holds no other shared resources. We call the part of J_i's execution in the time interval $[t_1, t_4)$ an *outermost critical section*. We define the *critical section length* to be the time required to execute a critical section in the absence of any blocking, and use $L_{i,a}$ to denote the length of J_i's longest outermost critical section that begins with an outermost request for ℓ_a. Note that $L_{i,a} \leq t_4 - t_1$, as J_i can experience scheduling or blocking delays during the execution of

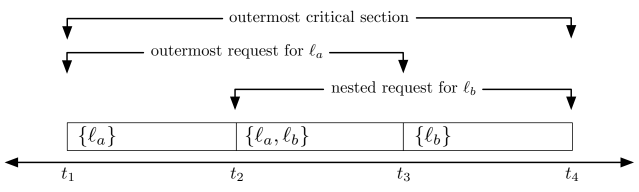

outermost critical section
outermost request for ℓ_a
nested request for ℓ_b
$\{\ell_a\}$
$\{\ell_a, \ell_b\}$
$\{\ell_b\}$
t_1
t_2
t_3
t_4

Figure 1 An outermost critical section spanning from time t_1 until time t_4. It begins with an outermost request for ℓ_a at time t_1, which completes at time t_3. The nested request for ℓ_b begins at time t_2 and completes at time t_4.

its outermost critical section. The maximal critical section length of J_i is $L_i^{\max} = \max_a L_{i,a}$ and the maximal critical section length among all tasks is $L^{\max} = \max_i L_i^{\max}$.

Scheduling We target *clustered scheduling* in this work. Under clustered scheduling, the m processors in the system are grouped into disjoint subsets of size c called *clusters*. For simplicity, we assume $m = k \cdot c$ where $k \in \mathbb{Z}^+$. Each task is statically assigned a *home cluster*, denoted by $C(T_i)$, which it will be scheduled in. The tasks in each cluster are scheduled by a global scheduling algorithm applied to the processors in that cluster. Global and partitioned scheduling are special cases of clustered scheduling where $c = m$ and $c = 1$, respectively.

We assume the use of *Job-Level Fixed Priority* (JLFP) scheduling algorithms such as *Earliest-Deadline First* (EDF) scheduling or *Fixed-Priority* (FP) Scheduling. The priorities assigned to jobs are assumed to be *unique* in each cluster, with any ties broken in favor of lower-indexed jobs. A job J_i has both an *effective priority* and a *base priority*. J_i's base priority, which is determined by the scheduling algorithm, never changes, but its effective priority may change due to interactions with a locking protocol; the scheduler uses J_i's current effective priority when scheduling jobs. We let $\mathrm{H}(J_i, t)$ be a predicate that indicates whether J_i is among the c highest effective-priority pending jobs at time t in its home cluster.

Priority Inversion Intuitively, a *priority inversion* is said to occur when the execution of a higher-priority job is delayed due to the execution of a lower-priority job [28, 29]. A typical example of this occurs when a lower-priority job holds a shared resource that a higher-priority job is requesting, and so the higher-priority job's execution is delayed until the resource is released. We refer to this type of blocking as *priority inversion blocking* (pi-blocking).

Under *suspension-oblivious analysis* (s-oblivious analysis) it is assumed that tasks never self-suspend (even though they may), and any self-suspension is treated as execution time. Conversely, *suspension-aware analysis* (s-aware analysis) explicitly accounts for self-suspensions. These two methods of analysis yield different lower-bounds on the pi-blocking incurred by a job due to requests for shared resources, which are $\Omega(m)$ and $\Omega(n)$, respectively [10]. Locking protocols that ensure $\mathcal{O}(m)$ and $\mathcal{O}(n)$ per-job pi-blocking under s-oblivious analysis and s-aware analysis, respectively, are said to be *asymptotically optimal* [10].

We focus on s-oblivious analysis in this work, and adapt the definition for s-oblivious priority inversion under clustered scheduling from previous work [6].

▶ **Definition 1.** J_i incurs an *s-oblivious priority inversion* at time t iff J_i is not scheduled and its priority is among the top c priorities of pending jobs in cluster $C(T_i)$, *i.e.*, if $\mathrm{H}(J_i, t)$.

Let b_i denote the maximum amount of s-oblivious pi-blocking that any job of T_i incurs. When deriving asymptotic bounds on b_i, we consider $L^{\max}$ to be a constant (i.e., not a function of m nor n). Following prior work [8, 10, 27, 28, 29], we consider pi-blocking to be *bounded* only if no b_i depends on any e_i (the e_i parameter is *not* considered to be a constant).

Independence Preservation The high-level idea of independence preservation is that tasks are isolated from "unrelated" critical sections. This can be easily pictured for locking protocols that do not permit nested locking: if a task never requests a shared resource ℓ_a, then it incurs no pi-blocking as a result of requests by other tasks for ℓ_a. This is of particular importance when considering latency-sensitive tasks, defined as follows.

▶ **Definition 2.** A task T_i is said to be *latency-sensitive* if its *slack*, the difference between its relative deadline and WCET, is less than the length of the longest critical section of some other task, i.e., $d_i - e_i < L^{\max}$.

If critical sections are permitted to execute non-preemptively, then a job of a latency-sensitive task necessarily misses its deadline if its release coincides with a lower-priority task executing non-preemptively for $L^{\max}$ time units, thus providing clear motivation for independence preservation. Prior work introduced the notion of independence preservation [6] among tasks under the assumption that nested resource requests are never made. For clarity, and to build upon it later, we restate the definition here.

▶ **Definition 3.** Let $b_{i,a}$ denote the maximum pi-blocking incurred by J_i due to requests for a shared resource ℓ_a, and $N_{i,a}$ be the number of times J_i requests ℓ_a. Under s-oblivious analysis, a locking protocol is *non-nested independence-preserving* iff $N_{i,a} = 0$ implies $b_{i,a} = 0$.

Priority Donation The progress mechanism *Replica-Request Priority Donation* (RRPD) [34] was introduced to realize the *Replica-Request Priority Donation Global Locking Protocol* (R^{2}DGLP) [34], a real-time k-exclusion lock for globally-scheduled systems. RRPD is a modification of the earlier *Job-Release Priority Donation* (JRPD) progress mechanism [11]; RRPD has a job donate its priority upon requesting a resource, whereas donation happens upon arrival (*i.e.*, release) under JRPD. Unlike JRPD, which was designed for clustered systems, RRPD relies on the ability to compare priorities among all jobs. Thus, RRPD applies only to globally-scheduled systems, as analytically speaking, numeric priority values are incomparable across clusters. This trade-off allows the R^{2}DGLP to realize non-nested independence preservation as jobs only donate their priority upon resource request. We revisit RRPD further in Section 4, and the R^{2}DGLP in Section 4.1.

Nested Locking Protocols The way locking protocols support nesting can be divided into two broad categories: *coarse-grained* locking and *fine-grained* locking. The FMLP [4] is an example of a real-time locking protocol that employs coarse-grained nested locking. Under the FMLP, resources are split into groups, and each group has a corresponding *group lock*. A job that holds a group lock has mutually-exclusive access to all the resources in the corresponding group; while simple, the loss in parallelism is clear.

In contrast to coarse-grained locking, fine-grained locking allows shared resources to be acquired incrementally. For example, the RNLP [31] allows jobs to issue nested requests for resources as they are needed, which provides more opportunities for parallelism when compared to simple group locks. However, the increased potential for parallelism comes at the cost of more complicated protocol rules and data structures when compared to group locks, as group locks can be realized with simpler non-nested locking protocols.

The RNLP [31] was a breakthrough in real-time nested locking, as it is the first, and to date only, asymptotically optimal fine-grained nested locking protocol for multiprocessor systems. It can be applied under clustered (and therefore global/partitioned) scheduling. The RNLP is in fact a "meta protocol" in the sense that it defines the properties that a *token lock* and a *request satisfaction mechanism* (RSM) must obey to realize an optimal nested locking protocol. The token lock restricts the number of jobs that can hold resources at any given time, and the RSM sequences requests of the token holders and ensures progress. The behavior of a particular *instantiation* (*i.e.*, a token lock/RSM combination) of the RNLP is largely determined by the progress mechanisms that token lock and RSM employ. Ward and Anderson demonstrate a number of possible instantiations of the RNLP [31]. When instantiated for s-oblivious analysis, the token lock is configured to provide m tokens. We examine the rules, requirements, and structure of the RNLP further in Section 4.

Summary If shared resources that will be held at the same time are protected by a group lock, then both the OMIP and R^{2}DGLP can realize asymptotically optimal coarse-grained nested locking under clustered and global scheduling, respectively, while still remaining non-nested independence-preserving.

Non-nested independence preservation is not trivially realized with the RNLP in the absence of nested locking. Fundamentally, the use of a token lock that arbitrates access to m tokens (and thus restricts the number of resource-holding jobs to m) precludes non-nested independence preservation; when all tokens in the system are held, a job must wait to acquire a token, even if the resource it requires is never accessed by any other task. In fact, to the best of our knowledge, prior work has not yet considered what it means to be independence-preserving in the context of fine-grained nested locking.

3 Nested Independence Preservation

The notion of independence preservation introduced in Definition 3 does not directly apply to nested locking, and there exists more than one way to generalize the notion in a conceptually analogous way, depending on when exactly resources involved in nesting are considered to be "related" (i.e., when they are considered "non-independent"). We consider two possible definitions in the following that we consider to be the most natural ways of expressing the idea.

3.1 Outer-Lock Independence Preservation

The core idea behind *outer-lock independence preservation* is that there is an asymmetric, transitive, and reflexive relation, which we call *dependence*, between a shared resource ℓ_a, and shared resources acquired via nested requests (with respect to an outer request for ℓ_a).

More precisely, for a shared resource $\ell_a \in \Gamma$, we say it *depends* on the set $[\ell_a]^{ol} = \{\ell_x \mid \ell_a \succ \ell_x\} \cup \{\ell_a\}$. Similarly, a task T_i depends on the set of shared resources $D_i^{ol} = \bigcup_{\ell_a \in \gamma_i} [\ell_a]^{ol}$. Based on this precise notion of dependence, we define outer-lock independence preservation.

▶ **Definition 4.** Let $b_{i,a}$ denote the maximum pi-blocking incurred by J_i due to requests by any task for a shared resource ℓ_a. Under s-oblivious analysis, a locking protocol is *outer-lock independence-preserving* iff $\ell_a \notin D_i^{ol}$ implies $b_{i,a} = 0$.

Outer-lock independence preservation as a notion for nested independence preservation has a fundamental impact on the pi-blocking incurred by a job under s-oblivious analysis. In fact, it turns out that for a large class of locking protocols (that arguably includes all "reasonable" locking protocols), the per-request pi-blocking bound is necessarily non-optimal under *rate monotonic* (RM), *deadline monotonic* (DM), or EDF scheduling (and s-oblivious analysis). Our proof of the non-optimality of outer-lock independence-preserving locking protocols requires the following seemingly obvious property.

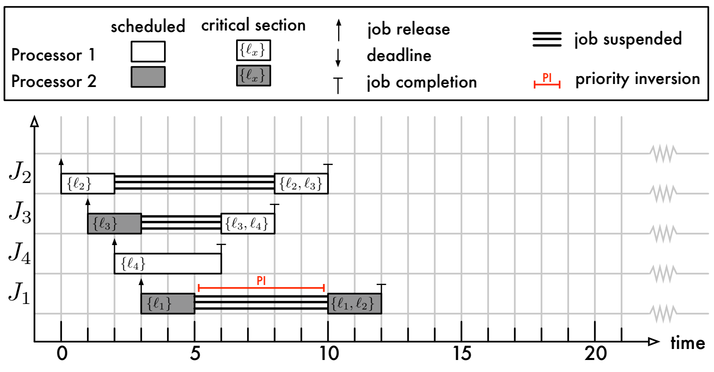


Figure 2 One possible G-FP schedule of $\tau^{ol}(4)$ on $m = 2$ processors. The jobs are in ascending priority from top to bottom (i.e., J_2 has the lowest priority, and J_1 has the highest priority).

▶ **Definition 5.** Let $\Gamma' \subseteq \Gamma$ denote the set of shared resources currently held by all tasks in the system. A locking protocol is *non-procrastinating* if any request for a shared resource (by one of the c highest-priority pending jobs in each cluster) is satisfied immediately if $|\Gamma'| = 0$.

We are not aware of *any* real-time locking protocol in the literature that does not satisfy a request $\mathcal{R}$ for a shared resource by one of the c highest-priority pending jobs when $|\Gamma'| = 0$. If we assume non-clairvoyance, and that tasks are sporadic (i.e., we cannot predict future job arrivals), then delaying the satisfaction of $\mathcal{R}$ is tantamount to willingly wasting CPU time; this is in direct contradiction to one of the most important goals of an effective real-time locking protocol. Non-procrastination is also a fairly weak property as it does not impose restrictions on how to arbitrate access to resources once contention is present.

We use a parameterized task set to lower-bound pi-blocking under RM, DM, and EDF.

▶ **Definition 6.** Let $\tau^{ol}(n) = \{T_1, \ldots, T_n\}$ be a task set of n tasks that share n resources $\{\ell_1, \ldots, \ell_n\}$, where $n \geq m \geq 2$, with the following properties: **(i)** $\ell_1 \succ \ell_2 \succ \ldots \succ \ell_{n-1} \succ \ell_n$; **(ii)** $\forall_{1 \leq i \leq n}\ e_i = 4$; **(iii)** $\forall_{1 \leq i \leq n}\ p_i = d_i = e_i \cdot n \cdot i$; **(iv)** $\forall_{1 \leq i < n}$ jobs of T_i require $\{\ell_i\}$ during the first two units of their execution, and then $\{\ell_i, \ell_{i+1}\}$ during the last two units of their execution; **(v)** jobs of T_n require $\{\ell_n\}$ throughout the four units of their execution.

▶ **Theorem 7.** *There exists an arrival sequence of $\tau^{ol}(n)$ such that $\max_{T_i \in \tau^{ol}(n)} b_i = \Omega(n)$ under s-oblivious analysis for any suspension-based incremental locking protocol that is non-procrastinating and outer-lock independence-preserving, when scheduled under RM, DM, or EDF scheduling (with respect to each cluster).*

Proof. Let $a_{i,1}$ denote the first arrival of T_i. Consider the arrival sequence of $\tau^{ol}(n)$ where $a_{i,1} = i - 2$ for $2 \leq i \leq n$ and $a_{1,1} = n - 1$. An example of $\tau^{ol}(4)$ with this arrival sequence is depicted in Figure 2. At time $t = 0$, J_2 requests and acquires ℓ_2, as we assume the use of a non-procrastinating locking protocol. At time $t = 1$, a request for ℓ_3 is made by J_3. If J_3 does not acquire ℓ_3 (and is therefore not scheduled) at $t = 1$, then the blocking that results from delaying J_3's request would result in a violation of outer-lock independence preservation as we would then have $b_{3,2} > 0$ despite $\ell_2 \notin D_3^{ol}$. The same argument analogously applies to all jobs released up until $t = n - 1$ when J_1 arrives and issues a request $\mathcal{R}_1$ for ℓ_1. There are then two cases to consider: $\mathcal{R}_1$ is satisfied immediately, or it is satisfied at a later time.

In the first case $\mathcal{R}_1$ is satisfied immediately and J_1 issues a nested request $\mathcal{R}_2$ for ℓ_2 at time $t = n + 1$. The maximum number of units of execution completed for jobs $J_2, \ldots, J_n$ up to $t = n+1$ is $(n-1)\cdot 2+1 = 2n-1$ for any $m \geq 2$. This is because jobs $J_2, \ldots, J_{n-1}$ can execute for at most 2 units of time before suspending due to a nested request for an already held resource, and because J_n can execute for at most 3 units of time until $\mathcal{R}_2$ is issued. Therefore, at the time $\mathcal{R}_2$ is issued, there are $(\sum_{i=2}^{n} e_i) - (2n-1) = (n-1) \cdot 4 - (2n-1) = 2n-3$ units of execution left before jobs $J_2, \ldots, J_n$ complete and ℓ_2 becomes available for acquisition by J_1. Furthermore, as jobs $J_2, \ldots, J_{n-1}$ are all waiting for the job with the next-highest index to release a resource, their executions are serialized. Thus J_1 incurs at the very least $2n - 3 = \Omega(n)$ time units of s-oblivious pi-blocking while waiting to acquire ℓ_2.

In the second case, we assume $\mathcal{R}_1$ is satisfied at time $t = n - 1 + \epsilon$ where $\epsilon > 0$ (*i.e.*, not immediately). Then, J_1 would begin to incur s-oblivious pi-blocking at $t = n - 1$, as it is the highest-priority job and not scheduled. This situation cannot result in a reduction of the s-oblivious pi-blocking that J_1 incurs in the first case because **(i)** J_1 is the highest-priority job by construction, and **(ii)** that the serialization of executions described in the first case enforces a minimum of $2n - 3$ units of execution before ℓ_2 is released. Therefore, the asymptotic lower-bound in the first case applies, as J_i still incurs a minimum of $2n - 3 = \Omega(n)$ units of s-oblivious pi-blocking while waiting to acquire ℓ_2. ◀

To conclude, under a standard set of assumptions and commonly used scheduling algorithms, any outer-lock independence-preserving protocol is necessarily non-optimal with respect to s-oblivious pi-blocking. In the rest of this paper, we focus on an alternative definition for nested independence preservation, which we call group independence preservation.

3.2 Group Independence Preservation

With group independence preservation, the relationships that exist among shared resources and tasks are defined by relaxing when resources are considered to be non-independent.

Let $\circ$ be a symmetric relation on the set of shared resources Γ. For any $\ell_a \in \Gamma$, let $\ell_a \circ \ell_a$, and for any $\ell_b, \ell_c \in \Gamma$ let $\ell_b \circ \ell_c$ if $\ell_b \succ \ell_c$ or $\ell_c \succ \ell_b$. The transitive closure of $\circ$ forms an equivalence relation on the resources in Γ, which we denote with $\sim$. Then the equivalence class $g(\ell_a) = \{\ell_x \in \Gamma \mid \ell_a \sim \ell_x\}$ is the set of resources that ℓ_a is *associated* with.

We refer to these equivalence classes as *groups*, and let $G = \{g_1, \ldots, g_r\}$ be the set of resource groups in the system. From the definition of a group, it naturally follows that the groups in G are disjoint, and that their union yields Γ. This definition of groups matches the notion of resource groups used in the FMLP [4]. Based on this notion of resource groups, we say that a task T_i is *associated* with the shared resources $D_i = \bigcup_{\ell_a \in \gamma_i} g(\ell_a)$.

▶ **Definition 8.** Let $b_{i,a}$ denote the pi-blocking incurred by J_i due to requests by any task for a shared resource ℓ_a. Under s-oblivious analysis, a locking protocol is *group independence-preserving* iff $\ell_a \notin D_i$ implies $b_{i,a} = 0$.

Stated differently, group independence is preserved if the overall s-oblivious pi-blocking $b_i = \sum_a b_{i,a}$ of each task does not depend on resources that the task is not associated with.

4 The Group Independence-Preserving Protocol

We show that group independence-preserving protocols do not necessarily suffer from the $\Omega(n)$ s-oblivious pi-blocking bound seen with outer-lock independence preservation. We demonstrate this through the construction of group independence-preserving fine-grained nested locking protocol that is asymptotically optimal under s-oblivious analysis: the *Group Independence-Preserving Protocol* (GIPP). We review the necessary background of the RNLP and RRPD in this section required to realize the GIPP, and then give a high-level overview of the GIPP before constructing its components in subsequent sections.

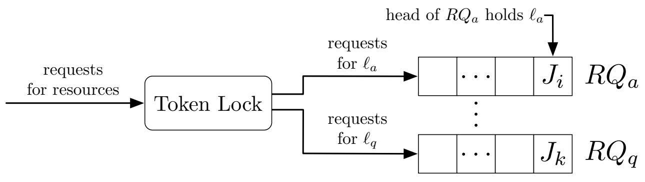


Figure 3 Under the RNLP, a job J_i that requests a resource ℓ_a first competes for a token. Once J_i acquires a token, a timestamp $ts(J_i)$ is recorded, and the request is enqueued into the priority queue RQ_a ordered by the token-acquisition timestamps. The job of the request that occupies the head of a queue holds the corresponding shared resource.

RNLP As a brief reminder, an instantiation of the RNLP consists of a token lock and an RSM. We speak in the context of a job J_i that requires a shared resource $\ell_a \in \Gamma$ when reviewing the following rules. Under the rules of the RNLP [31, Section 3], J_i first competes for a token λ in the token lock before it can actually issue a request for ℓ_a. Once J_i acquires λ, a timestamp $ts(J_i)$[1] of the current time is recorded. Conceptually, J_i now enters the RSM, and is placed in a priority queue RQ_a ordered by increasing timestamp[2]; such a priority queue exists for each shared resource in Γ. J_i waits until it becomes the head of RQ_a, and then it holds ℓ_a unless a job J_k holds a shared resource ℓ_b s.t. $\ell_b \succ \ell_a \land ts(J_k) < ts(J_i)$. Once J_i completes its request for ℓ_a it is dequeued from RQ_a and the new head of RQ_a (if any) acquires ℓ_a subject to the previous constraints. As requests need not be properly nested, J_i may continue to compete for shared resources in the RSM. Once J_i has completed its outermost critical section, it releases λ. This queuing structure is depicted in Figure 3.

The RNLP specifies properties [31] that a token lock and RSM must have to realize a valid instantiation of the RNLP. Of the following properties, **T1** and **T2** apply to the token lock, and **R1** applies to the RSM.

T1 There are at most c token-holding jobs per cluster at any time (and thus m system-wide).
T2 If a job is pi-blocked waiting for a token, then it makes progress (*i.e.*, the token holder is scheduled).
R1 If a job is pi-blocked by the RSM, then the job makes progress.

Finally, we restate one of the RNLP's theorems, as we use it to prove properties of the GIPP later in this work. We refer to the original RNLP paper for its proof [31].

▶ **Theorem 9** ([31, Theorem 1], paraphrased). *The maximum amount of pi-blocking in the RSM, when waiting is realized by suspending, is* $(m - 1) \cdot L^{max}$.

RRPD Under the rules of RRPD [34, Section 3], a job donates its priority upon requesting a resource, and as such only becomes a priority donor at most once per outermost critical section. Thus if a job does not request a resource, it never donates its priority to jobs requesting said resource. We speak in the context of a job J_i that requires a shared resource $\ell_a \in \Gamma$ when reviewing the following rules. Furthermore, we discuss RRPD in the context of a single cluster despite it having been designed for globally-scheduled systems, as we assume throughout this paper that clusters function independently with respect to RRPD. We refer the reader to Figure 4 for a visual depiction of the life-cycle of an RRPD request,

[1] The timestamps are assumed to be unique, and thus there is a total order on them.
[2] For clarity, the head of a queue RQ_a is the request with the oldest (*i.e.*, earliest) timestamp.

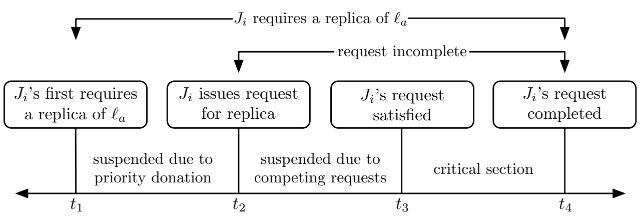

J_i requires a replica of ℓ_a
request incomplete
J_i's first requires a replica of ℓ_a
J_i issues request for replica
J_i's request satisfied
J_i's request completed
suspended due to priority donation
suspended due to competing requests
critical section
t_1
t_2
t_3
t_4

Figure 4 Life-cycle of a request under RRPD for a replica of a shared resource ℓ_a (adapted from [34]).

and the terminology used. Let $R(\ell_a, t)$ denote the (possibly empty) set of the c highest effective-priority jobs that require ℓ_a at time t. If J_i requires ℓ_a and $J_i \in R(\ell_a, t)$ then J_i may issue a request for ℓ_a, unless it becomes a priority donor first. J_i becomes the priority donor of a job J_d at time t_1 (see Figure 4) if **(i)** $J_i \in R(\ell_a, t)$, **(ii)** there are c jobs with an incomplete request for ℓ_a, and **(iii)** J_d is the lowest-effective priority job with an incomplete request for ℓ_a; while J_i donates its priority to J_d in the interval $[t_2, t_4)$ it is suspended and assumed to have no effective-priority. Should J_i be displaced from $R(\ell_a, t)$ while donating its priority to J_d by a job J_k, then J_i ceases to be a priority donor and J_k becomes J_d's priority donor. Finally, if J_i is J_d's priority donor when J_d completes its outermost critical section, then J_i ceases to be a priority donor.

We now restate two of the RRPD's lemmas as we use them later when constructing the components required to realize the GIPP. Their proofs are available in the original work [34].

▶ **Lemma 10** ([34, Lemma 2 (adapted for clustered scheduling)]). *There are at most c jobs per cluster with an incomplete request for a replica of a shared resource ℓ_a at any time.*

▶ **Lemma 11** ([34, Lemma 4]). *Under RRPD, if a job J_i that requires a replica of ℓ_a is pi-blocked waiting for a replica of ℓ_a it either has an incomplete request for a replica of ℓ_a or it is a priority donor.*

GIPP At a very high level, the GIPP works as follows. For each group, we instantiate a separate instance of the RNLP. Crucially, the choice of token lock and RSM used to instantiate each instance of the RNLP must not violate group independence preservation, that is, any progress mechanisms employed must lend themselves to group independence preservation. Progress mechanisms like priority boosting that rely on elevating a job's priority can cause jobs that never request shared resources to incur release-blocking, which precludes the property of group independence preservation; this is highly undesirable in the presence of latency-sensitive tasks [6]. Furthermore, progress mechanisms that rely on the ability to directly compare priorities across clusters can result in unbounded pi-blocking (i.e., the blocking depends on some other task's WCET) [6]. The challenges to realizing the GIPP are then to **(i)** construct an appropriate token lock and RSM, **(ii)** prove the token lock and RSM satisfy the required properties of the RNLP, **(iii)** prove the optimality of the GIPP under s-oblivious analysis, and **(iv)** prove that the GIPP is group independence-preserving.

We construct the token lock in Section 4.1, and the RSM in Section 4.2. We then show how to use these two components to realize the GIPP in Section 4.3.

4.1 An Independence-Preserving k-Exclusion Locking Protocol

To realize the GIPP we use a single token lock that is common to all the instantiations of the RNLP. If there are r groups (and therefore r instances of the RNLP), then a token lock that arbitrates access to r distinct token types, where each token type has m replicas, will suffice. However, as stated earlier, any such token lock must lend itself to independence preservation.

To the best our knowledge, no such k-exclusion locking protocol (i.e., token lock) exists for clustered scheduling. To realize such a token lock, we generalize the R^2DGLP [34], which satisfies the requirements just described, with the exception that it was designed for globally-scheduled systems. Ward et al. use the term replica instead of token when discussing shared resources in the context of RRPD and R^2DGLP; we use the terms interchangeably.

As discussed in Section 2, the R^2DGLP uses RRPD as a progress mechanism. Thus, to generalize the R^2DGLP to clustered scheduling, the requirement that priorities across all jobs are comparable must be lifted. Additionally, RRPD alone is not enough to ensure progress [34], which means that replica-holders (i.e., token holders) are not guaranteed to be scheduled without the aid of an additional progress mechanism. The R^2DGLP solves this with priority inheritance, as the protocol targets globally-scheduled systems. However, the R^2DGLP does not strictly mandate the use of priority inheritance, instead, any locking protocol that utilizes RRPD must satisfy the following property [34, Section 3].

P1 A job J_i with an incomplete replica request makes progress (*i.e.*, either J_i is scheduled itself or the replica-holding job that J_i is waiting for is scheduled) if J_i has sufficient priority to be scheduled in $C(T_i)$.

We introduce the *Clustered k-Exclusion Independence-Preserving Protocol* (CKIP) as a generalization of R^2DGLP that is non-nested independence preserving, asymptotically optimal under s-oblivious analysis, and employable under clustered scheduling. The CKIP is realized by having tasks compete amongst each other in their home clusters under the rules of RRPD, but not across clusters. This is possible as priorities can be directly compared within each cluster. However, this means that priority inheritance can no longer be used to ensure that replica holders make progress. To this end, we employ *allocation inheritance* [21, 22, 20] (sometimes referred to as *migratory priority inheritance* [6, 12]), an independence-preserving progress mechanism that works across clusters, and which is also used in the OMIP [6]. We define allocation inheritance in the context of the CKIP and GIPP as follows.

▶ **Definition 12** (allocation inheritance)**.** Let J_i be a job that holds a replica of a shared resource ℓ_a that has $k \geq 1$ replicas, and W_i be the set of jobs across all clusters waiting to a acquire a replica of ℓ_a. Under allocation inheritance (AI), if J_i is not scheduled and there exists a job $J_k \in W_i \cup \{J_i\}$ that has sufficient priority to be scheduled in $C(T_k)$, then J_i migrates to $C(T_k)$ (if necessary) and runs with J_k's priority. While J_i executes in $C(T_k)$ with J_k's priority, we call J_k an *allocation donor.* Once J_i releases ℓ_a, it migrates back to $C(T_i)$ (if necessary) and resumes execution when it has sufficient priority. Finally, J_i's allocation donor (if any) ceases to be an allocation donor when J_i releases ℓ_a.

Now armed with an independence-preserving progress mechanism [6],[3] we can construct the CKIP by adapting the rules that define the R^2DGLP [34, Section 4]. The rules and structure of the CKIP differ enough from the R^2DGLP that its rules do not directly apply. Therefore, we present the modified rules and structure in full below.

Structure Tasks compete for a set of q shared resources $\Gamma = \{\ell_1, \ldots, \ell_q\}$ where each resource ℓ_a has $k \geq 1$ replicas. Nested requests are not permitted. Each of the k replicas has an

[3] One might wonder whether the CKIP and hence the GIPP could also be realized with a progress mechanism that does not result in inter-cluster migrations. Unfortunately, that is not the case: it is generally impossible for a protocol to ensure bounded pi-blocking, be independence-preserving, *and* avoid inter-cluster migrations at the same time even in the non-nested case [6].

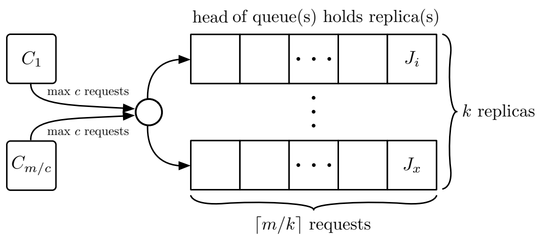


Figure 5 Queuing structure of the CKIP, which has been adapted from the R^{2}DGLP [34]. At any time, each of the m/c clusters has at most c incomplete requests for a replica of a given shared resource. Requests are enqueued into the replica queue with the least number of requests in it.

associated FIFO queue of size $\lceil m/k \rceil$ that jobs are placed in when requesting a replica; we use KQ$_a$ to refer to any one of the queues for ℓ_a. The queuing structure of the CKIP closely resembles the R^{2}DGLP and is depicted in Figure 5. The following rules for CKIP focus on a single replicated resource $\ell_a \in \Gamma$, though they directly apply to all resources in Γ.

K1 Jobs issue requests subject to the rules of RRPD. When J_i requests ℓ_a, it is enqueued into a KQ$_a$ with the fewest number of jobs in it, and suspends while it waits.
K2 J_i's request for ℓ_a is satisfied when it becomes the head of KQ$_a$, and thus becomes ready.
K3 While J_i is the head of KQ$_a$, it benefits from AI, but only with respect to the other jobs in the same KQ$_a$ as J_i (*i.e.*, W_i is comprised of the jobs in KQ$_a$ that J_i is the head of).
K4 When J_i's request for ℓ_a is completed, it is dequeued from KQ$_a$ and the new head (if any) acquires the replica. If J_i had benefited from AI, it returns to its home cluster and assumes its former (possibly donated) priority. If J_i has a priority donor due to RRPD in $C(T_i)$, the donor may now issue a request subject to the rules of RRPD.

▶ **Lemma 13.** *Rule **K3** ensures property **P1**.*

Proof. If J_i in KQ$_a$ has sufficient priority to be scheduled in $C(T_i)$, then under AI, the head of KQ$_a$ can migrate to $C(T_i)$ and execute with J_i's priority (if necessary). Therefore, the replica-holder is scheduled and J_i makes progress. ◀

▶ **Lemma 14.** *A job J_i that incurs pi-blocking while acting as a priority donor under the rules of RRPD makes progress.*

Proof. Let J_x be the job that J_i donates its priority to. Then J_x has an incomplete request for a replica of a shared resource ℓ_a that both jobs require. Because J_i has sufficient priority to be scheduled in $C(T_i)$—otherwise it would not incur s-oblivious pi-blocking—J_x does as well, as $C(T_i) = C(T_x)$ and since J_x receives its effective priority from J_i. Therefore, J_x makes progress by Lemma 13, which means J_i does as well. ◀

▶ **Lemma 15.** *The CKIP ensures property **T1** with respect to each replicated resource.*

Proof. RRPD is orchestrated on a per-cluster basis under the CKIP, and so we can reason about each cluster individually as if it were a lone globally-scheduled system with c processors. Then, by Lemma 10 there are at most c incomplete requests for a given replicated resource per cluster, and therefore at most m across a clustered system as $\frac{m}{c} \cdot c = m$. ◀

▶ **Lemma 16.** *The CKIP ensures property **T2**.*

Proof. By Lemma 11, a job J_i that requires a replica of a shared resource ℓ_a has an incomplete request, or is a priority donor. By Lemma 14, J_i makes progress while acting as a priority donor, and by Lemma 13, J_i makes progress while it has an incomplete request. Thus, J_i makes progress if it incurs pi-blocking while waiting for a token (*i.e.*, replica of ℓ_a). ◀

We next establish the CKIP's bound of $O(m/k)$ maximum s-oblivious pi-blocking, which follows analogously to the proof of asymptotic optimality of the R^{2}DGLP [34] due to their similar structures and acquisition rules.

▶ **Lemma 17.** *The length of KQ_a does not exceed $\lceil m/k \rceil$.*

Proof. By rule K1, a request is always enqueued into (one of) the queue(s) containing the fewest number of requests. Thus, if KQ_a were to be longer than $\lceil m/k \rceil$, there would necessarily need to be more than m incomplete requests for the shared resource. However, by Lemma 10, there are never more than c incomplete requests per cluster, and as there are $\frac{m}{c}$ clusters in total, there are never more than $\frac{m}{c} \cdot c = m$ incomplete requests for ℓ_a at any point in time. ◀

▶ **Lemma 18.** *A job J_i incurs at most $(\lceil m/k \rceil - 1) \cdot L^{max}$ s-oblivious pi-blocking in KQ_a.*

Proof. By Lemma 17, there are at most $\lceil m/k \rceil - 1$ jobs ahead of J_i in KQ_a, and by Lemma 13 the head of KQ_a is scheduled if any other job in KQ_a has sufficient priority to be scheduled in its own cluster. Therefore, a job incurs at most $(\lceil m/k \rceil - 1) \cdot L^{\max}$ s-oblivious pi-blocking while in KQ_a. ◀

We restate the following property of RRPD in preparation for the CKIP's proof of asymptotic optimality under s-oblivious analysis, and refer the reader to the original paper [34] for the proof of Lemma 19.

▶ **Lemma 19** ([34, Lemma 5])**.** *A priority donor J_d can be pi-blocked during priority donation for at most the maximum duration of time that a job can be pi-blocked with an incomplete request for a replica of ℓ_a (refer to the timeline in Figure 4), plus the length of one critical section.*

▶ **Theorem 20.** *A job J_i incurs at most $(2\lceil m/k \rceil - 1) \cdot L^{max}$ s-oblivious pi-blocking while waiting to acquire a replica of a shared resource ℓ_a.*

Proof. By Lemma 19, a priority donor can be pi-blocked for only one critical section plus the maximum amount of time a job can be pi-blocked with an incomplete request, and by Lemma 18 a jobs incurs at most $(\lceil m/k \rceil - 1) \cdot L^{\max}$ s-oblivious pi-blocking while in KQ_a. Thus, a priority donor incurs at most $\lceil m/k \rceil \cdot L^{\max}$ s-oblivious pi-blocking while waiting to acquire ℓ_a. Furthermore, by Lemma 11 a job that incurs s-oblivious pi-blocking while waiting for a replica of a resource ℓ_a either has an incomplete request for ℓ_a, or it is a priority donor. Thus, while waiting to acquire ℓ_a, the total amount of pi-blocking a job J_i incurs is at most the sum of the pi-blocking it is subject to as a priority donor, and the pi-blocking it is subject to while traversing KQ_a, which is $\lceil m/k \rceil \cdot L^{\max} + (\lceil m/k \rceil - 1) \cdot L^{\max} = (2\lceil m/k \rceil - 1) \cdot L^{\max}$. ◀

Finally, we observe that the CKIP is indeed non-nested independence-preserving under clustered scheduling due to its reliance on AI as the underlying progress mechanism.

▶ **Theorem 21.** *The CKIP is non-nested independence-preserving under any JFLP scheduler.*

Proof. Under the CKIP, requests for replicas of shared resources are arbitrated under the rules of RRPD in each cluster. The rules of RRPD are such that jobs do *not* incur pi-blocking for resources they do not access [34]. Thus, any pi-blocking J_i incurs due to requests for a resource $\ell_a \notin \gamma_i$ would need to be the result of the use of AI as a cross-cluster progress mechanism. However, any job that benefits from AI only executes with the priority of another job currently waiting on a replica of the same resource, which precludes J_i from incurring pi-blocking due to jobs inheriting allocations [6]. Thus, if $N_{i,a} = 0$ then $b_{i,a} = 0$. ◀

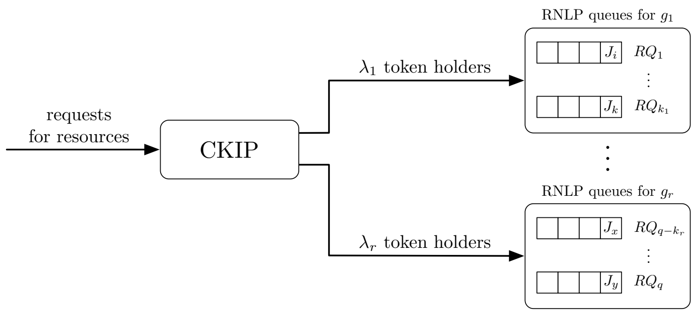


Figure 6 Queuing structure of the GIPP. A request for a token of a group is first arbitrated by the CKIP before the request is passed to the group's corresponding instance of the RNLP.

4.2 An Independence-Preserving RSM

The GIPP requires that its RSM lends itself to independence preservation, and no such suitable RSM for clustered scheduling has been proposed in prior work. Thus, we introduce the *Allocation Inheritance Resource Satisfaction Mechanism* (AI-RSM). The AI-RSM applies to clustered scheduling, and utilizes AI to ensure progress among jobs competing for shared resources. Let $ts(J_i)$ be the time that J_i acquired its token (and therefore entered the RSM), and let $sr(J_i, t)$ be the set of resources J_i holds at time t. Finally, we let $A_{i,a,t} = \{J_k \mid ts(J_k) < ts(J_i) \wedge (\ell_a \in sr(J_k, t) \vee \exists \ell_b \in sr(J_k, t) \text{ s.t. } \ell_b \succ \ell_a)\}$ denote the set of jobs that can prevent J_i from acquiring ℓ_a at time t, which follows from the rules of the RNLP.

A1 When the AI-RSM prevents J_i from acquiring a shared resource ℓ_a at time t, J_i donates its allocation to the job in $A_{i,a,t}$ with the earliest timestamp under the rules of AI.

▶ **Lemma 22.** *The AI-RSM ensures property **R1** under clustered scheduling when waiting is realized by suspending.*

Proof. Let J_i be a job that is pi-blocked by the RSM at time t while it waits to acquire a shared resource ℓ_a. Then, there must exist some job $J_k \in A_{i,a,t}$ that prevents J_i from acquiring ℓ_a by the rules of the RNLP. By Rule **A1**, the job $J_k \in A_{i,a,t}$ with the earliest timestamp is eligible to inherit J_i's priority in J_i's home cluster. Since J_i incurs s-oblivious pi-blocking, it has one of the c highest priorities in its cluster, and hence the inherited priority enables J_k to be scheduled in J_i's home cluster. Thus, at least one job preventing J_i from acquiring ℓ_a is scheduled and J_i therefore makes progress. ◀

We now have a group independence-preserving token lock, and an RSM with an independence-preserving progress mechanism. We use these components to realize the GIPP.

4.3 Structure and Analysis of The GIPP

We next define the structure of the GIPP and then establish its asymptotic optimality with respect to s-oblivious pi-blocking, and that it is group independence-preserving.

There are m tokens for each group $g_x \subseteq \Gamma$; a token for g_x is denoted with λ_x. A single instance of the CKIP arbitrates access to the set $\Lambda = \{\lambda_1, \ldots, \lambda_r\}$ of replicated tokens, and an instance of the RNLP with the AI-RSM is instantiated for each group. The CKIP instance serves as a common token lock among all the instances of the RNLP. To execute an outermost critical section under the GIPP for resources in g_x a job must **(i)** compete for and acquire a token λ_x under the CKIP, **(ii)** compete in g_x's instance of the AI-RSM under the rules of the RNLP [31, Section 3], and **(iii)** release λ_x upon completing its outermost critical section and exiting the AI-RSM. The queuing structure of the GIPP is depicted in Figure 6.

▶ **Theorem 23.** *The maximum amount of s-oblivious pi-blocking incurred per outermost request under the GIPP is* $(2m - 1) \cdot L^{max} = \mathcal{O}(m)$ *under any JLFP scheduler.*

Proof. The CKIP satisfies property **T1** by Lemma 15, and the AI-RSM satisfies property **R1** by Lemma 22. Therefore, the maximum amount of s-oblivious pi-blocking a job incurs while in the AI-RSM is $L^{\text{RSM}} = (m-1) \cdot L^{\max}$ by Theorem 9, as the corresponding RNLP proof generalizes to any protocol that satisfies these two properties.

Under the rules of the RNLP, a job holds a token for the entire duration it is in the RSM, and releases its token after completing its outermost critical section. The RNLP proof of Theorem 9 establishes that a job is pi-blocked for at most $m - 1$ outermost critical sections while in any RSM. Thus, after a job completes its outermost critical section, the maximum amount of time the job holds a token is $L^{\text{token}} = m \cdot L^{\max}$. By Theorem 20, a job waiting to acquire a token under the CKIP incurs at most $(2\lceil m/k \rceil - 1) \cdot L^{\text{token}}$ units of s-oblivious pi-blocking. Under the GIPP, there are m tokens for each group (*i.e.*, $k = m$), so the pi-blocking incurred while waiting for a token simplifies to L^{token} as $(2\lceil m/m \rceil - 1) = 1$. The total s-oblivious pi-blocking a job occurs per outermost request is then the sum of the pi-blocking incurred while waiting to acquire a token and while competing in the AI-RSM, which is $L^{\text{token}} + L^{\text{RSM}} = m \cdot L^{\max} + (m-1) \cdot L^{\max} = (2m-1) \cdot L^{\max}$. ◀

▶ **Theorem 24.** *The GIPP is group independence-preserving under any JLFP scheduler.*

Proof. By the structure of the GIPP, a job interacts with the CKIP for the entire duration it interacts with the GIPP. Under the CKIP, nested requests are not permitted, so each shared resource (*e.g.*, token type) forms its own group. When each group consists of a single resource, the definition of group independence preservation trivially reduces to non-nested independence preservation. Thus, it follows that the CKIP is group independence-preserving.

To prove the claim, it hence suffices to show that the GIPP remains group independence-preserving while token holders compete for shared resources in the AI-RSM. We prove this by contradiction: suppose a job J_i that does not request a token λ_x for a group g_x incurs pi-blocking due to a request for λ_x by a job J_k. Under the AI-RSM, J_k's effective priority is raised to that of another job competing for resources in g_x. Thus, if J_k's effective priority in $C(T_i)$ is greater than J_i's, there must be another job J_h in $C(T_i)$ that requires resources in g_x and has a higher base priority than J_i. As this argument applies to any such job J_k, and since AI establishes a one-to-one relationship among donors and recipients (*i.e.*, donor priorities are not "duplicated"), it follows there are at least c higher-base-priority jobs in $C(T_i)$, and hence J_i does not incur pi-blocking according to Definition 1. Contradiction. ◀

It is worth noting that, in special cases, the GIPP emulates the behavior of the RNLP and the OMIP, respectively. When there is just a single group (*i.e.*, $r = 1$), the GIPP effectively reduces to the RNLP in the sense that there is a single, global token lock. Conversely, when $r = |\Gamma|$, the GIPP behaves like the OMIP. These cases are examined further in Section 6.

5 Fine-Grained Pi-Blocking Analysis

We next introduce a fine-grained, non-asymptotic pi-blocking analysis for the GIPP, which we formulate as a *Linear Programming* (LP) problem as in prior work [7, 6, 35]. The asymptotic bound presented in Section 4.3 is coarse-grained as it does not reflect the exact resources each task requests, individual critical section lengths, nor the frequency of critical sections. The following analysis is fine-grained in the sense that it considers these workload-specific aspects to obtain a less pessimistic, but still safe upper-bound on s-oblivious pi-blocking.

In the following, we let T_i denote the task under analysis and let J_i denote an arbitrary job of T_i. For each other task T_x, we let θ_x^i denote a bound on the maximum number of jobs of T_x that overlap with J_i (i.e., that are pending while J_i is pending). Let r_i and r_x be the maximum response times of T_i and T_x, respectively. Then $\theta_x^i = \left\lceil \frac{r_i + r_x}{p_x} \right\rceil$ [5, 7].

We denote T_x's y^{th} outermost critical section as $O_{x,y}$, its length as $L_{x,y}^O$, the set of resources it accesses as $S_{x,y}$, and define $O_x(g) \triangleq \{O_{x,y} \mid S_{x,y} \subseteq g\}$. Note that the index y is used only for enumeration purposes and does *not* imply an order; each job of T_x may execute its outermost critical sections in any order. For each task $T_x \neq T_i$, each outermost request $O_{x,y}$, and $v \in \{1, \ldots, \theta_x^i\}$, we introduce two real-valued variables $X_{x,y,v}^T$ and $X_{x,y,v}^R$, each with domain $[0, 1]$. These variables are called *blocking fractions* [7] and serve to encode the portion of T_x's v^{th} overlapping instance of $O_{x,y}$ that contributes to the pi-blocking that J_i incurs. We use $X_{x,a,v}^T$ and $X_{x,a,v}^R$ to respectively encode the *token* and *RSM blocking* that J_i incurs, where token blocking refers to the time spent waiting to acquire a token, and RSM blocking refers to time spent waiting for a resource within the AI-RSM.

With these definitions in place, the pi-blocking incurred by J_i can be stated as

$$b_i = \sum_{T_x \neq T_i} \sum_{O_{x,y}} \sum_{v=1}^{\theta_x^i} (X_{x,y,v}^T + X_{x,y,v}^R) \cdot L_{x,y}^O. \tag{1}$$

By interpreting Equation (1) as the **objective function** of an LP maximization problem, we obtain an upper bound b_i on the maximum pi-blocking incurred by any J_i [7, 6, 35]. To avoid excessive pessimism, we introduce in the following LP constraints that reflect both the invariants of the GIPP and properties of the specific task set under analysis.

To start, we prevent any blocking critical section from being counted twice.

▶ **Constraint 25.** $\forall T_x \neq T_i : \forall O_{x,y} : \forall v : X_{x,y,v}^T + X_{x,y,v}^R \leq 1$

Proof. A single critical section of T_x cannot cause J_i to experience token blocking and RSM blocking *simultaneously*: to wait for a resource within the AI-RSM, J_i must already hold a token, but while J_i competes for a token it cannot yet interact with the RSM. Thus, the combined token and RSM blocking induced by one of T_x's critical sections cannot exceed the length of the critical section (i.e., the blocking fractions sum to at most one). ◀

Next, we bound the maximum amount of token blocking that J_i incurs. In preparation, let τ_k be the set of tasks assigned to cluster C_k, and $\tau_k' = \tau_k \setminus \{T_i\}$. Furthermore, $\phi_{i,g} \triangleq |\{O_{i,y} \mid S_{i,y} \cap g \neq \emptyset\}|$ denotes the number of times J_i issues an outermost request for a resource in g, and $\beta_{k,g} \triangleq \left| \left\{ T_x \;\middle|\; T_x \in \tau_k \wedge \bigcup_{O_{x,y}} S_{x,y} \cap g \neq \emptyset \right\} \right|$ is the number of tasks in C_k that request a resource in g. Based on these definitions, we state a bound on the number of times that J_i must wait for a token. In Equation (2), let k denote the index of $C(T_i)$.

$$W_{i,g} \triangleq \begin{cases} 0 & \beta_{k,g} \leq c \\ \min(\phi_{i,g}, \phi_{i,g}') & \text{otherwise} \end{cases} \quad \text{where } \phi_{i,g}' \triangleq \left(\sum_{T_x \in \tau_k'} (\phi_{x,g} \cdot \theta_x^i) \right) - c + 1 \tag{2}$$

▶ **Lemma 26.** *$W_{i,g}$ upper-bounds the number of times J_i must wait for a token of group g.*

Proof. By case analysis. Let k denote the index of $C(T_i)$. First, if $\beta_{k,g} \leq c$, then there are at most c tasks in $C(T_i)$ that ever require a token for group g (including T_i). There are never more than c token holders per cluster under the CKIP, which effectively reserves c tokens for each cluster. Thus, whenever J_i requires a token for group g, one is always available, and J_i never needs to wait for a token: $W_{i,g} = 0$ if $\beta_{k,g} \leq c$.

Otherwise, if $\beta_{k,g} > c$, then J_i requires a token no more than $\phi_{i,g}$ times, and hence clearly $W_{i,g} \leq \phi_{i,g}$. To obtain $W_{i,g} \leq \phi'_{i,g}$, consider the number of times that other tasks require a token while J_i is pending, which is bounded by $\sum_{T_x \in \tau'_k} (\phi_{x,g} \cdot \theta^i_x)$. Since J_i must wait for a token only if all c tokens are currently held by other tasks, the worst case occurs when $c-1$ tokens are held "indefinitely" (i.e., if they remain unavailable throughout the interval during which J_i is pending) and the remaining $\phi'_{i,g} = \left(\sum_{T_x \in \tau'_k} (\phi_{x,g} \cdot \theta^i_x)\right) - c + 1$ requests must all share a single token, and thus $W_{i,g} \leq \phi'_{i,g}$. ◀

We further restrict under which conditions J_i incurs token blocking at all.

▶ **Lemma 27.** *J_i incurs token blocking (i.e., it incurs pi-blocking while waiting to acquire a token for a group g) only if it is a priority donor under the rules of the RRPD.*

Proof. Recall that GIPP allocates group tokens using the CKIP, and that the CKIP employs RRPD. As there are $k = m$ tokens per group (i.e., replicas per token type), the CKIP's per-replica FIFO queues have length $\lceil \frac{m}{k} \rceil = 1$. Since by Rule K2 the head of each per-replica FIFO queue holds the replica (i.e., a token), and jobs enter a queue immediately when they issue a request (Rule K1), it follows that J_i can be waiting for a token only *before* it issues its request for a token, that is, while it serves as a priority donor under the rules of the RRPD in the time span between requiring a token and actually issuing a request (recall Figure 4). ◀

Lemmas 26 and 27 allow us to establish a constraint on token blocking due to each task.

▶ **Constraint 28.** $\forall g \in G : \forall T_x \neq T_i : \sum\limits_{O_{x,y} \in O_x(g)} \sum\limits_{v=1}^{\theta^i_x} X^T_{x,y,v} \leq W_{i,g}$

Proof. Suppose not. Then there exists a task T_x that token-blocks J_i with more than $W_{i,g}$ outermost critical sections (w.r.t. some group g). If $W_{i,g} = 0$, then by Lemma 26 J_i must never wait to acquire a token for group g, which immediately yields a contradiction. Hence assume $W_{i,g} > 0$. As J_i waits for a token for g at most $W_{i,g}$ times (Lemma 26), this implies that there exists an outermost critical section $O_{i,z}$ executed by J_i such that J_i is delayed, while waiting to acquire a token for g in preparation of $O_{i,z}$, by at least two outermost critical sections of T_x. By Lemma 27, J_i is a priority donor while it incurs token blocking. According to the rules of the RRPD (recall Section 2), J_i becomes a priority donor at most once per request, and only for a single other request: either immediately when J_i requires a token to commence $O_{i,z}$, or not at all. It follows that T_x must pi-block J_i with *two* distinct outermost critical sections while J_i continuously serves as the priority donor of some job J_l. Since under the rules of the RRPD J_i ceases to be a priority donor as soon as J_l finishes its outermost critical section (i.e., when J_l releases its token), J_l cannot be a job of T_x. Hence there remains only one other way for an outermost critical section of T_x to delay J_i, namely by delaying one or more requests of J_l within the RSM, which transitively causes J_i to incur pi-blocking. Consider the later of T_x's two outermost critical sections that cause J_i to incur pi-blocking while donating its priority to J_l. Since it is the second *outermost* critical section of T_x in this interval, T_x necessarily must have acquired a token for group g strictly after the beginning of the interval, when J_l was already holding its token. However, the RSM satisfies resource requests strictly in order of increasing token-acquisition timestamps, and thus T_x's second outermost critical section cannot delay J_l. Contradiction. ◀

We similarly bound the aggregate token blocking across all tasks in each cluster.

▶ **Constraint 29.** $\forall g \in G : \forall k \in \{1, \ldots, \frac{m}{c}\} : \sum\limits_{T_x \in \tau'_k} \sum\limits_{O_{x,y} \in O_x(g)} \sum\limits_{v=1}^{\theta^i_x} X^T_{x,y,v} \leq W_{i,g} \cdot \min(c, \beta_{k,g})$

Proof. Again the case of $W_{i,g} = 0$ is trivial; hence assume $W_{i,g} > 0$ and suppose the invariant does not hold. Then analogously to the proof of Constraint 28, there exists a contiguous interval $[t_1, t_2)$ and a cluster C_k such that both **(i)** J_i serves as the priority donor of some job J_l throughout $[t_1, t_2)$, and **(ii)** J_i incurs pi-blocking during $[t_1, t_2)$ due to at least $\min(c, \beta_{k,g}) + 1$ outermost critical sections executed by tasks in τ_k. Also analogously to the proof of Constraint 28, no task delays J_i with more than one outermost critical section during $[t_1, t_2)$. Because the RSM satisfies requests strictly in order of increasing token-acquisition timestamps, any job that delays J_l within the RSM (and hence transitively causes J_i to incur token blocking) must have acquired its token for group g before J_l did so, and hence no later than at time t_1. Furthermore, any such job necessarily releases its token only some time after t_1. At time t_1 there hence exist at least $\min(c, \beta_{k,g}) + 1$ token-holding jobs in cluster C_k. However, the CKIP ensures that no more than c jobs in C_k hold a token at any time, and by definition at most $\beta_{k,g}$ tasks in τ_k require a token for group g. Contradiction. ◀

This concludes the constraints on token blocking. We next constrain RSM blocking, that is, the pi-blocking incurred by J_i while it holds a token and waits for individual resources. First, we observe on a per-cluster basis that RSM blocking across all tasks is limited by the number of resource requests that J_i issues because the RSM serves jobs in timestamp order.

▶ **Constraint 30.** $\forall g \in G : \forall k \in \{1, \ldots, \frac{m}{c}\}$:

$$\sum_{T_x \in \tau'_k} \sum_{O_{x,y} \in O_x(g)} \sum_{v=1}^{\theta^i_x} X^R_{x,y,v} \leq \begin{cases} \phi_{i,g} \cdot \min(c, \beta_{k,g}) & T_i \notin \tau_k \\ \phi_{i,g} \cdot \min(c-1, \beta_{k,g} - 1) & \text{otherwise} \end{cases}$$

Proof. If $\phi_{i,g} = 0$, then J_i does not access resources in group g and the invariant is trivial. Hence, suppose the invariant does not hold. First consider the case $T_i \notin \tau_k$: then there exists an interval $[t_1, t_2)$ during which J_i holds a token for group g such that J_i incurs RSM blocking due to more than $\min(c, \beta_{k,g})$ outermost critical sections executed by jobs in C_k. Analogously to the proof of Constraint 29, it follows that more than $\min(c, \beta_{k,g})$ jobs must hold a token for group g at time t_1, which is impossible. In the second case, if $T_i \in \tau_k$, then J_i necessarily holds one of the c available tokens (otherwise it could not interact with the RSM), so that there are only $c - 1$ tokens available to other tasks, and only $\beta_{k,g} - 1$ other tasks in τ_k that are also accessing resources in group g. ◀

Next we constrain RSM blocking in a more detailed fashion by considering which critical sections actually conflict within the RSM. The resulting constraint is essential to realizing the benefits of the increased parallelism in nested locking protocols (relative to coarse-grained group-locking) at analysis time, and not just at runtime. To this end, we require some further terminology. First, we say that a set of resources s is *possibly conflicting* with another set of resources s' if either **(i)** $s \cap s' \neq \emptyset$ or **(ii)** $\exists \ell_b \in s, \ell_a \in s'$ such that $\ell_a \succ \ell_b$. Second, we define $F_i(s) \triangleq |\{O_{i,y} \mid S_{i,y} \textit{ possibly conflicts with } s\}|$ to count the number of outermost critical sections of T_i in which it needs resources that the RSM may have to withhold due to other jobs holding resources in s. Finally, we let $S^i(g) \triangleq \{S_{x,y} \mid T_x \neq T_i \land S_{x,y} \cap g \neq \emptyset\}$ denote the set of all combinations of resources in group g acquired by other tasks. Based on these definitions, we constrain RSM blocking as follows.

▶ **Constraint 31.** $\forall g \in G : \forall s \in S^i(g) : \forall k \in \{1, \ldots, \frac{m}{c}\}$:

$$\sum_{T_x \in \tau'_k} \sum_{\substack{O_{x,y} \text{ s.t.} \\ S_{x,y} \subseteq s}} \sum_{v=1}^{\theta^i_x} X^R_{x,y,v} \leq \begin{cases} F_i(s) \cdot \min(c, \beta_{k,g}) & T_i \notin \tau_k \\ F_i(s) \cdot \min(c-1, \beta_{k,g} - 1) & \text{otherwise} \end{cases}$$

Proof. Consider any group g, set of resources $s \in S^i(g)$, and cluster C_k. For J_i to incur RSM blocking when issuing a request for some resource $\ell_b \in g$, there must exist a job J_x with an earlier token-acquisition time that either already holds ℓ_b, or that holds a resource $\ell_a \in g$ such that $\ell_a \succ \ell_b$. In other words, J_i incurs RSM blocking only if $\{\ell_b\}$ is possibly conflicting with the set of resources already held by jobs with earlier timestamps. Recall that $F_i(s)$ counts the number of outermost critical sections of T_i accessing resources that are possibly conflicting with s. It follows that J_i executes at most $F_i(s)$ outermost critical sections that may encounter RSM blocking due to outermost critical sections of tasks in τ'_k that access s or a subset of s (i.e., the requests represented on the left-hand side of the constraint). As in the proof of Constraint 30, it is easy to show that no more than $\min(c, \beta_{k,g})$ (respectively, $\min(c-1, \beta_{k,g}-1)$) outermost critical sections can cause RSM blocking per each outermost critical section of J_i if $T_i \notin \tau_k$ (respectively, $T_i \in \tau_k$). The bound follows. ◀

6 Schedulability Experiments

We compared the GIPP against the OMIP and the RNLP as **(i)** they are both asymptotically optimal with respect to s-oblivious pi-blocking, **(ii)** the OMIP [6] is the only prior independence-preserving locking protocol for clustered scheduling, and **(iii)** the RNLP [31] is the only prior fine-grained nested locking protocol that ensures asymptotically optimal pi-blocking bounds under clustered scheduling.

To conduct meaningful experiments, one requires a fine-grained (i.e., non-asymptotic) pi-blocking analysis. The OMIP has such analysis, which is also formulated as an LP [6], which we use in these experiments. However, for the RNLP, there are surprisingly no fine-grained bounds available in prior work. We therefore had to adapt our analysis of the GIPP (Section 5) to the RNLP. To this end, we created an instantiation of the RNLP called the CA-RNLP that uses the the CKIP as its token lock and AI-RSM as its RSM. As the RNLP and hence the CA-RNLP uses only a single, global token lock, our analysis in Section 5 is applicable to the CA-RNLP if one presumes that *all* resources are part of a *single* group.

We generated task sets with Emberson et al.'s method [17] via the SchedCAT [9] library and considered all combinations of the following parameter choices. Each task set consisted of $n \in \{2.0m, 3.0m\}$ tasks with total utilization $U \in \{0.4m, 0.6m\}$ to be scheduled on $m \in \{4, 8, 16\}$ processors under partitioned EDF scheduling ($c = 1$). There were $n^{\text{nl}} \in \{0.0m, 0.5m, 1.0m\}$ latency-sensitive tasks in the task set, and $n - n^{\text{nl}}$ non-latency-sensitive (*i.e.*, regular) tasks. Periods were drawn uniformly at random from the set $\{1ms, 2ms, 4ms, 5ms, 8ms\}$ for latency-sensitive tasks, and from the set $\{10ms, 20ms, 25ms, 40ms, 50ms, 100ms, 125ms, 200ms, 250ms, 500ms, 1000ms\}$ for regular tasks; the specific period values were inspired by Kramer et al.'s work on producing real-world automotive benchmarks [24]. Latency-sensitive tasks shared three resources in a single group, and each issues one or two outermost requests for resources in their group at random. The regular tasks shared twelve resources split into equally sized groups of $g^{\text{size}} = \{1, 2, 3, 4\}$ resources. Each regular task accessed just one group, and issued from $\{1, \ldots, N^{\max}\}$ outermost requests for resources in that group at random, where $N^{\max} \in \{1, 2, 3\}$. The outermost critical section lengths of latency-sensitive tasks were drawn uniformly at random from $[1\mu s, 15\mu s]$, and from $[1\mu s, mcsl]$ for regular tasks, where $mcsl$ was varied across $[5\mu s, 1000\mu s]$ in increments of $5\mu s$.

Each resource group was of one of two types $g^{\text{type}} \in \{\textit{wide}, \textit{deep}\}$. More precisely, we say a shared resource $\ell_b \in \Gamma$ is a *top-level resource* if $\nexists \ell_a \in \Gamma$ s.t. $\ell_a \succ \ell_b$, and consider a group to be *wide* if at least half of its resources are top-level, and *deep* otherwise. Tasks were assigned a minimal set of requests to form the desired groups; afterwards tasks were assigned outermost requests for resources in their corresponding group at random until each made $N^{\max}$ requests (per job); each outermost request contained a nested request with probability 0.5.

There are 864 combinations of the varied parameters $(n, n^{\mathrm{nl}}, U, m, N^{\max}, g^{\mathrm{size}}, g^{\mathrm{type}})$. Some combinations are duplicate in effect (*i.e.*, wide and deep groups of size one are the same), or not possible to generate. After removing such combinations from consideration, there remain 522 combinations. For each combination, we generated 1000 task sets per *mcsl* value (*i.e.*, per point on the x-axis), and then tested each task set for schedulability under the GIPP, OMIP, and the CA-RNLP. Figures 7 to 9 show three select scenarios out of the 522 combinations; all graphs are shown in Appendix A. (Figure 10 further shows a hand-crafted example to highlight a specific behavior as discussed below.)

In our large-scale experiments, both the GIPP and the OMIP retained a high level of schedulability for most parameter configurations. In most cases, the CA-RNLP provided a substantially lower level of schedulability than the GIPP or the OMIP. We now outline some key observations drawn from the large-scale schedulability experiments. As our **first observation**, we notice that the GIPP performs noticeably better than the OMIP and the CA-RNLP in most of our experiments, and never worse. In corner cases, the performance of the GIPP approaches that of the OMIP when $g^{\mathrm{size}} = 1$, and the CA-RNLP when $g^{\mathrm{size}} = |\Gamma|$ (*i.e.*, the total number of resources). As a result, the GIPP never performs worse than the better-performing of the two baselines. This is apparent in Figure 7 and Figure 8. Our **second observation** is that the isolation of latency-sensitive tasks greatly impacts schedulability. When latency-sensitive tasks must compete with regular tasks for the same set of tokens, schedulability quickly drops under the CA-RNLP as *mcsl* increases (see Figure 7). As our **third observation**, we note that even in the absence of latency-sensitive tasks, schedulability is greatly affected by the use of a single, global token lock (*i.e.*, if tokens are not group-specific). As *mcsl* increases, schedulability under the CA-RNLP drops at a roughly linear rate in Figure 8, whereas task sets remain schedulable for the entire range of *mcsl* under the GIPP and the OMIP. This demonstrates that a single token lock ultimately becomes a bottleneck for otherwise schedulable task sets. As a **fourth observation**, the benefits of (group) independence preservation diminish under high contention for all resources. This is shown in Figure 9, where roughly the same pattern of schedulability is seen under all three protocols. In contrast, the benefits of independence preservation are more clearly seen when there is a greater degree of isolation as in Figure 8.

This concludes our discussion of the large-scale schedulability study. As the workload generator with the chosen set of parameters did not generate a sufficient number of task sets that hit weak points of the OMIP, we further set up an experiment with a hand-crafted task set; the result is shown in Figure 10. The four tasks of the task set share a single wide group of size four. Three of the tasks access only top-level resources, while the fourth task makes the necessary requests to form the group. The rules and structure of the GIPP and the CA-RNLP allow for top-level resources to be acquired independently, which is not possible with the OMIP's group locks. Thus, as a final and **fifth observation**, we note that fine-grained nested locking offers a noticeable increase in schedulability when compared to group locks for heavily asymmetric resource access patterns.

7 Conclusion

We have examined the concept of independence preservation in the context of fine-grained nested locking. On the one hand, *outer-lock independence preservation* yields non-optimal bounds on s-oblivious pi-blocking. On the other hand, *group independence preservation* can

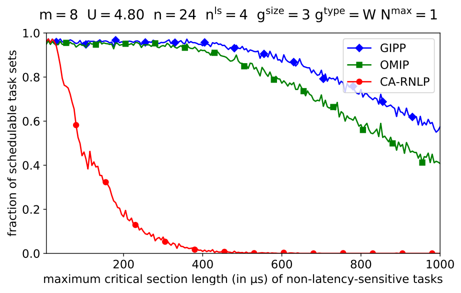


Figure 7 The presence of latency-sensitive tasks dominates schedulability.

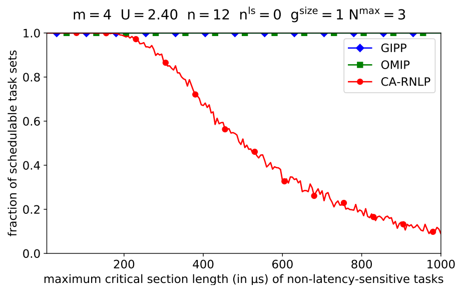


Figure 8 Token contention dominates schedulability.

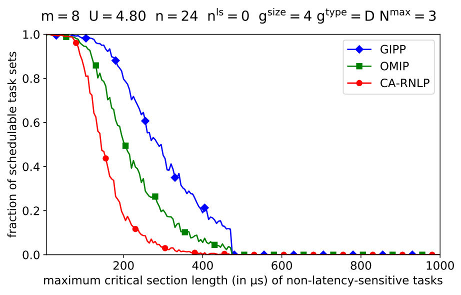


Figure 9 High contention among few groups dominates schedulability.

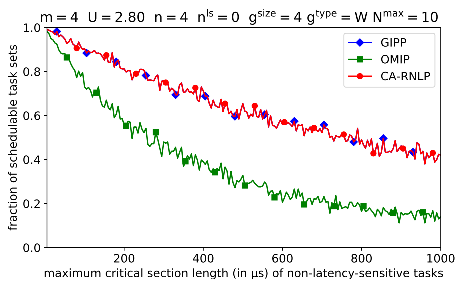


Figure 10 A single group lock limits schedulability for certain asymmetric access patterns.

be realized with asymptotically optimal pi-blocking bounds (under s-oblivious analysis), as demonstrated with the GIPP. To obtain the GIPP, we constructed the CKIP as a building block, which is noteworthy in itself as it is the first asymptotically optimal, non-nested independence-preserving, k-exclusion lock for clustered scheduling. Finally, we demonstrated with empirical experiments using a fined-grained pi-blocking analysis of the GIPP that it avoids the bottleneck imposed by the RNLP's single token lock (or group locks under the OMIP), thereby allowing latency-sensitive tasks to be accommodated.

In future work, it would be interesting to extend the GIPP to semi-partitioned scheduling [1, 2, 13]. It will also be necessary to study the real-world overheads (*e.g.*, cache misses, TLB flushes, inter-processor interrupts, etc.), which the GIPP is particularly exposed to due to its use of allocation inheritance, in a practical system such as LITMUS^{RT} [5, 15].

References

1 James H. Anderson, Vasile Bud, and UmaMaheswari C. Devi. An EDF-based scheduling algorithm for multiprocessor soft real-time systems. In *17th Euromicro Conference on Real-Time Systems (ECRTS'05)*, 2005. doi:10.1109/ECRTS.2005.6.

2 Andrea Bastoni, Björn B. Brandenburg, and James H. Anderson. Is Semi-Partitioned Scheduling Practical? In *23rd Euromicro Conference on Real-Time Systems (ECRTS'11)*, 2011. doi:10.1109/ECRTS.2011.20.

3 Alessandro Biondi, Björn B. Brandenburg, and Alexander Wieder. A Blocking Bound for Nested FIFO Spin Locks. In *38th Real-Time Systems Symposium (RTSS'17)*, 2017. doi:10.1109/RTSS.2016.036.

4 Aaron Block, Hennadiy Leontyev, Björn B. Brandenburg, and James H Anderson. A Flexible Real-Time Locking Protocol for Multiprocessors. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07)*. IEEE, 2007. doi:10.1109/RTCSA.2007.8.

5 Björn B. Brandenburg. *Scheduling and Locking in Multiprocessor Real-Time Operating Systems*. PhD thesis, University of North Carolina at Chapel Hill, 2011.

6 Björn B. Brandenburg. A Fully Preemptive Multiprocessor Semaphore Protocol for Latency-Sensitive Real-Time Applications. In *25th Euromicro Conference on Real-Time Systems (ECRTS'13)*, 2013. doi:10.1109/ECRTS.2013.38.

7 Björn B. Brandenburg. Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling. In *19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'13)*, 2013. doi:10.1109/RTAS.2013.6531087.

8 Björn B. Brandenburg. Multiprocessor Real-Time Locking Protocols: A Systematic Review. *arXiv:1909.09600 [cs]*, 2019. URL: http://arxiv.org/abs/1909.09600, arXiv:1909.09600.

9 Björn B. Brandenburg. SchedCAT: The schedulability test collection and toolkit., January 2020. URL: https://github.com/brandenburg/schedcat.

10 Björn B. Brandenburg and James H. Anderson. Optimality Results for Multiprocessor Real-Time Locking. In *31st IEEE Real-Time Systems Symposium (RTSS'10)*. IEEE, 2010. doi:10.1109/RTSS.2010.17.

11 Björn B. Brandenburg and James H. Anderson. Real-Time Resource-Sharing under Clustered Scheduling: Mutex, Reader-Writer, and k-Exclusion Locks. In *9th ACM International Conference on Embedded Software (EMSOFT'11)*, 2011. doi:10.1145/2038642.2038655.

12 Björn B. Brandenburg and Andrea Bastoni. The Case for Migratory Priority Inheritance in Linux: Bounded Priority Inversions on Multiprocessors. In *14th Real Time Linux Workshop (RTLWS'12)*, 2012.

13 Björn B. Brandenburg and Mahircan Gul. Global Scheduling Not Required: Simple, Near-Optimal Multiprocessor Real-Time Scheduling with Semi-Partitioned Reservations. In *37th IEEE Real-Time Systems Symposium (RTSS'16)*. IEEE, 2016. doi:10.1109/RTSS.2016.019.

14 Alan Burns and Andy J. Wellings. A Schedulability Compatible Multiprocessor Resource Sharing Protocol - MrsP. In *25th Euromicro Conference on Real-Time Systems (ECRTS'13)*, 2013. doi:10.1109/ECRTS.2013.37.

15 John M. Calandrino, Hennadiy Leontyev, Aaron Block, UmaMaheswari C. Devi, and James H. Anderson. LITMUS^{RT} : A Testbed for Empirically Comparing Real-Time Multiprocessor Schedulers. In *27th IEEE Real-Time Systems Symposium (RTSS'06)*, 2006. doi:10.1109/RTSS.2006.27.

16 Arvind Easwaran and Björn Andersson. Resource Sharing in Global Fixed-Priority Preemptive Multiprocessor Scheduling. In *30th IEEE Real-Time Systems Symposium (RTSS'09)*, 2009. doi:10.1109/RTSS.2009.37.

17 Paul Emberson, Roger Stafford, and Robert I. Davis. Techniques for the Synthesis of Multiprocessor Tasksets. In *Proceedings of the 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS'10)*, 2010.

18 Dario Faggioli, Giuseppe Lipari, and Tommaso Cucinotta. The Multiprocessor Bandwidth Inheritance Protocol. In *22nd Euromicro Conference on Real-Time Systems (ECRTS'10)*, 2010. `doi:10.1109/ECRTS.2010.19`.

19 Dario Faggioli, Giuseppe Lipari, and Tommaso Cucinotta. Analysis and Implementation of the Multiprocessor Bandwidth Inheritance Protocol. *Real-Time Systems*, 48(6):789–825, 2012. `doi:10.1007/s11241-012-9162-0`.

20 Philip Holman. *On the Implementation of Pfair-Scheduled Multiprocessor Systems.* PhD thesis, University of North Carolina at Chapel Hill, 2004.

21 Philip Holman and James H. Anderson. Object sharing in Pfair-Scheduled Multiprocessor Systems. In *14th Euromicro Conference on Real-Time Systems (ECRTS'02)*, 2002. `doi:10.1109/EMRTS.2002.1019191`.

22 Philip Holman and James H. Anderson. Locking Under Pfair Scheduling. *ACM Transactions on Computer Systems*, 24(2):140–174, 2006. `doi:10.1145/1132026.1132028`.

23 Catherine E. Jarrett, Bryan C. Ward, and James H. Anderson. A Contention-Sensitive Fine-Grained Locking Protocol for Multiprocessor Real-Time Systems. In *23rd International Conference on Real Time and Networks Systems (RTNS'15)*, 2015. `doi:10.1145/2834848.2834874`.

24 Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real World Automotive Benchmarks for Free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS'15)*, 2015.

25 Catherine E. Nemitz, Tanya Amert, and James H. Anderson. Using Lock Servers to Scale Real-Time Locking Protocols: Chasing Ever-Increasing Core Counts. In Sebastian Altmeyer, editor, *30th Euromicro Conference on Real-Time Systems (ECRTS'18)*, 2018. `doi:10.4230/LIPIcs.ECRTS.2018.25`.

26 Catherine E. Nemitz, Tanya Amert, and James H. Anderson. Real-time Multiprocessor Locks with Nesting: Optimizing the Common Case. *Real-Time Systems*, 55(2):296–348, 2019. `doi:10.1007/s11241-019-09328-w`.

27 R. Rajkumar. Real-Time Synchronization Protocols for Shared Memory Multiprocessors. In *10th International Conference on Distributed Computing Systems (ICDCS'90)*, 1990. `doi:10.1109/ICDCS.1990.89257`.

28 Ragunathan Rajkumar. *Synchronization in Real-Time Systems: A Priority Inheritance Approach.* Kluwer Academic Publishers, USA, 1991.

29 Liu Sha, Ragunathan Rajkumar, and John P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990. `doi:10.1109/12.57058`.

30 Hiroaki Takada and Ken Sakamura. Real-time Scalability of Nested Spin Locks. In *2nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'95)*, 1995. `doi:10.1109/rtcsa.1995.528766`.

31 Bryan C. Ward and James H. Anderson. Supporting Nested Locking in Multiprocessor Real-Time Systems. In *24th Euromicro Conference on Real-Time Systems (ECRTS'12)*, 2012. `doi:10.1109/ECRTS.2012.17`.

32 Bryan C. Ward and James H. Anderson. Fine-Grained Multiprocessor Real-Time Locking with Improved Blocking. In *21st International Conference on Real Time and Networks Systems (RTNS'13)*, 2013. `doi:10.1145/2516821.2516843`.

33 Bryan C. Ward and James H. Anderson. Multi-Resource Real-Time Reader/Writer Locks for Multiprocessors. In *28th IEEE International Parallel and Distributed Processing Symposium (IPDPS'14)*, 2014. `doi:10.1109/IPDPS.2014.29`.

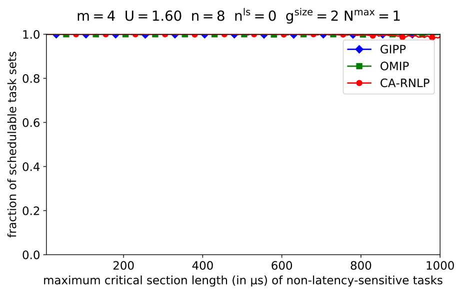
34 Bryan C. Ward, Glenn A. Elliott, and James H. Anderson. Replica-Request Priority Donation: A Real-Time Progress Mechanism for Global Locking Protocols. In *18th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2012. `doi:10.1109/RTCSA.2012.26`.

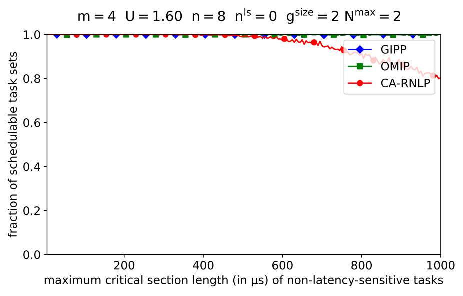
35 Alexander Wieder and Björn B. Brandenburg. On Spin Locks in AUTOSAR: Blocking Analysis of FIFO, Unordered, and Priority-Ordered Spin Locks. In *34th IEEE Real-Time Systems Symposium (RTSS'13)*, 2013. `doi:10.1109/RTSS.2013.13`.

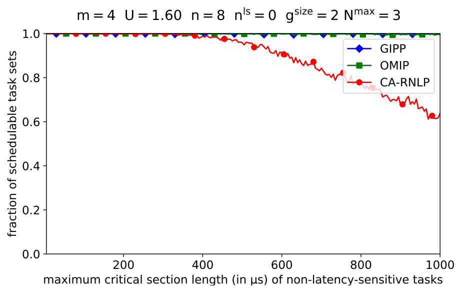
36 Maolin Yang, Alexander Wieder, and Björn B. Brandenburg. Global Real-Time Semaphore Protocols: A Survey, Unified Analysis, and Comparison. In *36th IEEE Real-Time Systems Symposium (RTSS'15)*, 2015. `doi:10.1109/RTSS.2015.8`.

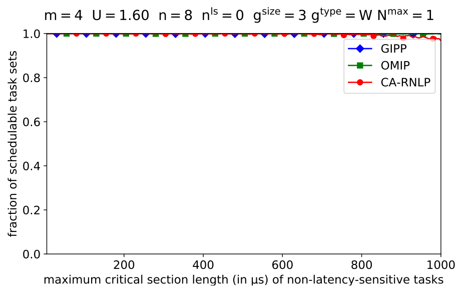
A Full Results of Large-Scale Schedulability Experiments

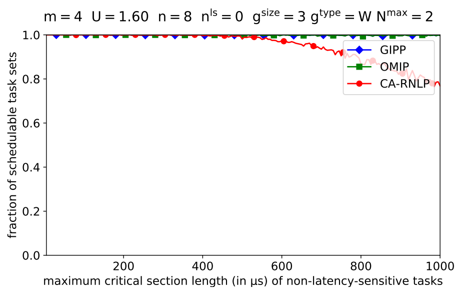
For the sake of completeness and transparency, in the following we provide the complete set of schedulability plots that resulted from the experiments described in Section 6.

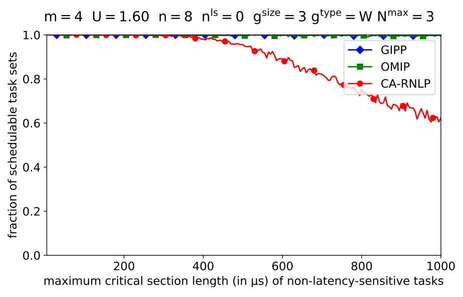


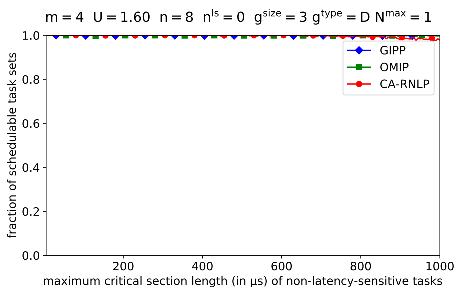


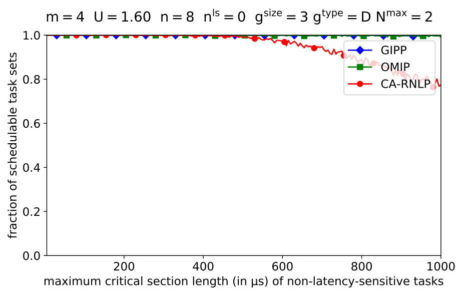


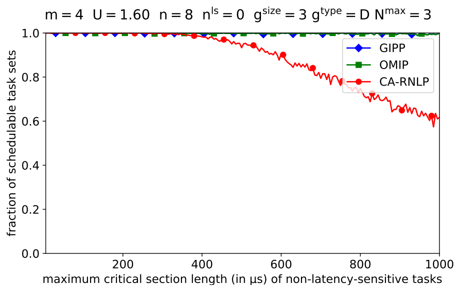


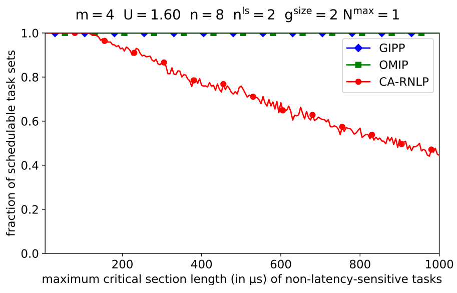


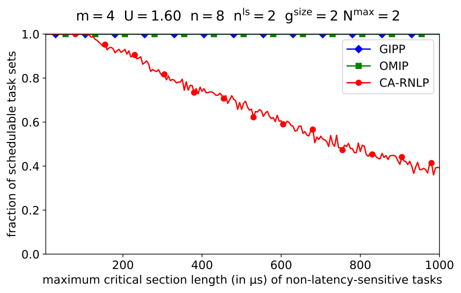


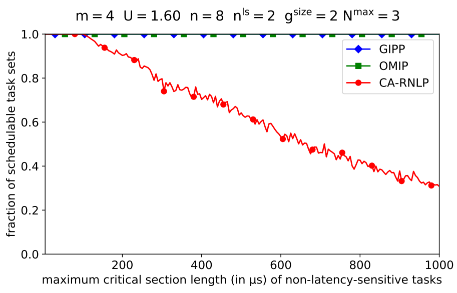


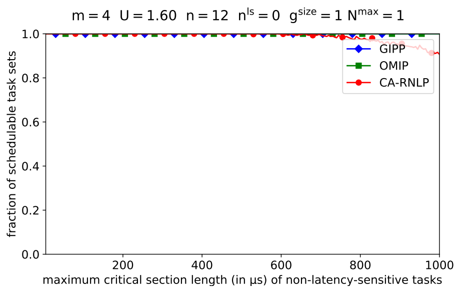


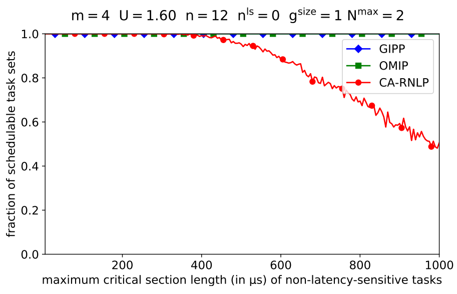

m = 4 U = 1.60 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

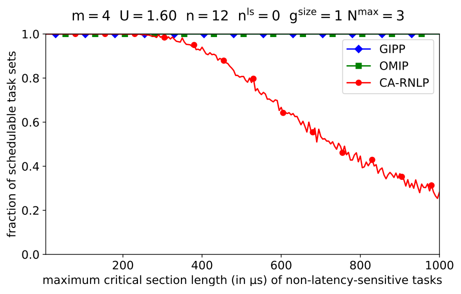

m = 4 U = 1.60 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

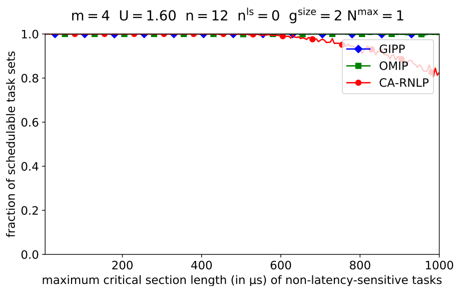

m = 4 U = 1.60 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

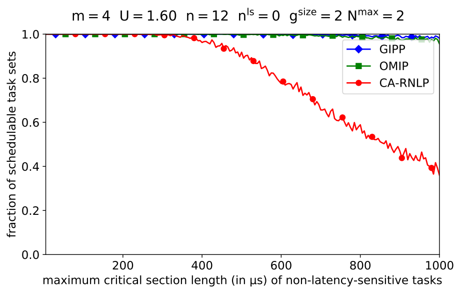

m = 4 U = 1.60 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

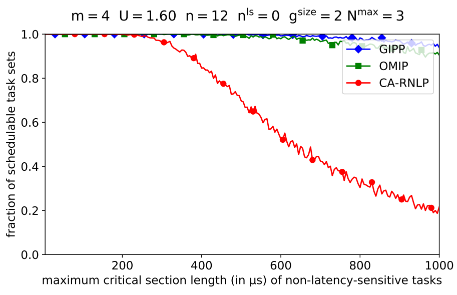

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

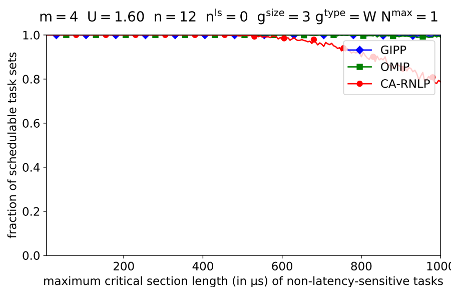

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

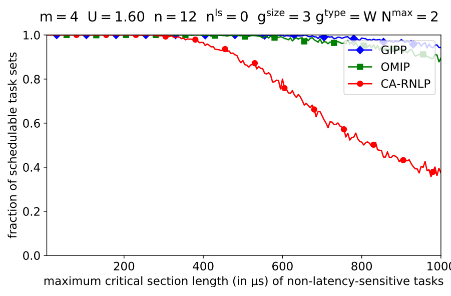

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

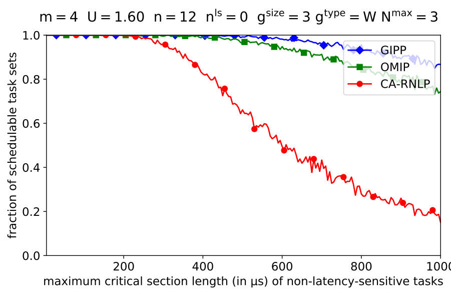

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

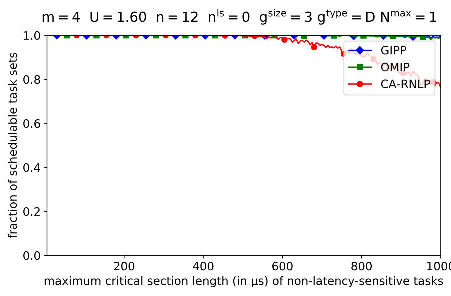

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

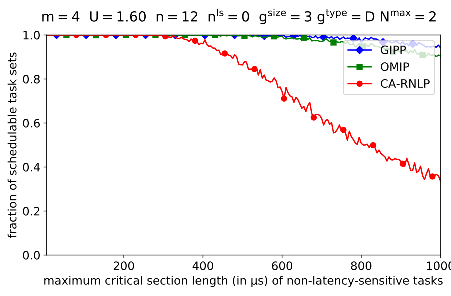

m = 4 U = 1.60 n = 12 $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

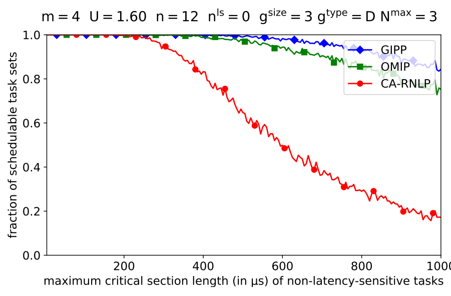

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

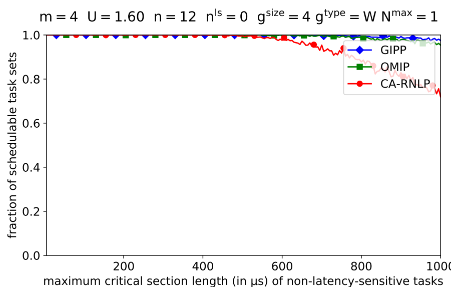

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

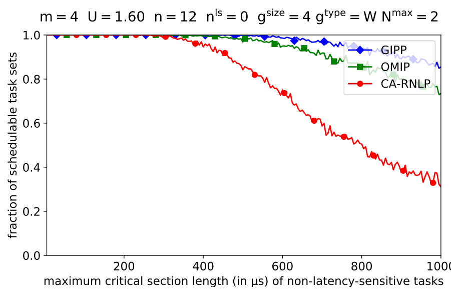

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

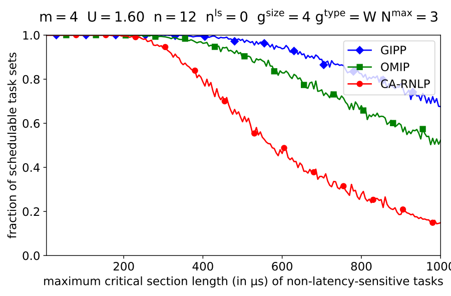

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

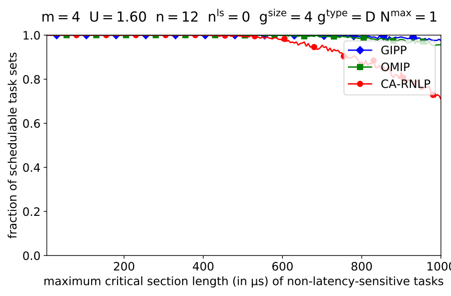

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

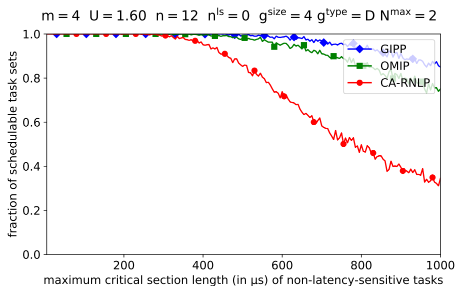

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

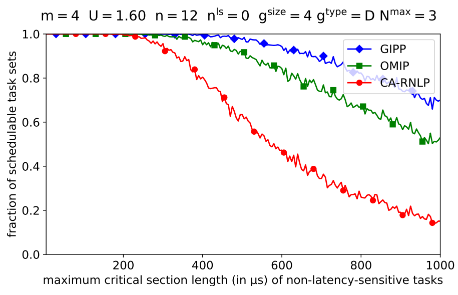

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

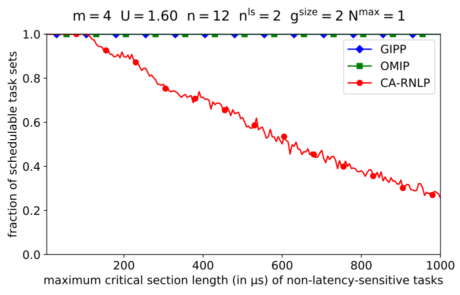

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

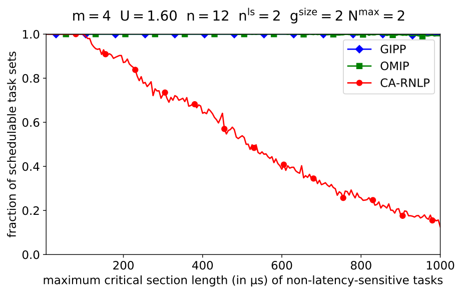

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

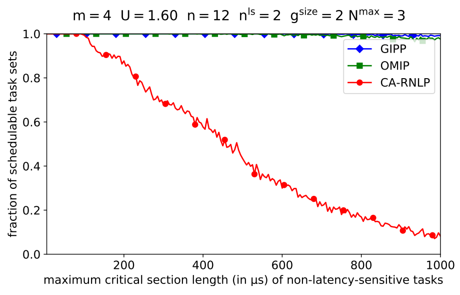

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

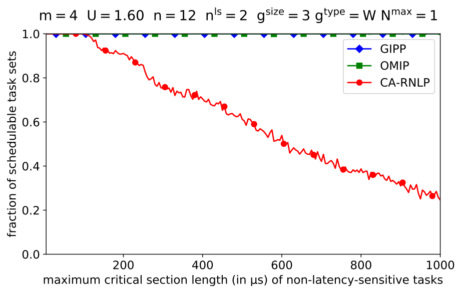

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

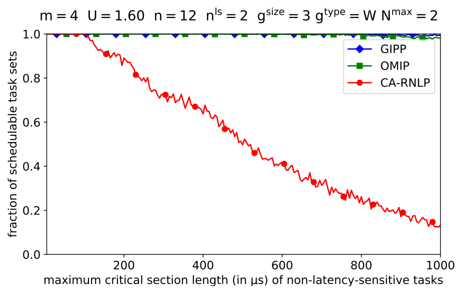

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

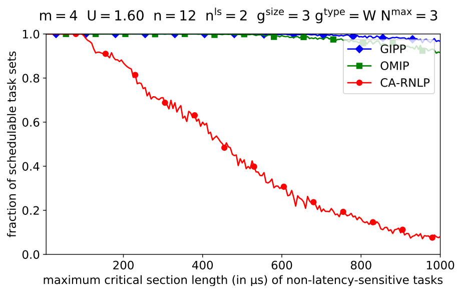

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

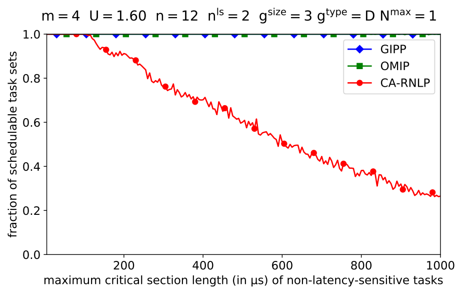

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

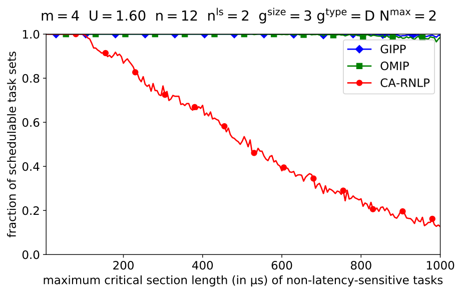

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

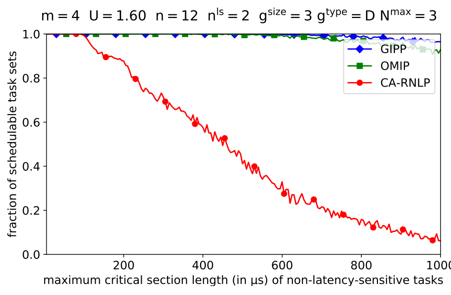

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

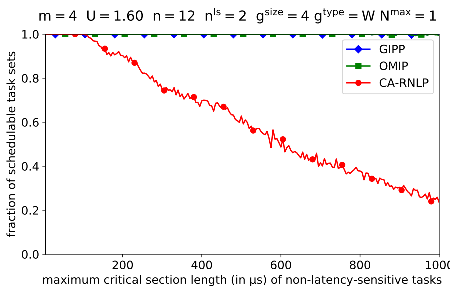

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

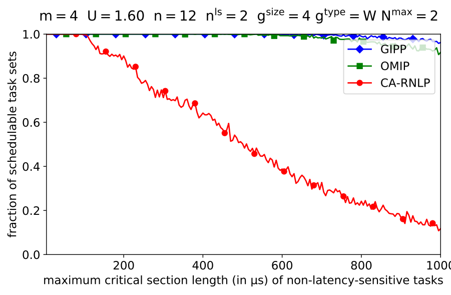

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

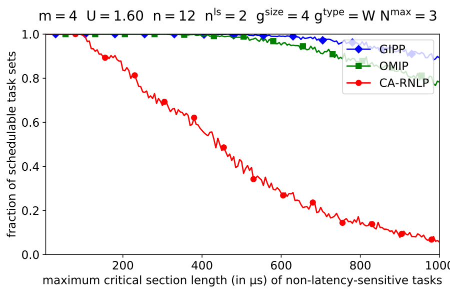

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

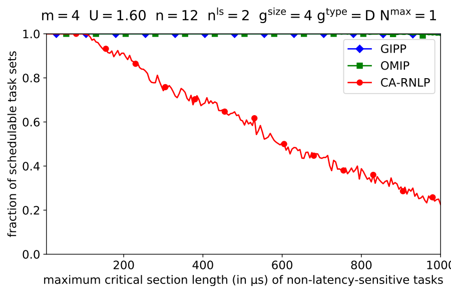

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

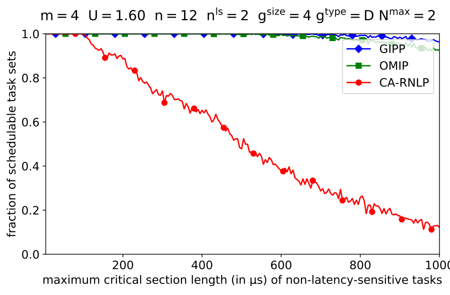

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

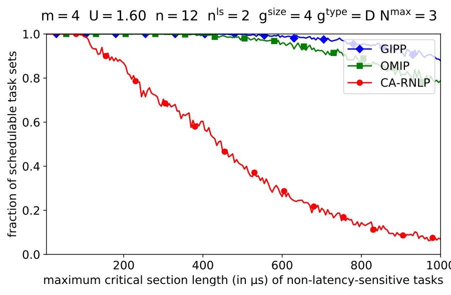

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

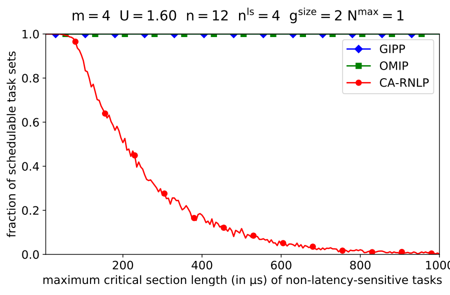

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

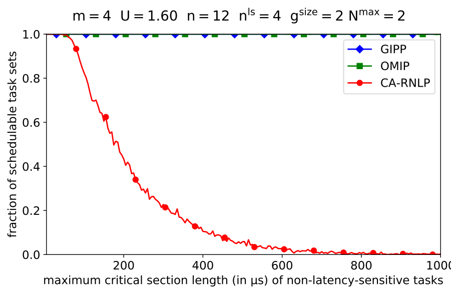

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

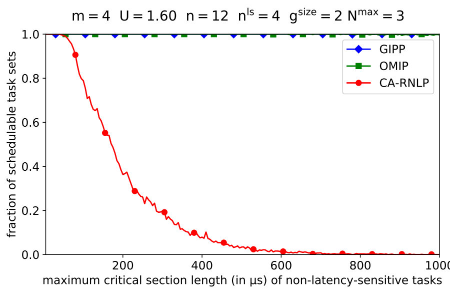

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

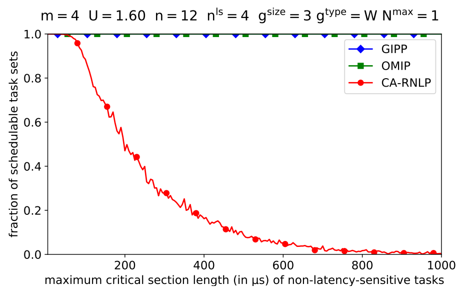

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

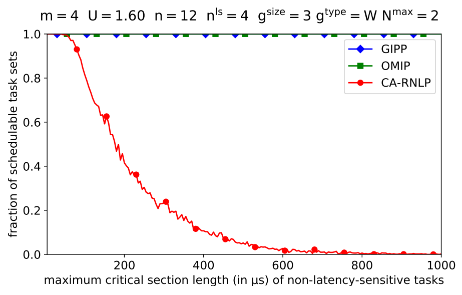

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 2$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

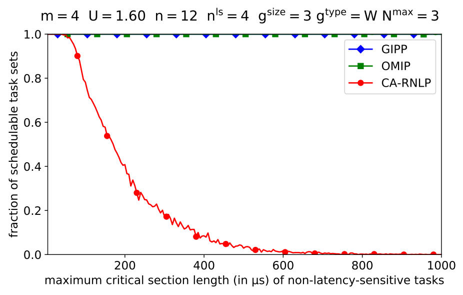

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

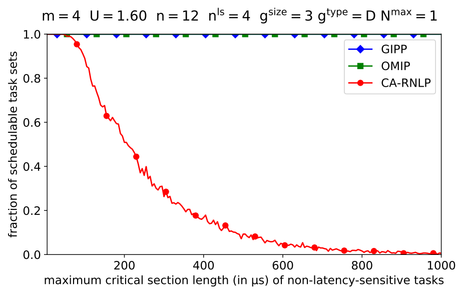

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

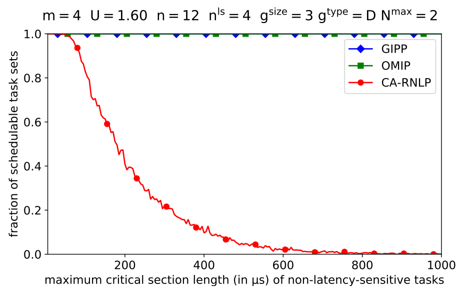

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

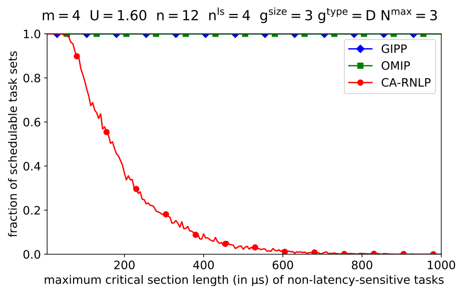

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

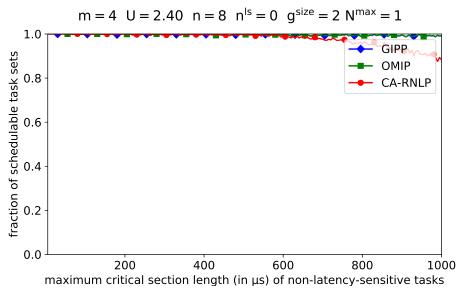

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

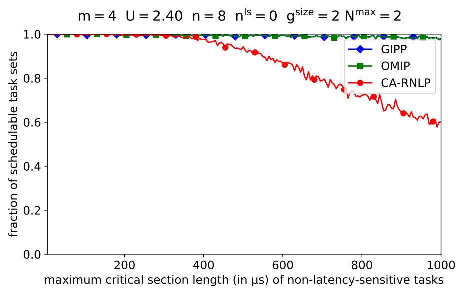

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

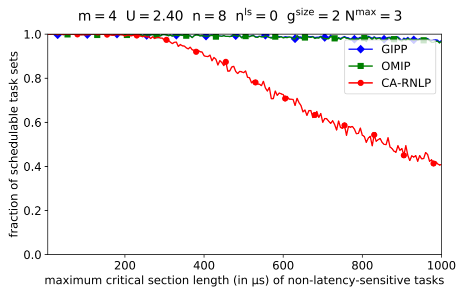

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

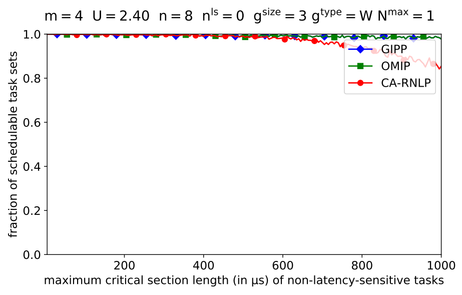

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

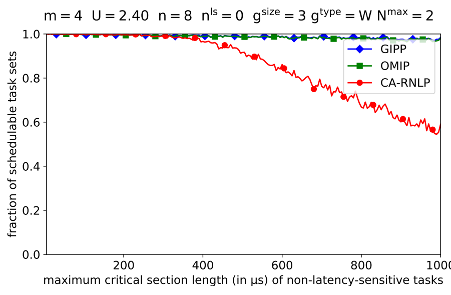

$m = 4$ $U = 1.60$ $n = 12$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

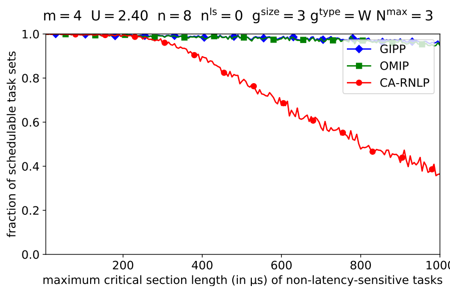

$m = 4$ $U = 2.40$ $n = 8$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

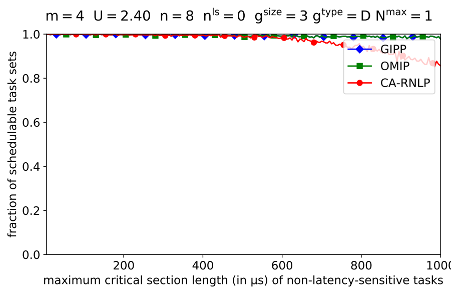

$m = 4$ $U = 2.40$ $n = 8$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

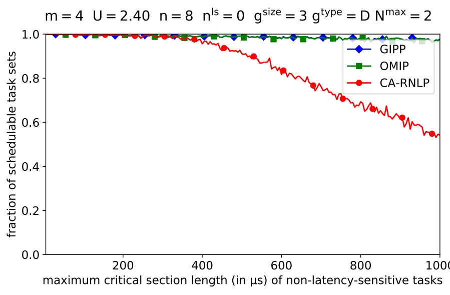

$m = 4$ $U = 2.40$ $n = 8$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

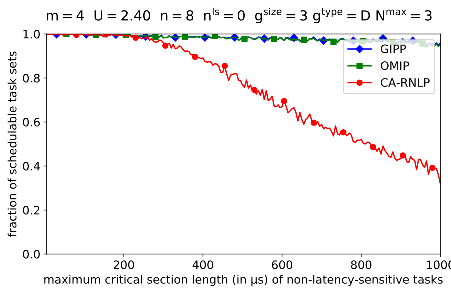

$m = 4$ $U = 2.40$ $n = 8$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

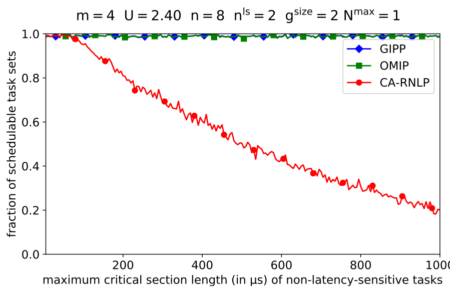

m = 4 U = 2.40 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

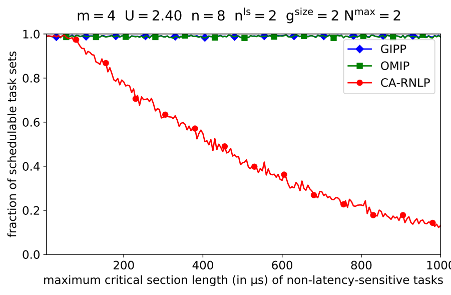

m = 4 U = 2.40 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

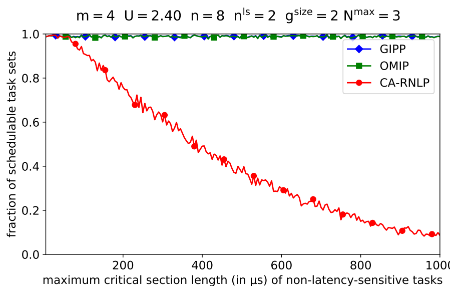

m = 4 U = 2.40 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

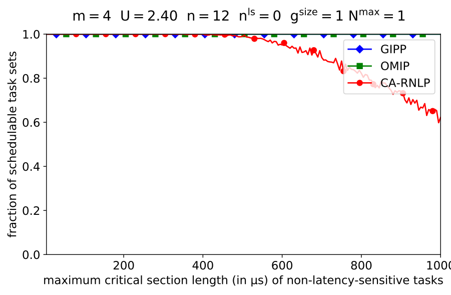

m = 4 U = 2.40 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

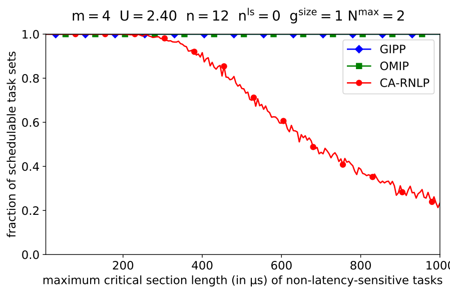

m = 4 U = 2.40 n = 8 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

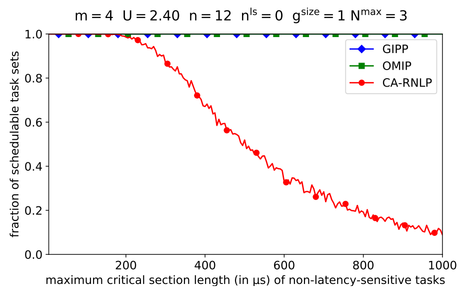

m = 4 U = 2.40 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

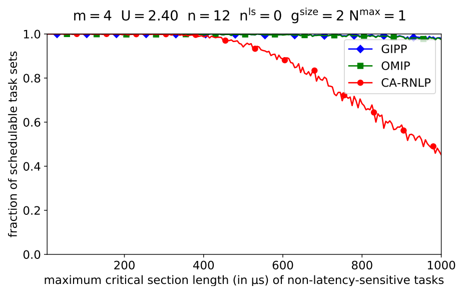

m = 4 U = 2.40 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

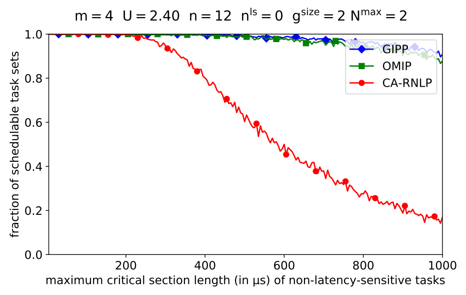

m = 4 U = 2.40 n = 8 $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

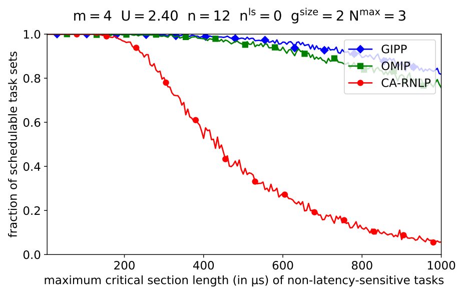

m = 4 U = 2.40 n = 12 $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

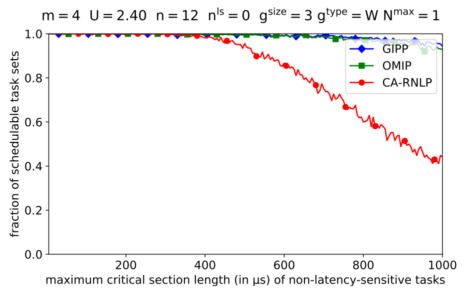

m = 4 U = 2.40 n = 12 $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

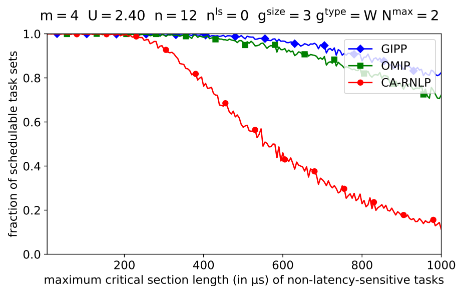

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

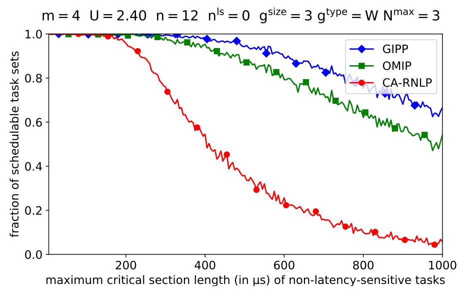

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

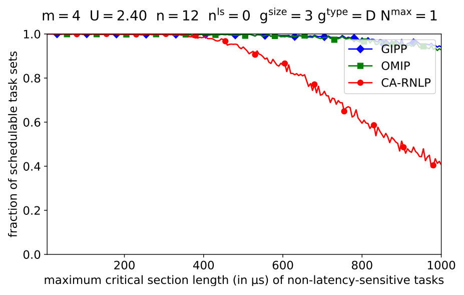

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

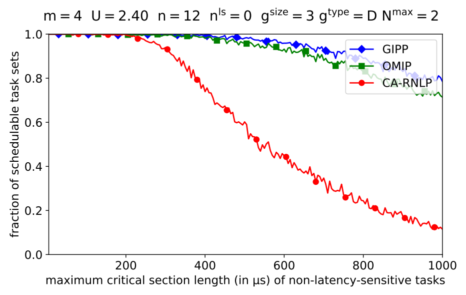

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

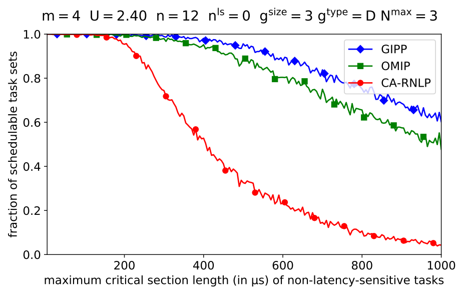

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

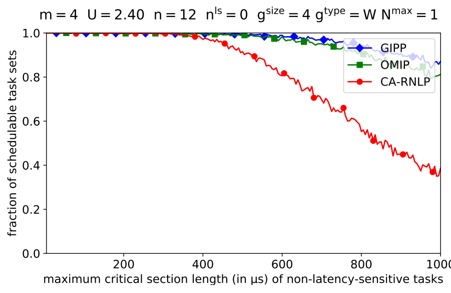

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

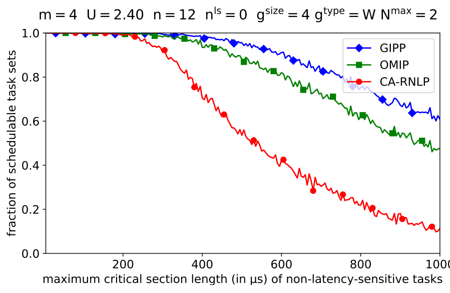

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

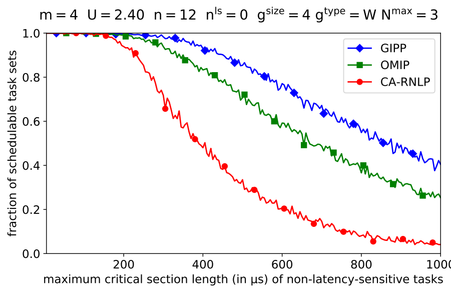

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

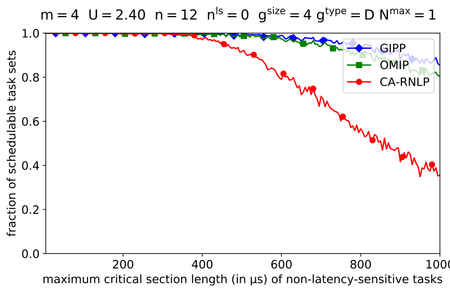

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

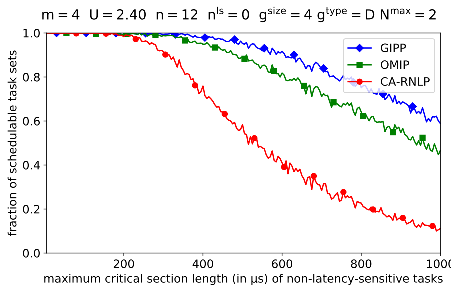

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

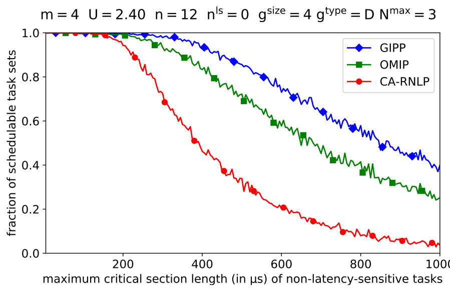

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

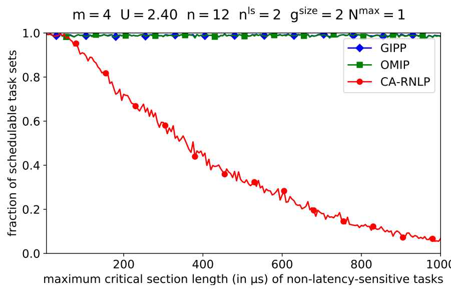

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

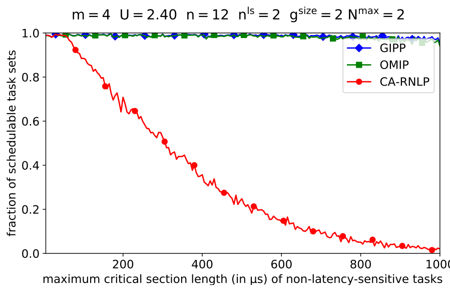

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

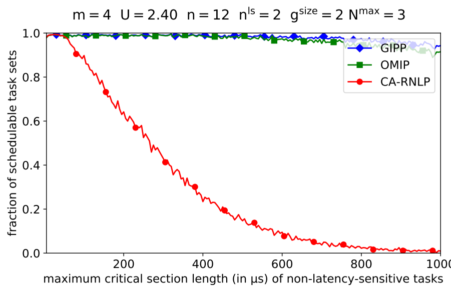

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

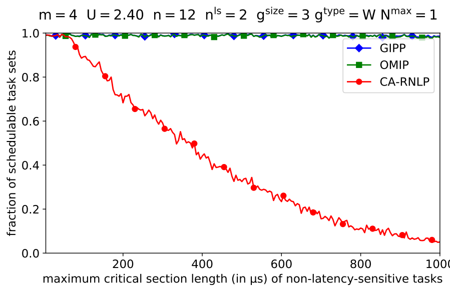

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

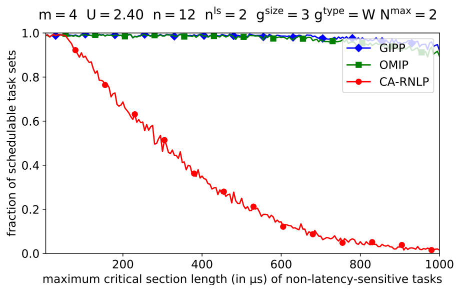

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

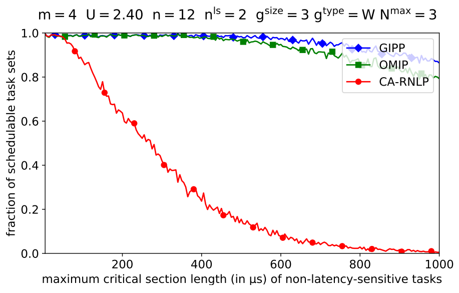

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

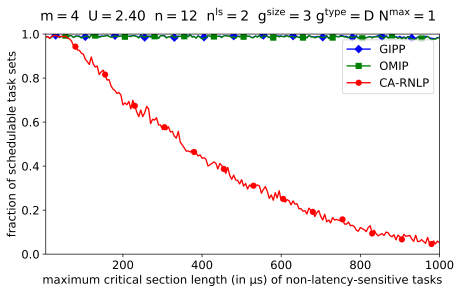

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

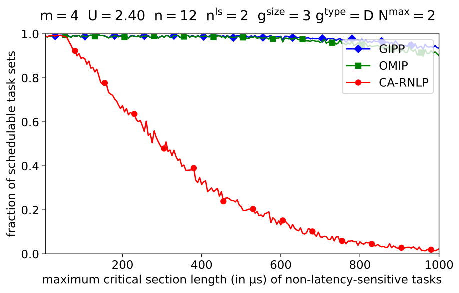

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 2$ $g^{size} = 2$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

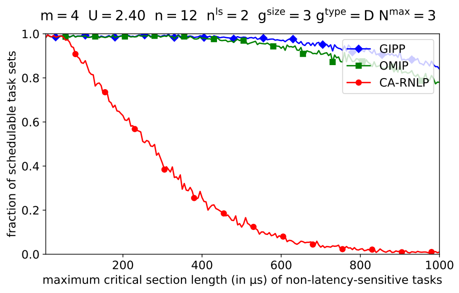

$m = 4$ $U = 2.40$ $n = 12$ $n^{ls} = 2$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

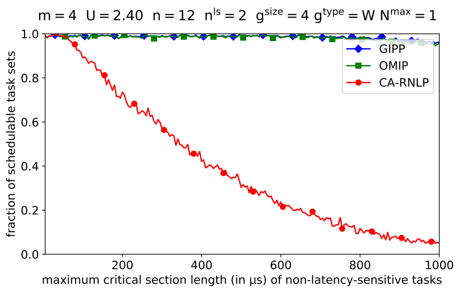

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

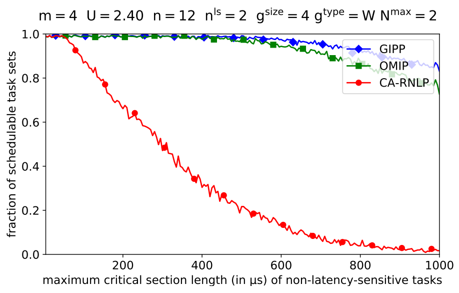

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

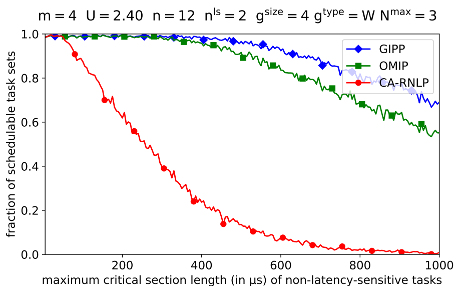

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

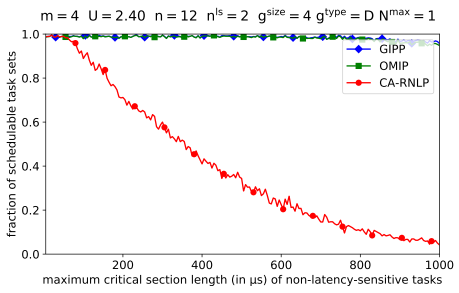

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

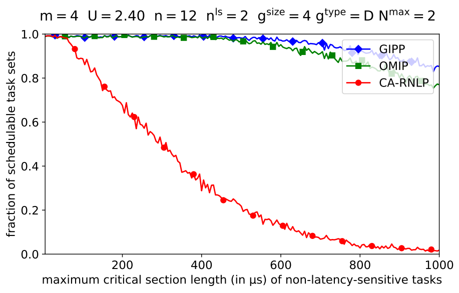

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

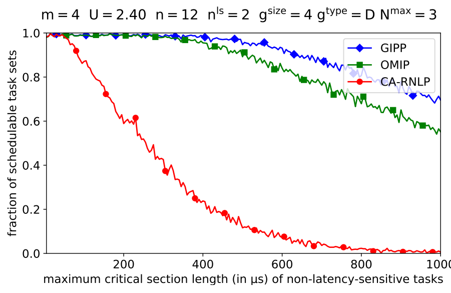

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

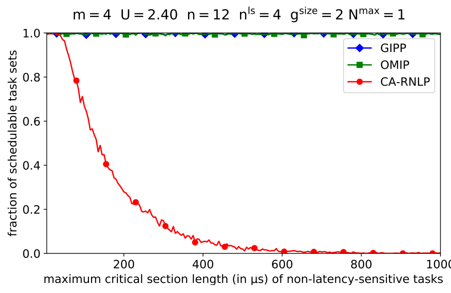

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

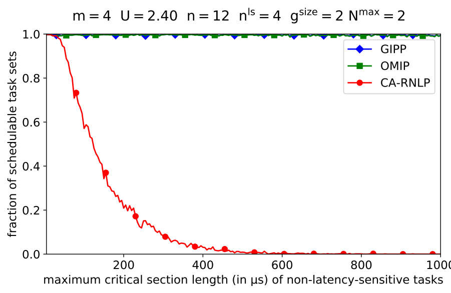

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

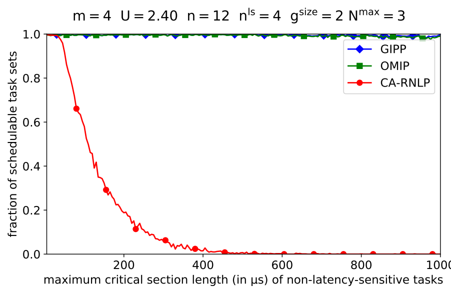

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

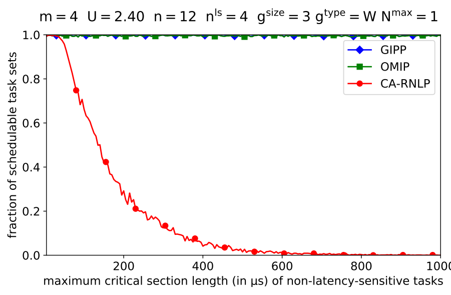

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

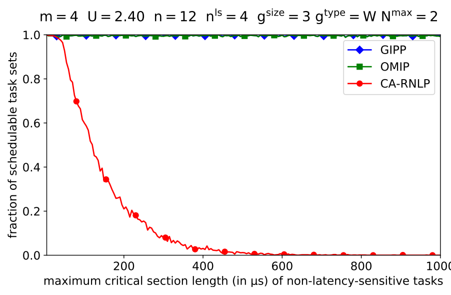

m = 4 U = 2.40 n = 12 n^ls = 2 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

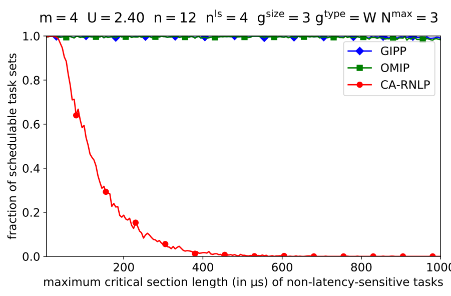

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

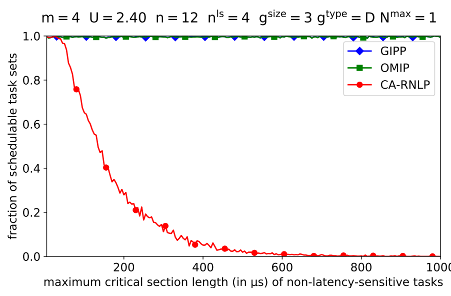

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

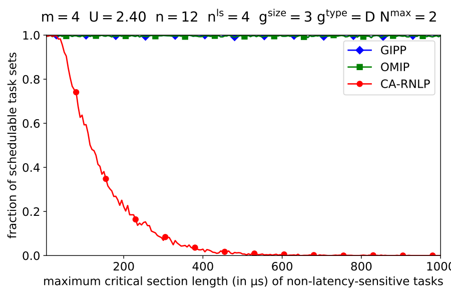

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

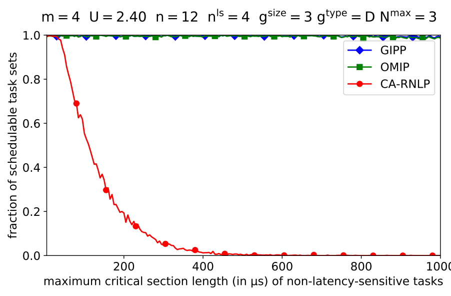

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

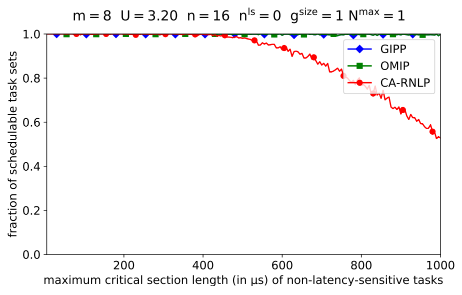

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

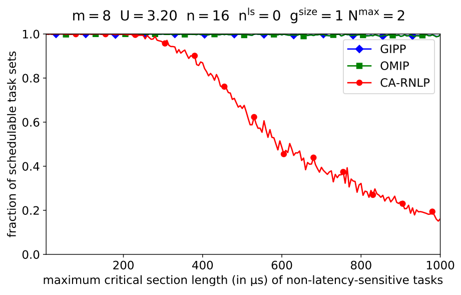

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

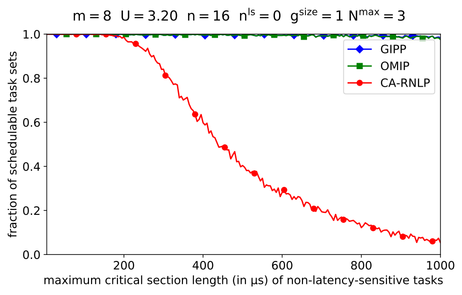

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

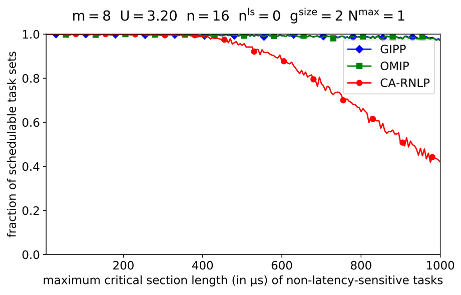

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

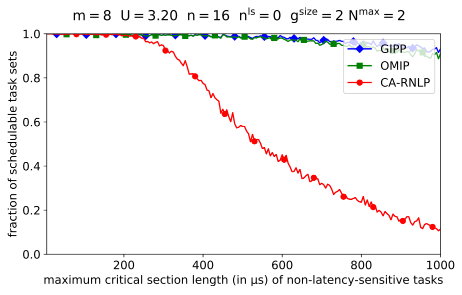

m = 4 U = 2.40 n = 12 n^ls = 4 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

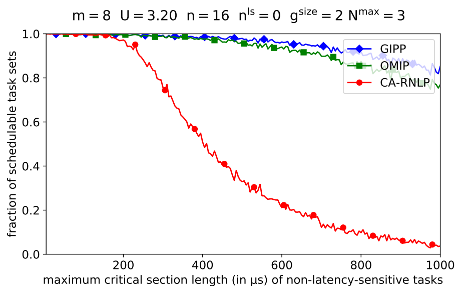

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

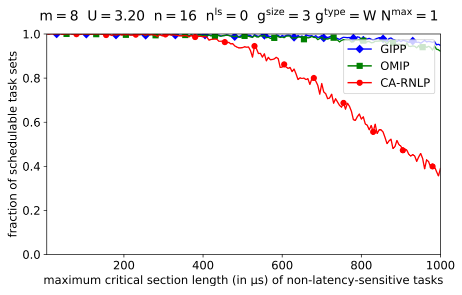

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

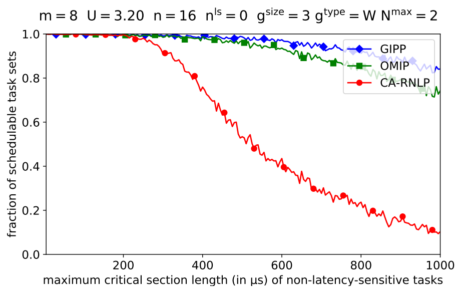

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

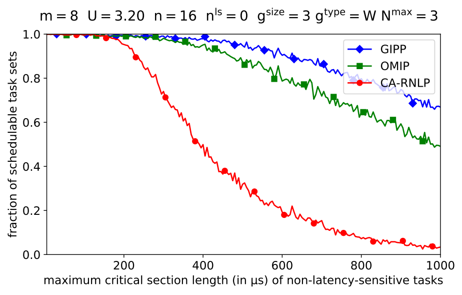

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

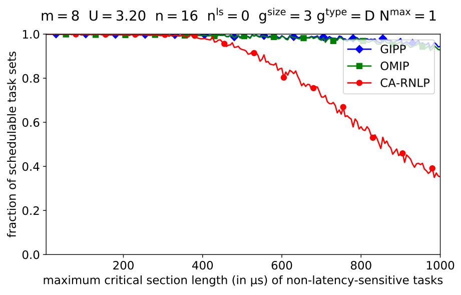

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

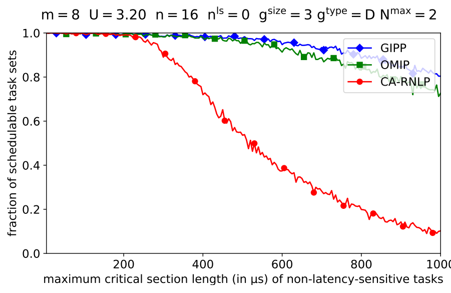

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

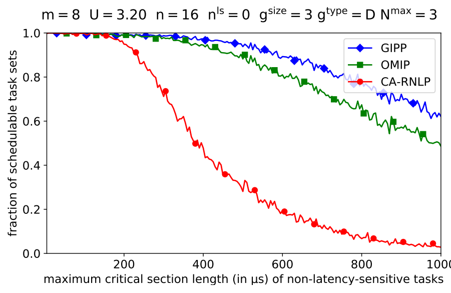

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

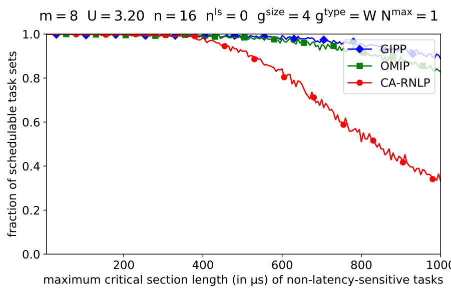

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

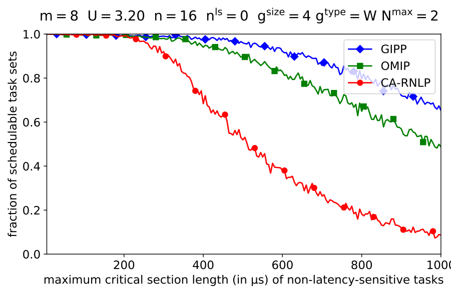

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

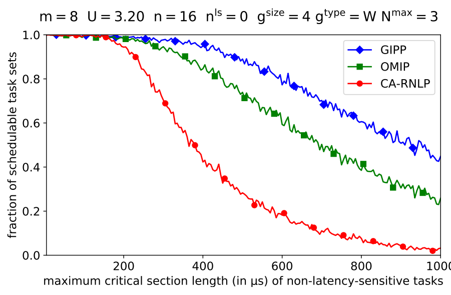

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

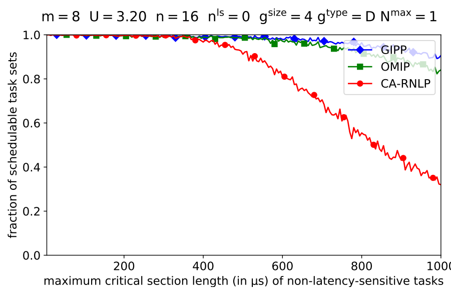

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

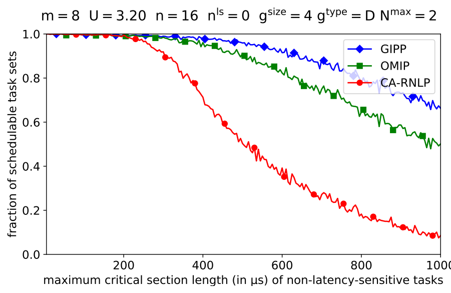

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

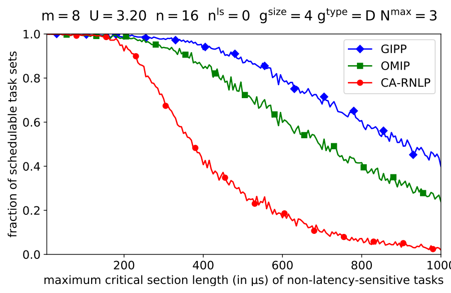

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

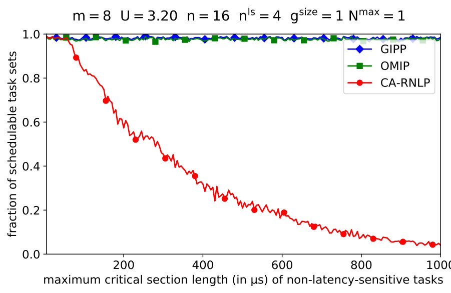

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

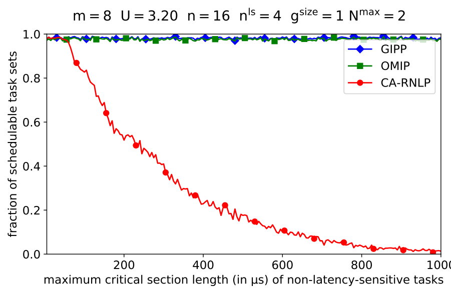

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

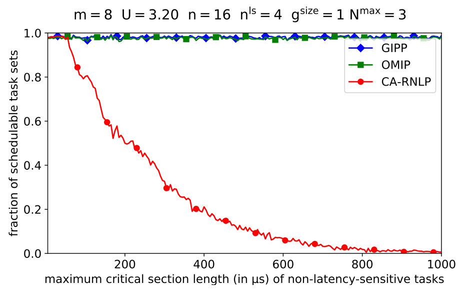

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

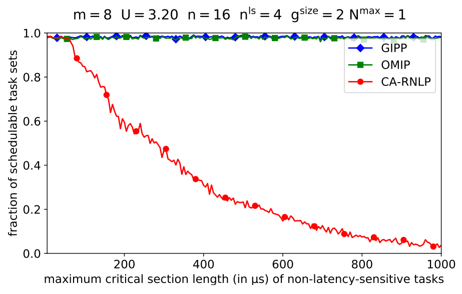

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

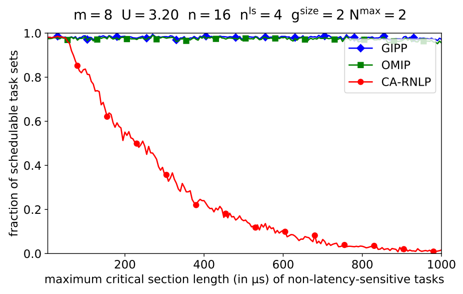

m = 8 U = 3.20 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

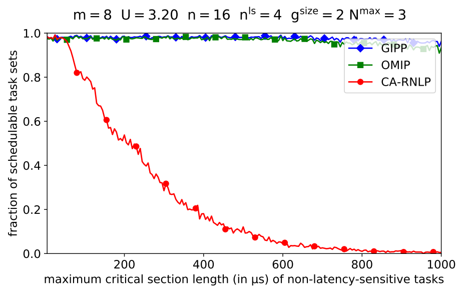

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

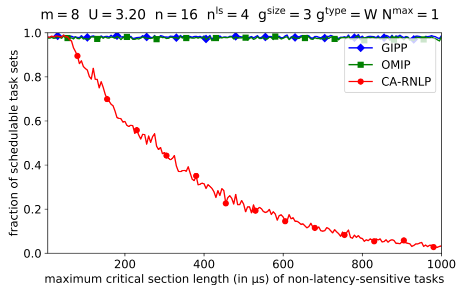

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

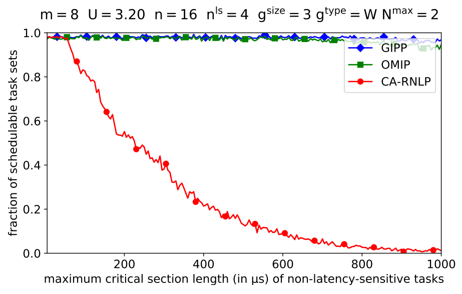

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

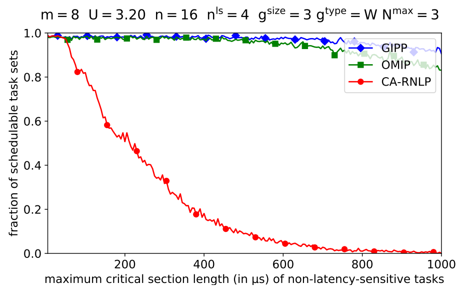

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

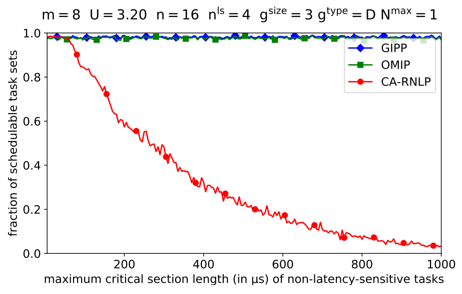

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

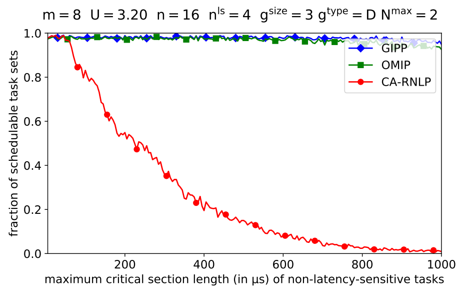

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

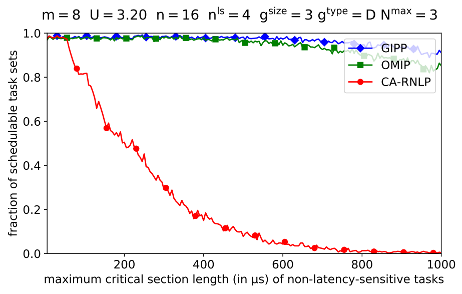

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

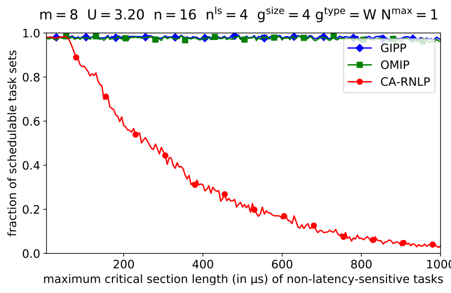

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

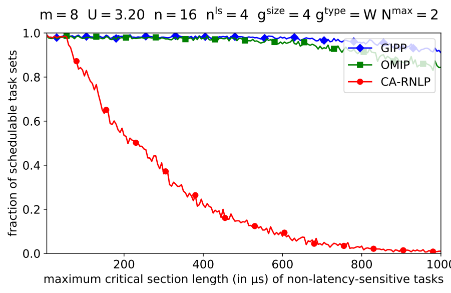

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

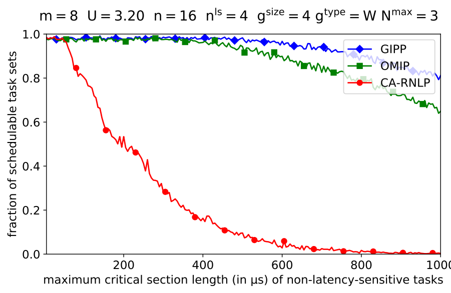

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

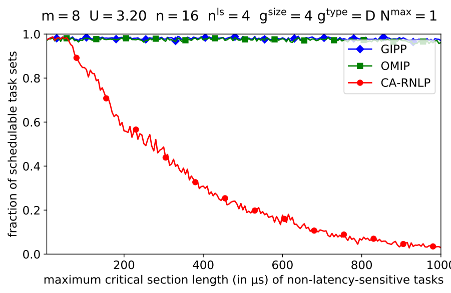

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

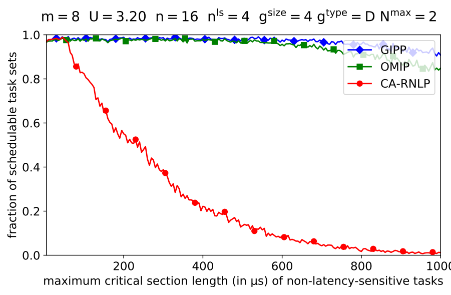

m = 8 U = 3.20 n = 16 n^ls = 4 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

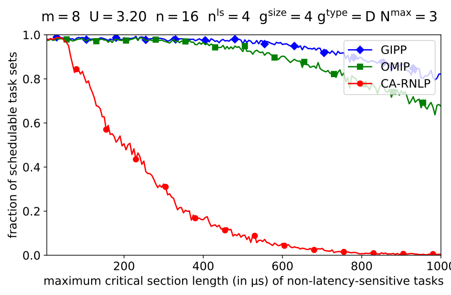

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = W N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

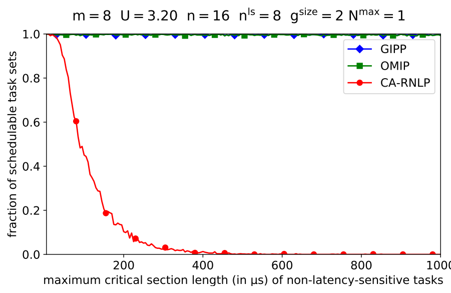

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = W N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

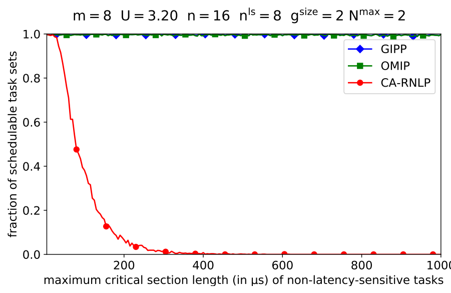

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = W N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

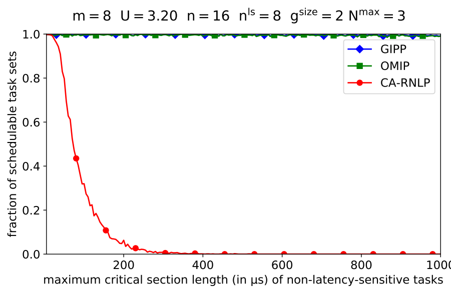

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = D N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

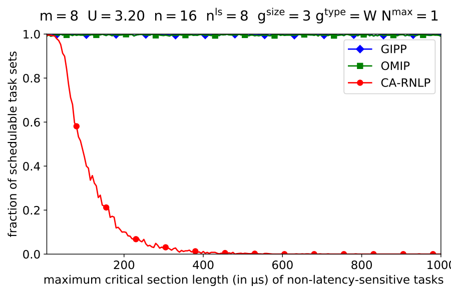

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = D N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

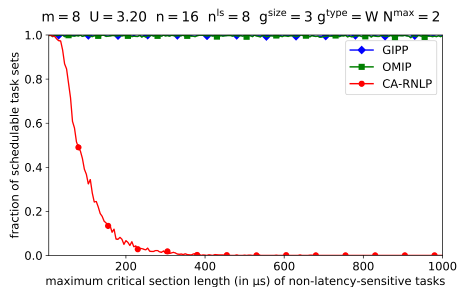

m = 8 U = 3.20 n = 16 n^{ls} = 4 g^{size} = 4 g^{type} = D N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

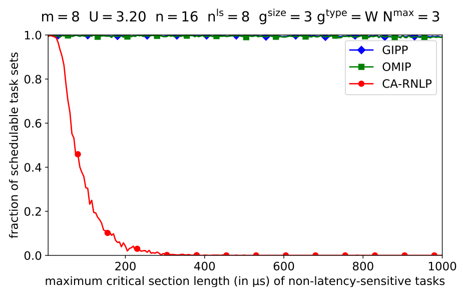

m = 8 U = 3.20 n = 16 n^{ls} = 8 g^{size} = 2 N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

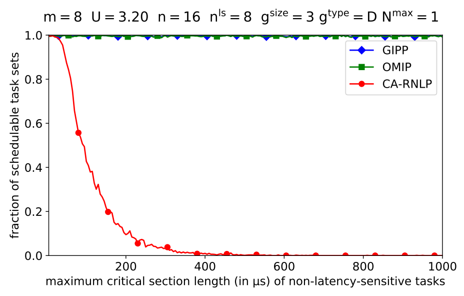

m = 8 U = 3.20 n = 16 n^{ls} = 8 g^{size} = 2 N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

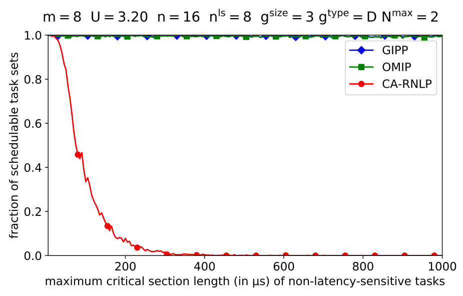

m = 8 U = 3.20 n = 16 n^{ls} = 8 g^{size} = 2 N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

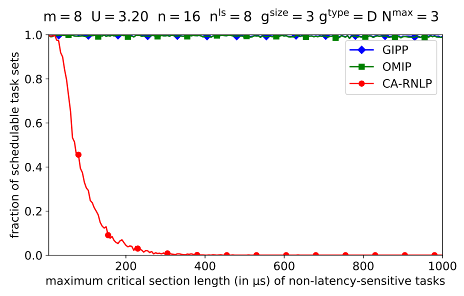

m = 8 U = 3.20 n = 16 n^{ls} = 8 g^{size} = 3 g^{type} = W N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

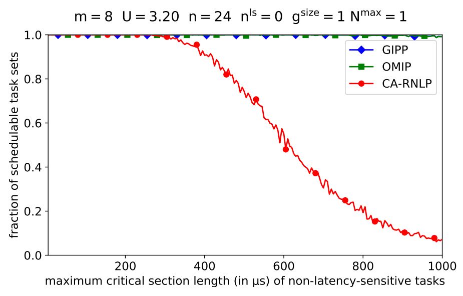

m = 8 U = 3.20 n = 16 n^ls = 8 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

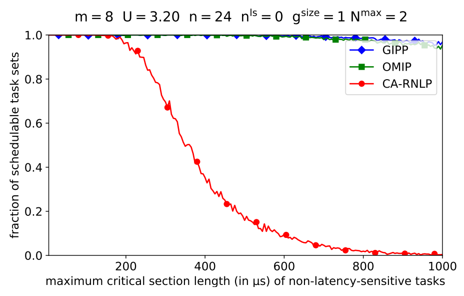

m = 8 U = 3.20 n = 16 n^ls = 8 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

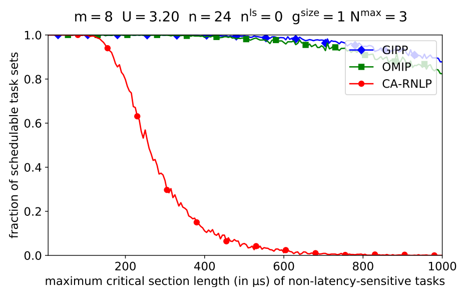

m = 8 U = 3.20 n = 16 n^ls = 8 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

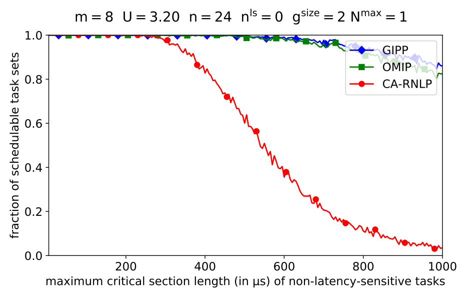

m = 8 U = 3.20 n = 16 n^ls = 8 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

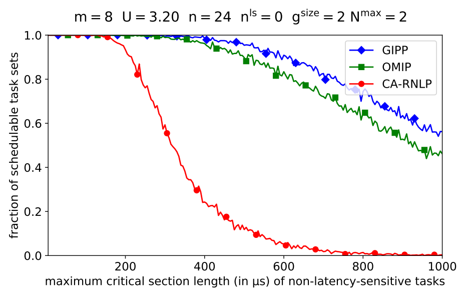

m = 8 U = 3.20 n = 16 n^ls = 8 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

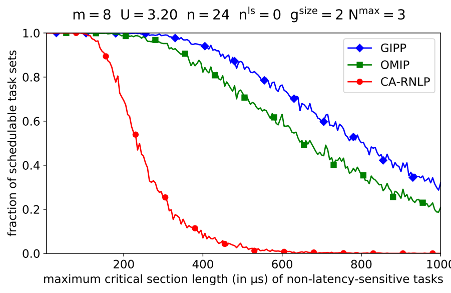

m = 8 U = 3.20 n = 24 n^ls = 0 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

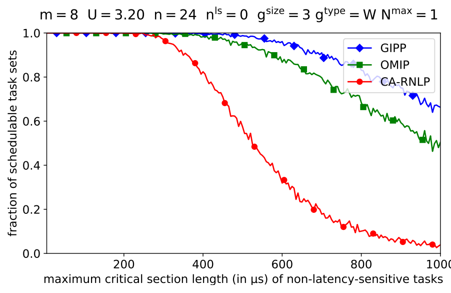

m = 8 U = 3.20 n = 24 n^ls = 0 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

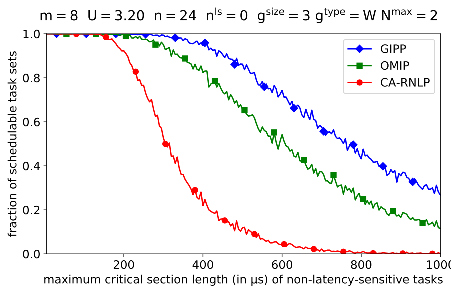

m = 8 U = 3.20 n = 24 n^ls = 0 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

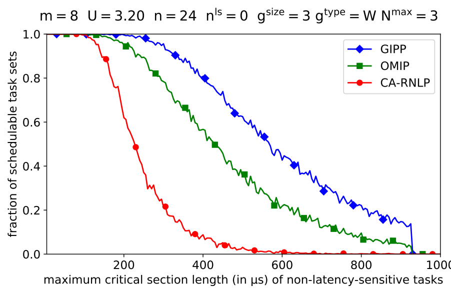

m = 8 U = 3.20 n = 24 n^ls = 0 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

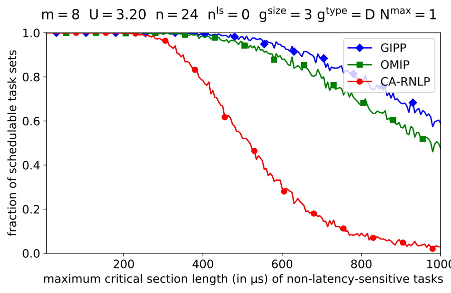

m = 8 U = 3.20 n = 24 n^ls = 0 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

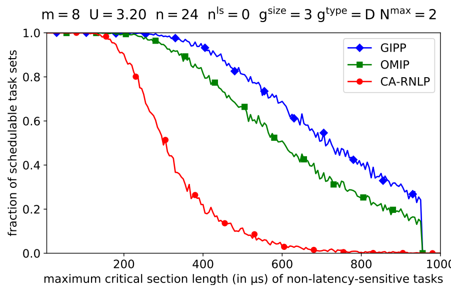

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 2 \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

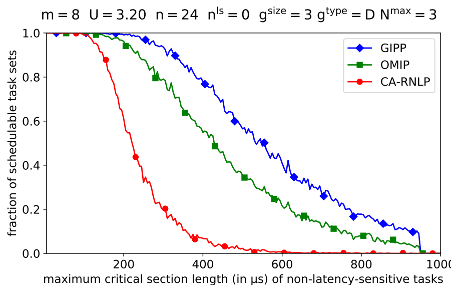

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = W \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

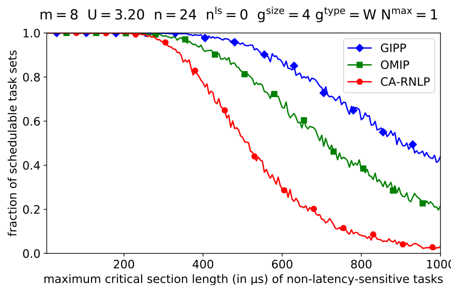

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = W \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

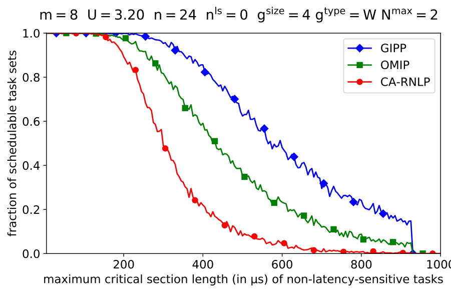

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = W \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

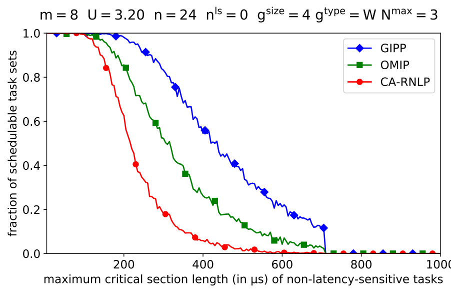

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = D \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

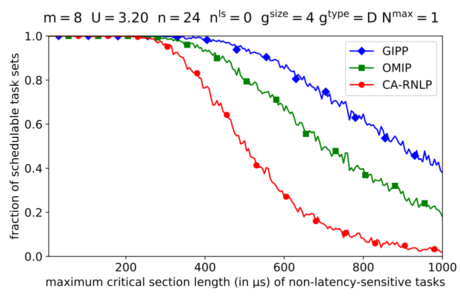

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = D \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

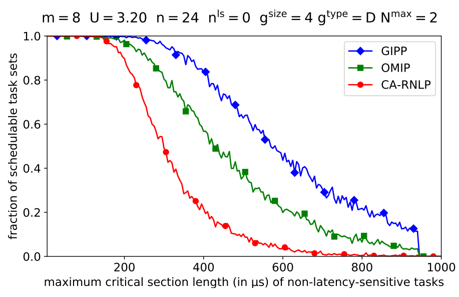

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 3 \; g^{type} = D \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

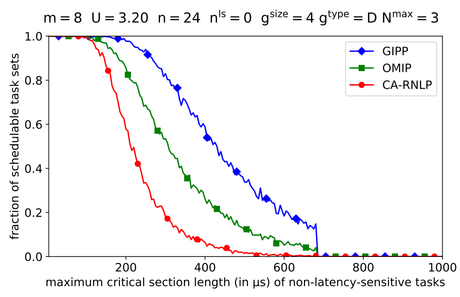

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 4 \; g^{type} = W \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

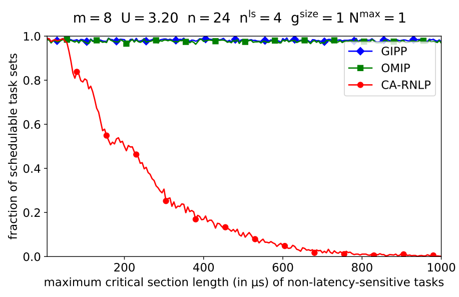

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 4 \; g^{type} = W \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

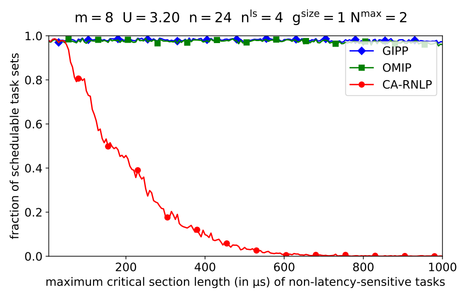

$m = 8 \;\; U = 3.20 \;\; n = 24 \;\; n^{ls} = 0 \;\; g^{size} = 4 \; g^{type} = W \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

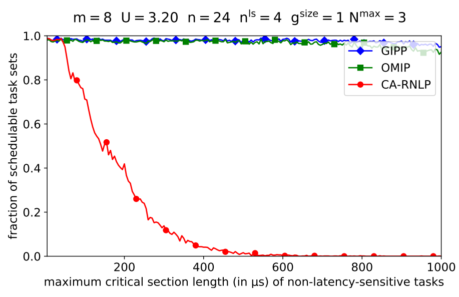

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

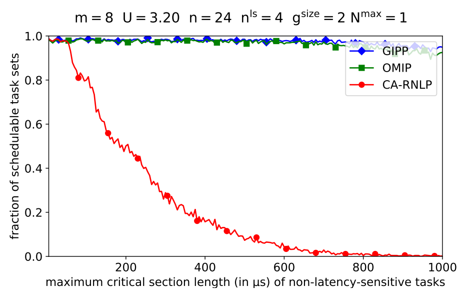

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

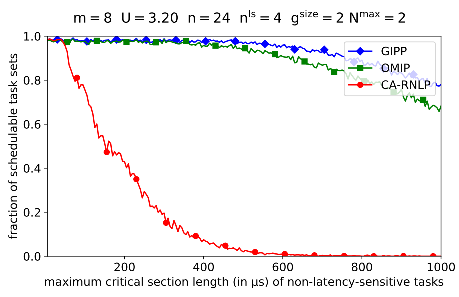

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

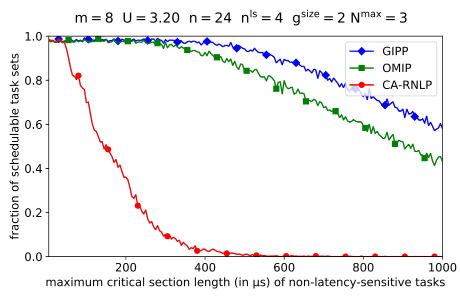

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

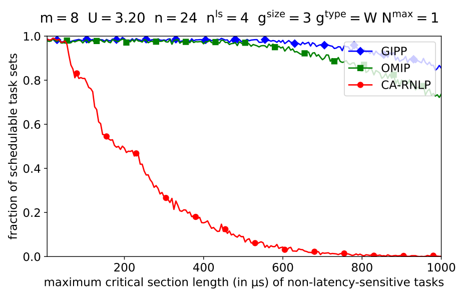

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

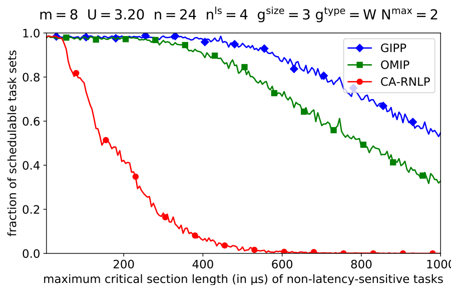

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 1$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

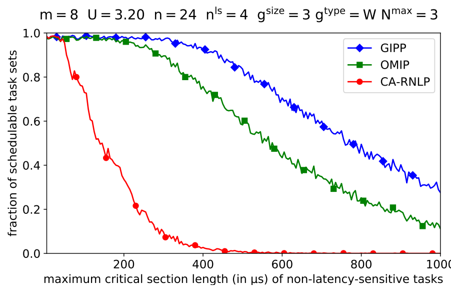

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

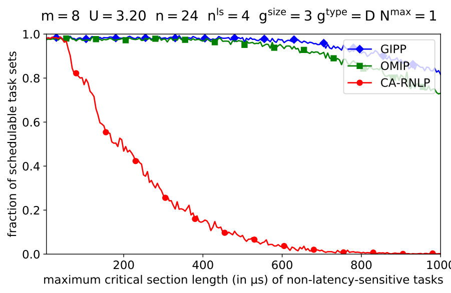

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

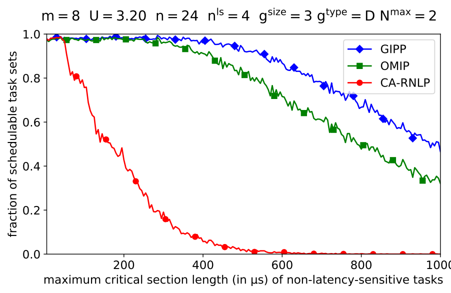

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 2$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

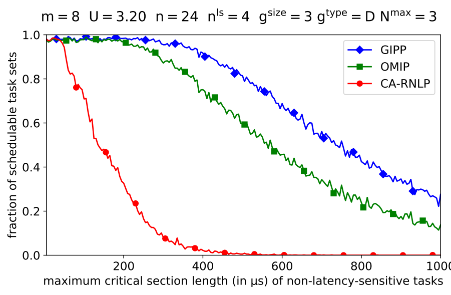

$m = 8$ $U = 3.20$ $n = 24$ $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

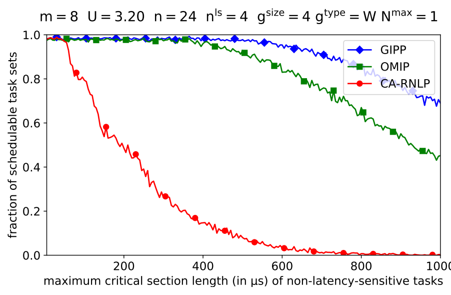

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

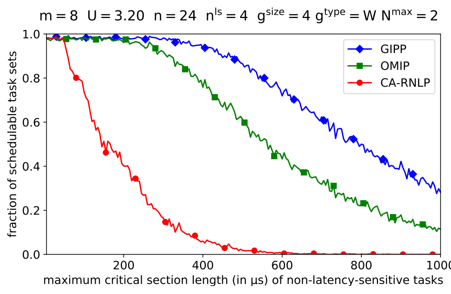

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

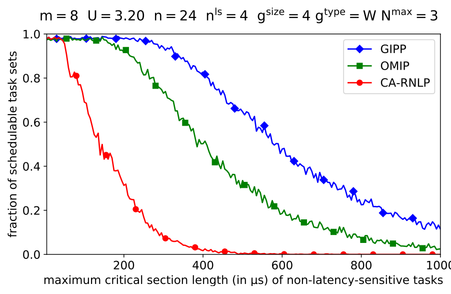

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

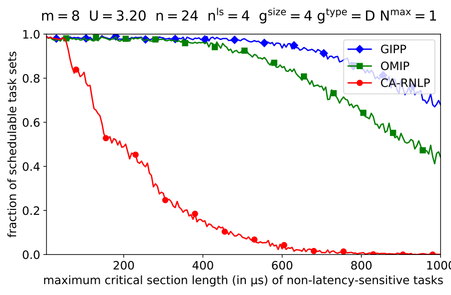

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

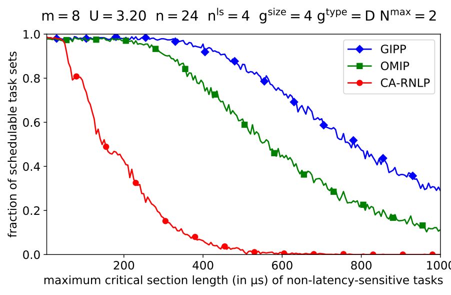

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

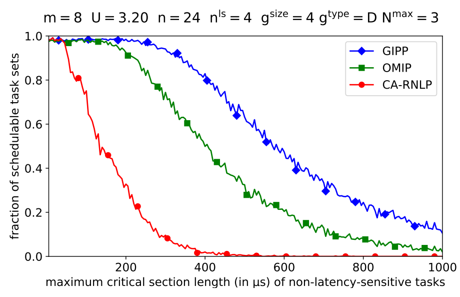

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

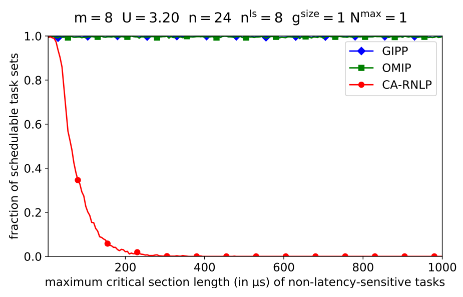

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

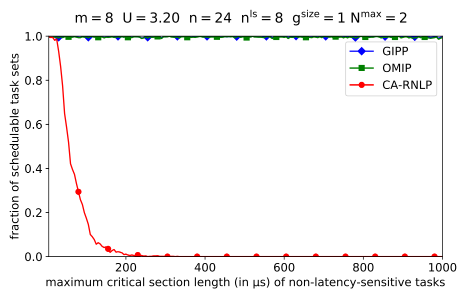

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

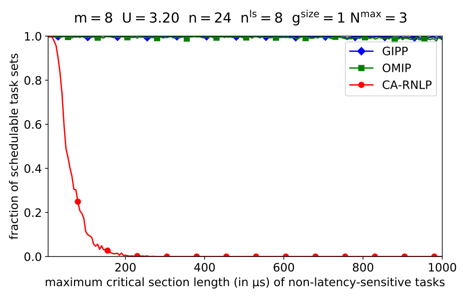

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

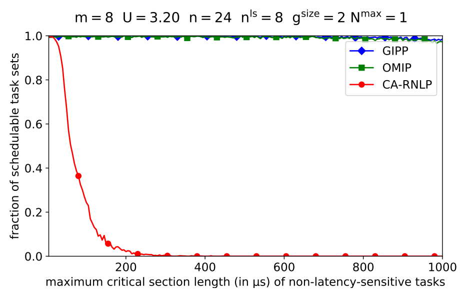

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

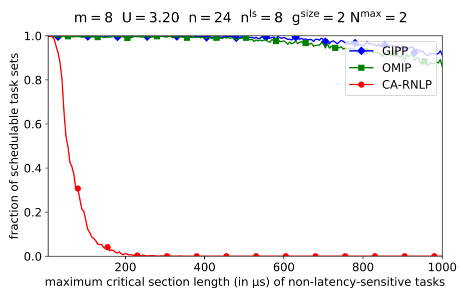

m = 8 U = 3.20 n = 24 n^ls = 4 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

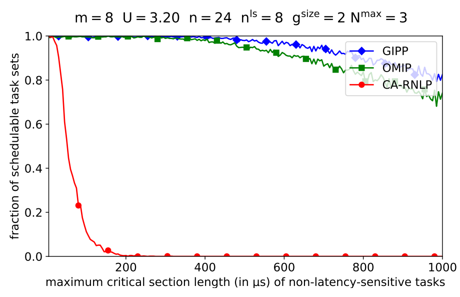

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

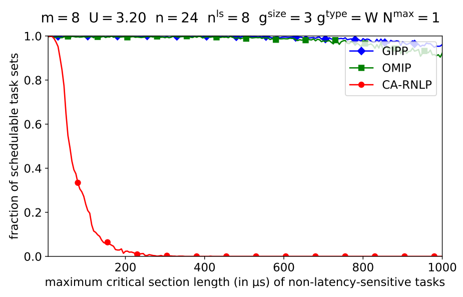

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

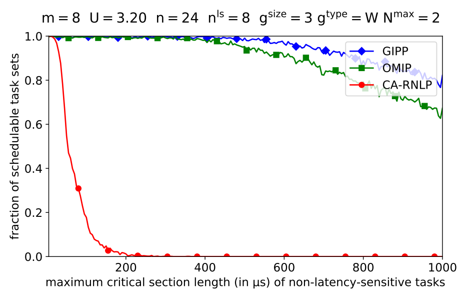

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

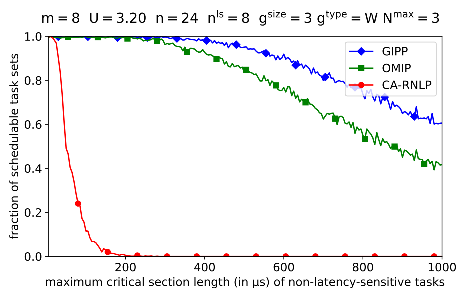

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

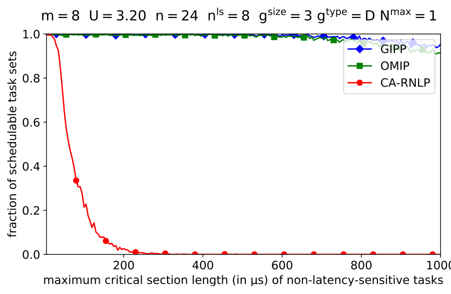

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

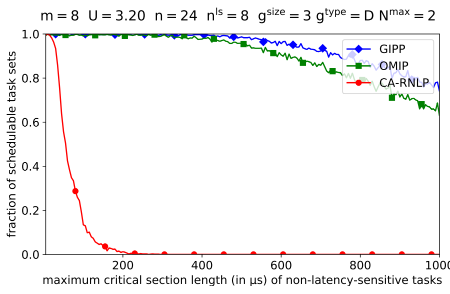

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

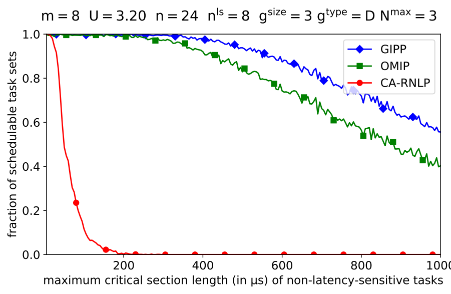

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

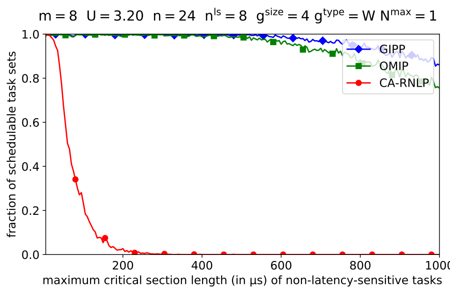

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

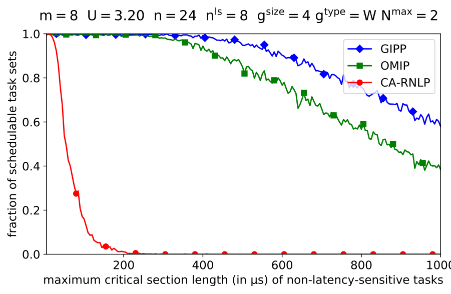

m = 8 U = 3.20 n = 24 n^ls = 8 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

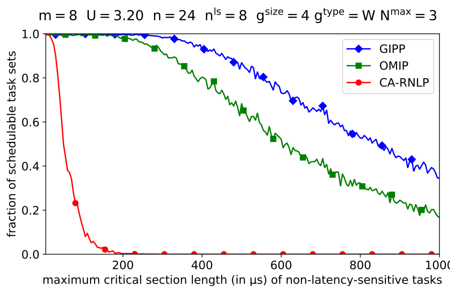

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = D N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

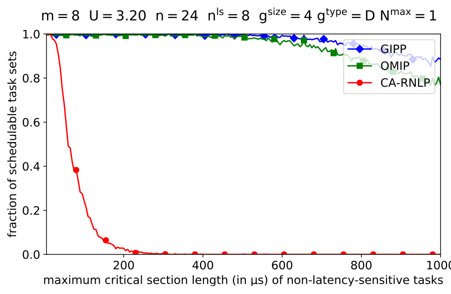

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = D N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

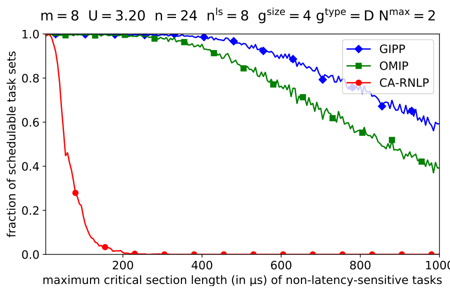

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = D N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

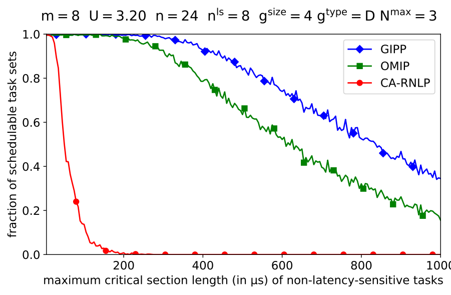

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = W N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

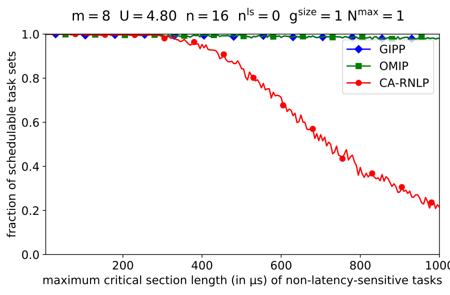

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = W N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

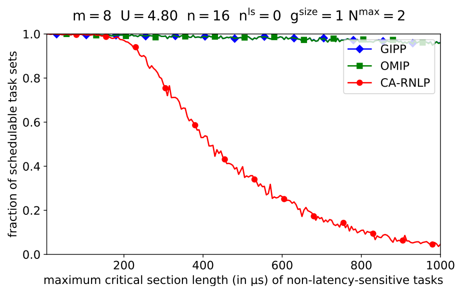

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = W N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

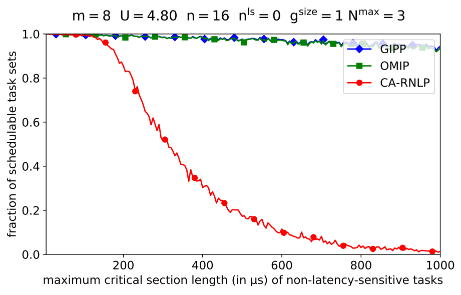

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = D N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

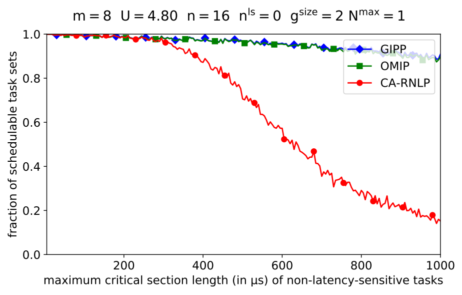

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = D N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

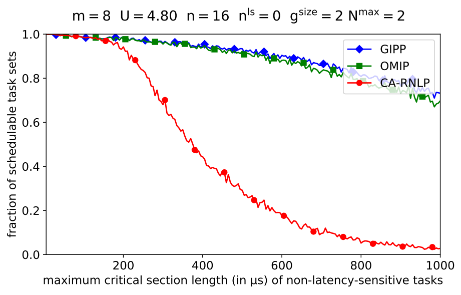

m = 8 U = 3.20 n = 24 n^{ls} = 8 g^{size} = 4 g^{type} = D N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

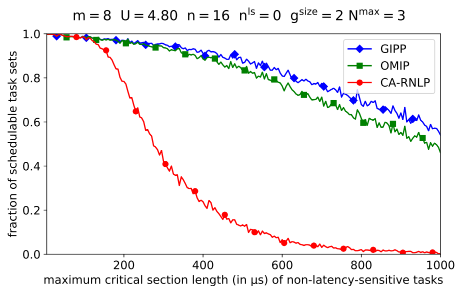

m = 8 U = 4.80 n = 16 n^{ls} = 0 g^{size} = 1 N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

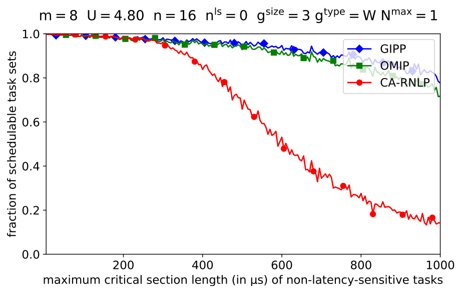

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

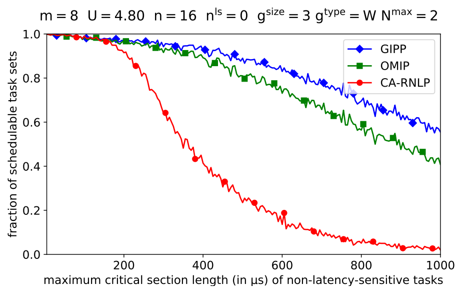

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

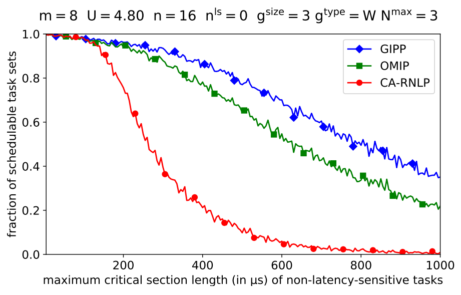

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

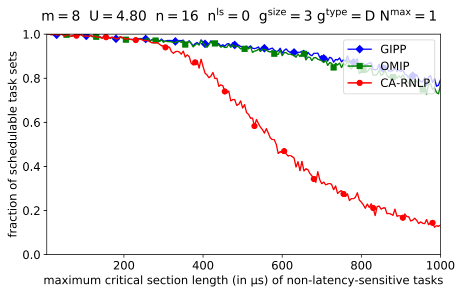

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

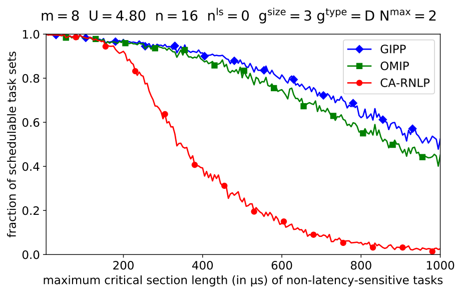

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

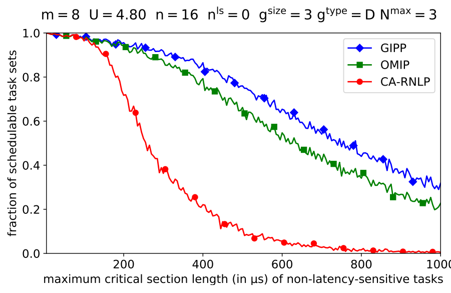

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

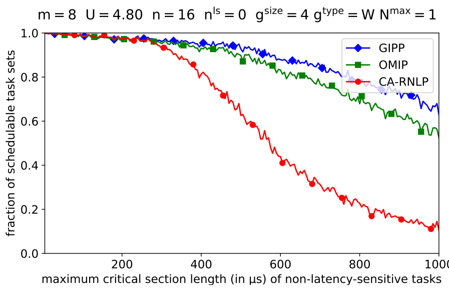

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

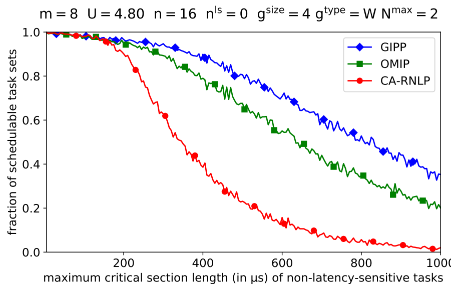

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

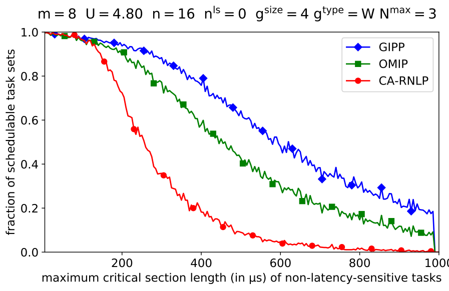

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

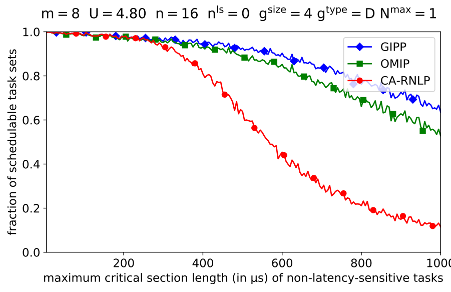

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

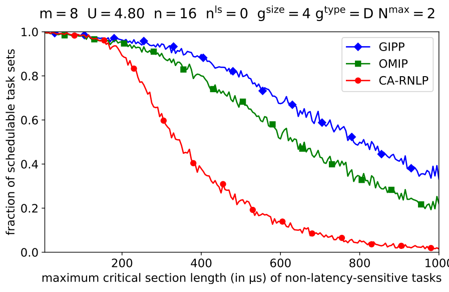

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

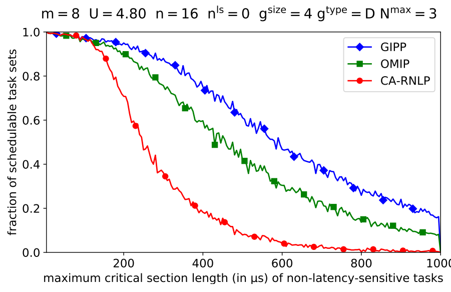

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

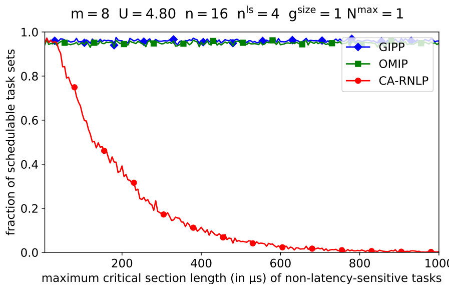

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

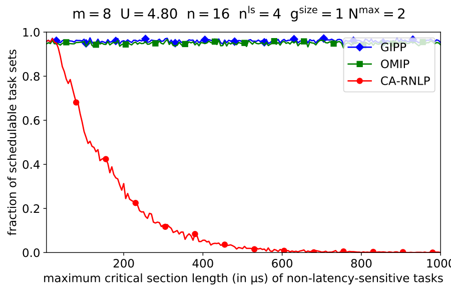

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

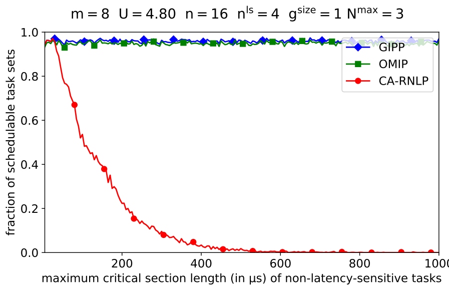

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

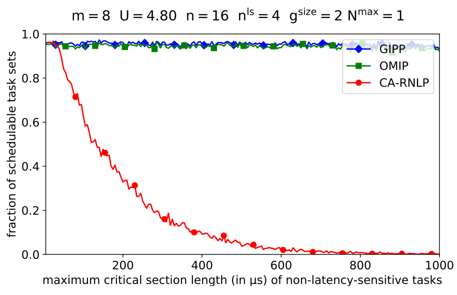

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

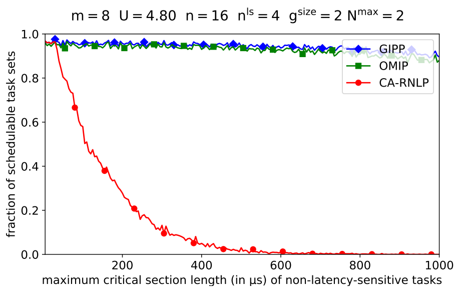

m = 8 U = 4.80 n = 16 n^ls = 0 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

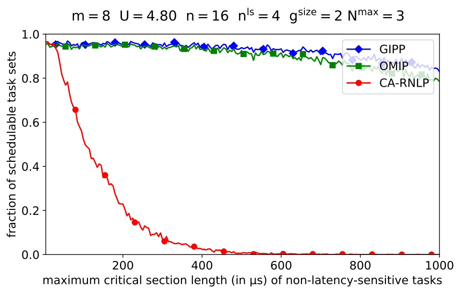

m = 8 U = 4.80 n = 16 n^ls = 4 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

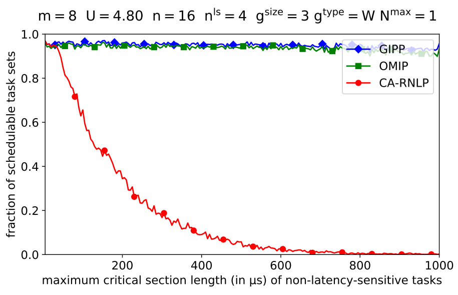

m = 8 U = 4.80 n = 16 n^ls = 4 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

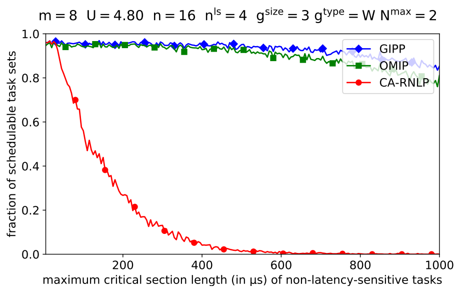

m = 8 U = 4.80 n = 16 n^ls = 4 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

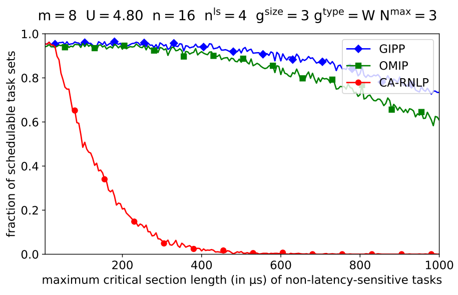

m = 8 U = 4.80 n = 16 nls = 4 gsize = 2 Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

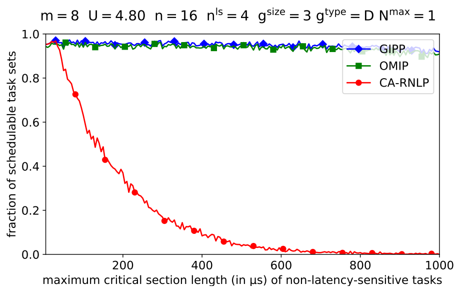

m = 8 U = 4.80 n = 16 nls = 4 gsize = 2 Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

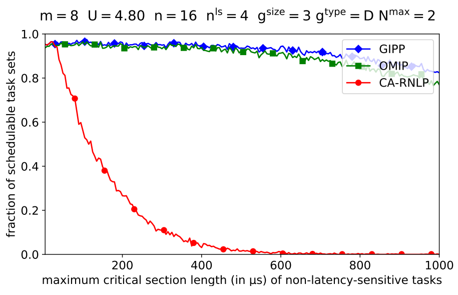

m = 8 U = 4.80 n = 16 nls = 4 gsize = 2 Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

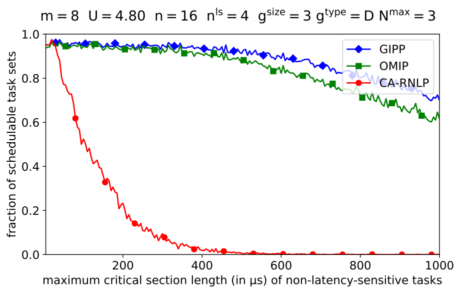

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = W Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

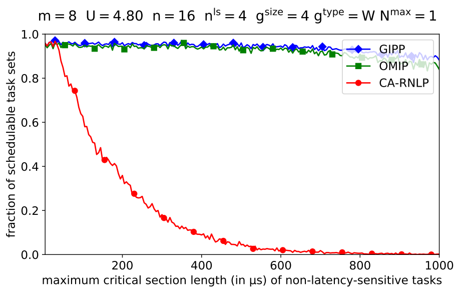

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = W Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

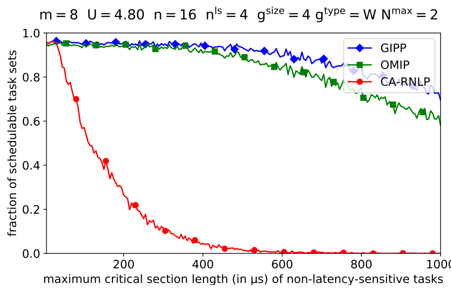

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = W Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

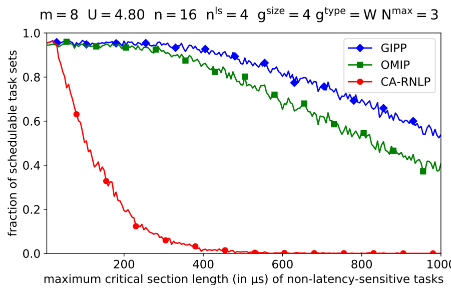

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = D Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

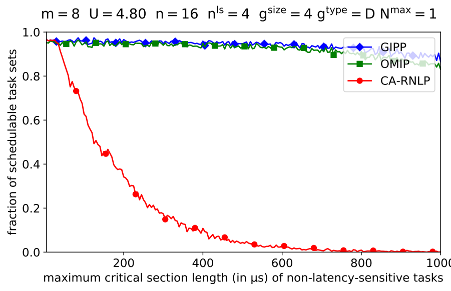

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = D Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

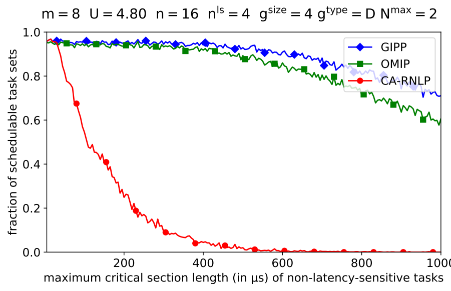

m = 8 U = 4.80 n = 16 nls = 4 gsize = 3 gtype = D Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

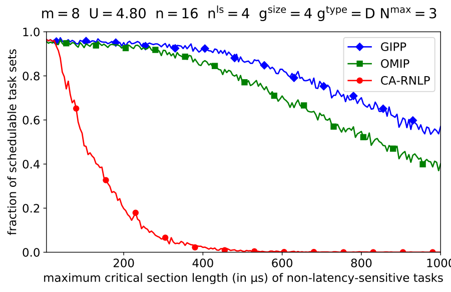

m = 8 U = 4.80 n = 16 nls = 4 gsize = 4 gtype = W Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

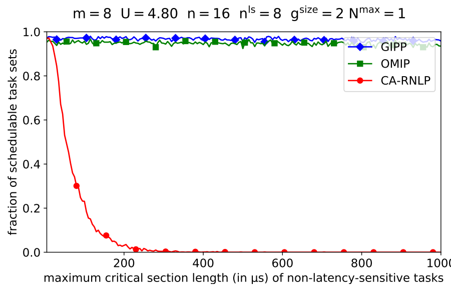

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

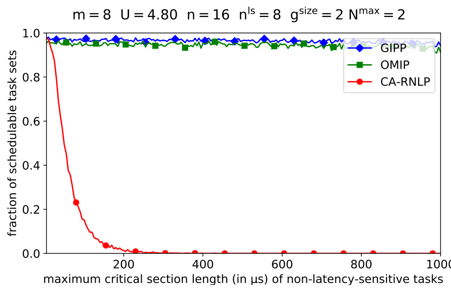

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

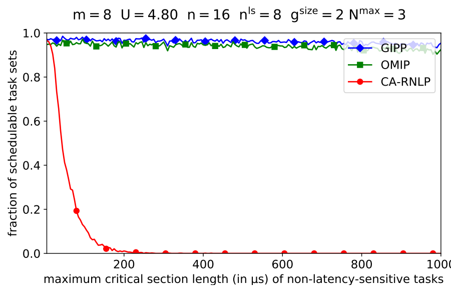

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

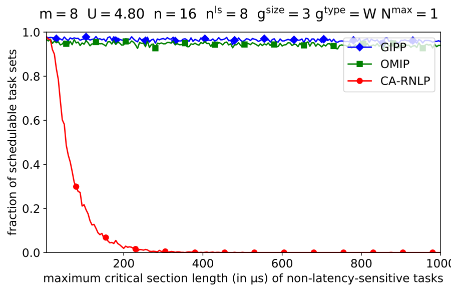

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

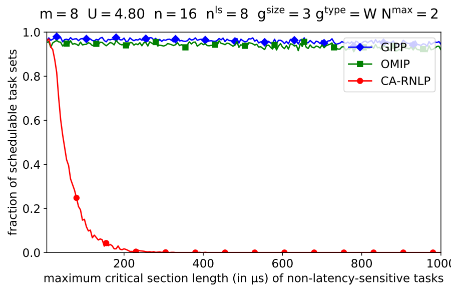

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

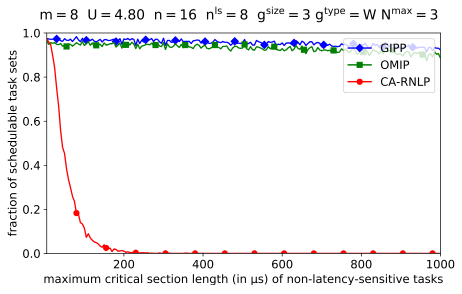

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

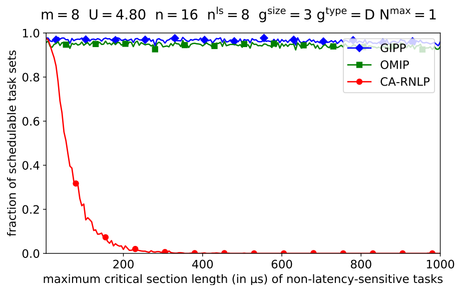

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

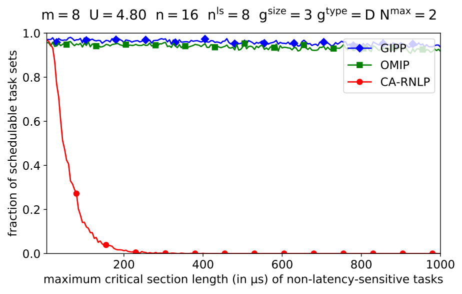

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

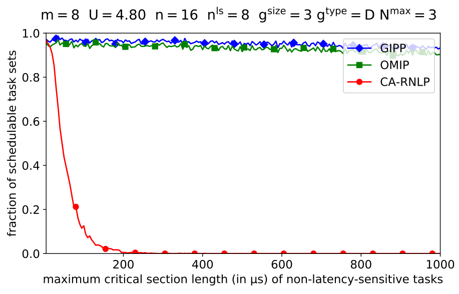

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

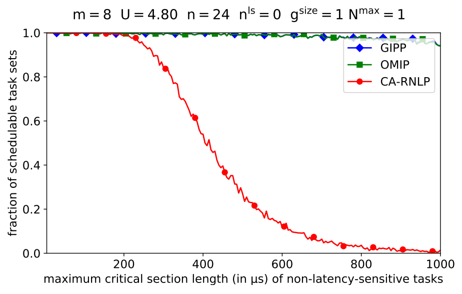

$m = 8$ $U = 4.80$ $n = 16$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

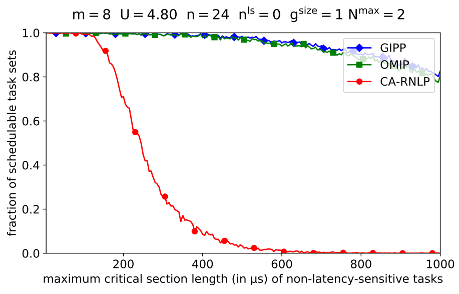

m = 8 U = 4.80 n = 16 nls = 8 gsize = 3 gtype = W Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

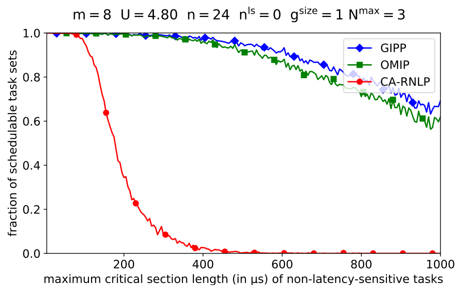

m = 8 U = 4.80 n = 16 nls = 8 gsize = 3 gtype = D Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

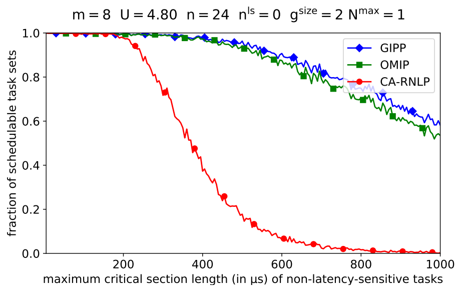

m = 8 U = 4.80 n = 16 nls = 8 gsize = 3 gtype = D Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

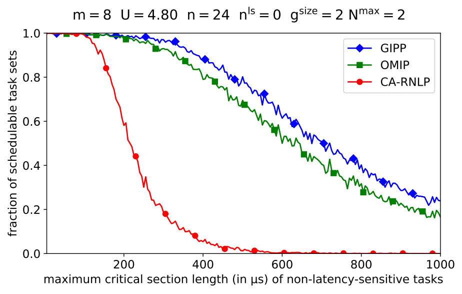

m = 8 U = 4.80 n = 16 nls = 8 gsize = 3 gtype = D Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

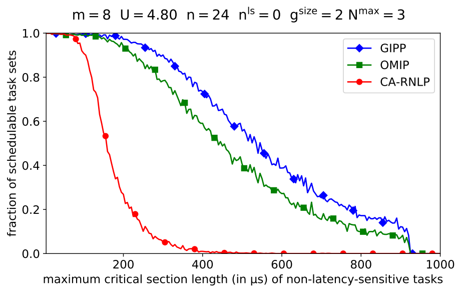

m = 8 U = 4.80 n = 24 nls = 0 gsize = 1 Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

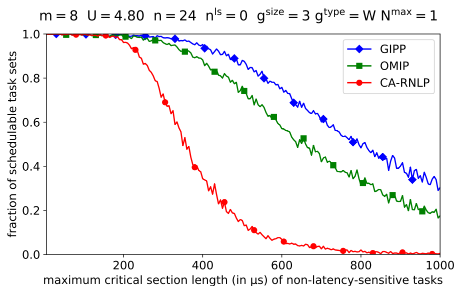

m = 8 U = 4.80 n = 24 nls = 0 gsize = 1 Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

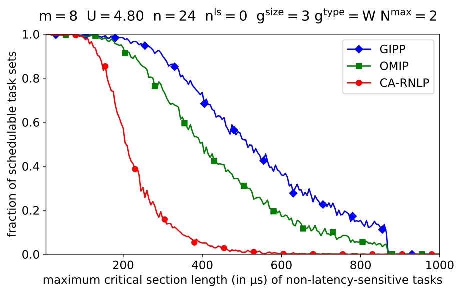

m = 8 U = 4.80 n = 24 nls = 0 gsize = 1 Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

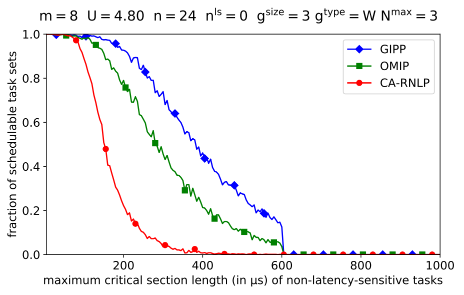

m = 8 U = 4.80 n = 24 nls = 0 gsize = 2 Nmax = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

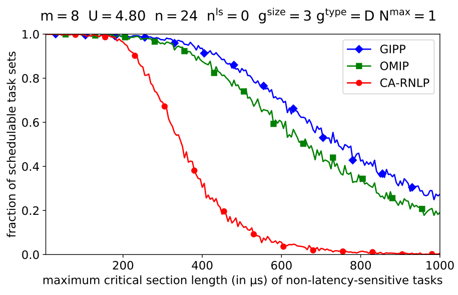

m = 8 U = 4.80 n = 24 nls = 0 gsize = 2 Nmax = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

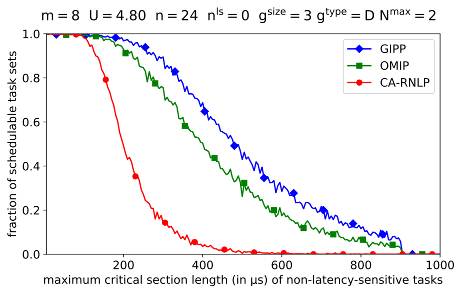

m = 8 U = 4.80 n = 24 nls = 0 gsize = 2 Nmax = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

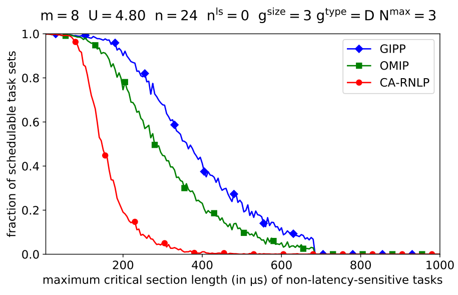

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

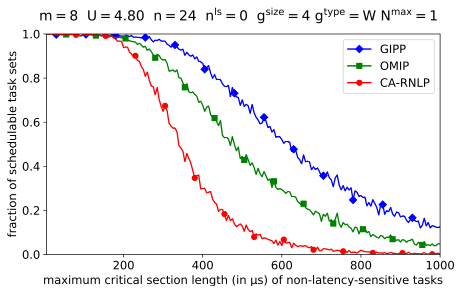

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

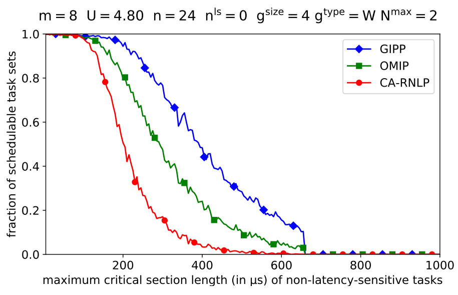

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

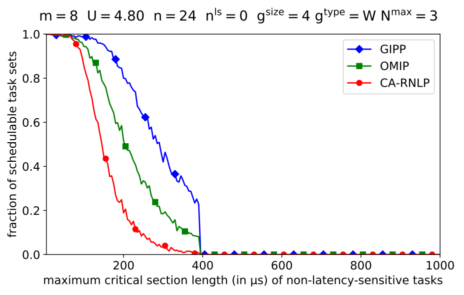

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

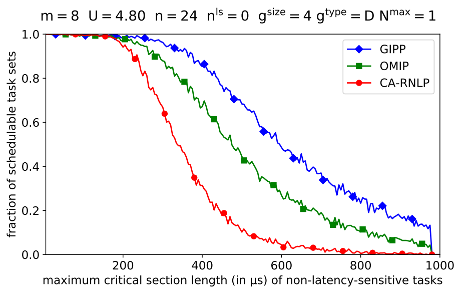

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

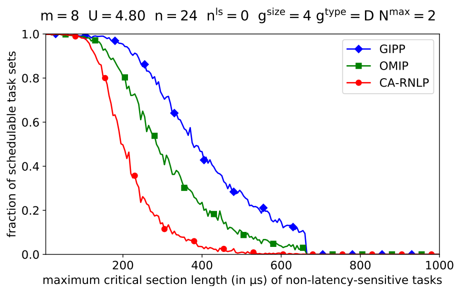

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

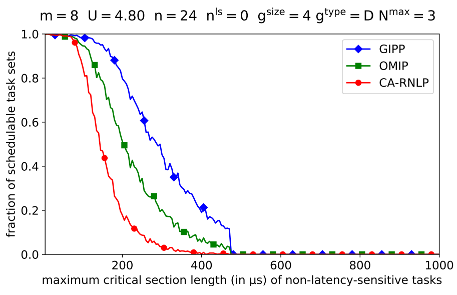

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

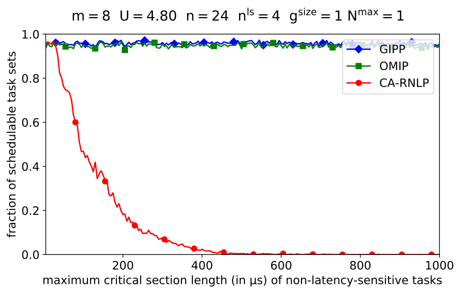

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

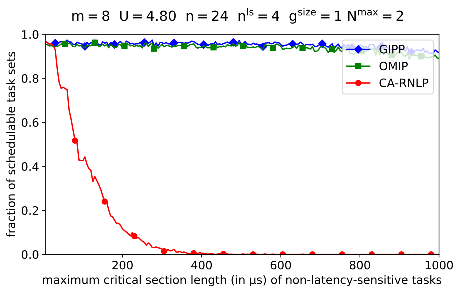

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

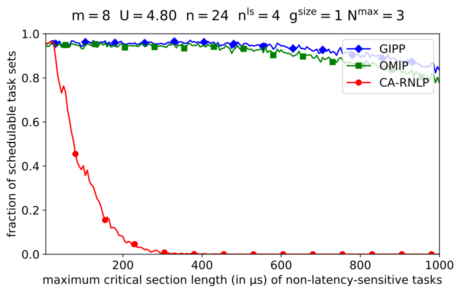

m = 8 U = 4.80 n = 24 $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

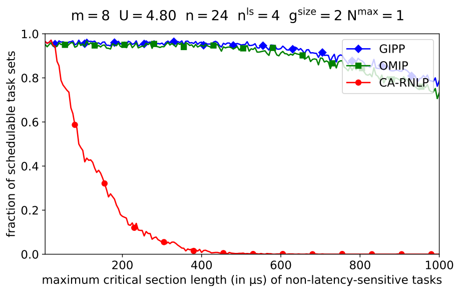

m = 8 U = 4.80 n = 24 n^ls = 0 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

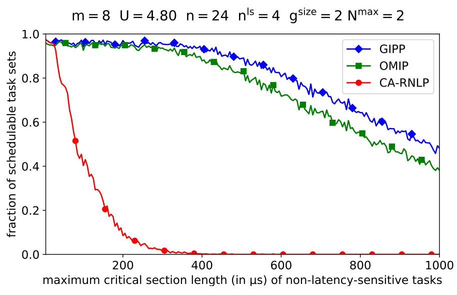

m = 8 U = 4.80 n = 24 n^ls = 0 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

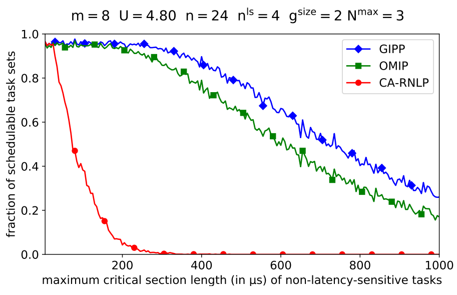

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

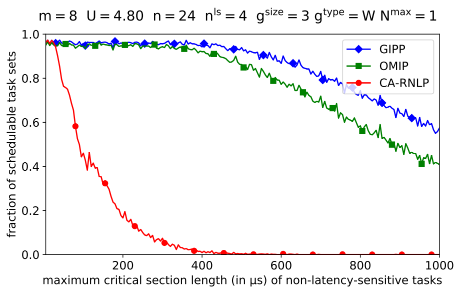

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

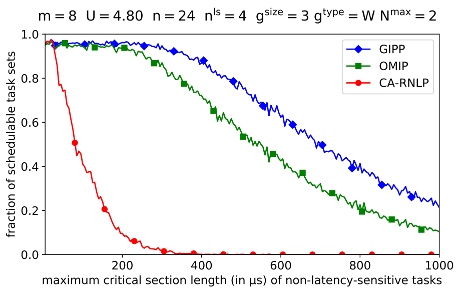

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

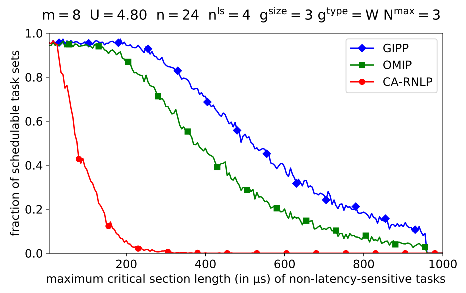

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

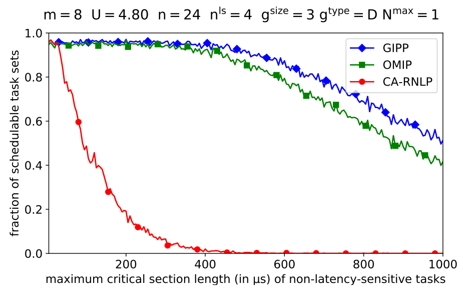

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

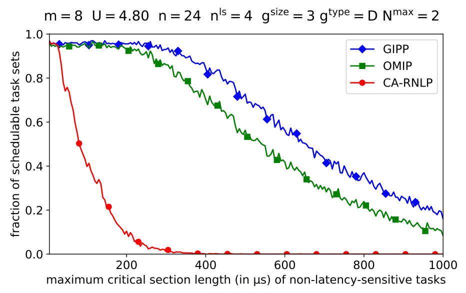

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

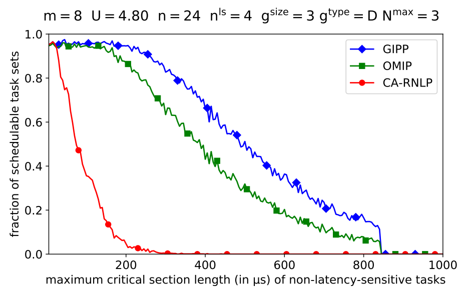

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

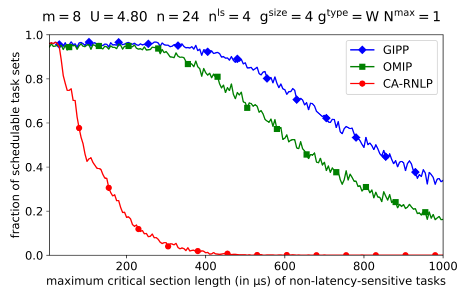

m = 8 U = 4.80 n = 24 n^ls = 4 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

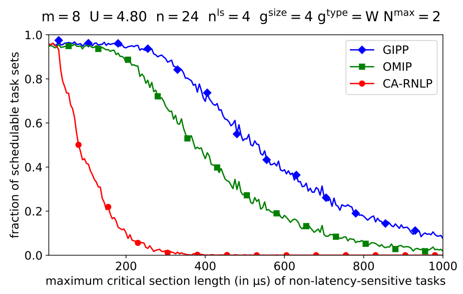

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

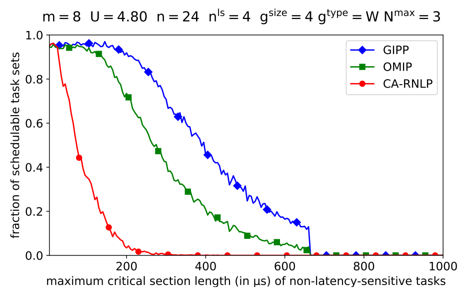

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

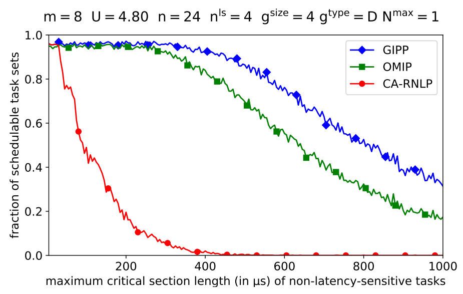

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

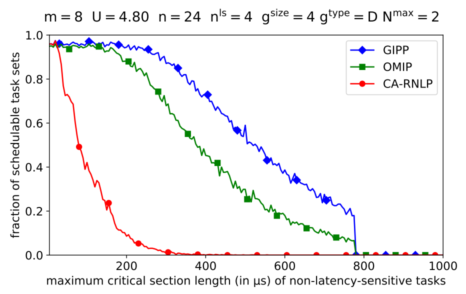

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

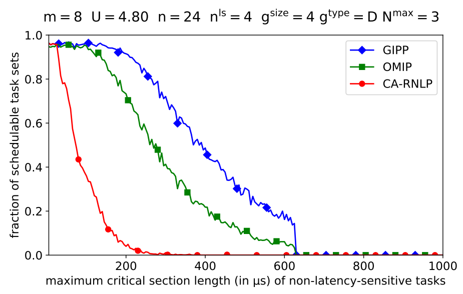

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

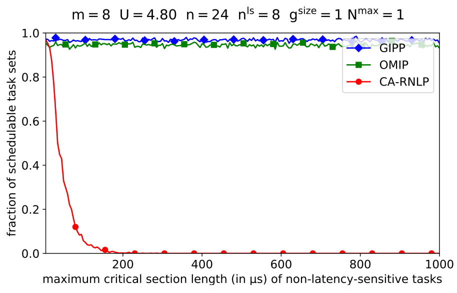

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

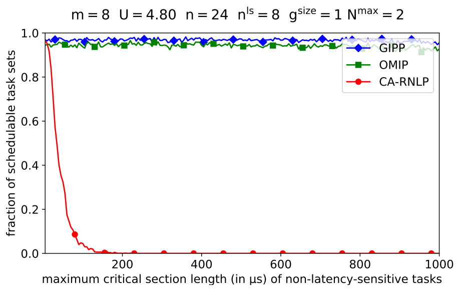

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

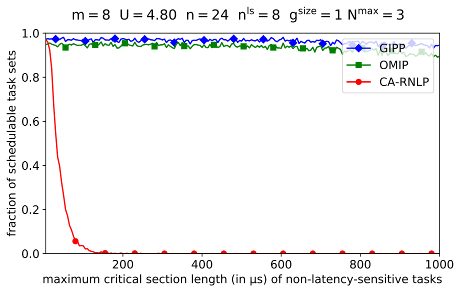

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

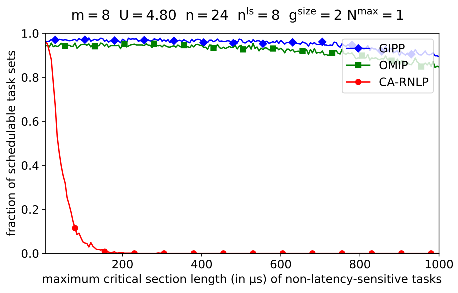

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

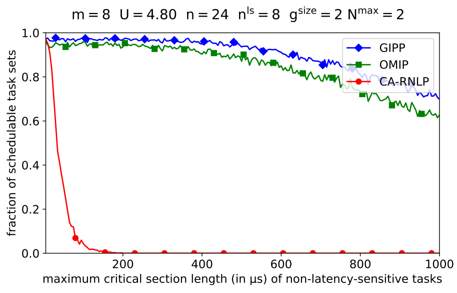

m = 8 U = 4.80 n = 24 $n^{ls} = 4$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

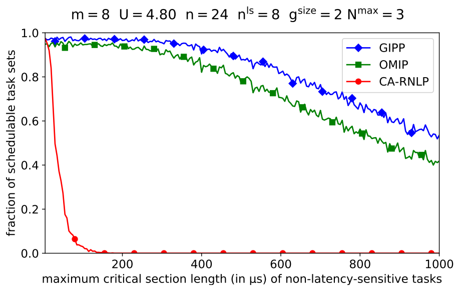

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 1 N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

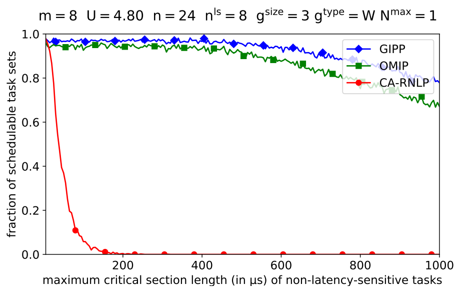

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 1 N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

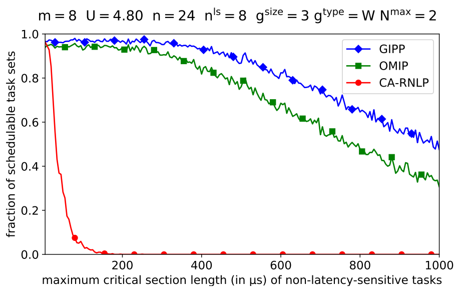

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 1 N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

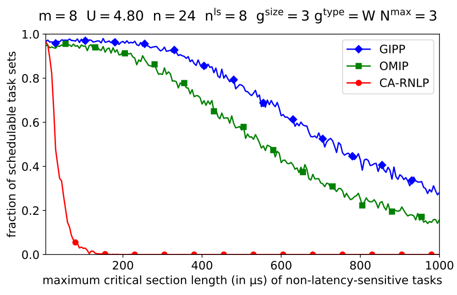

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 2 N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

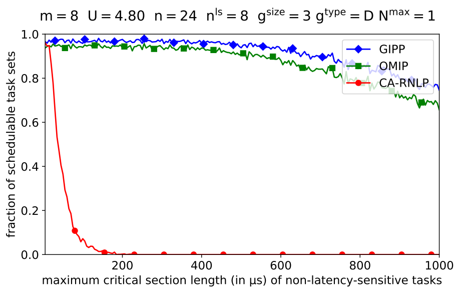

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 2 N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

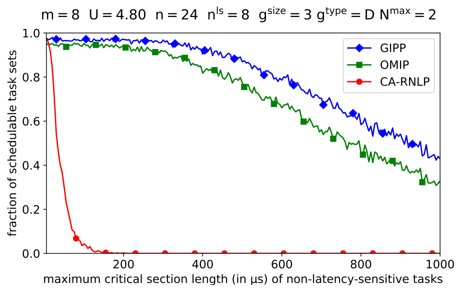

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 2 N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

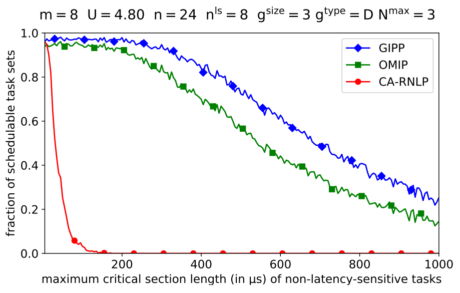

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = W N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

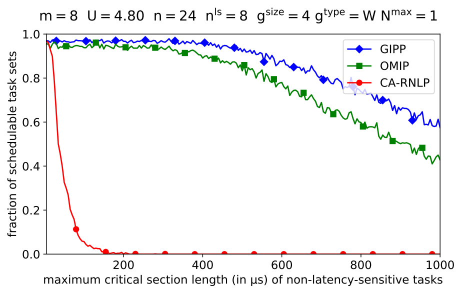

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = W N^{max} = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

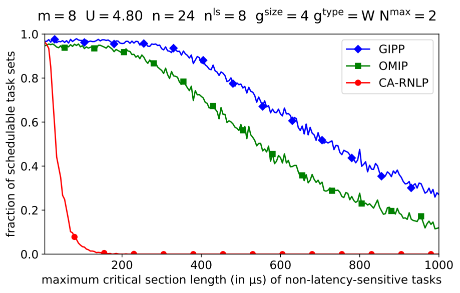

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = W N^{max} = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

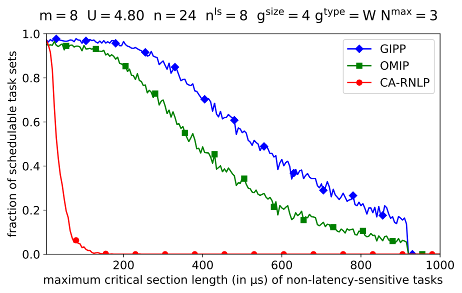

m = 8 U = 4.80 n = 24 n^{ls} = 8 g^{size} = 3 g^{type} = D N^{max} = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

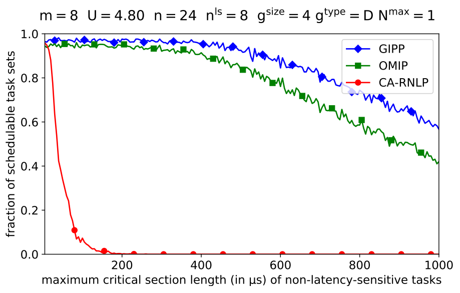

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

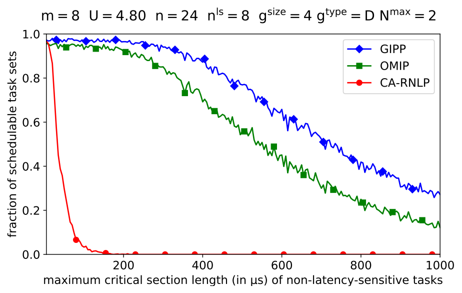

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

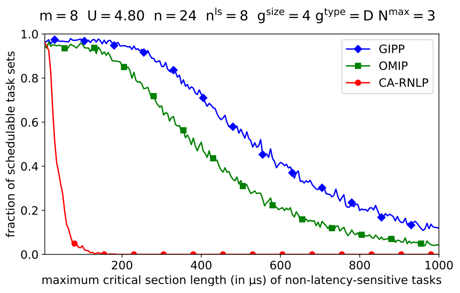

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

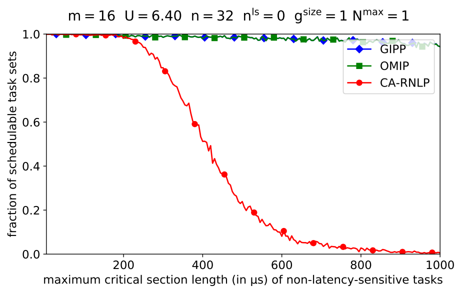

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

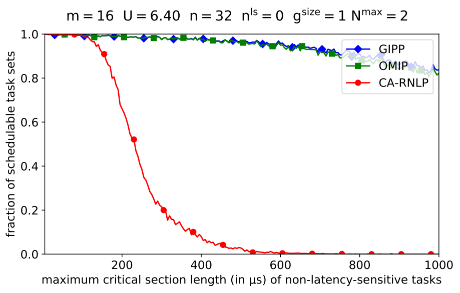

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

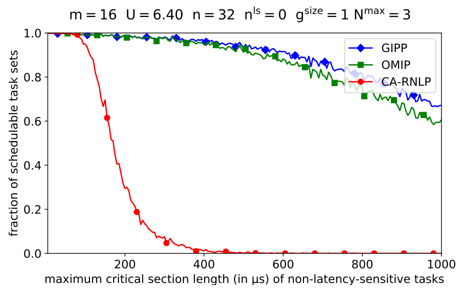

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

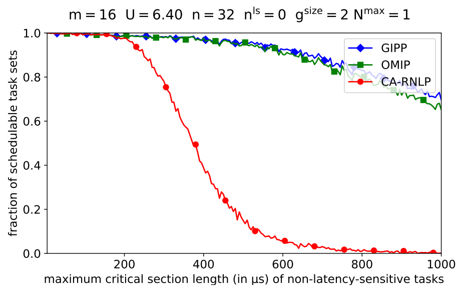

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

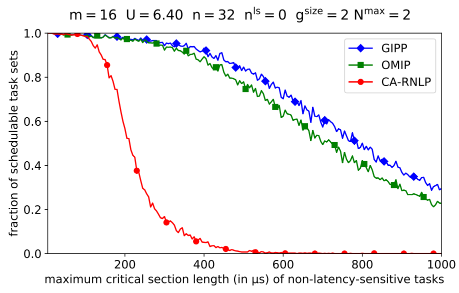

$m = 8$ $U = 4.80$ $n = 24$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

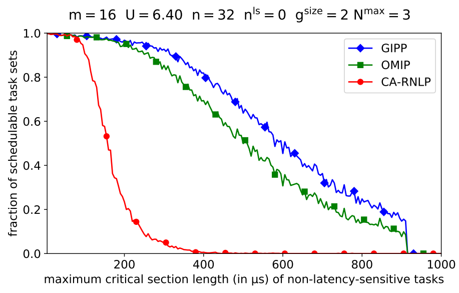

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

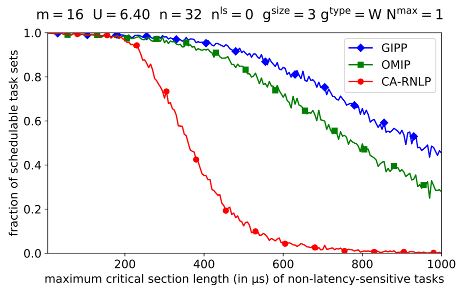

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

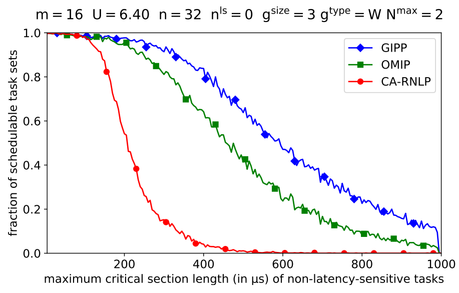

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

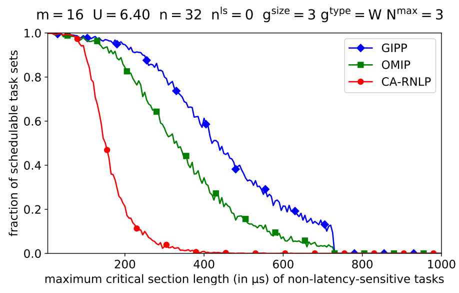

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

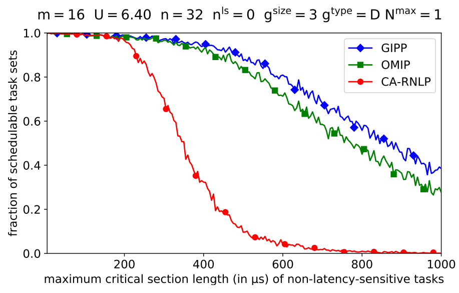

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

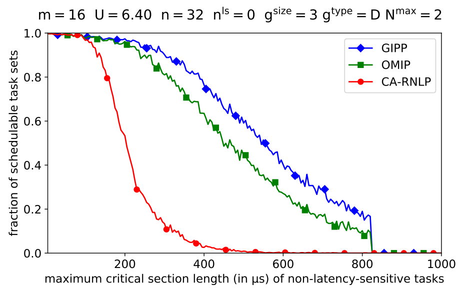

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

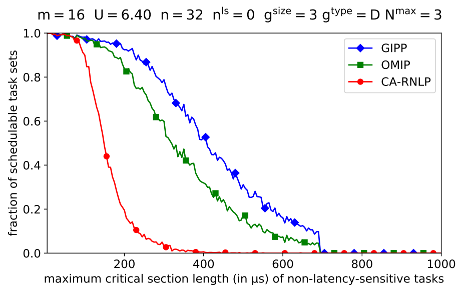

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

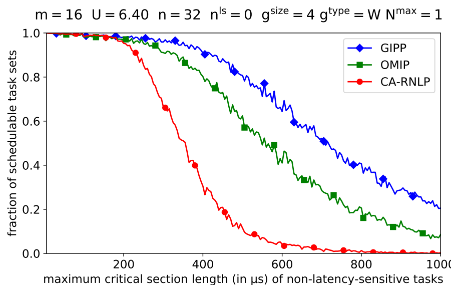

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

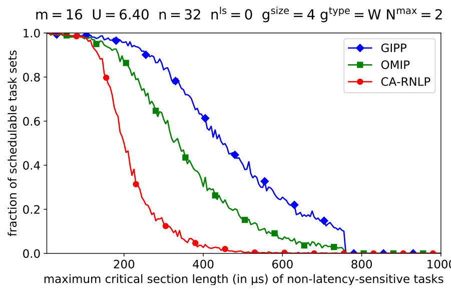

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

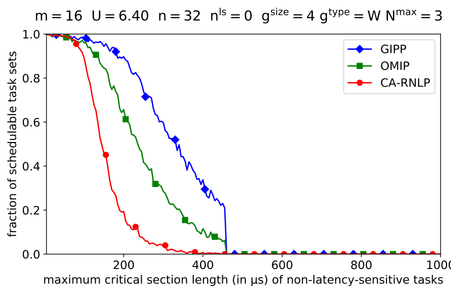

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

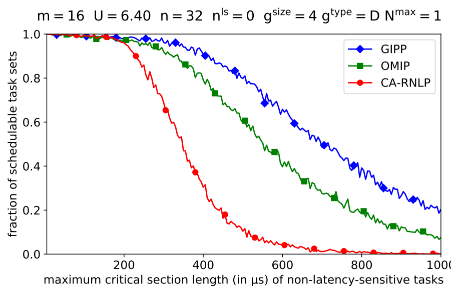

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

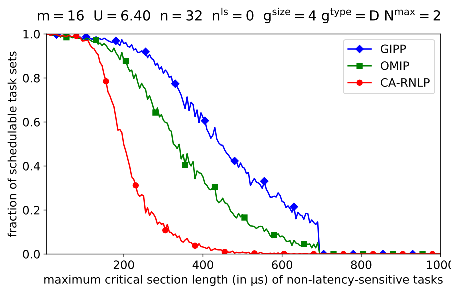

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in μs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

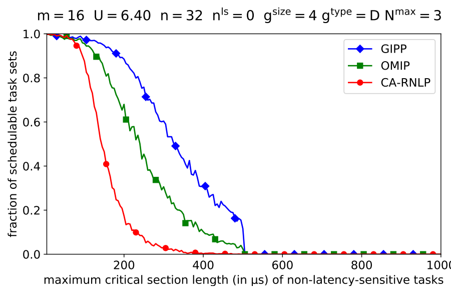

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

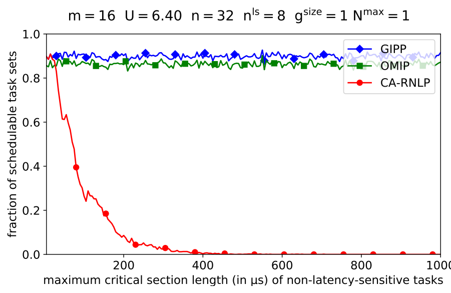

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

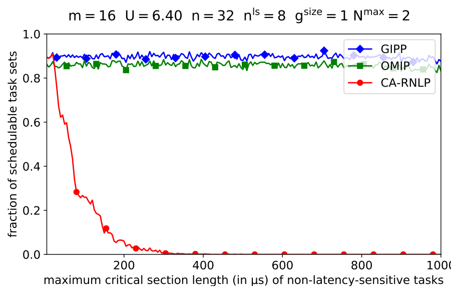

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

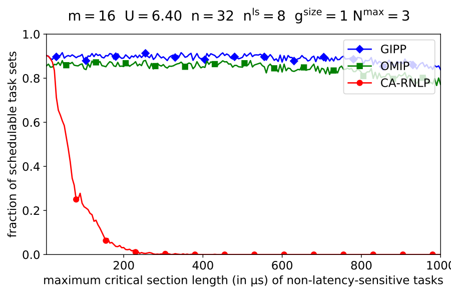

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

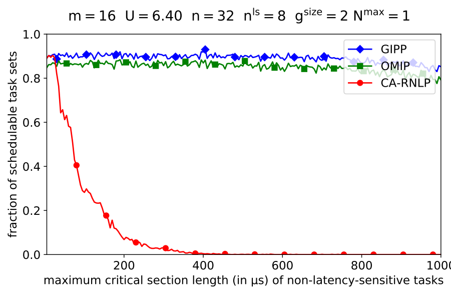

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

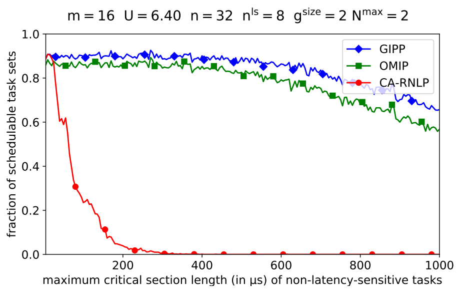

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

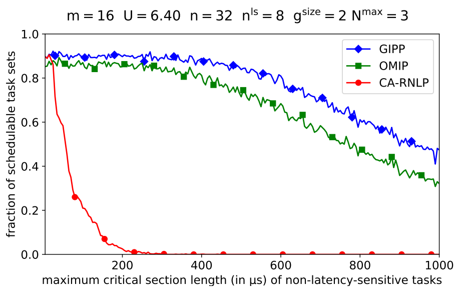

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 1$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

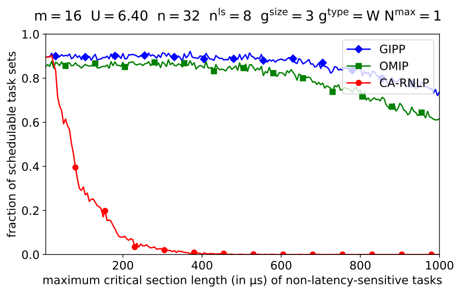

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

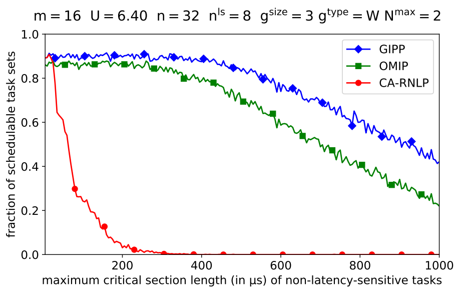

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

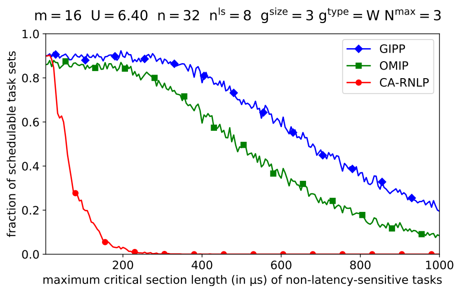

$m = 16$ $U = 6.40$ $n = 32$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
GIPP
OMIP
CA-RNLP
maximum critical section length (in µs) of non-latency-sensitive tasks

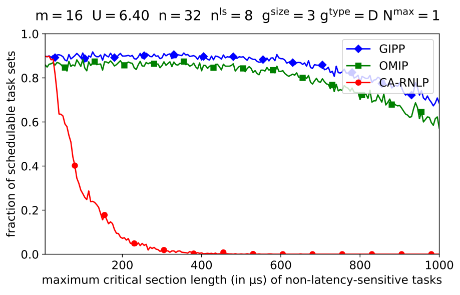

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

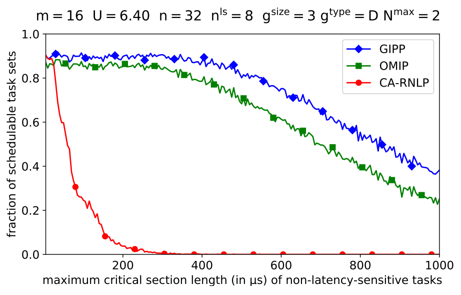

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

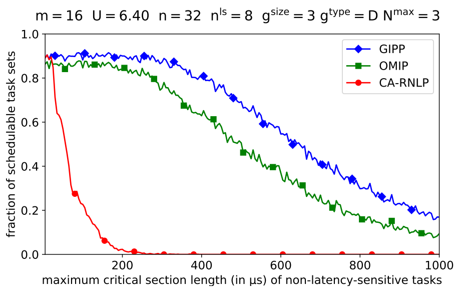

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

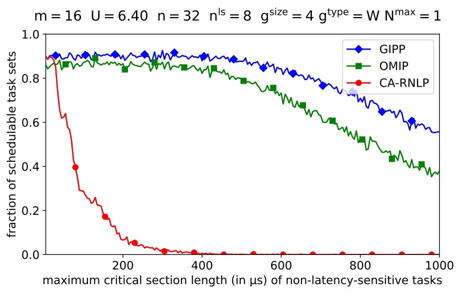

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

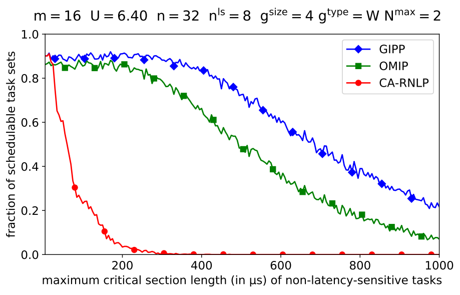

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

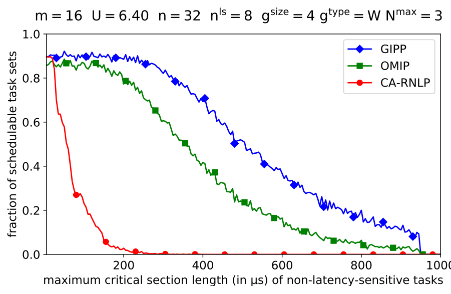

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

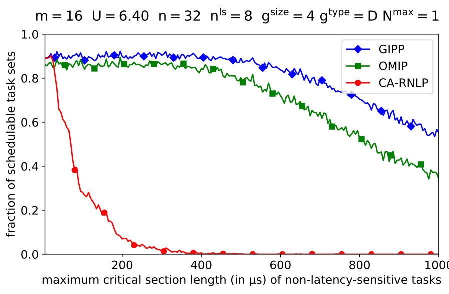

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

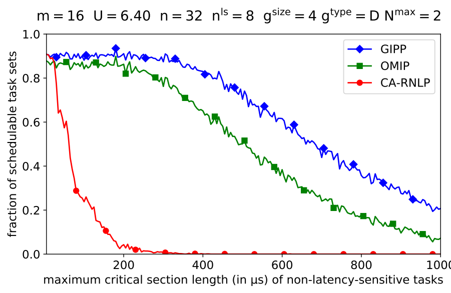

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

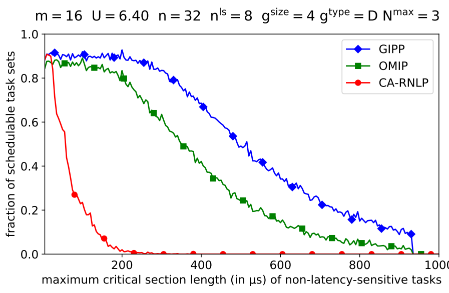

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

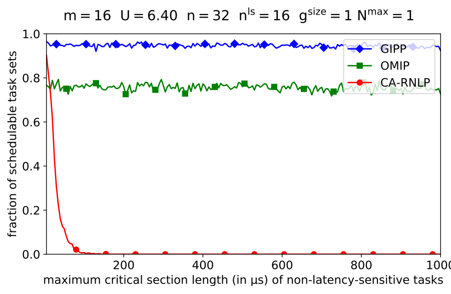

m = 16 U = 6.40 n = 32 $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

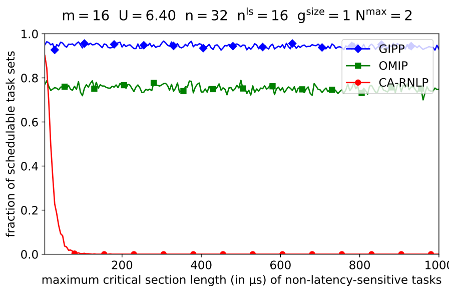

m = 16 U = 6.40 n = 32 n^ls = 8 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

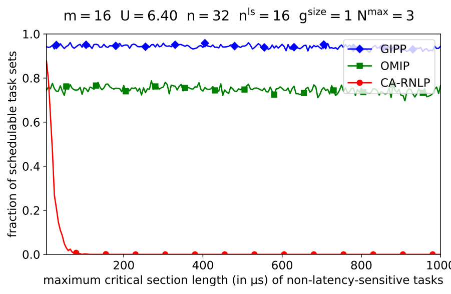

m = 16 U = 6.40 n = 32 n^ls = 8 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

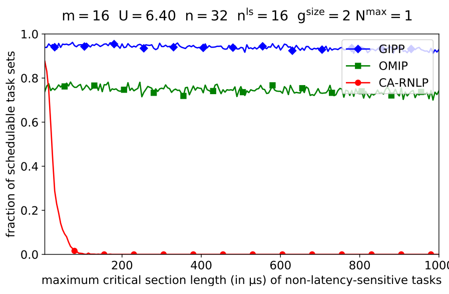

m = 16 U = 6.40 n = 32 n^ls = 8 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

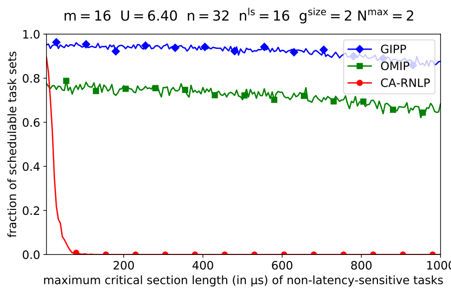

m = 16 U = 6.40 n = 32 n^ls = 8 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

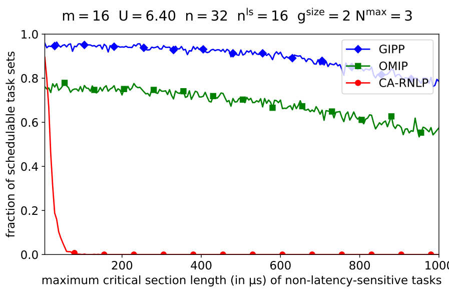

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

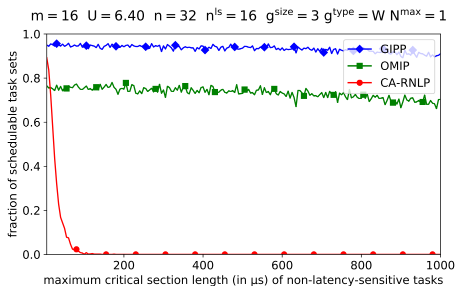

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

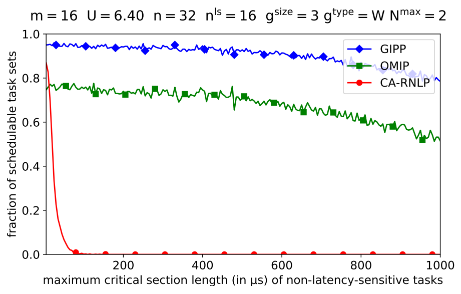

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

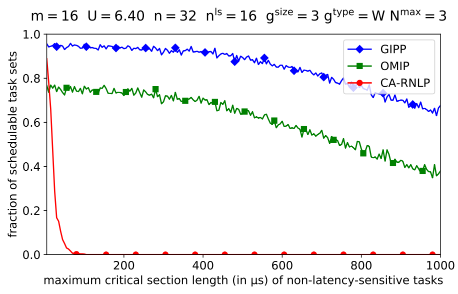

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

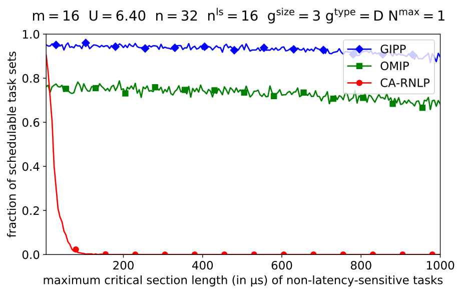

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

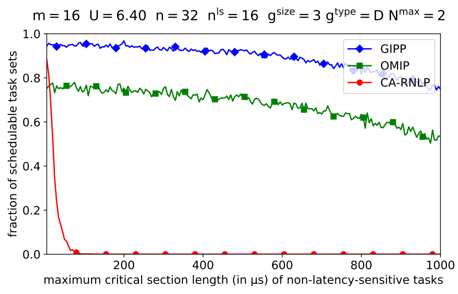

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

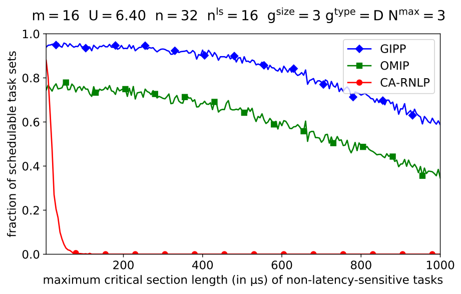

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = W \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

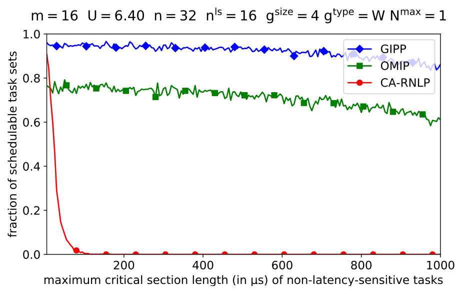

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = W \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

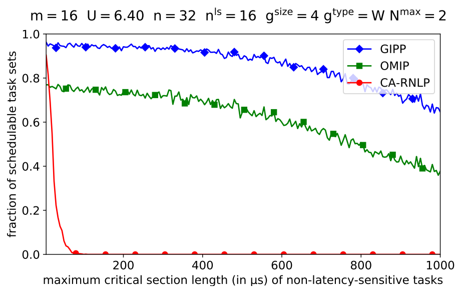

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = W \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

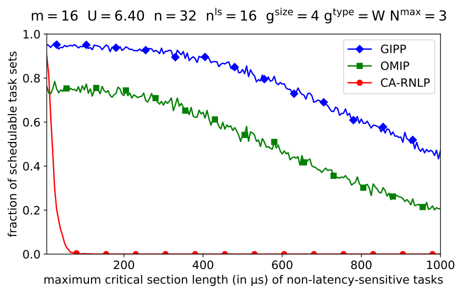

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = D \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

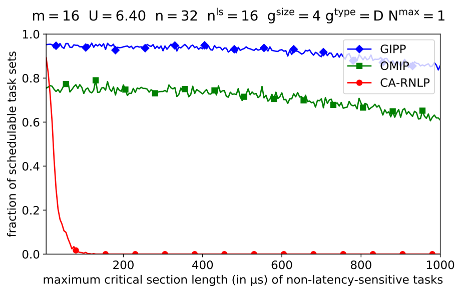

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = D \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

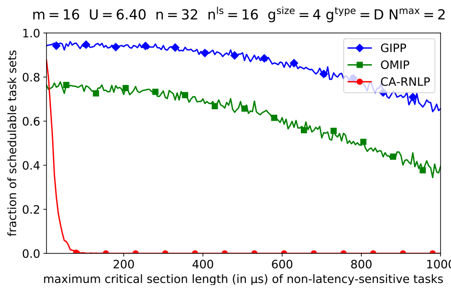

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 3 \;\; g^{type} = D \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

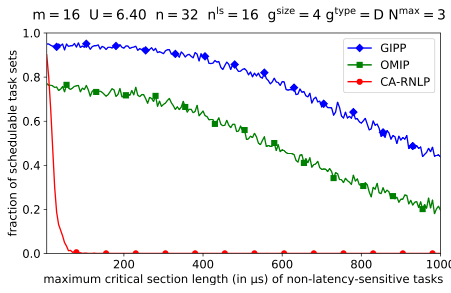

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 4 \;\; g^{type} = W \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

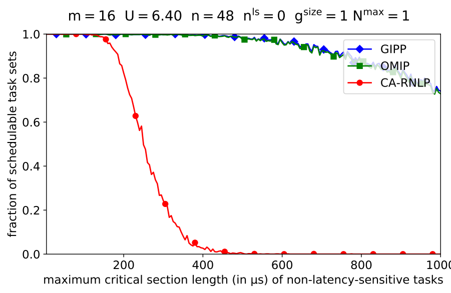

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 4 \;\; g^{type} = W \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

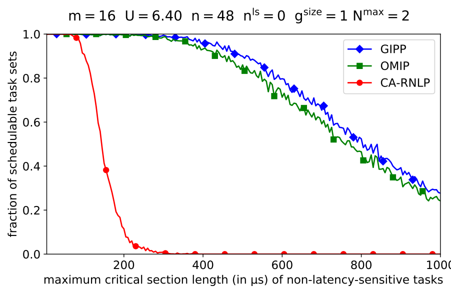

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 4 \;\; g^{type} = W \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

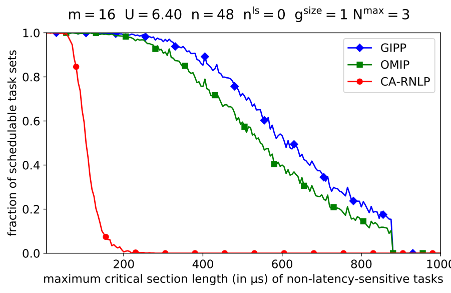

$m = 16 \;\; U = 6.40 \;\; n = 32 \;\; n^{ls} = 16 \;\; g^{size} = 4 \;\; g^{type} = D \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

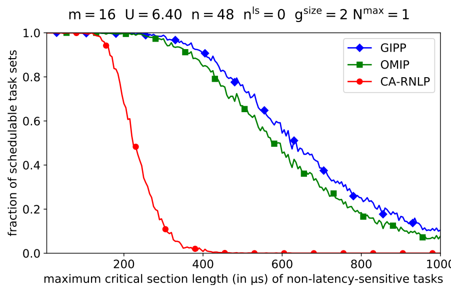

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

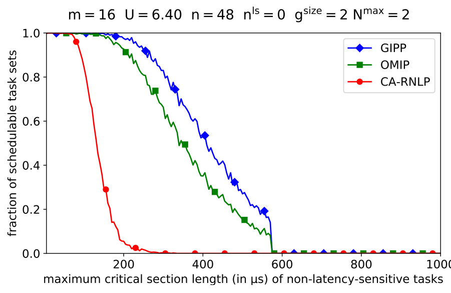

m = 16 U = 6.40 n = 32 n^ls = 16 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

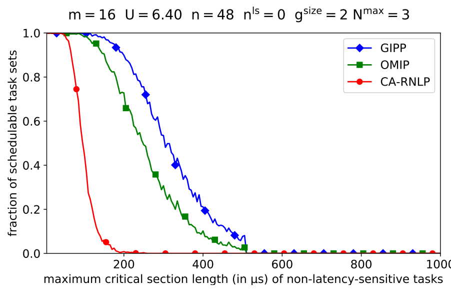

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 1 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

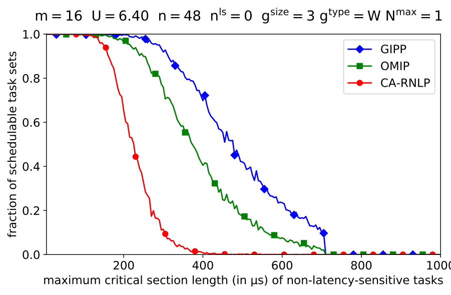

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 1 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

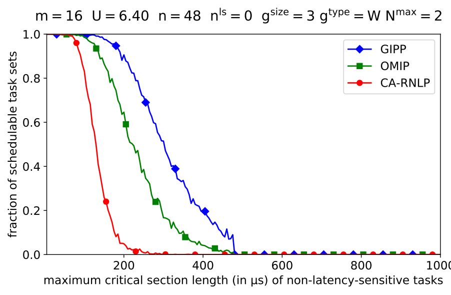

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 1 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

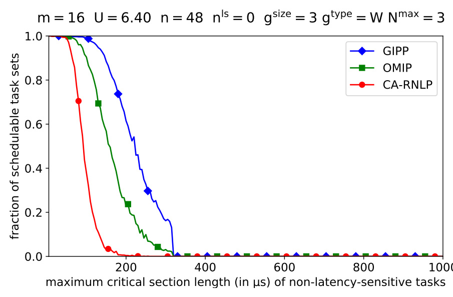

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 2 N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

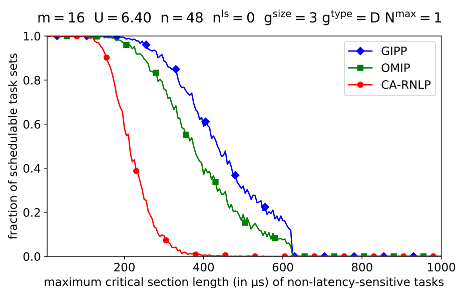

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 2 N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

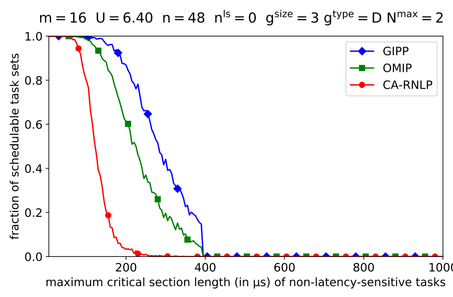

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 2 N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

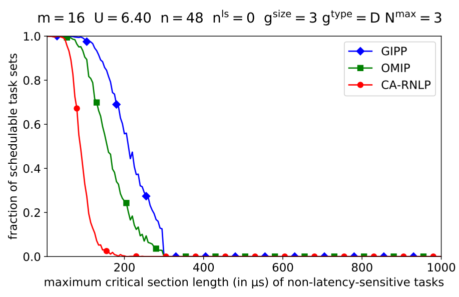

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 3 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

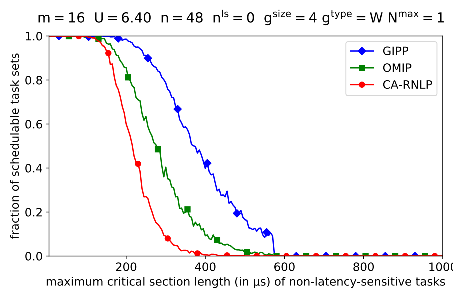

m = 16 U = 6.40 n = 48 n^ls = 0 g^size = 3 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

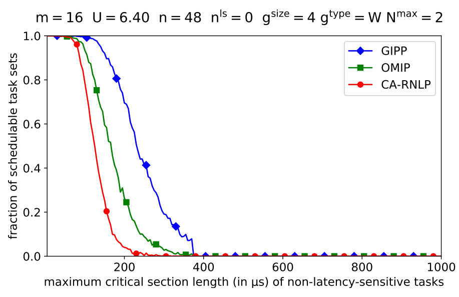

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

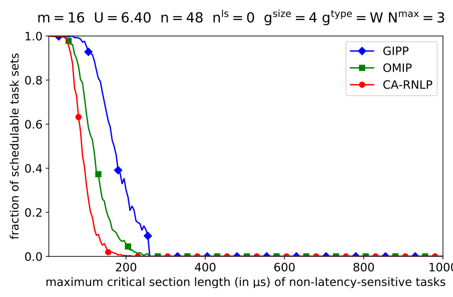

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

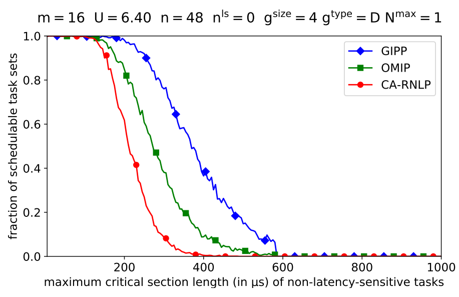

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

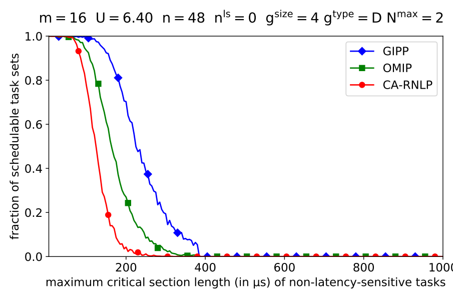

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

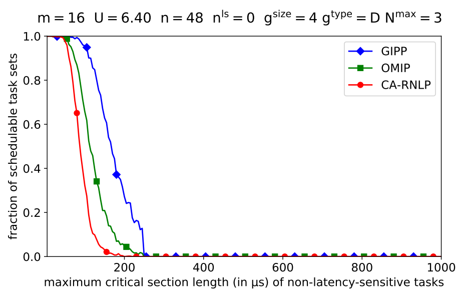

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

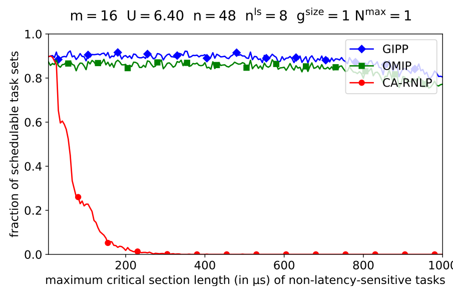

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

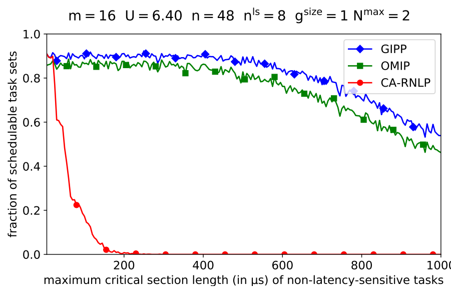

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

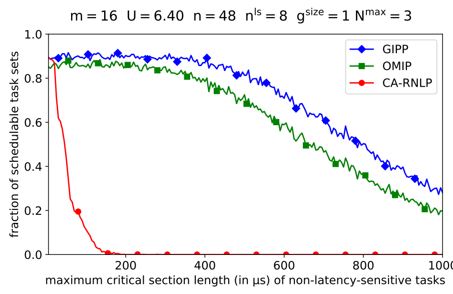

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

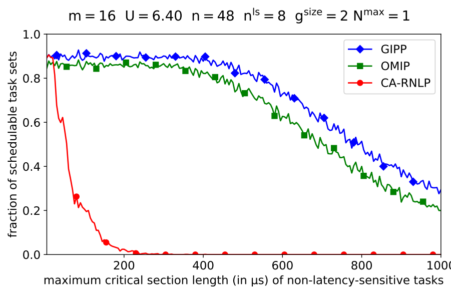

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

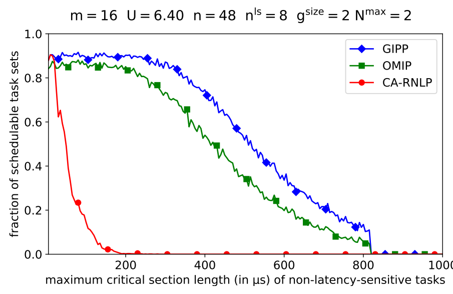

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

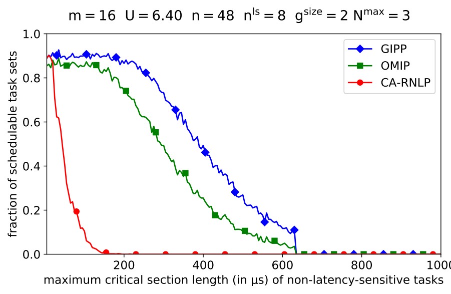

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 1 \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

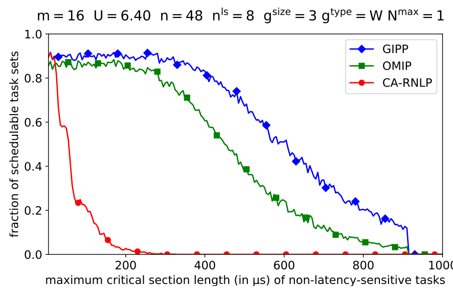

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 1 \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

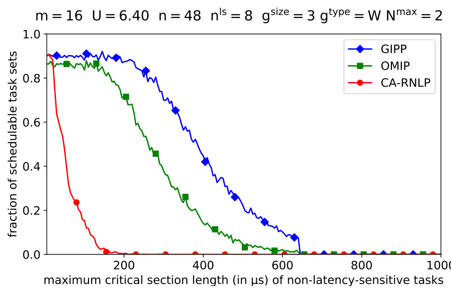

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 1 \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

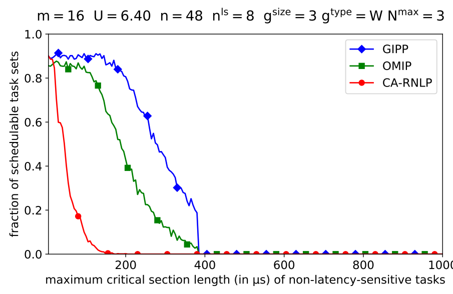

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 2 \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

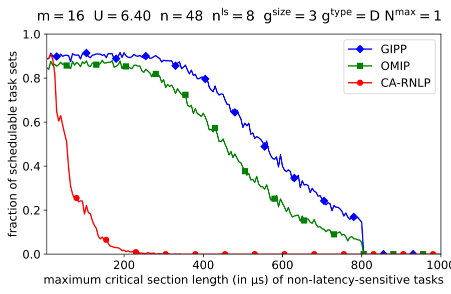

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 2 \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

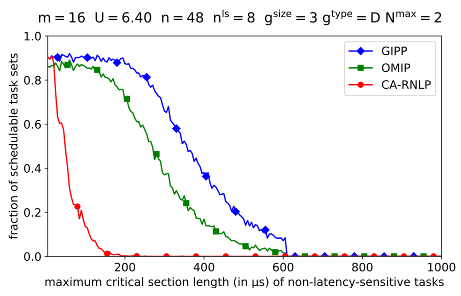

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 2 \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

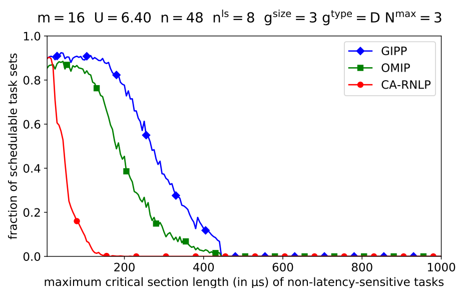

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 3 \; g^{type} = W \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

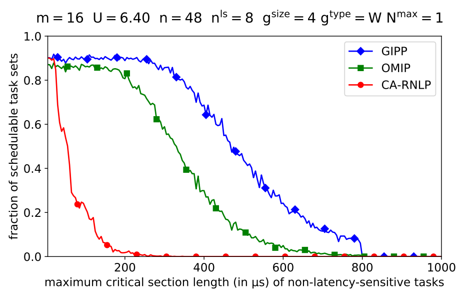

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 3 \; g^{type} = W \; N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

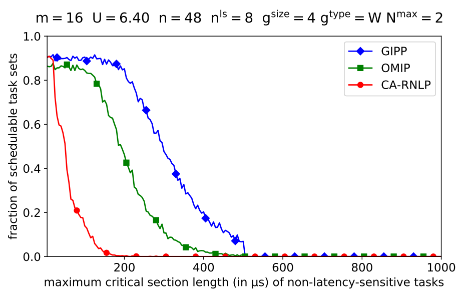

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 3 \; g^{type} = W \; N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

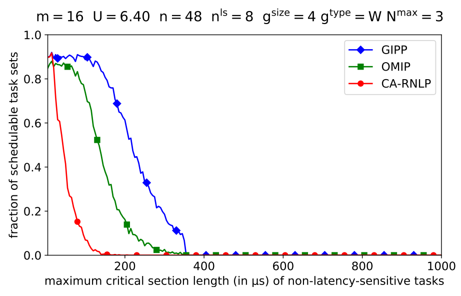

$m = 16 \quad U = 6.40 \quad n = 48 \quad n^{ls} = 8 \quad g^{size} = 3 \; g^{type} = D \; N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

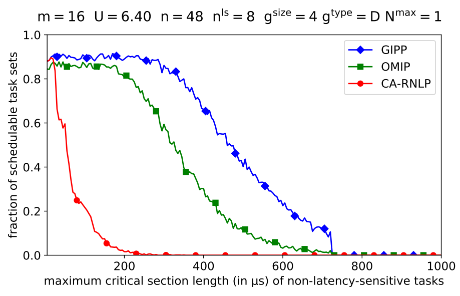

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

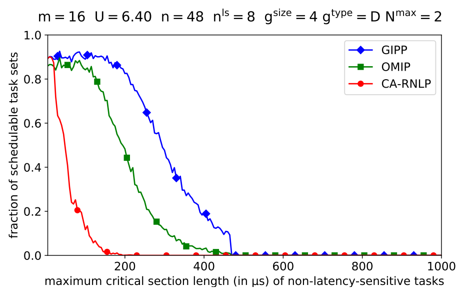

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

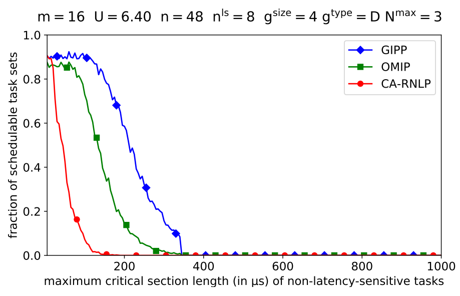

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

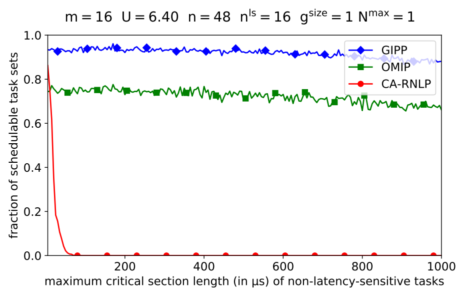

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

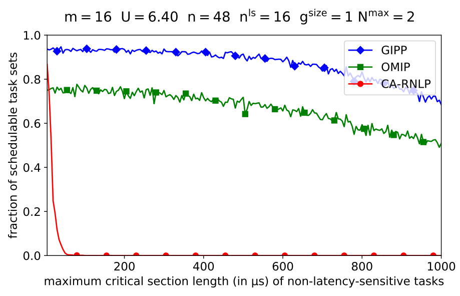

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

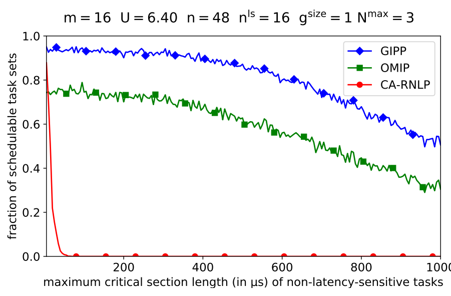

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

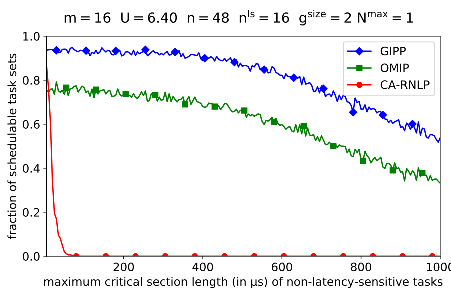

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

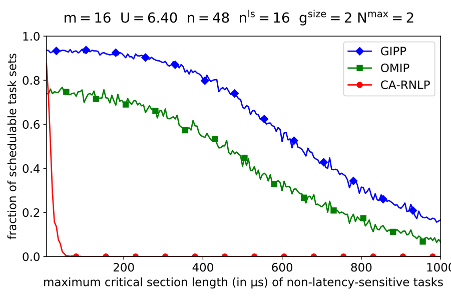

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

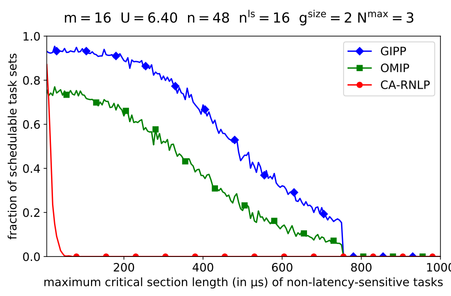

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 1$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

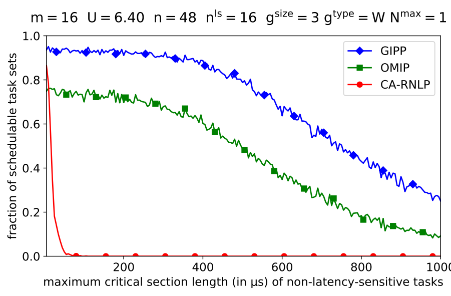

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

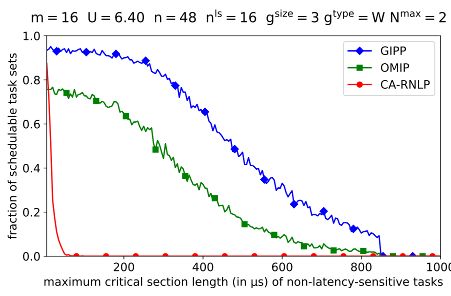

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 1 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

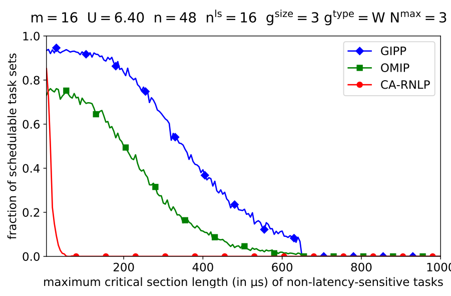

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 2 N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

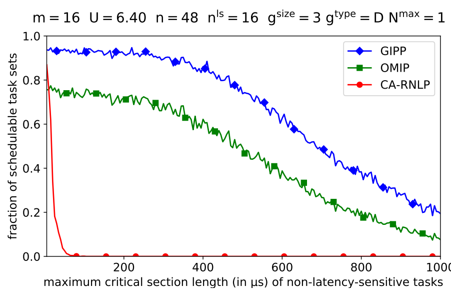

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 2 N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

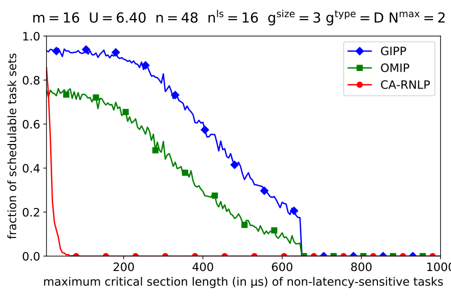

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 2 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

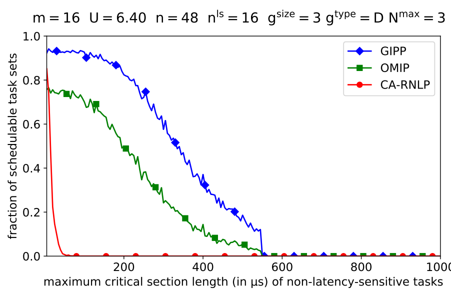

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = W N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

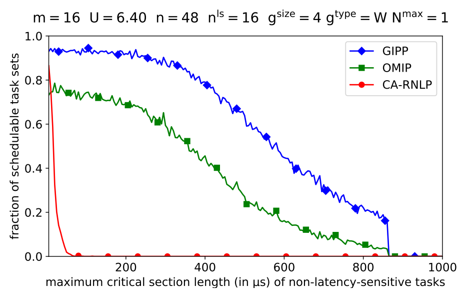

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = W N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

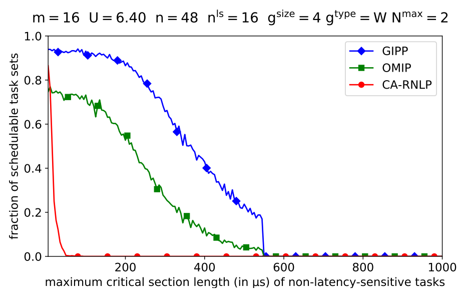

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = W N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

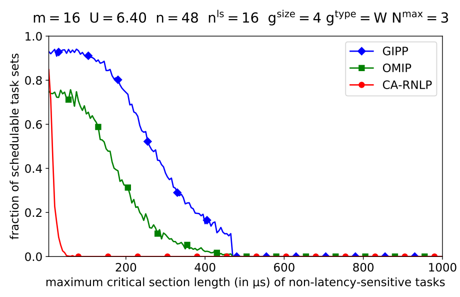

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = D N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

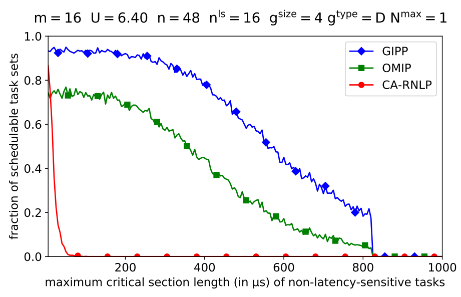

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = D N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

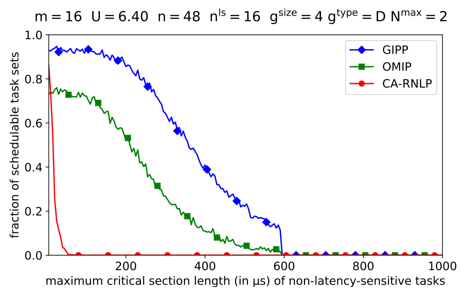

m = 16 U = 6.40 n = 48 n^ls = 16 g^size = 3 g^type = D N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

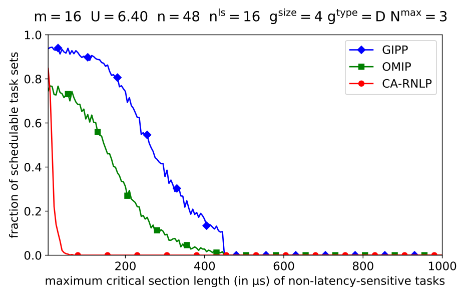

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

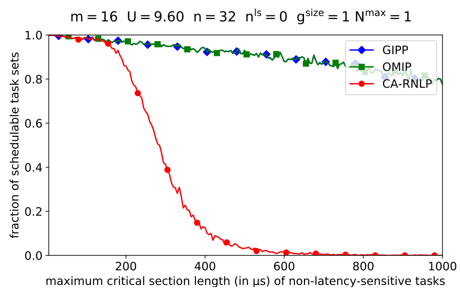

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

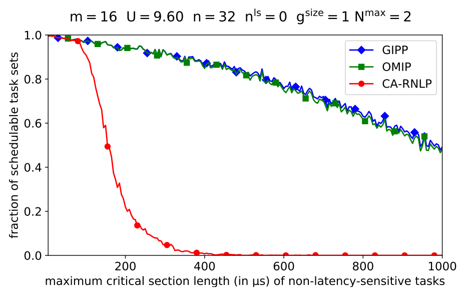

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

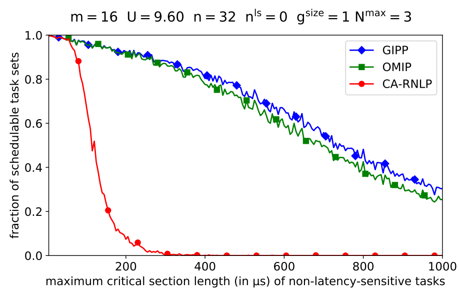

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

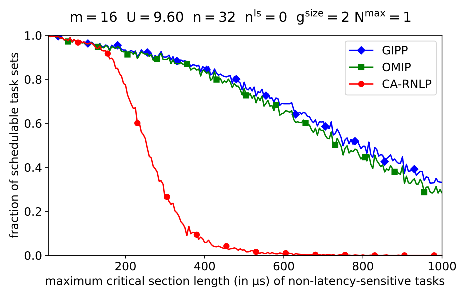

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

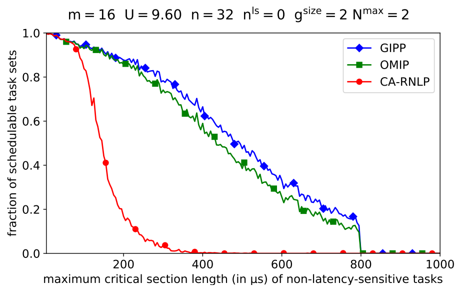

$m = 16$ $U = 6.40$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

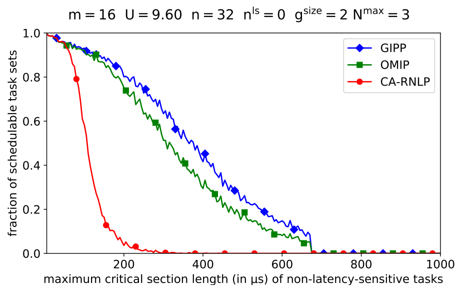

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

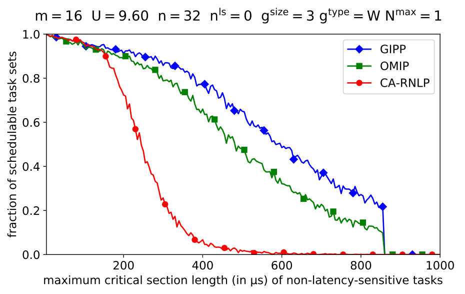

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

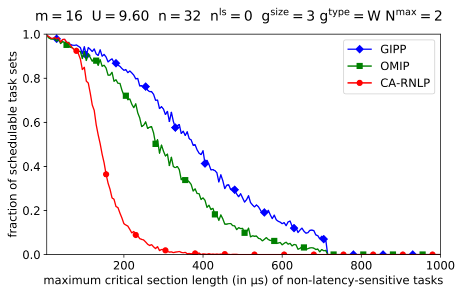

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

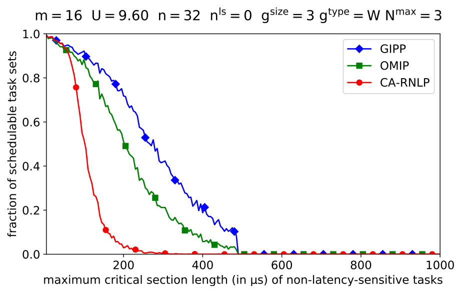

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

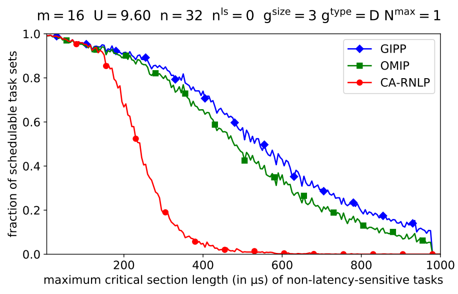

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 2 N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

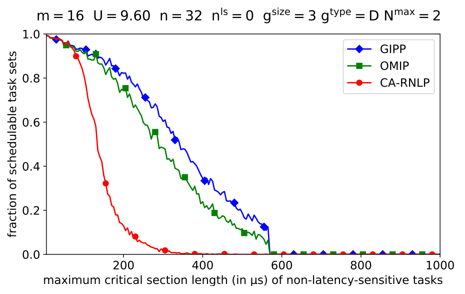

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 2 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

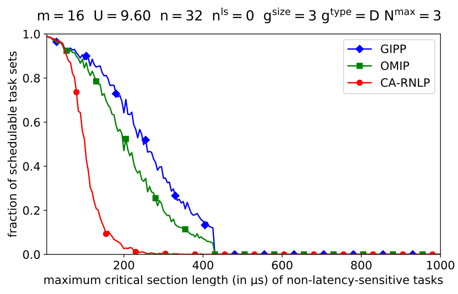

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = W N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

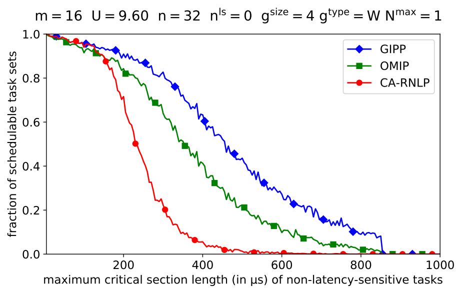

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = W N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

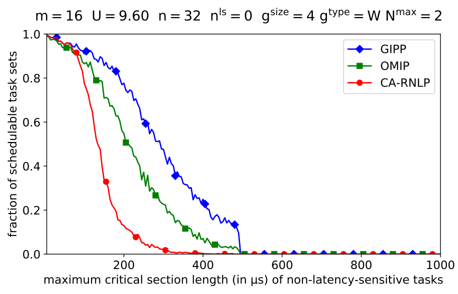

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = W N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

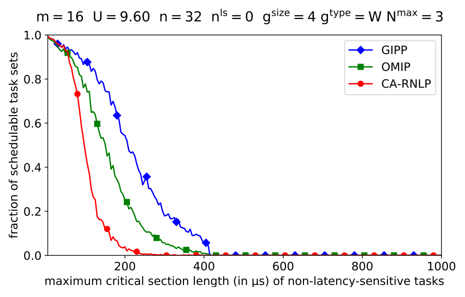

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = D N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

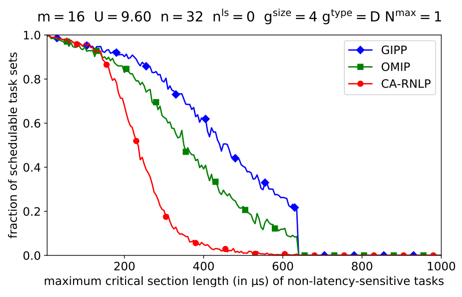

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = D N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

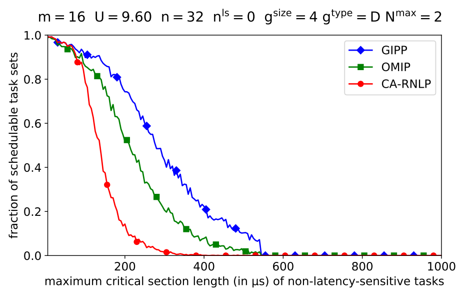

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 3 g^type = D N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

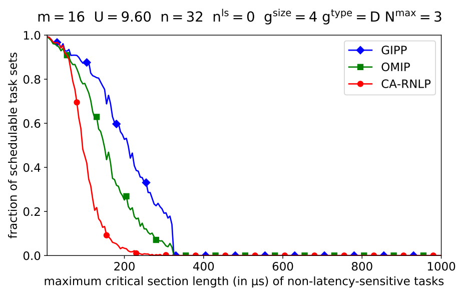

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 4 g^type = W N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

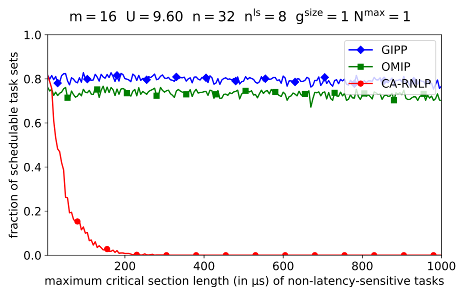

m = 16 U = 9.60 n = 32 n^ls = 0 g^size = 4 g^type = W N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

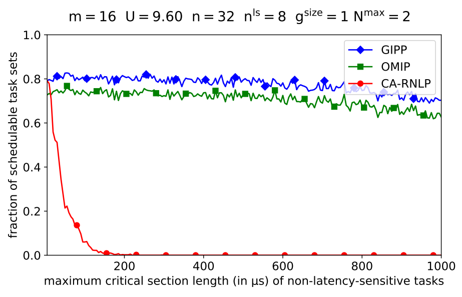

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

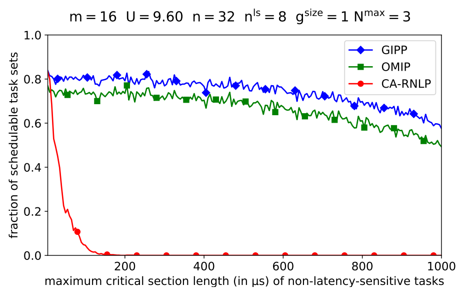

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

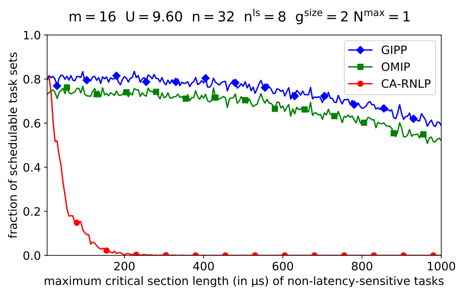

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

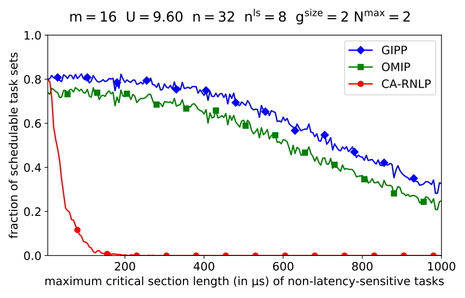

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

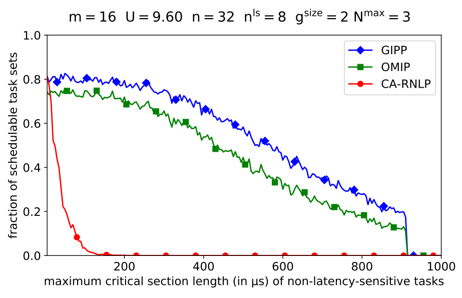

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

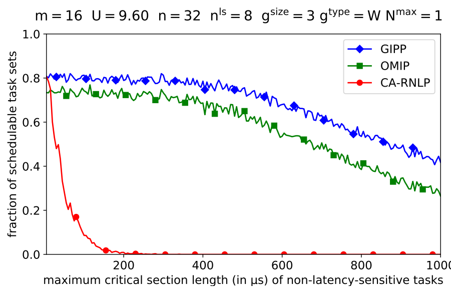

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

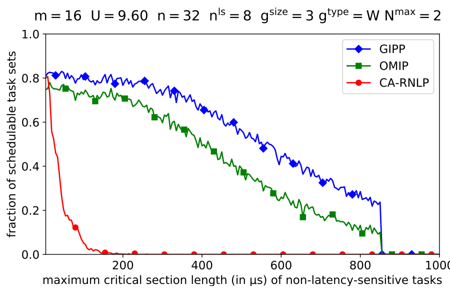

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

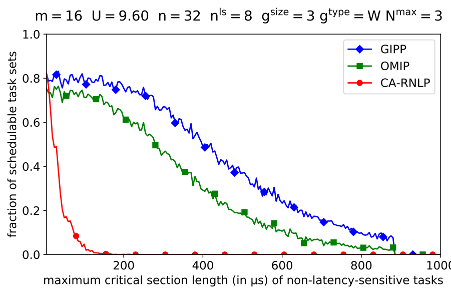

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

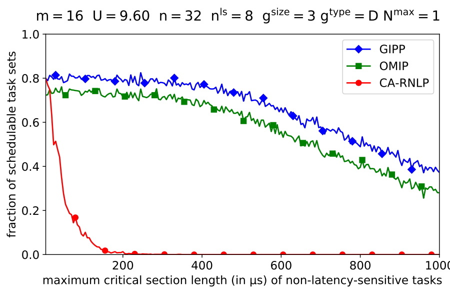

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

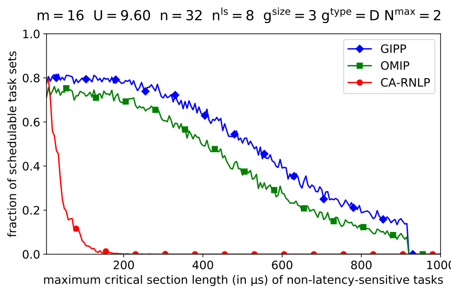

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

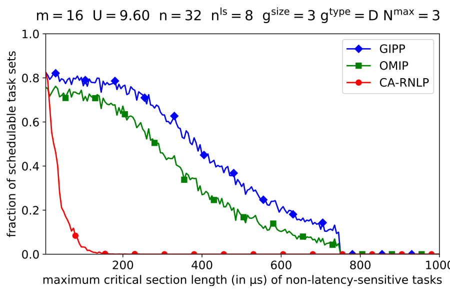

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

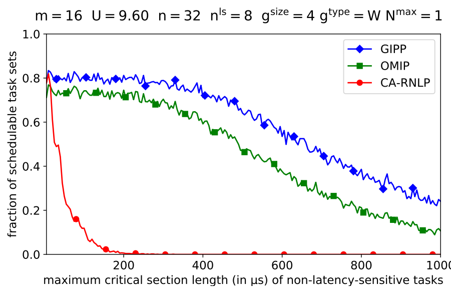

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

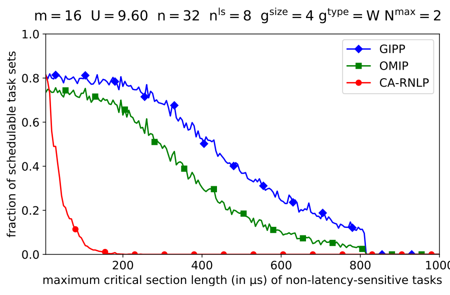

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

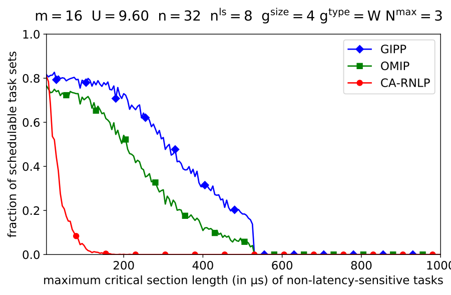

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

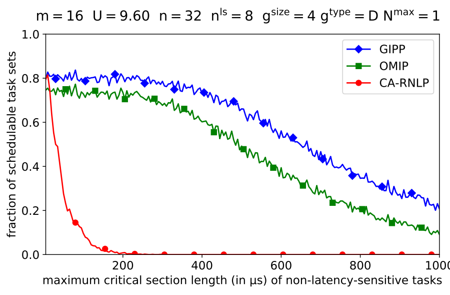

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

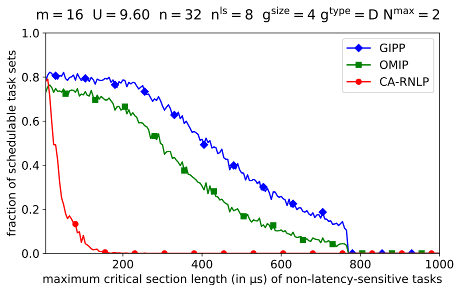

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

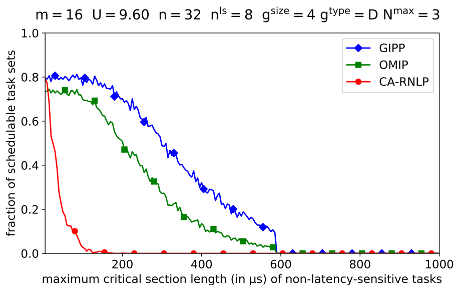

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

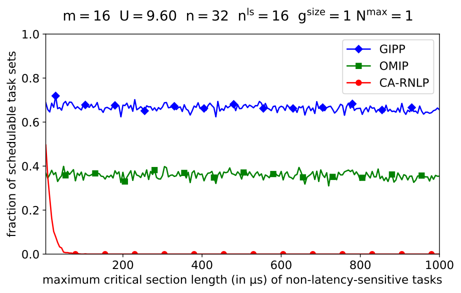

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

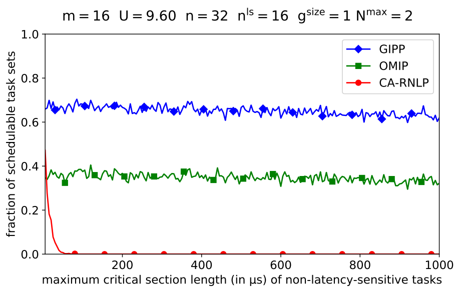

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

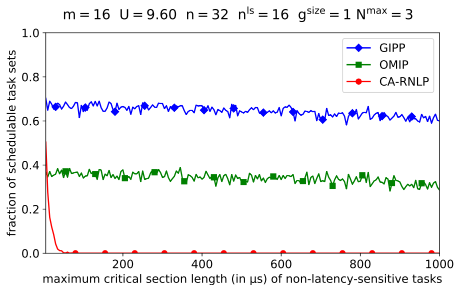

$m = 16$ $U = 9.60$ $n = 32$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

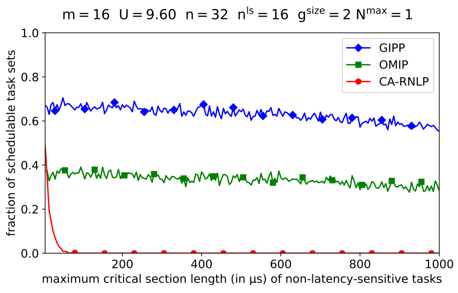

m = 16 U = 9.60 n = 32 n^ls = 8 g^size = 4 g^type = D N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

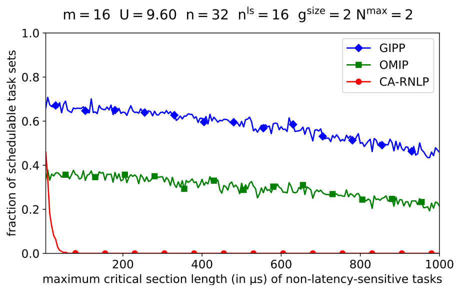

m = 16 U = 9.60 n = 32 n^ls = 8 g^size = 4 g^type = D N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

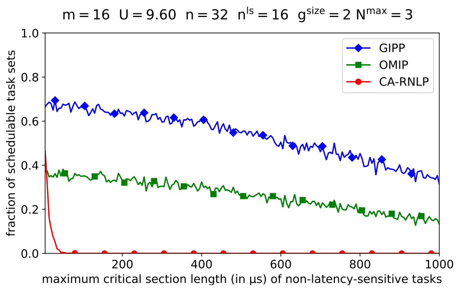

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 1 N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

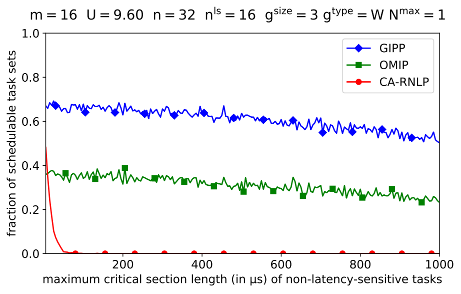

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 1 N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

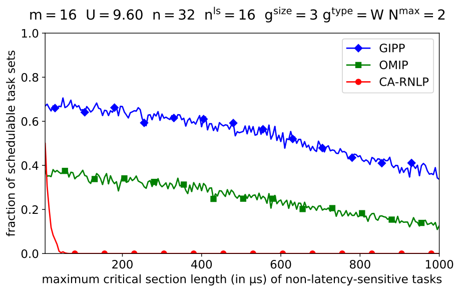

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 1 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

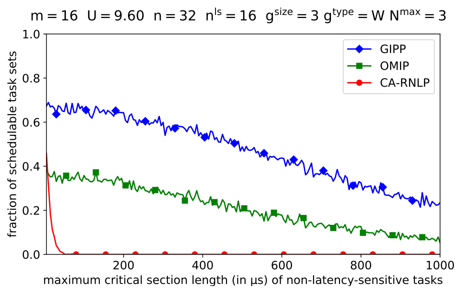

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 2 N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

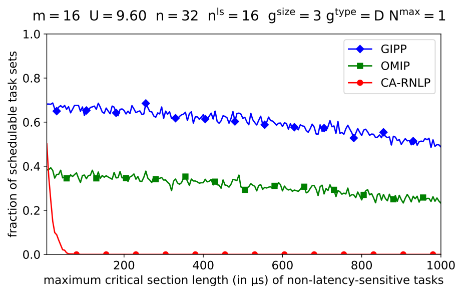

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 2 N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

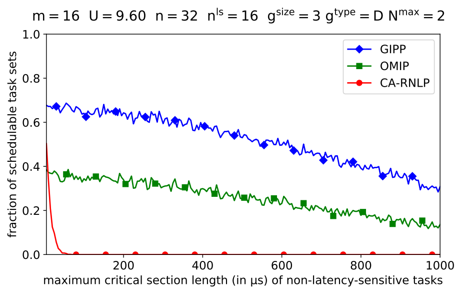

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 2 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

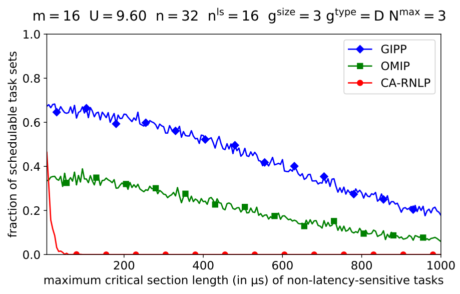

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = W N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

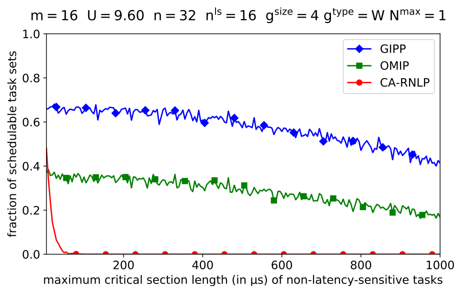

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = W N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

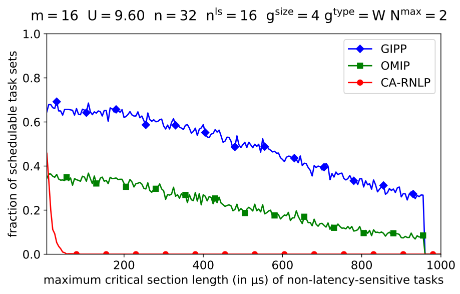

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

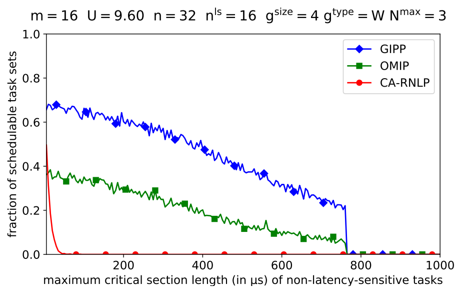

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

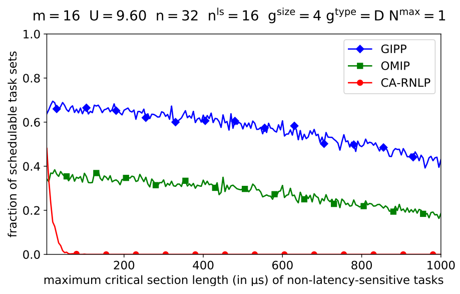

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

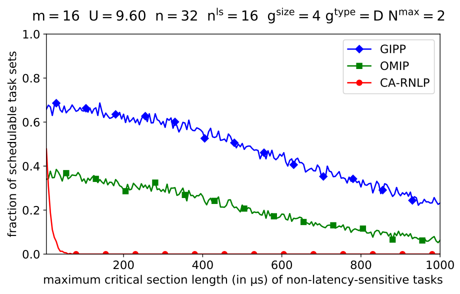

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 3 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

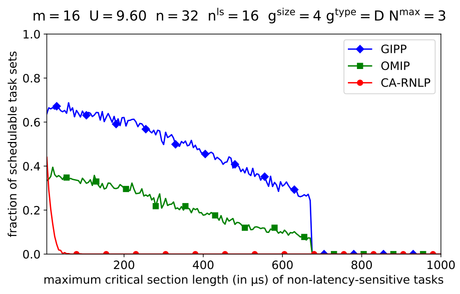

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = W N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

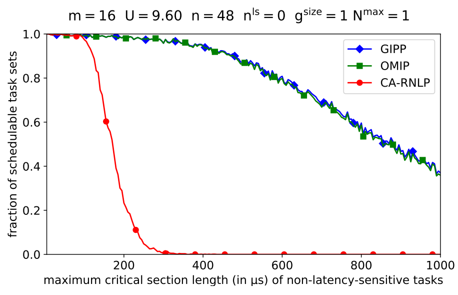

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = W N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

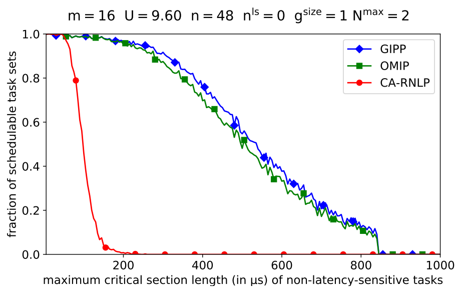

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

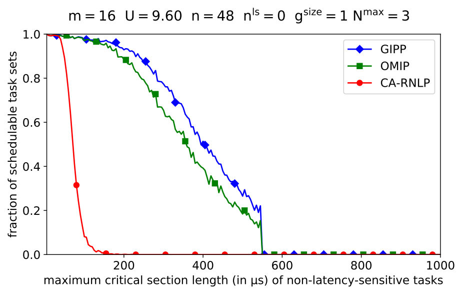

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

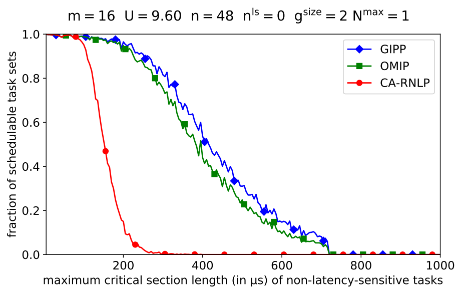

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

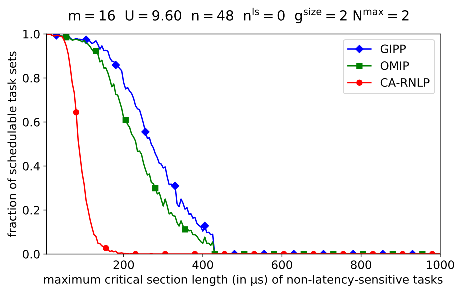

m = 16 U = 9.60 n = 32 n^ls = 16 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

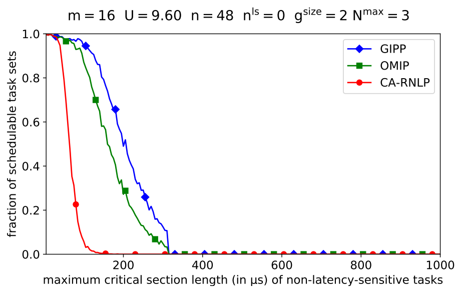

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

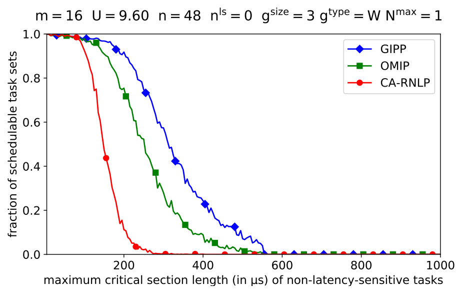

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

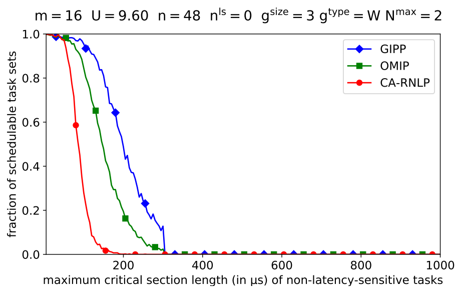

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 1$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

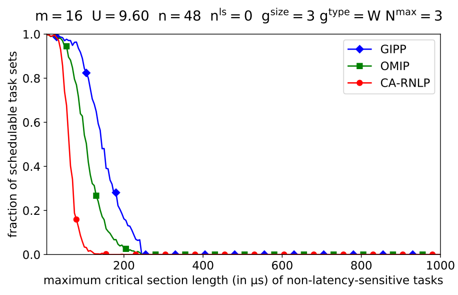

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

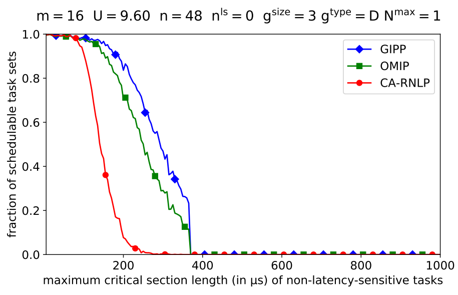

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

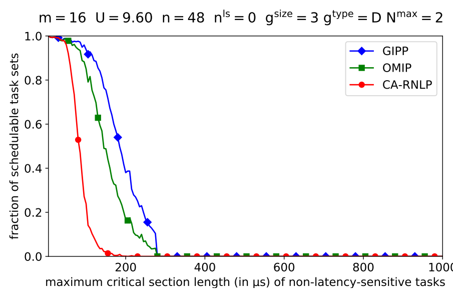

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 2$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

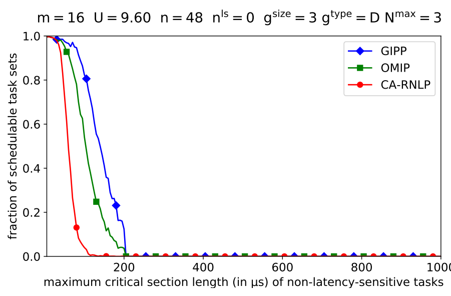

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

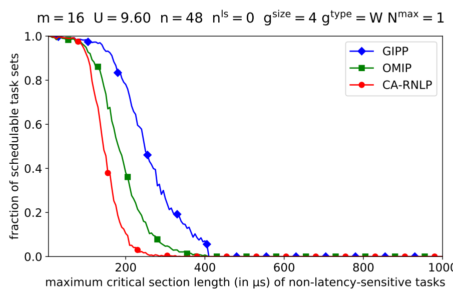

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

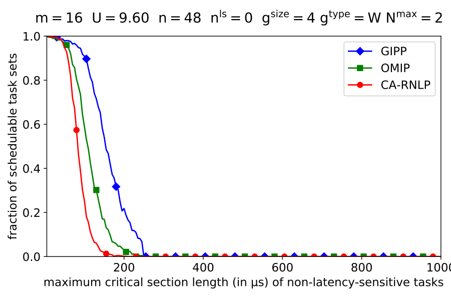

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

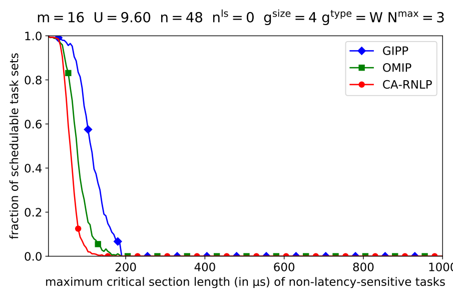

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
0.0 0.2 0.4 0.6 0.8 1.0
200 400 600 800 1000

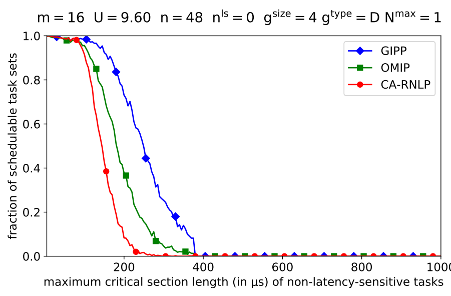

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

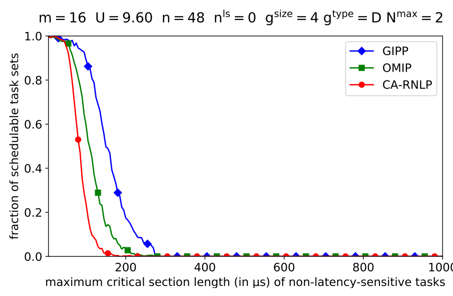

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

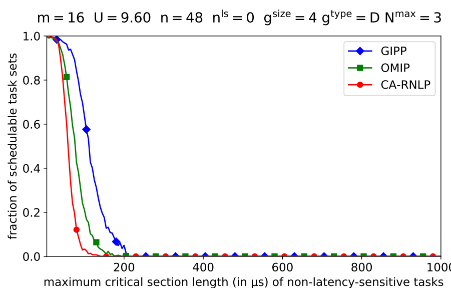

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

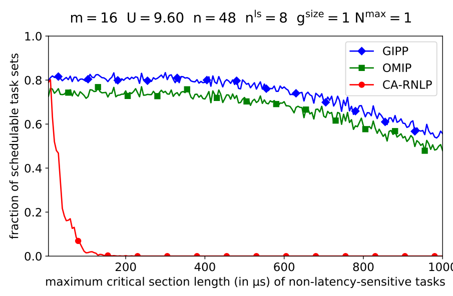

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

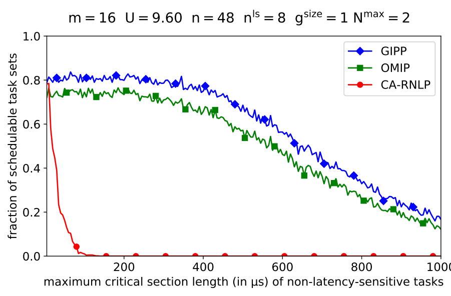

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

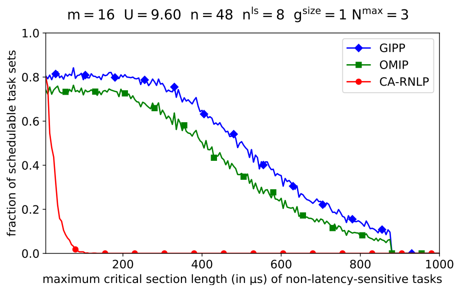

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

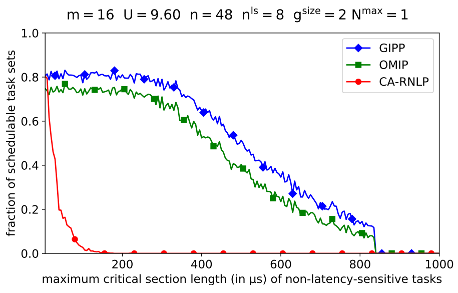

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

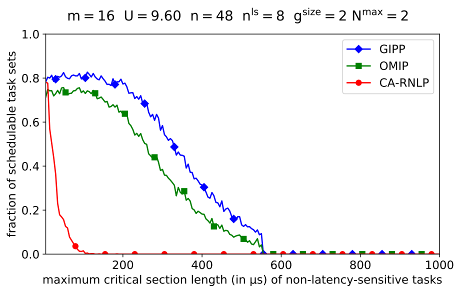

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 0$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

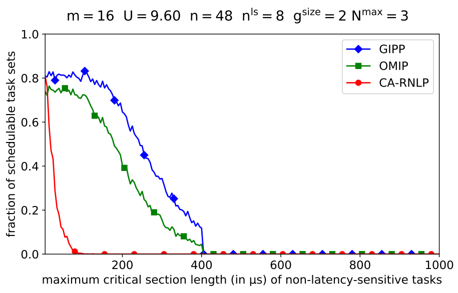

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

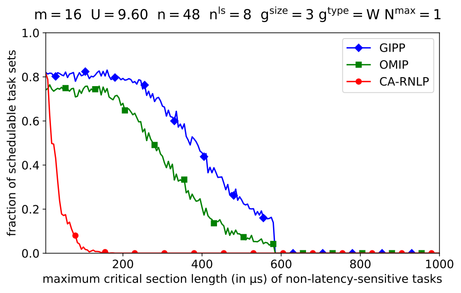

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

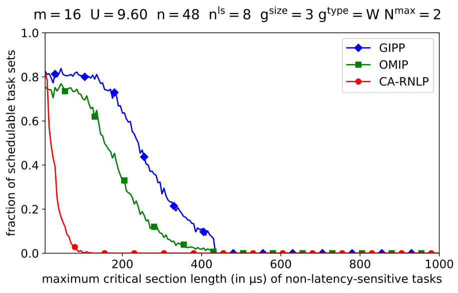

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 1 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

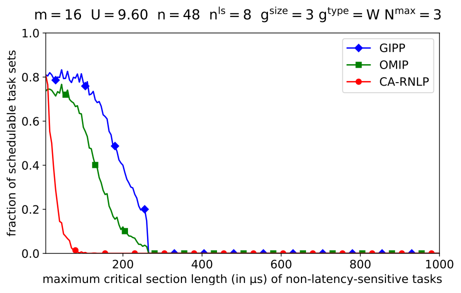

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 2 N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

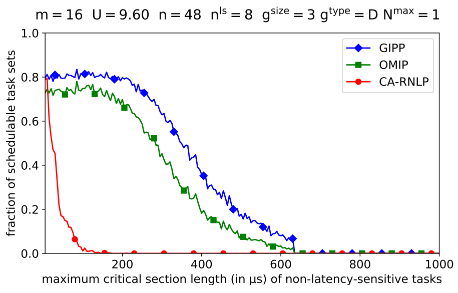

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 2 N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

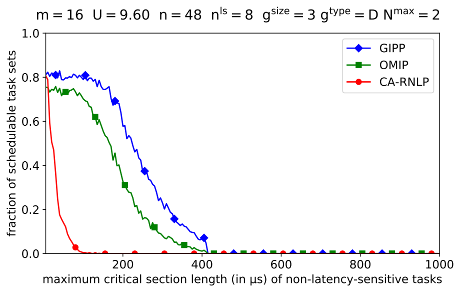

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 2 N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

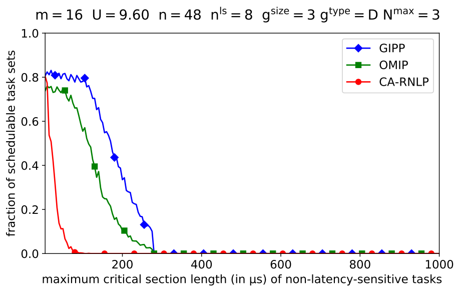

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = W N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

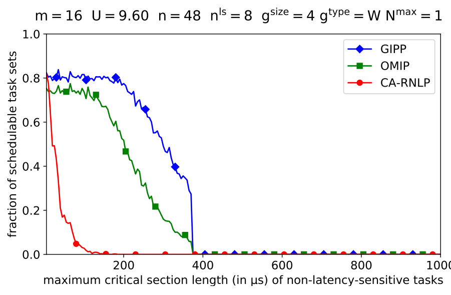

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = W N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

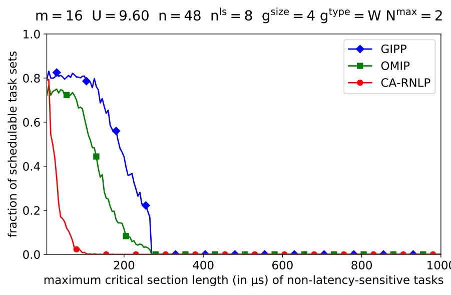

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = W N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

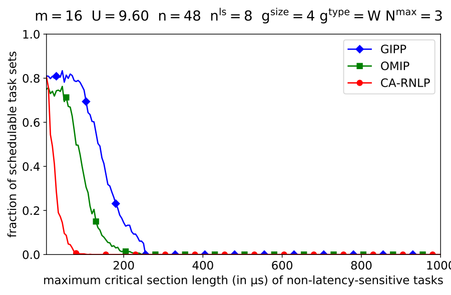

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = D N^max = 1
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

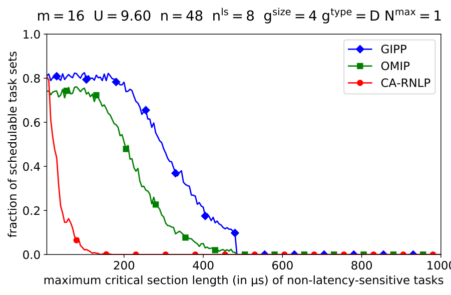

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = D N^max = 2
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

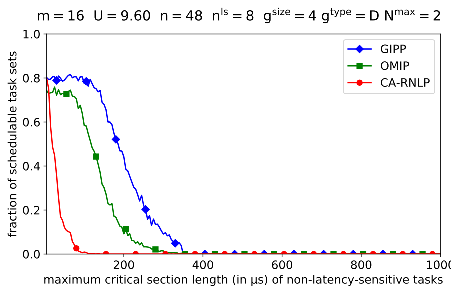

m = 16 U = 9.60 n = 48 n^ls = 8 g^size = 3 g^type = D N^max = 3
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

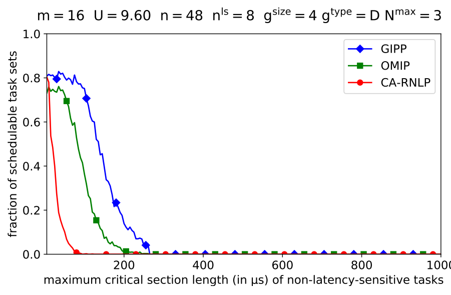

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

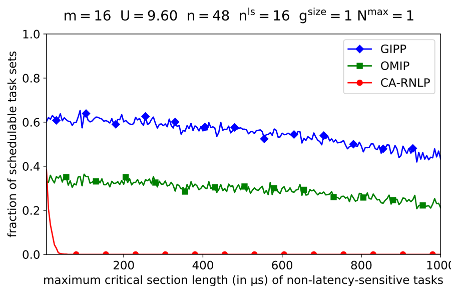

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

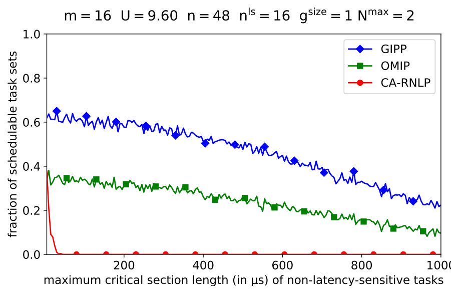

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

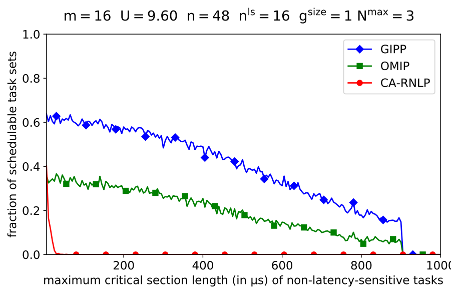

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

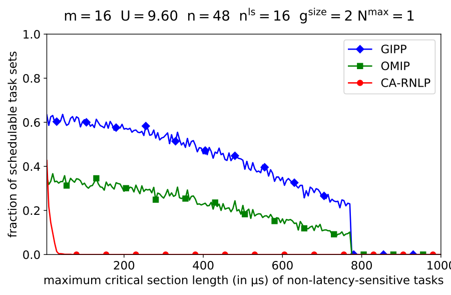

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

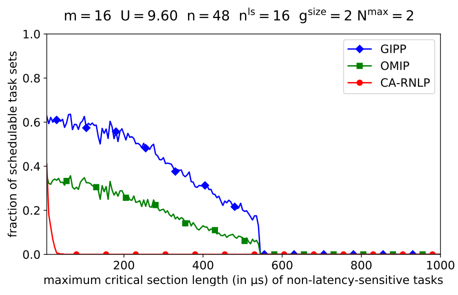

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 8$ $g^{size} = 4$ $g^{type} = D$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

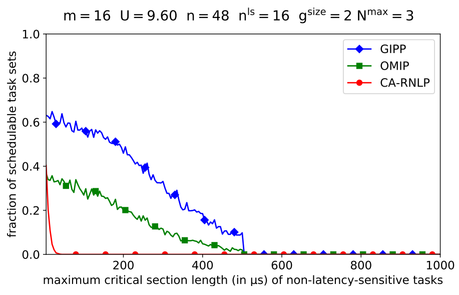

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 1$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

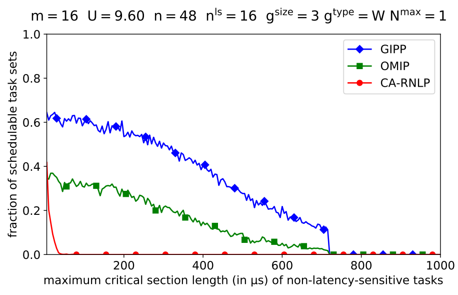

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 1$ $N^{max} = 2$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

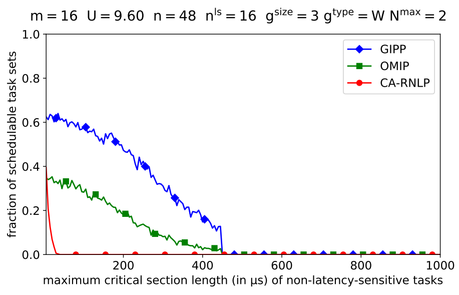

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 1$ $N^{max} = 3$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

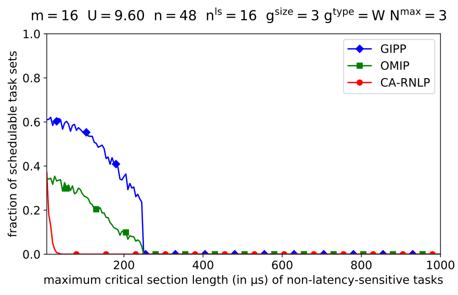

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 2$ $N^{max} = 1$
GIPP
OMIP
CA-RNLP
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks

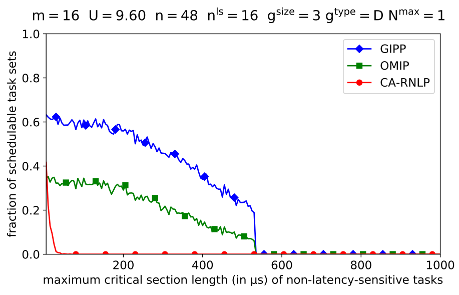

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 2$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

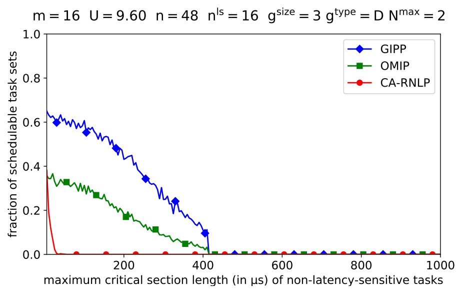

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 2$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

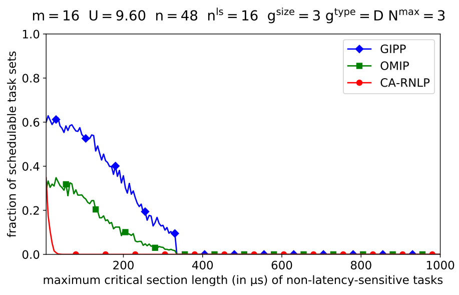

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

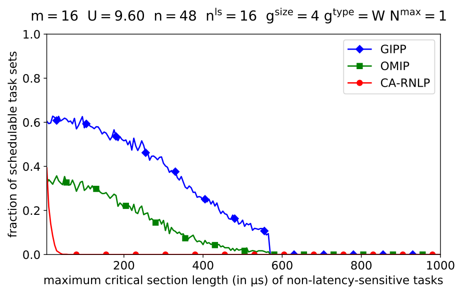

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

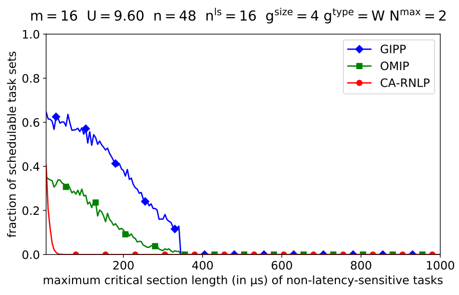

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = W$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

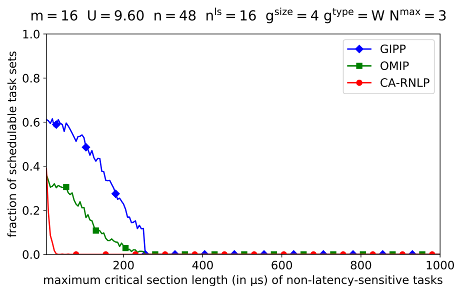

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

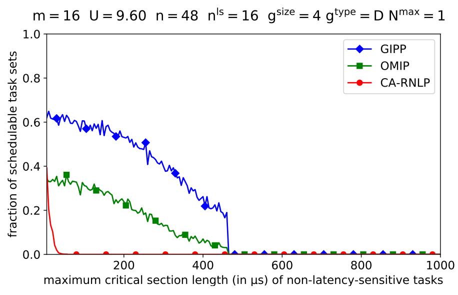

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

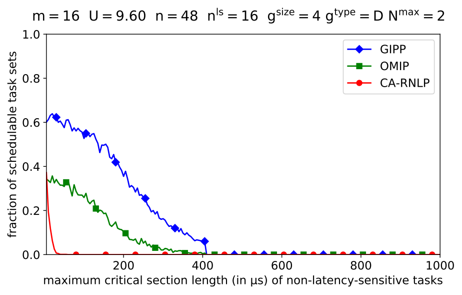

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 3$ $g^{type} = D$ $N^{max} = 3$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP

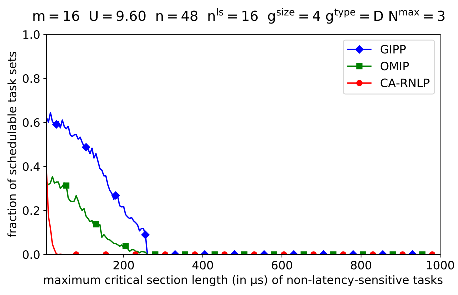

$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 1$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP


$m = 16$ $U = 9.60$ $n = 48$ $n^{ls} = 16$ $g^{size} = 4$ $g^{type} = W$ $N^{max} = 2$
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP


m = 16 U = 9.60 n = 48 n^ls = 16 g^size = 4 g^type = W N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP


m = 16 U = 9.60 n = 48 n^ls = 16 g^size = 4 g^type = D N^max = 1
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP


m = 16 U = 9.60 n = 48 n^ls = 16 g^size = 4 g^type = D N^max = 2
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP


m = 16 U = 9.60 n = 48 n^ls = 16 g^size = 4 g^type = D N^max = 3
fraction of schedulable task sets
maximum critical section length (in µs) of non-latency-sensitive tasks
GIPP
OMIP
CA-RNLP