Symbiotic Task Scheduling and Data Prefetching

Gilead Posluns Mark C. Jeffrey

UNIVERSITY OF TORONTO

October 20, 2025

Hardware task parallelism meets prefetching

- Task parallel runtimes parallelize irregular algorithms
  - OpenCilk, OpenMP 4.0, Galois, Intel TBB, …

Hardware task parallelism meets prefetching

- Task parallel runtimes parallelize irregular algorithms
  - OpenCilk, OpenMP 4.0, Galois, Intel TBB, …
  - But a **software** runtime can be as expensive as the tasks themselves

Hardware task parallelism meets prefetching

- Task parallel runtimes parallelize irregular algorithms
  - OpenCilk, OpenMP 4.0, Galois, Intel TBB, …
  - But a **software** runtime can be as expensive as the tasks themselves
- Offloading the runtime to HW fixes this problem
  - Swarm[Jeffrey+ MICRO'15], HD-CPS[Shan+ HPCA'22], Dalorex[Orenes-Vera+ HPCA'23], Carbon [Kumar+ ISCA'07], …
  - >100X faster than parallel SW alone [Jeffrey et al. MICRO'16]

Hardware task parallelism meets prefetching

- Task parallel runtimes parallelize irregular algorithms
  - OpenCilk, OpenMP 4.0, Galois, Intel TBB, …
  - But a **software** runtime can be as expensive as the tasks themselves
- Offloading the runtime to HW fixes this problem
  - Swarm[Jeffrey+ MICRO'15], HD-CPS[Shan+ HPCA'22], Dalorex[Orenes-Vera+ HPCA'23], Carbon [Kumar+ ISCA'07], …
  - >100X faster than parallel SW alone [Jeffrey et al. MICRO'16]
  - But prior data prefetchers are incompatible with these HW task schedulers, leaving performance on the table

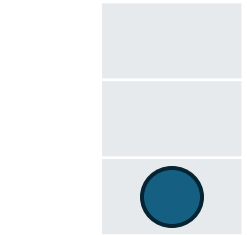
Hardware task parallelism meets prefetching

- Task parallel runtimes parallelize irregular algorithms
  - OpenCilk, OpenMP 4.0, Galois, Intel TBB, …
  - But a **software** runtime can be as expensive as the tasks themselves
- Offloading the runtime to HW fixes this problem
  - Swarm[Jeffrey+ MICRO'15], HD-CPS[Shan+ HPCA'22], Dalorex[Orenes-Vera+ HPCA'23], Carbon [Kumar+ ISCA'07], …
  - >100X faster than parallel SW alone [Jeffrey et al. MICRO'16]
  - But prior data prefetchers are incompatible with these HW task schedulers, leaving performance on the table
- Our Task Seeded Prefetcher and the Memory Response Scheduler
  - first data prefetcher and HW task scheduler that cooperate
  - Up to 3.1X faster than task parallel HW

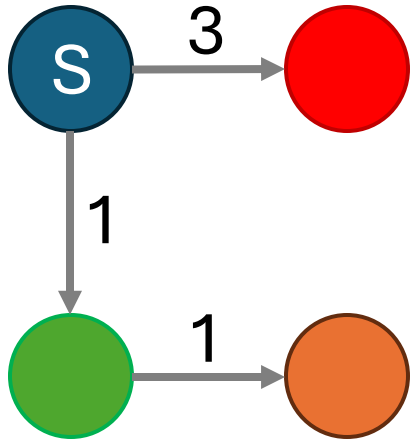
Understanding HW Task Architectures and Prefetching

Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){



}}
```



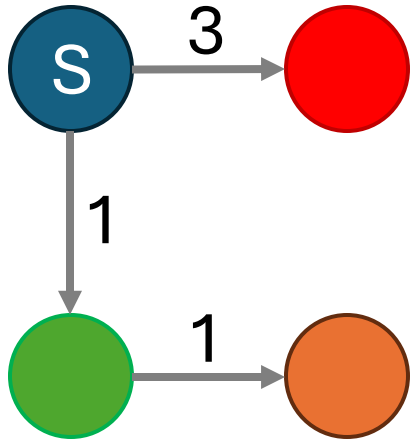
Scheduler



Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){

}}
```

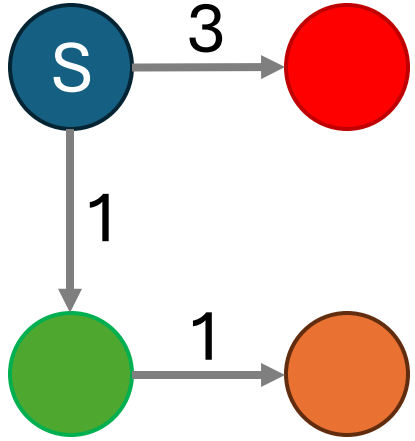

Scheduler



Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;



}}
```

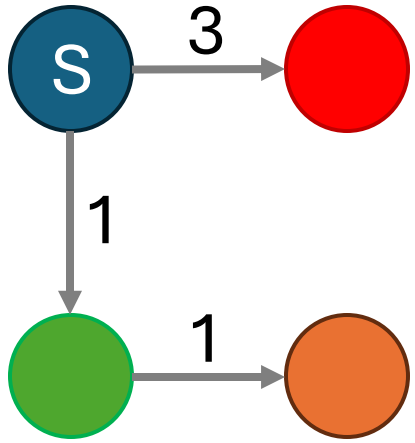

Scheduler



Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist


}}
```

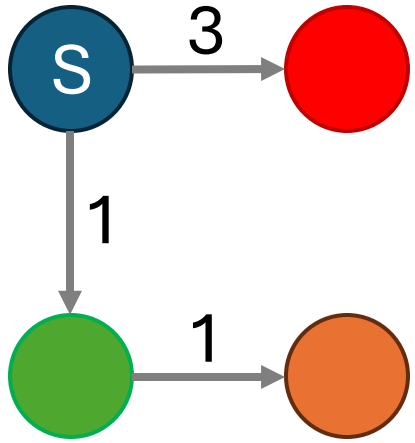

Scheduler



Task Parallelism enables irregular parallelism

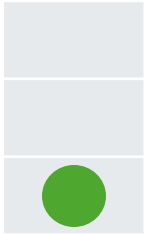
```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){

}}
```


Scheduler

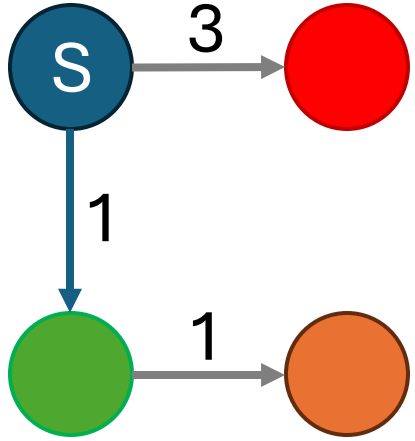


Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```

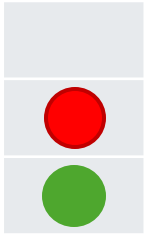


Scheduler

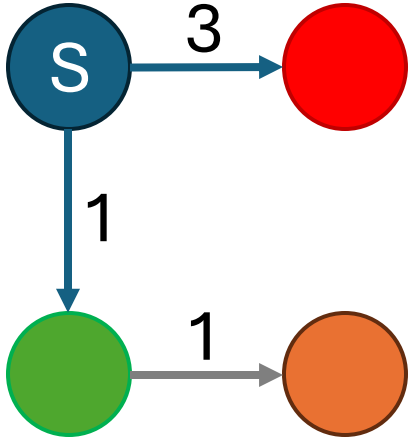


Task Parallelism enables irregular parallelism

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```



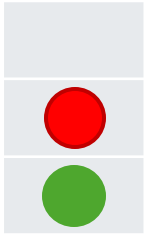
Scheduler



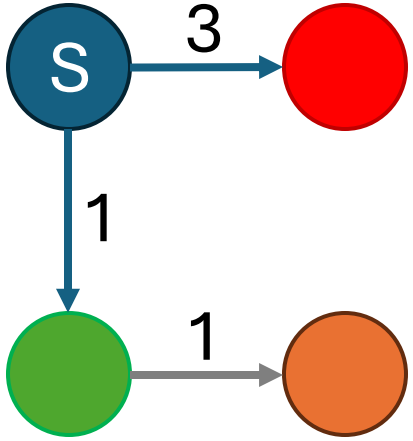
Task Parallelism enables irregular parallelism

<100 Cycles

```
void SSSPTask(int dist, int node){
    if (dist >= dists[node]) return;
    dists[node] = dist
    for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
    runtime::enqueue(dsts[nbr], dist + weights[nbr])
    }}
```

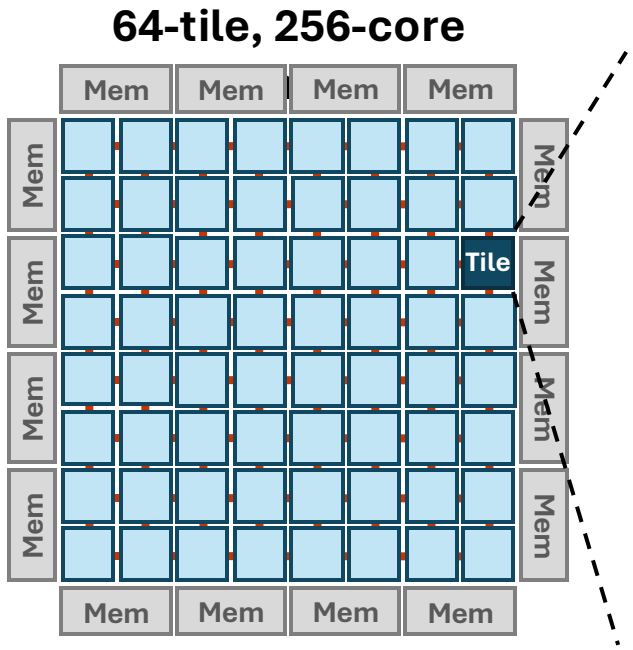


Scheduler

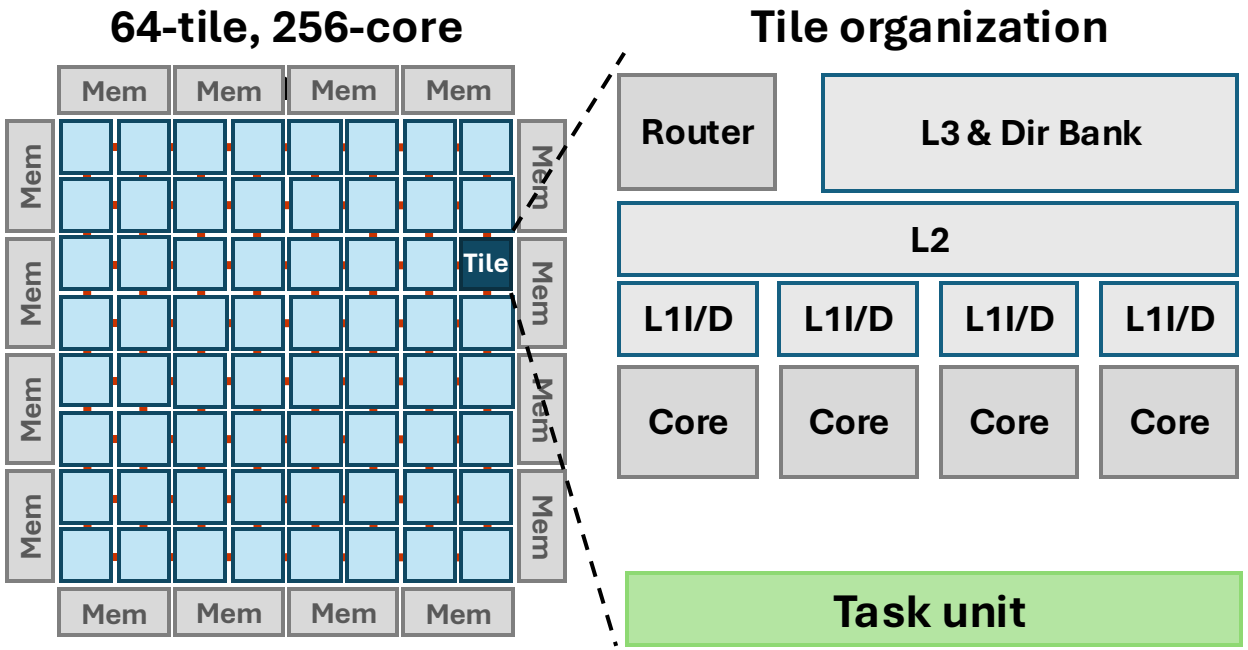


Tasks are very short, motivating hardware schedulers

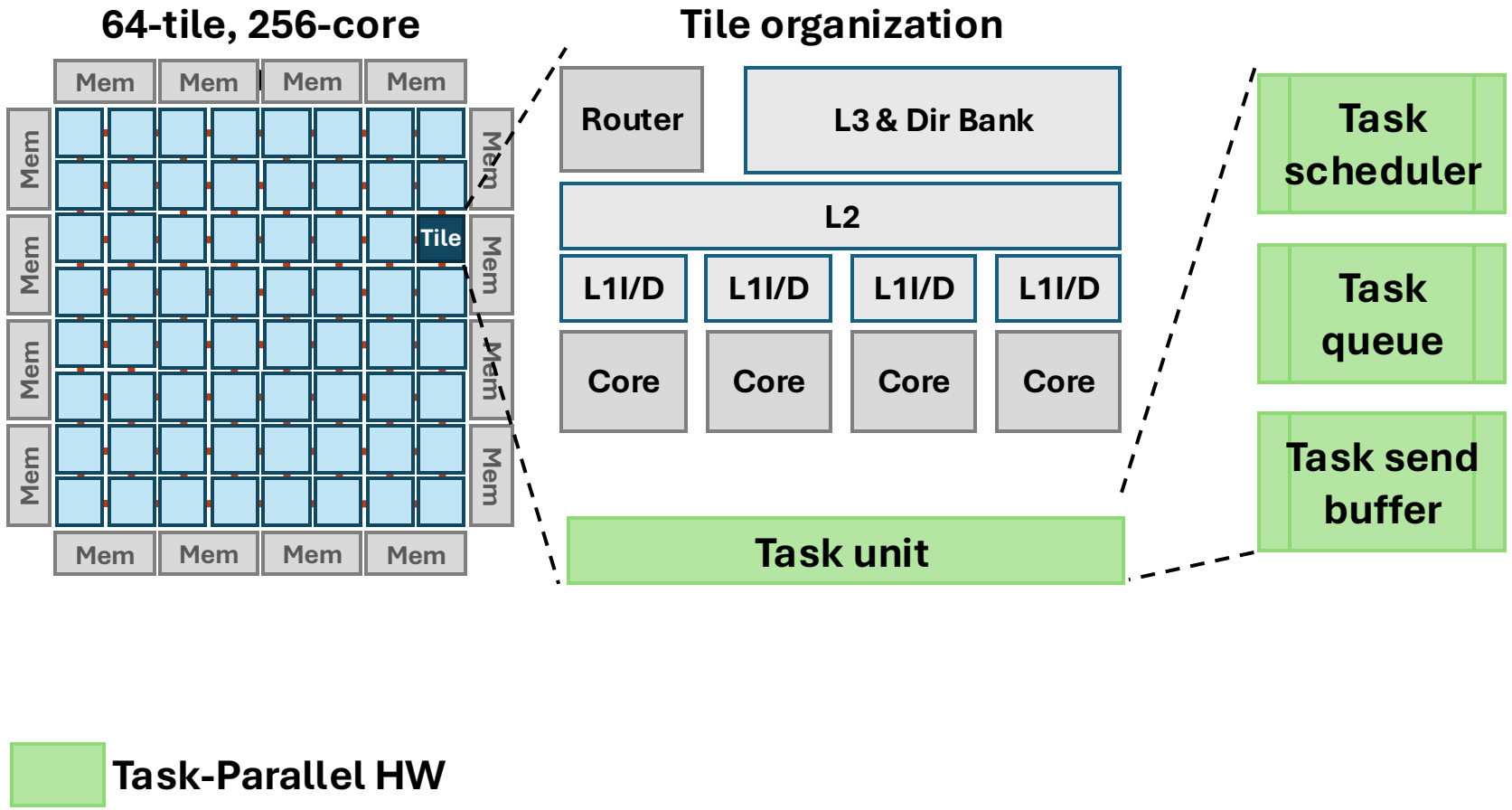
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



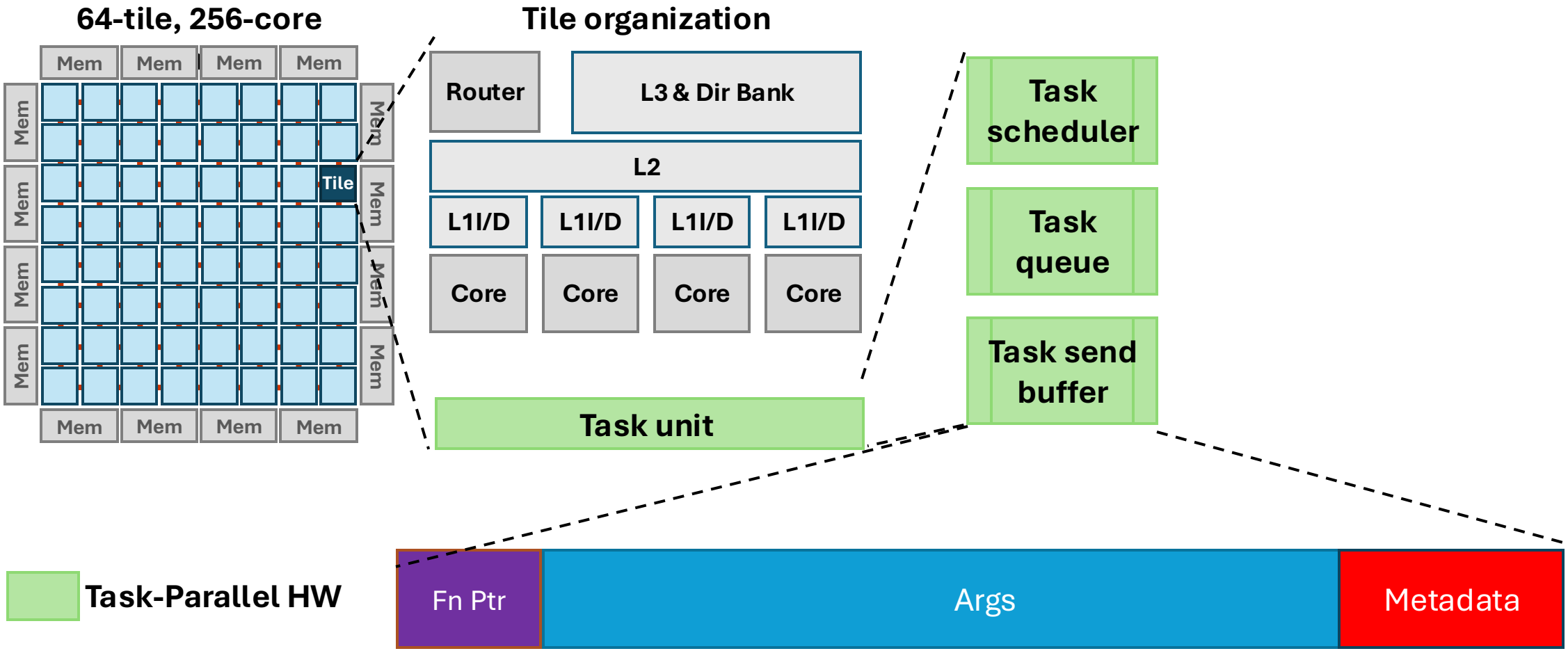
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



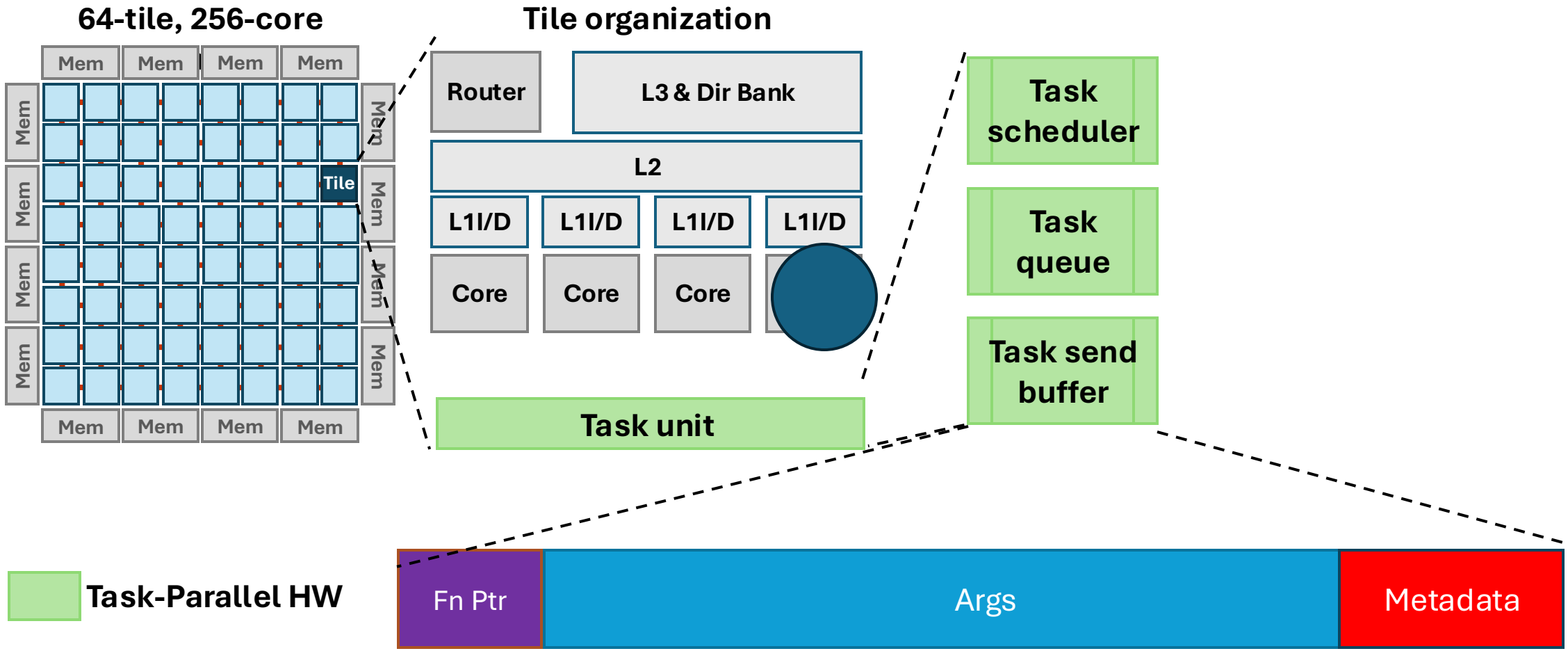
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



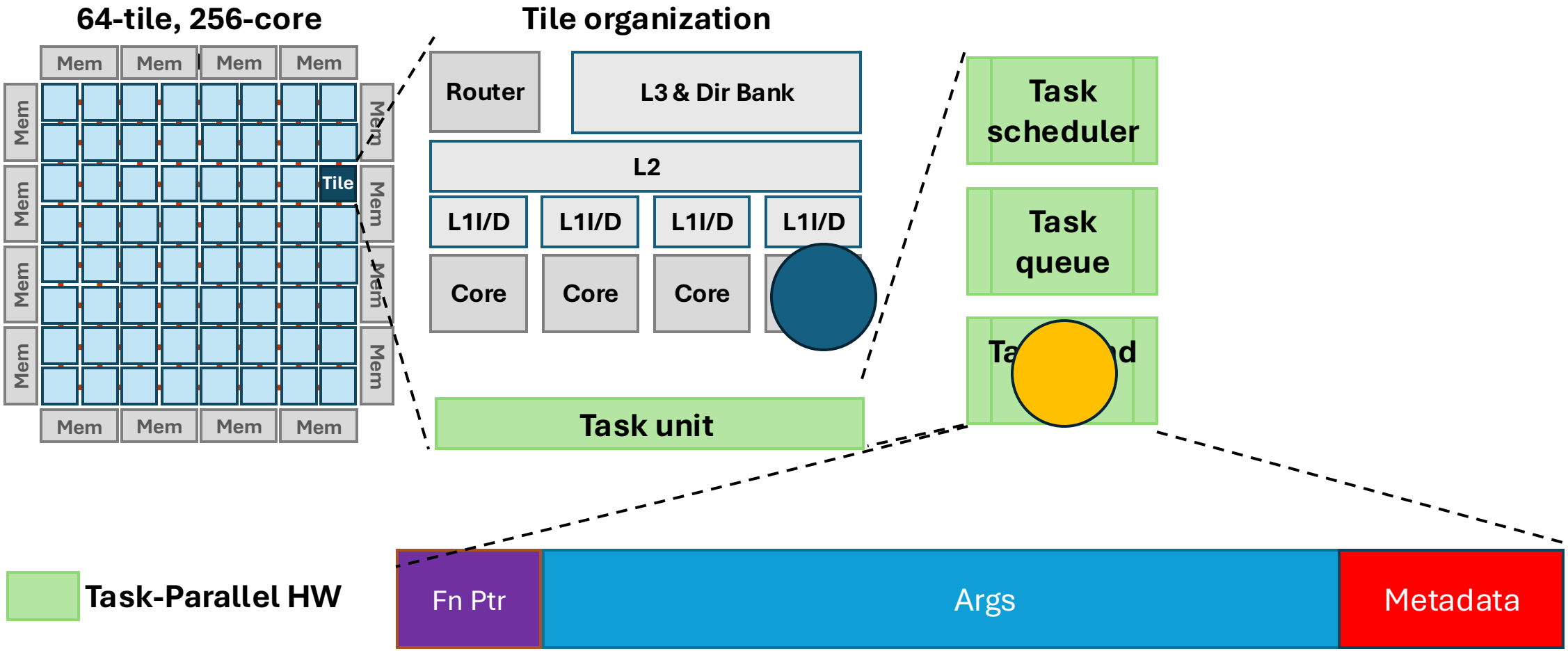
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



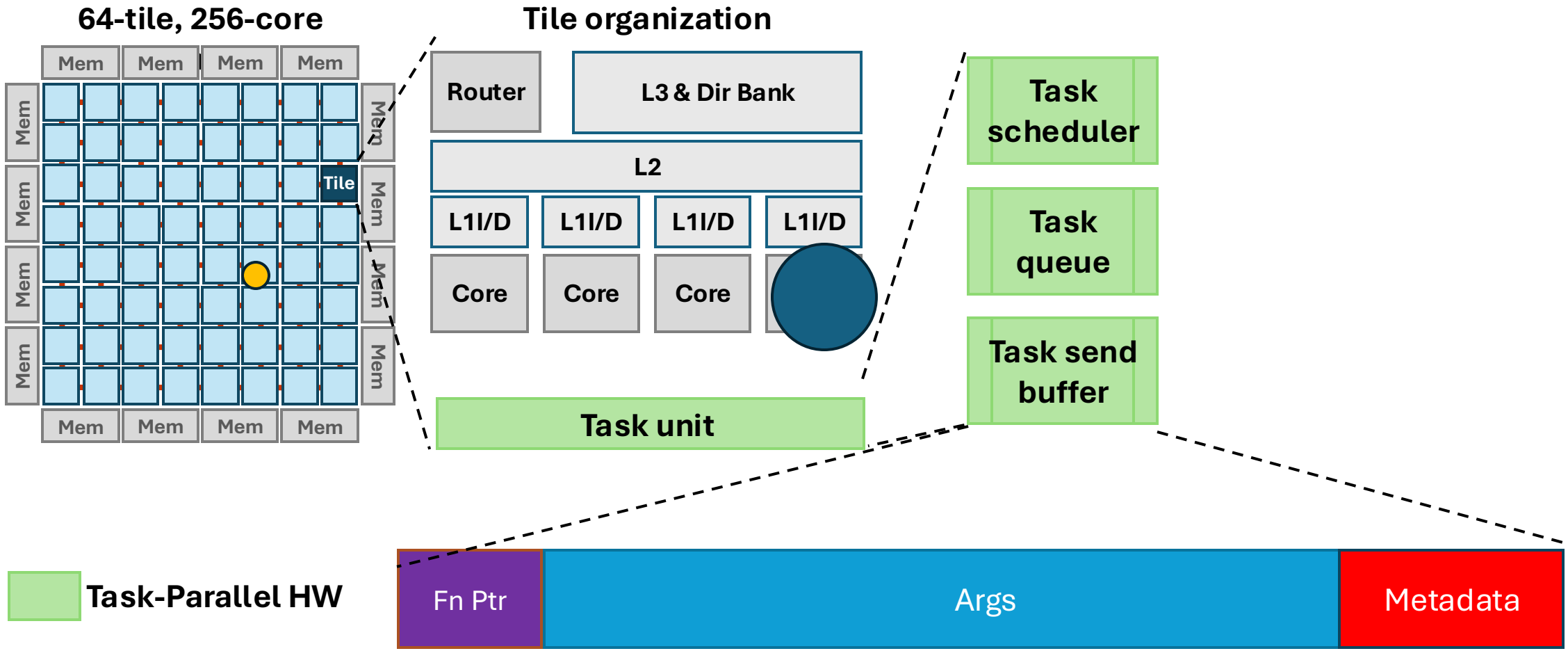
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



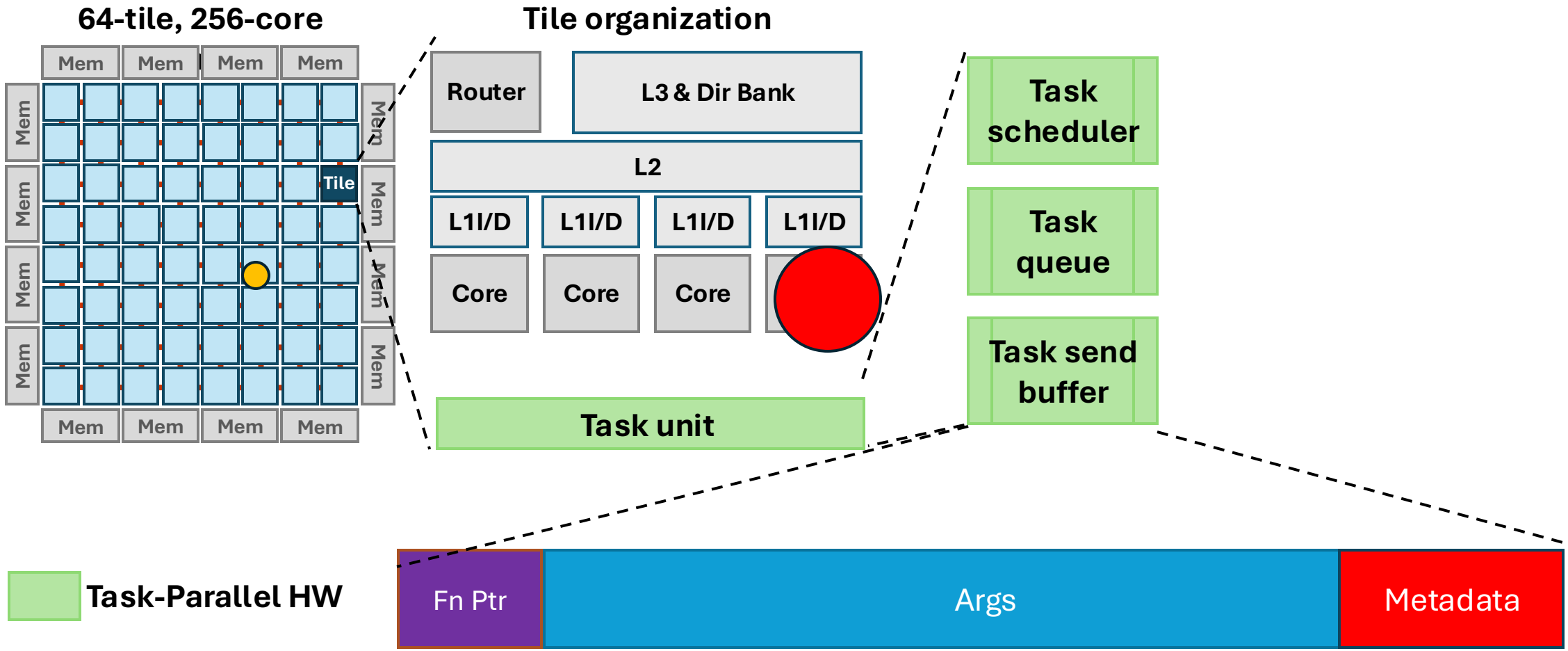
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



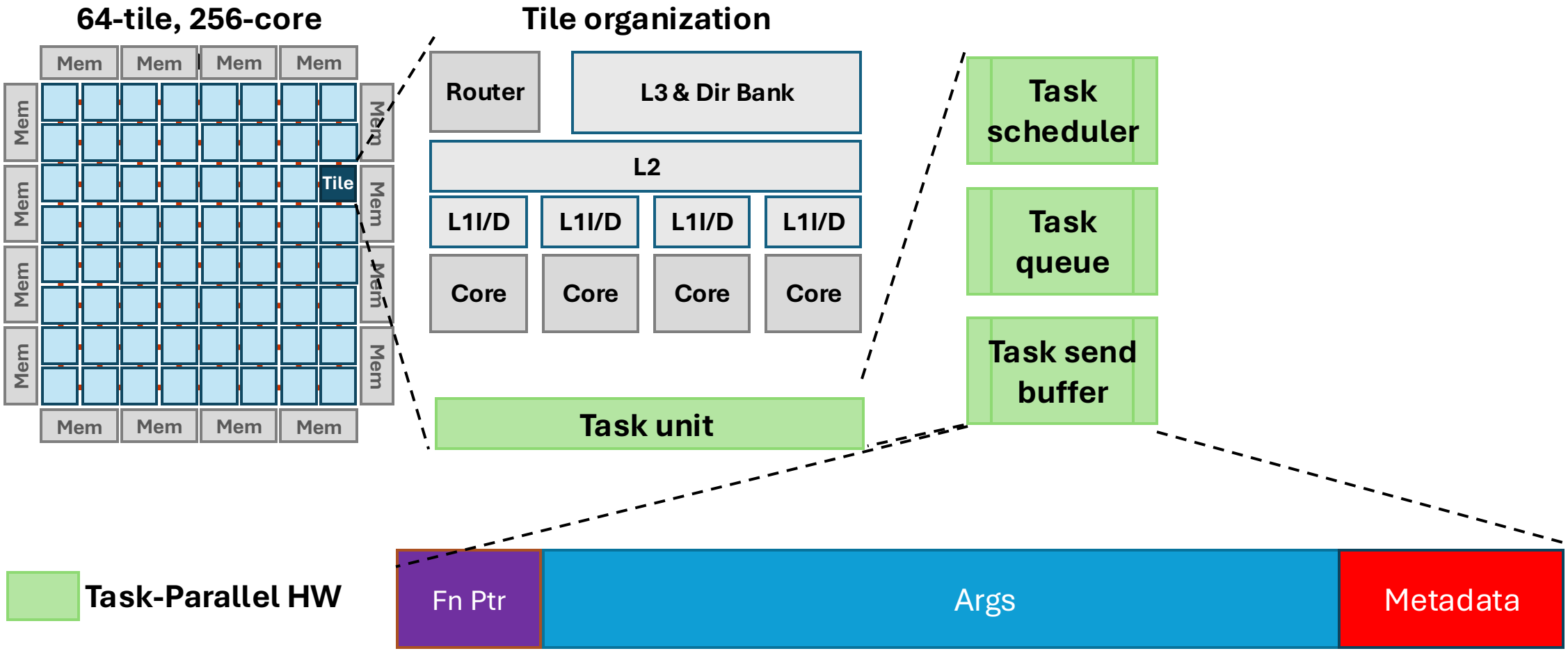
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



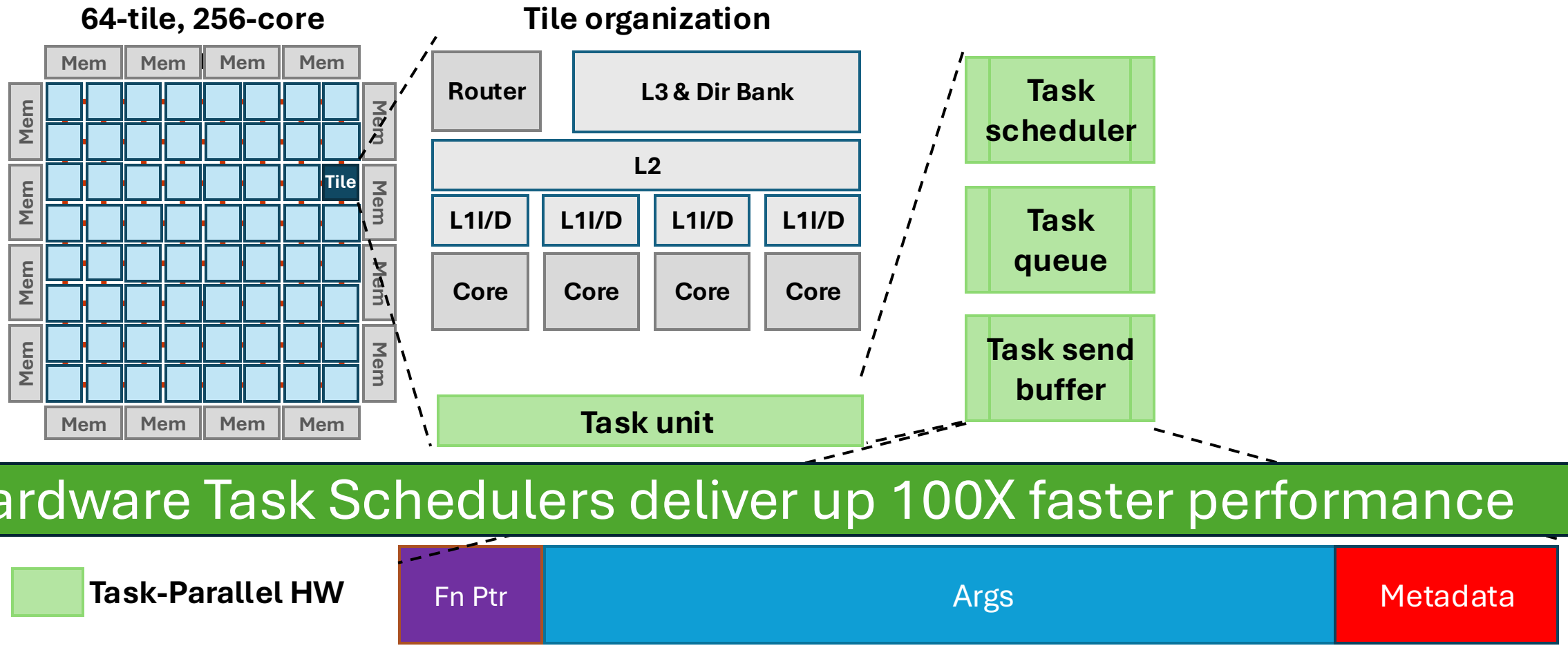
Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



Hardware-Assisted Task Scheduling [Jeffrey et al. MICRO'15]



Tasks suffer high DRAM latencies

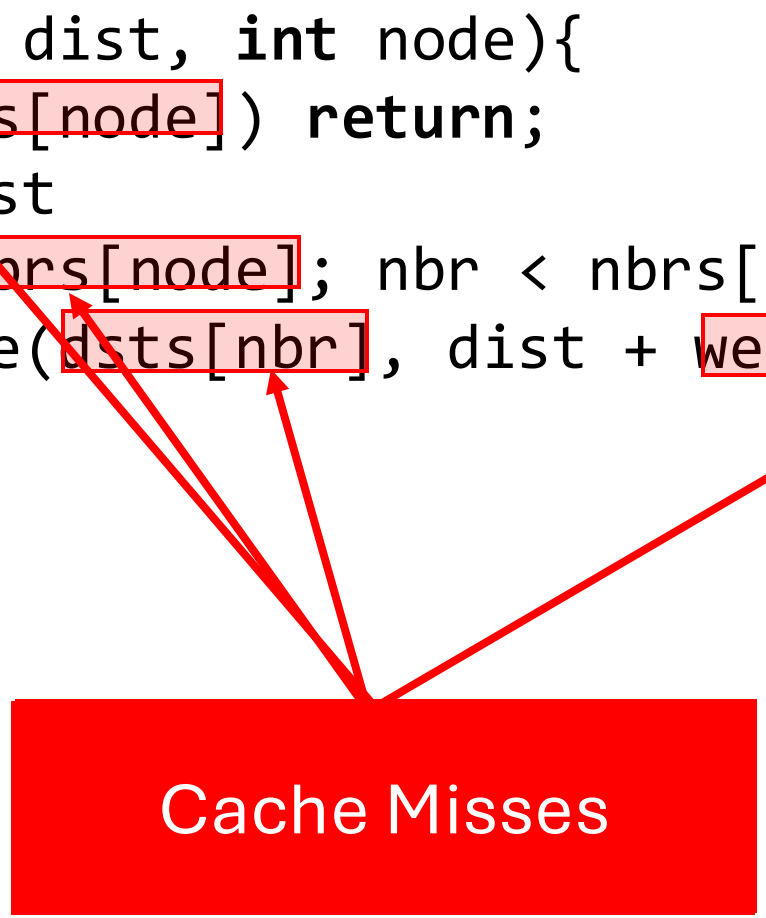
```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```

Tasks suffer high DRAM latencies

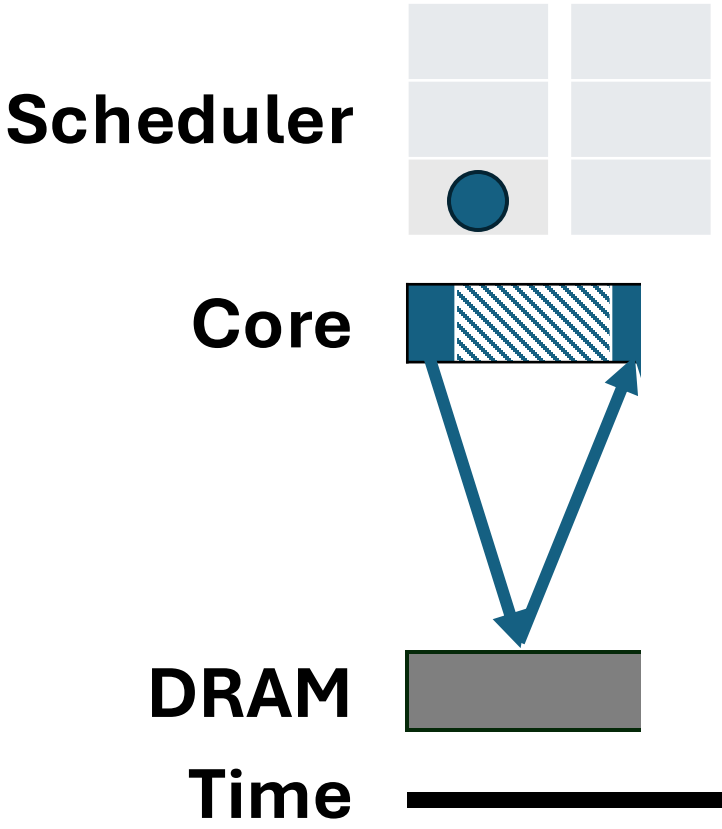
```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```

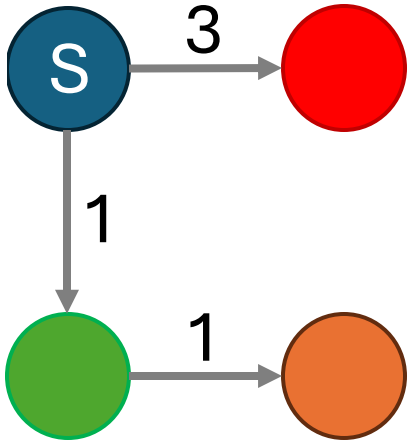
Tasks suffer high DRAM latencies

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```

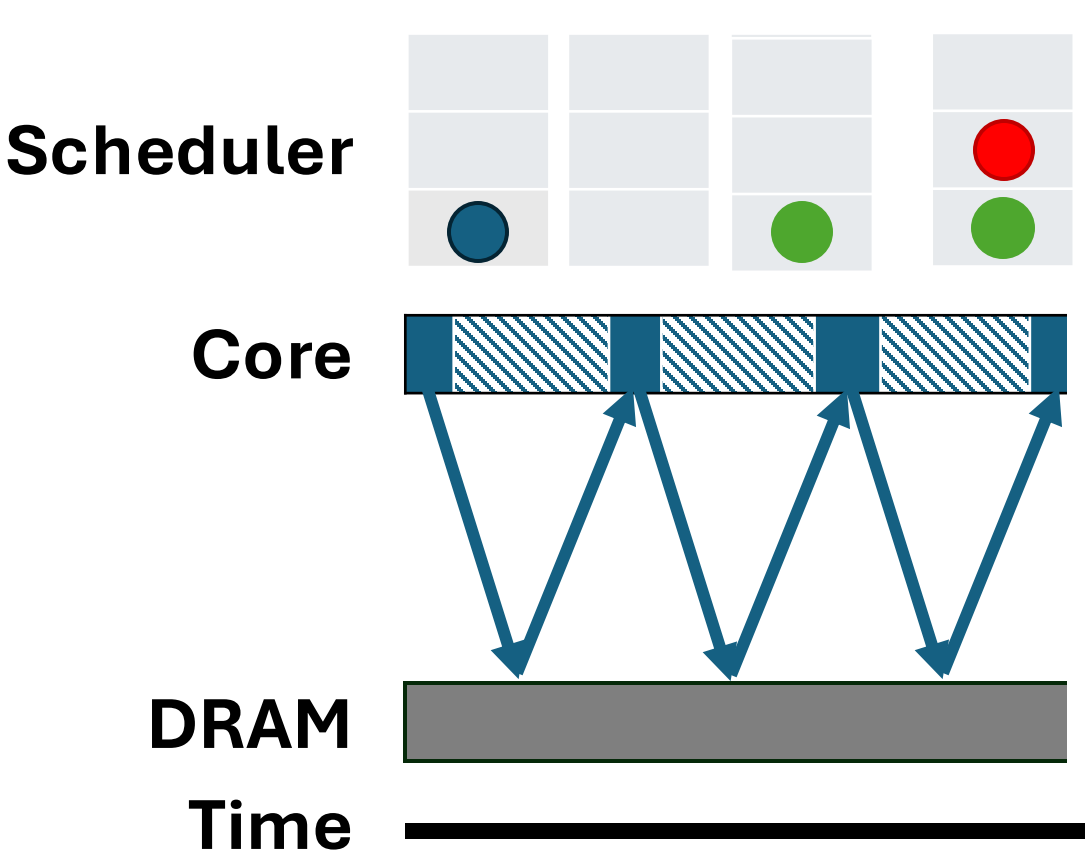


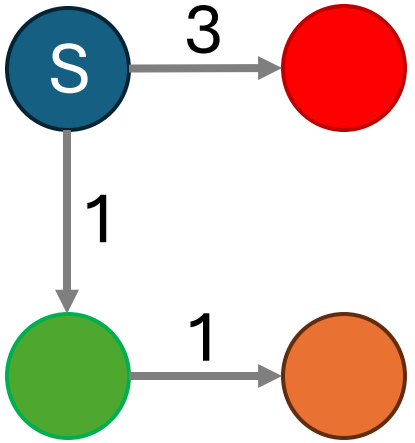
One-Core Task Timeline



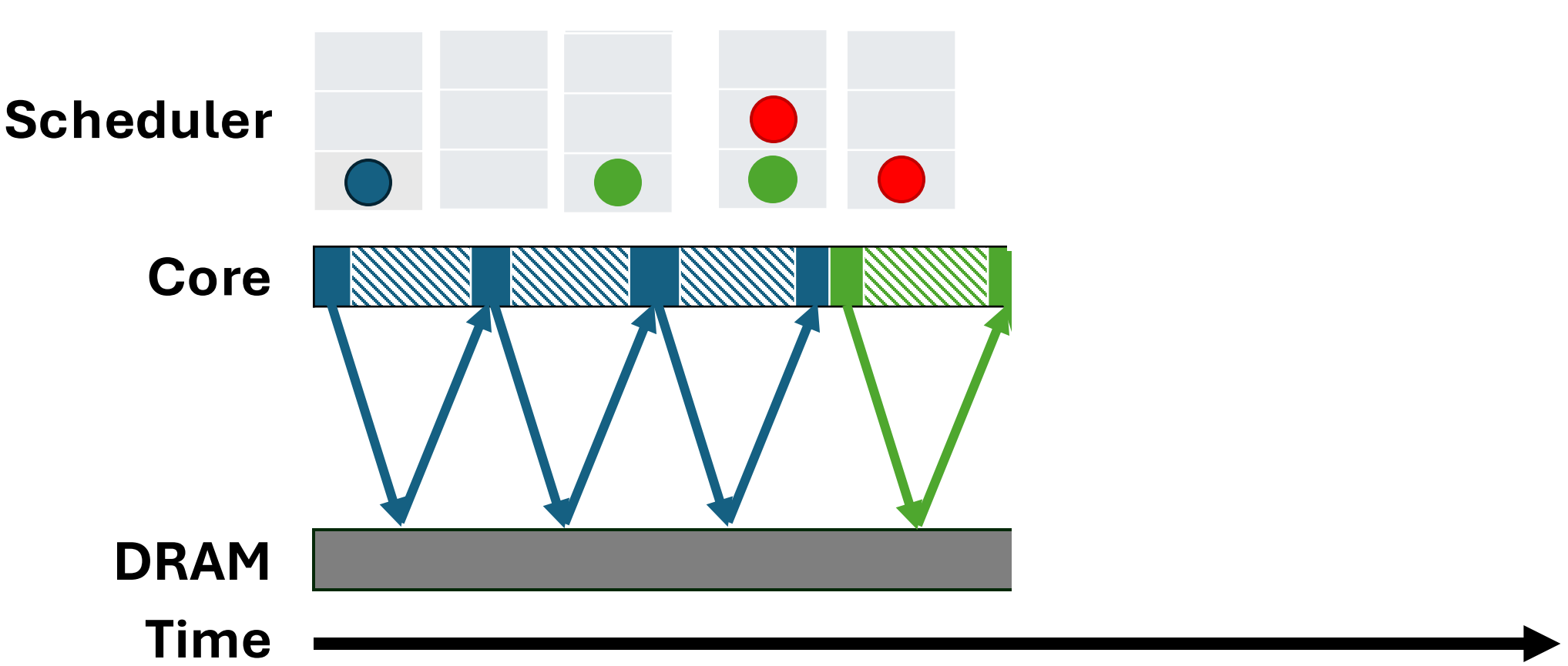


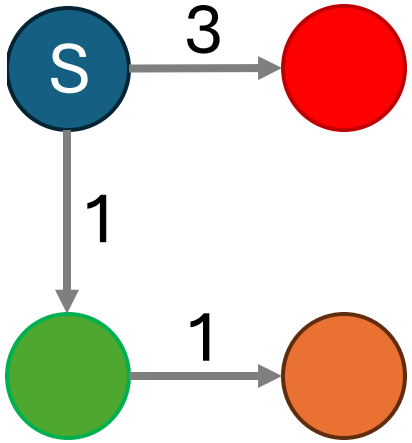
One-Core Task Timeline



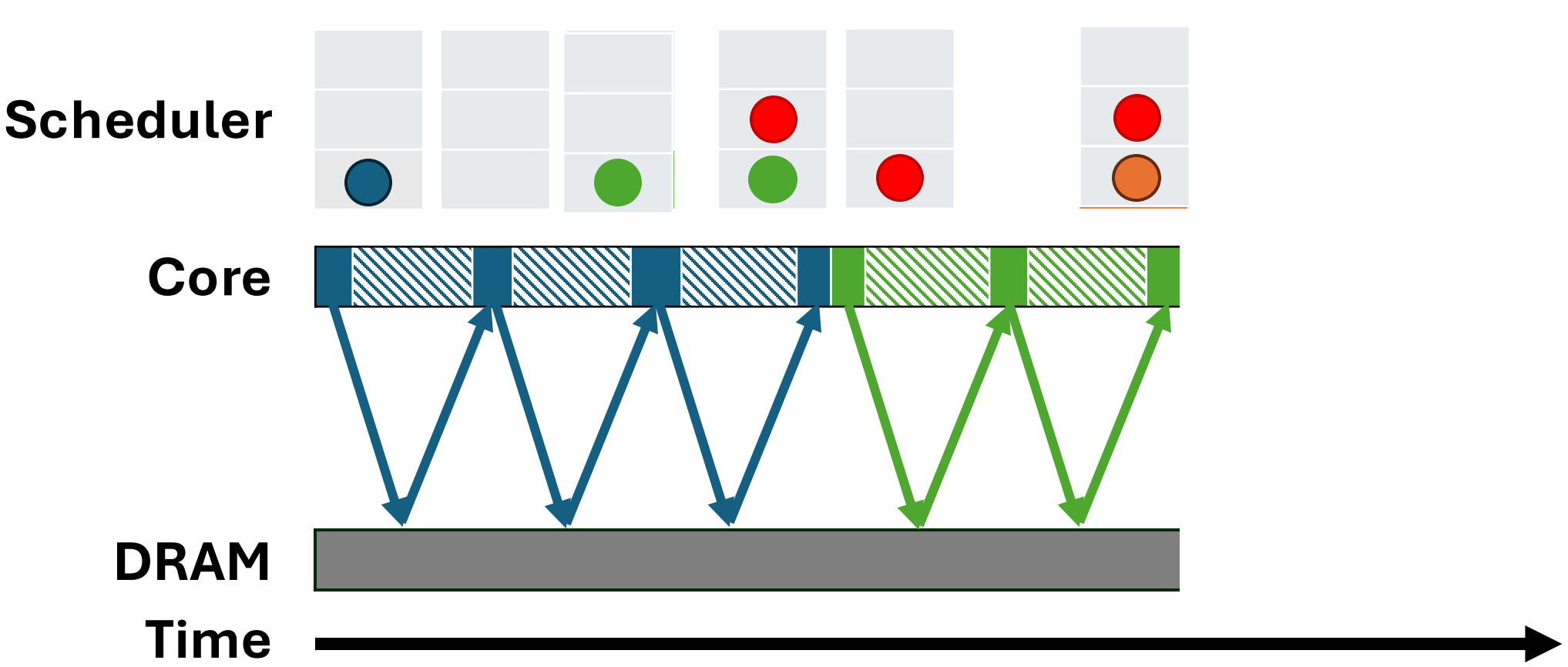


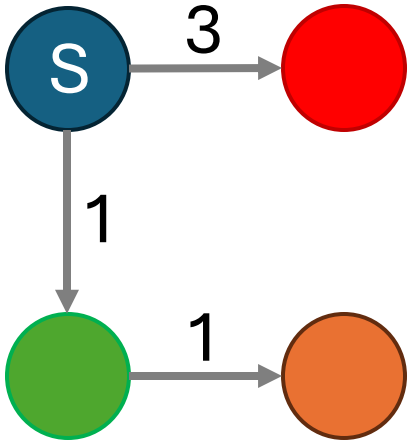
One-Core Task Timeline



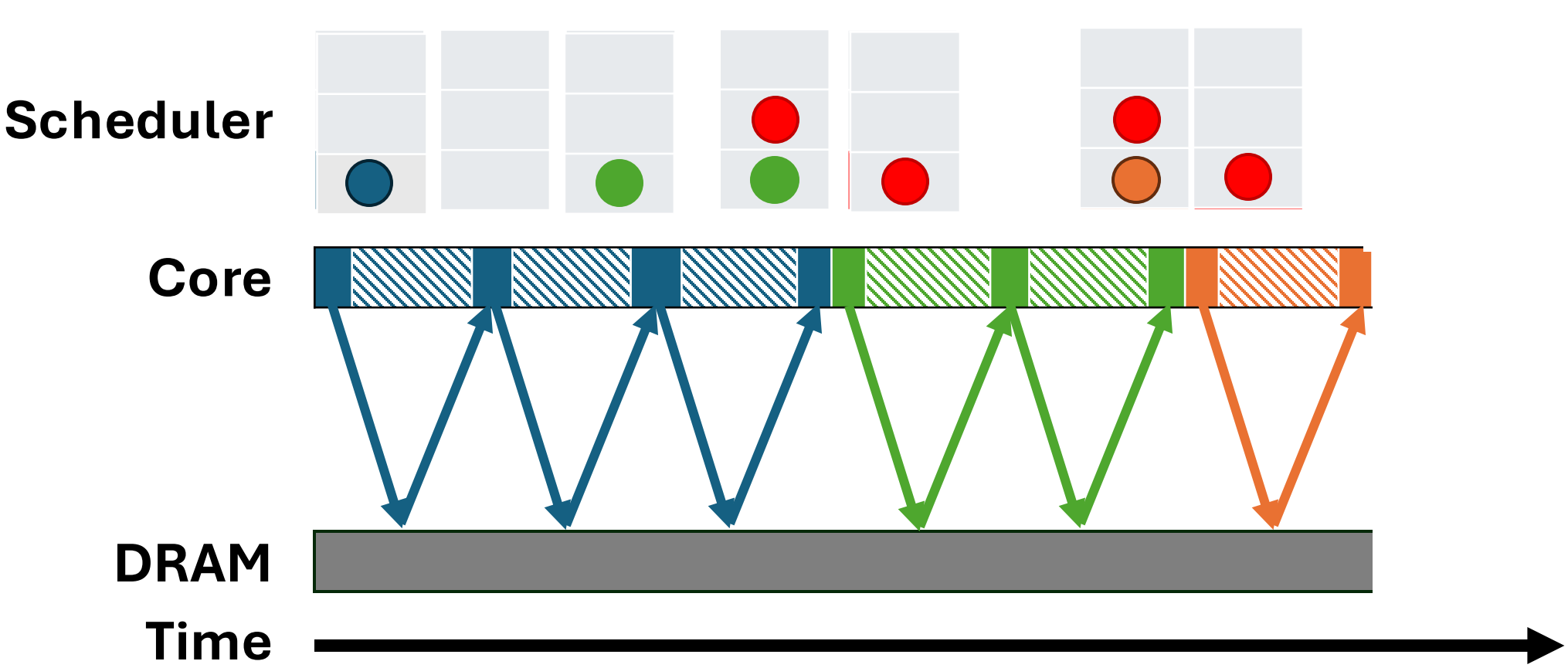


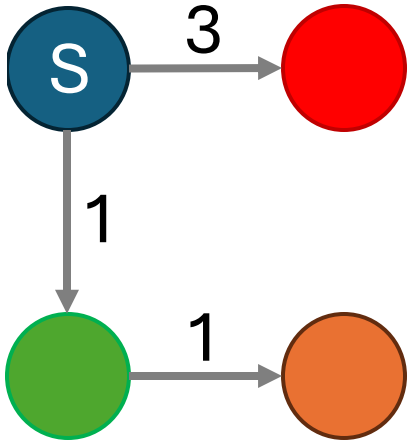
One-Core Task Timeline



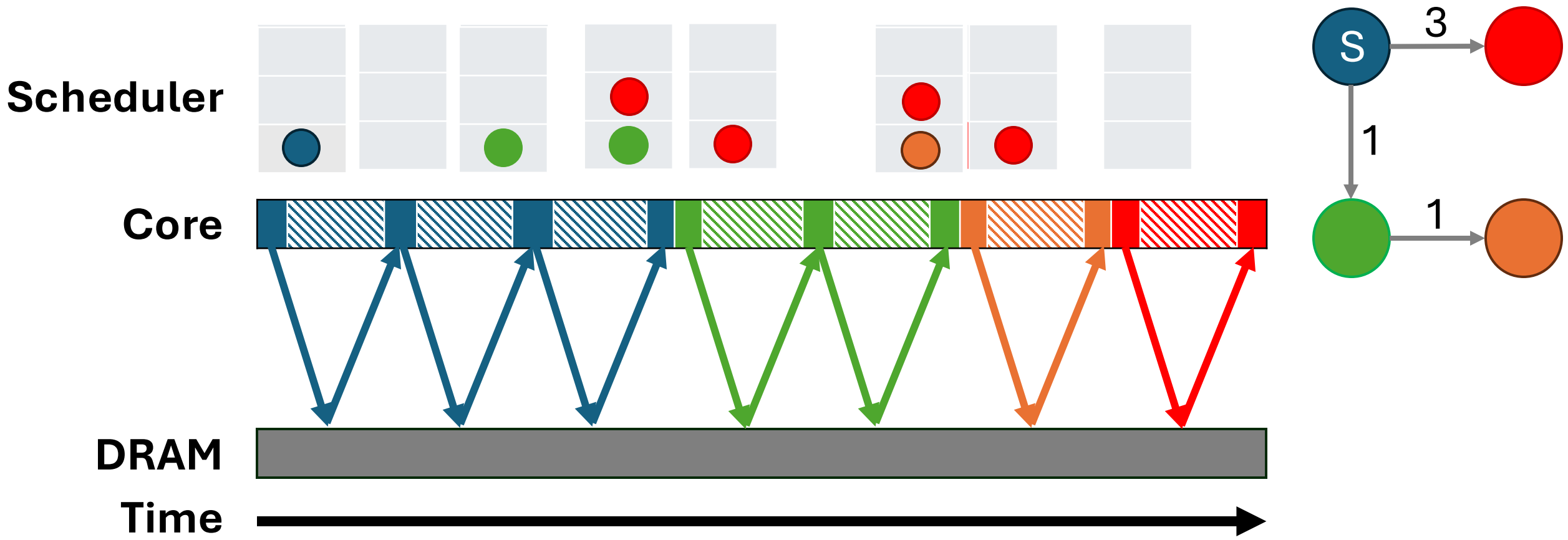


One-Core Task Timeline

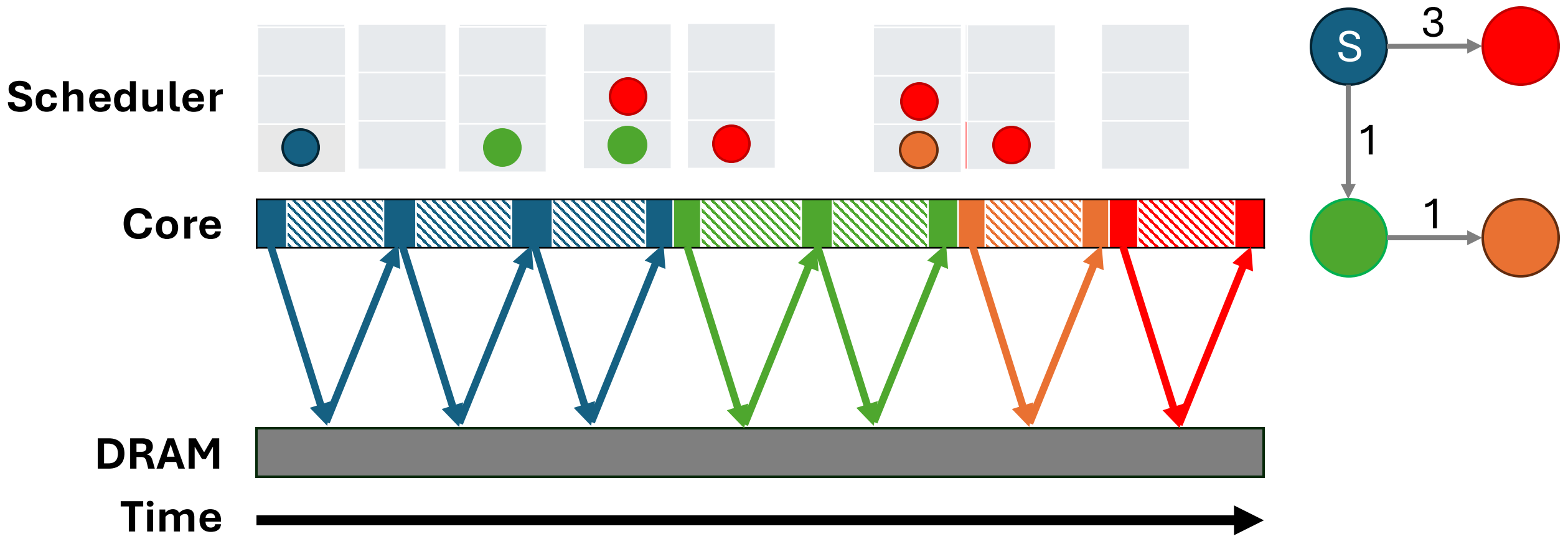




One-Core Task Timeline



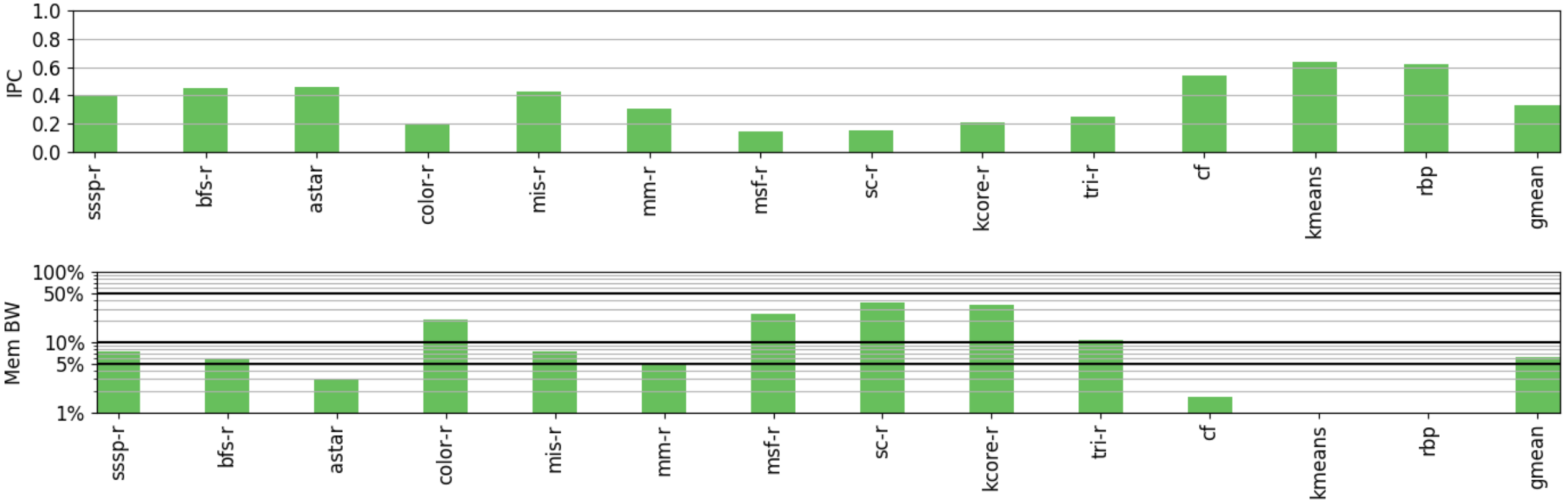
One-Core Task Timeline



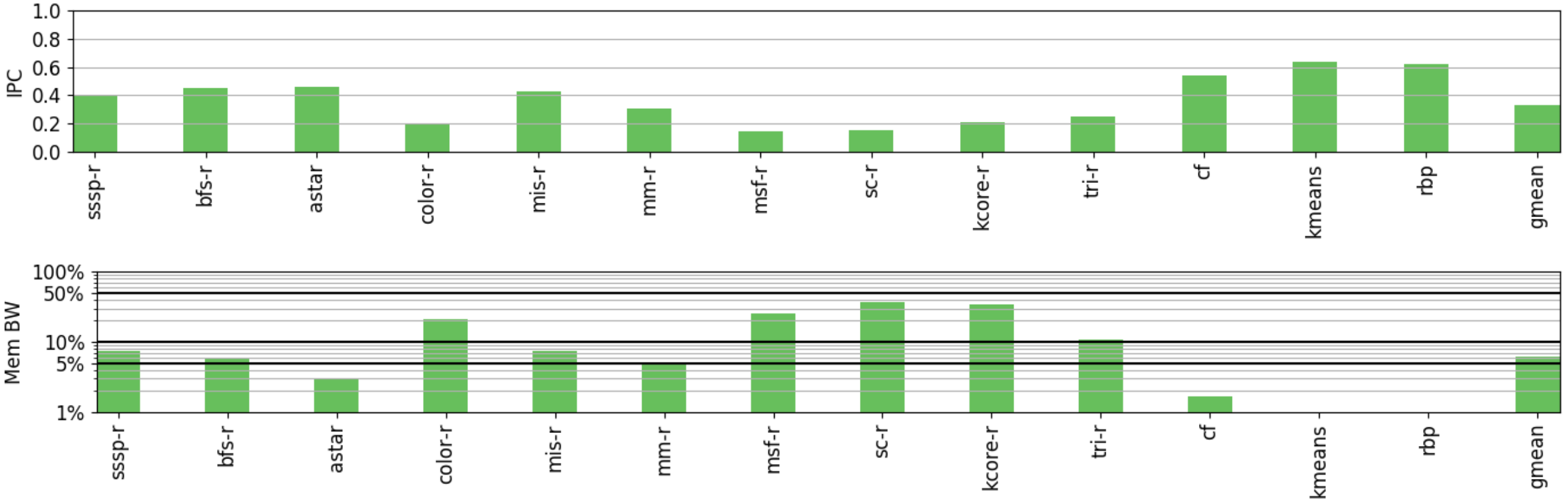
SSSP IPC: 0.4

SSSP BW Utilization: < 10%

SSSP is typical



SSSP is typical



Tiny tasks miss in the cache and have BW to spare

Why do prior prefetchers fail task parallel HW?

Prefetchers like IMP[Yu et al. MICRO'15] and Vector Runahead [Naithani et al. ISCA'21] can prefetch irregular applications…

Why do prior prefetchers fail task parallel HW?

Prefetchers like IMP [Yu et al. MICRO'15] and Vector Runahead [Naithani et al. ISCA'21] can prefetch irregular applications…

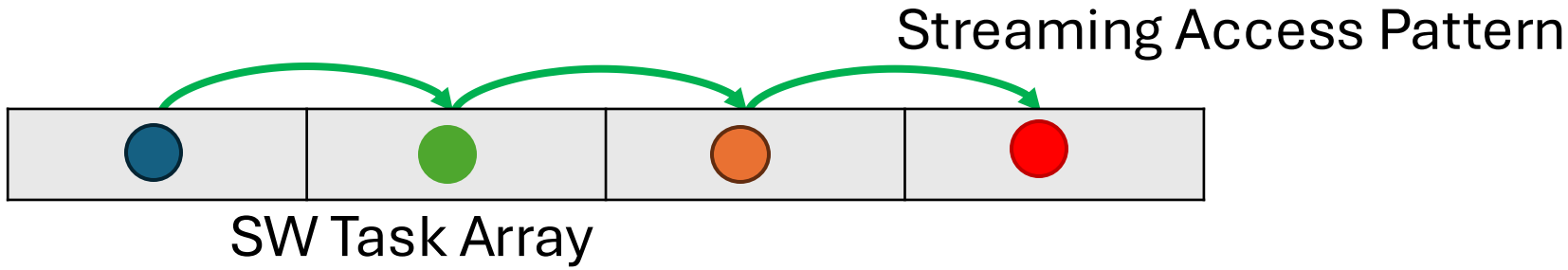
- But only by looking ahead in a SW data structure

SW Task Array

Why do prior prefetchers fail task parallel HW?

Prefetchers like IMP [Yu et al. MICRO'15] and Vector Runahead [Naithani et al. ISCA'21] can prefetch irregular applications…

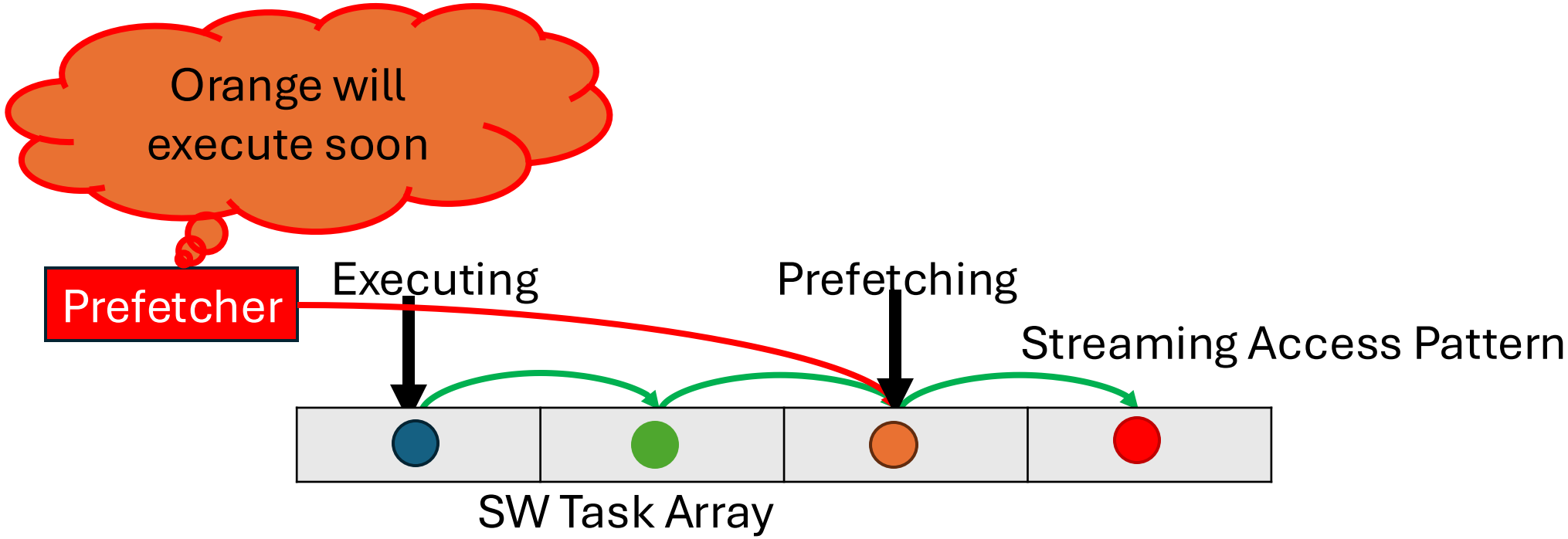
- But only by looking ahead in a SW data structure



Why do prior prefetchers fail task parallel HW?

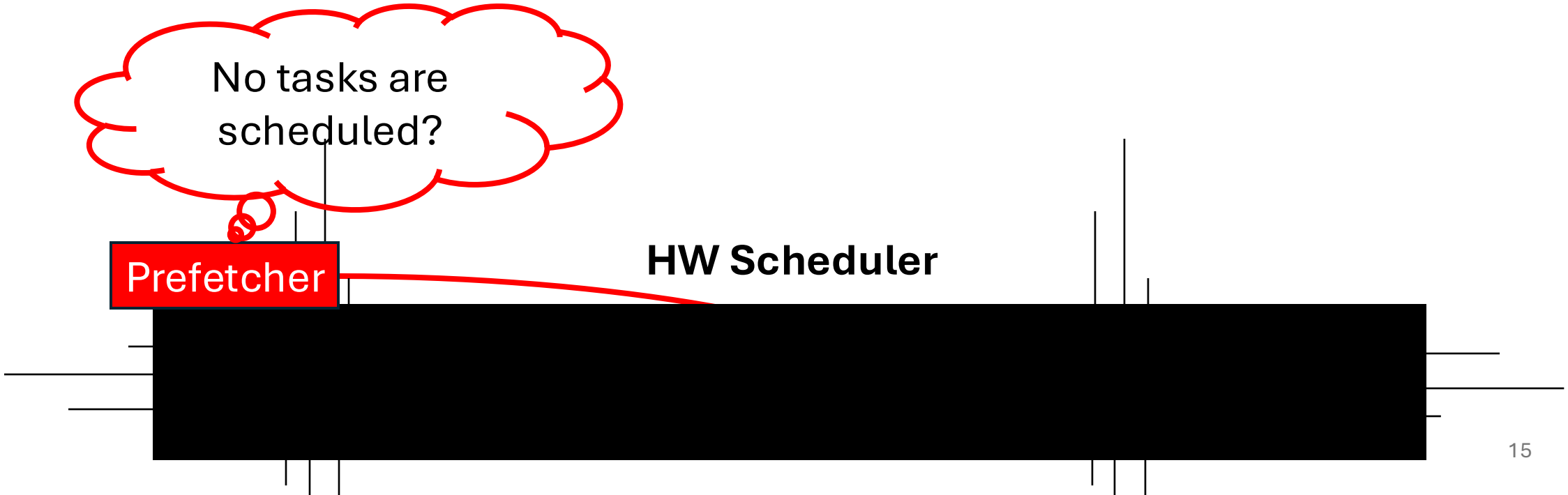
Prefetchers like IMP [Yu et al. MICRO'15] and Vector Runahead [Naithani et al. ISCA'21] can prefetch irregular applications…

- But only by looking ahead in a SW data structure



Why do prior prefetchers fail task parallel HW?

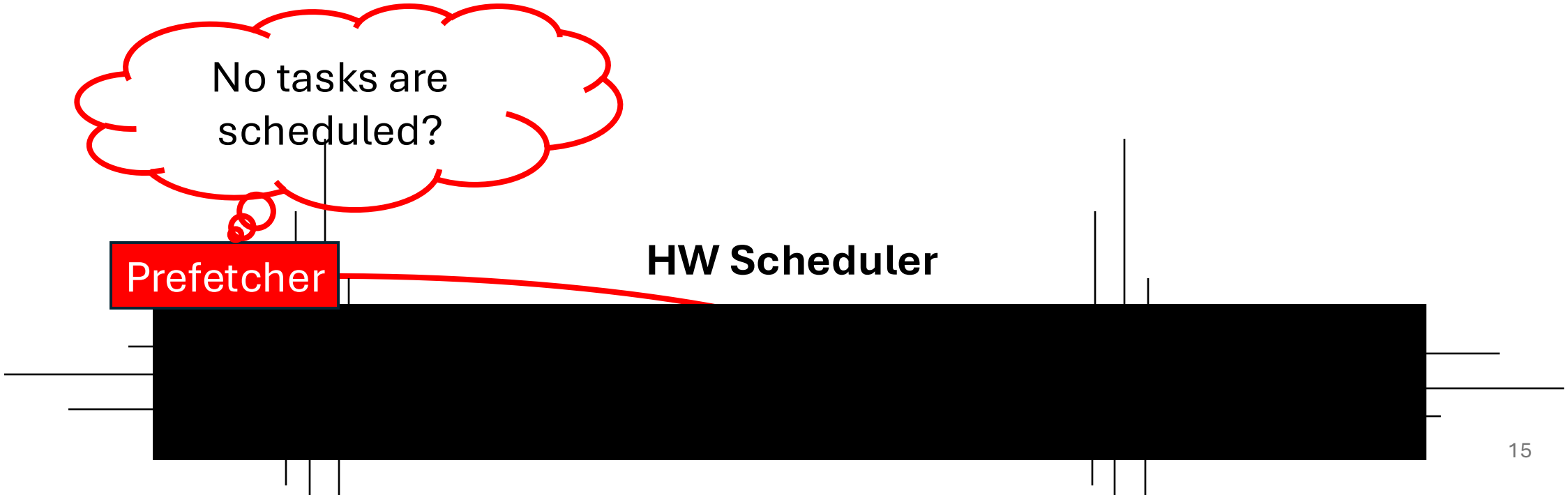
HW Task schedulers eliminate the task array, leaving conventional prefetchers blind!



Why do prior prefetchers fail task parallel HW?

HW Task schedulers eliminate the task array, leaving conventional prefetchers blind!

HW Task schedulers hide the information needed for prefetching



Introducing TSP and MRS

Introducing TSP and MRS

- Task-Seeded Prefetcher

- Memory Response Scheduler

Introducing TSP and MRS

- Task-Seeded Prefetcher
  - Scheduler seeds it with a task descriptor

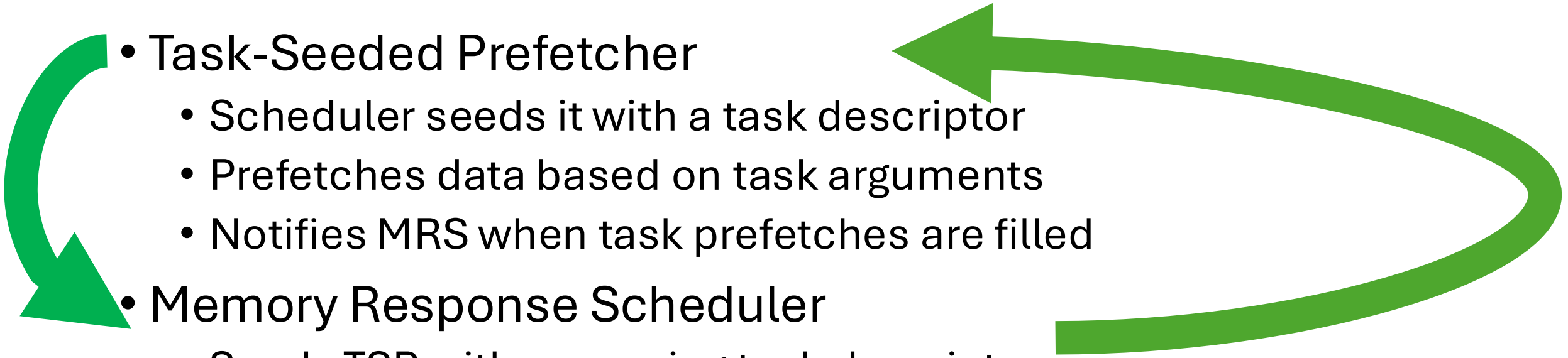
- Memory Response Scheduler

Introducing TSP and MRS

- Task-Seeded Prefetcher
  - Scheduler seeds it with a task descriptor
  - Prefetches data based on task arguments
  - Notifies MRS when task prefetches are filled
- Memory Response Scheduler

Introducing TSP and MRS

- Task-Seeded Prefetcher
  - Scheduler seeds it with a task descriptor
  - Prefetches data based on task arguments
  - Notifies MRS when task prefetches are filled
- Memory Response Scheduler
  - Seeds TSP with upcoming task descriptors
  - Augments a baseline scheduling policy
  - Prioritizes tasks with ready data



TSP turns task arguments to addresses

TSP uses an affine function, like IMP [Yu et al. MICRO'15]

$$addr = offset + scaling \times [ref]$$

TSP turns task arguments to addresses

TSP uses an affine function, like IMP [Yu et al. MICRO'15]

But gets its value directly from task arguments

Learned during Training

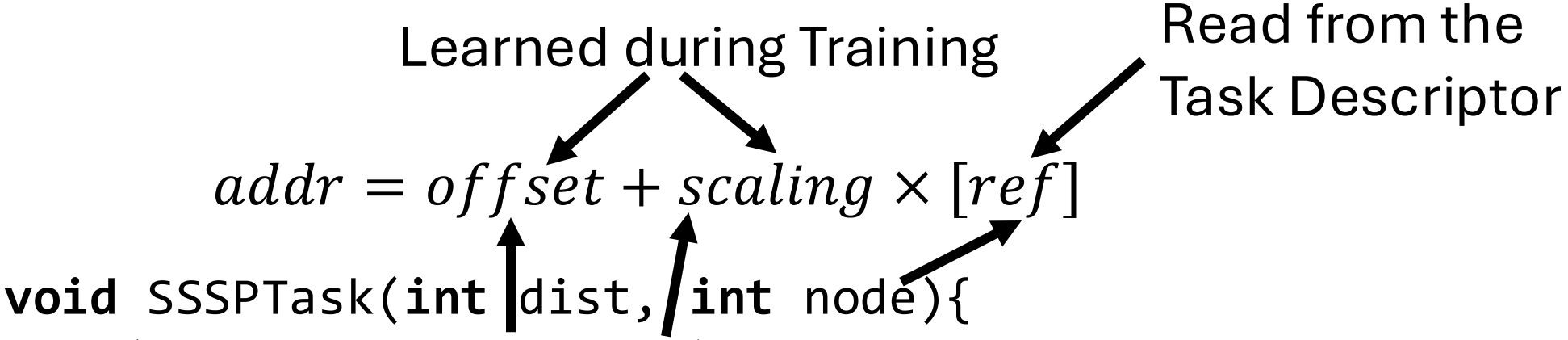
Read from the Task Descriptor

$$addr = offset + scaling \times [ref]$$

TSP turns task arguments to addresses

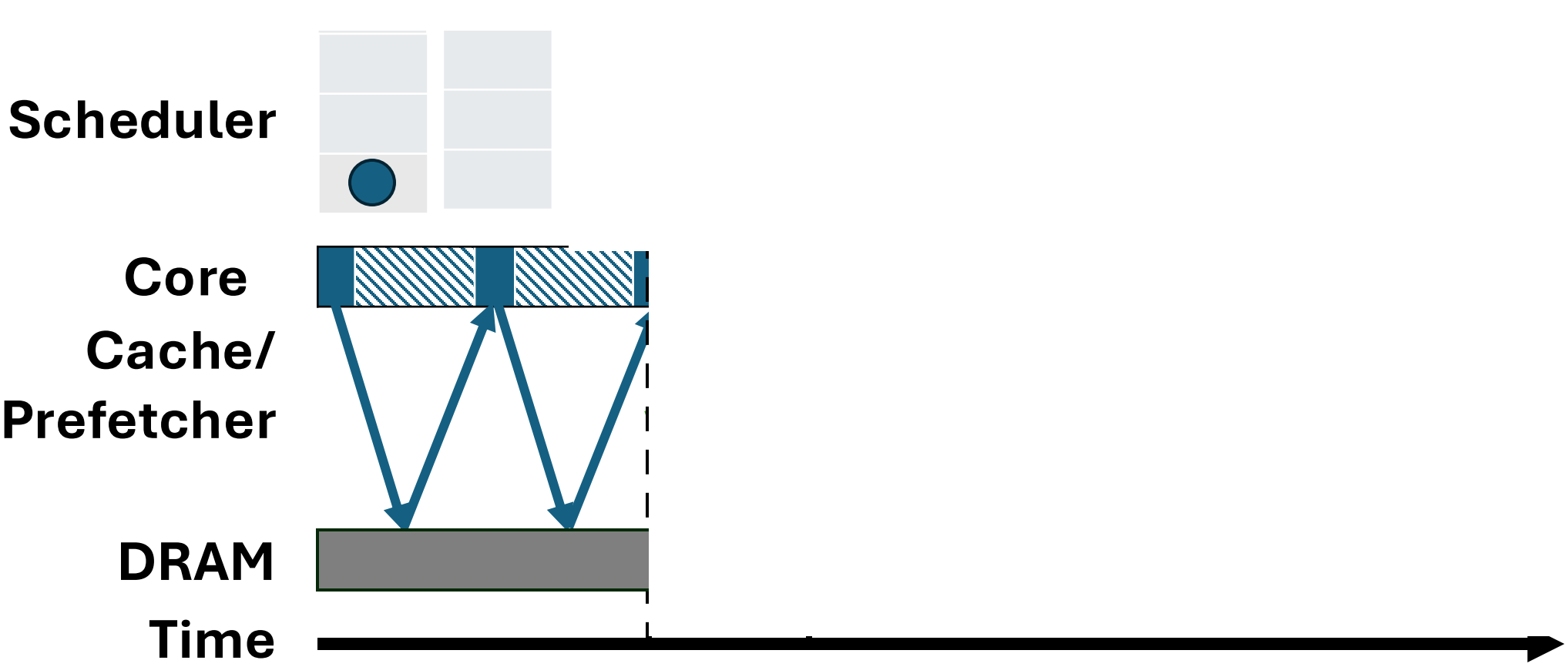
TSP uses an affine function, like IMP [Yu et al. MICRO'15]

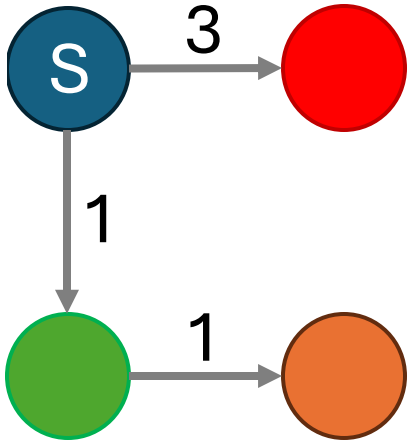
But gets its value directly from task arguments



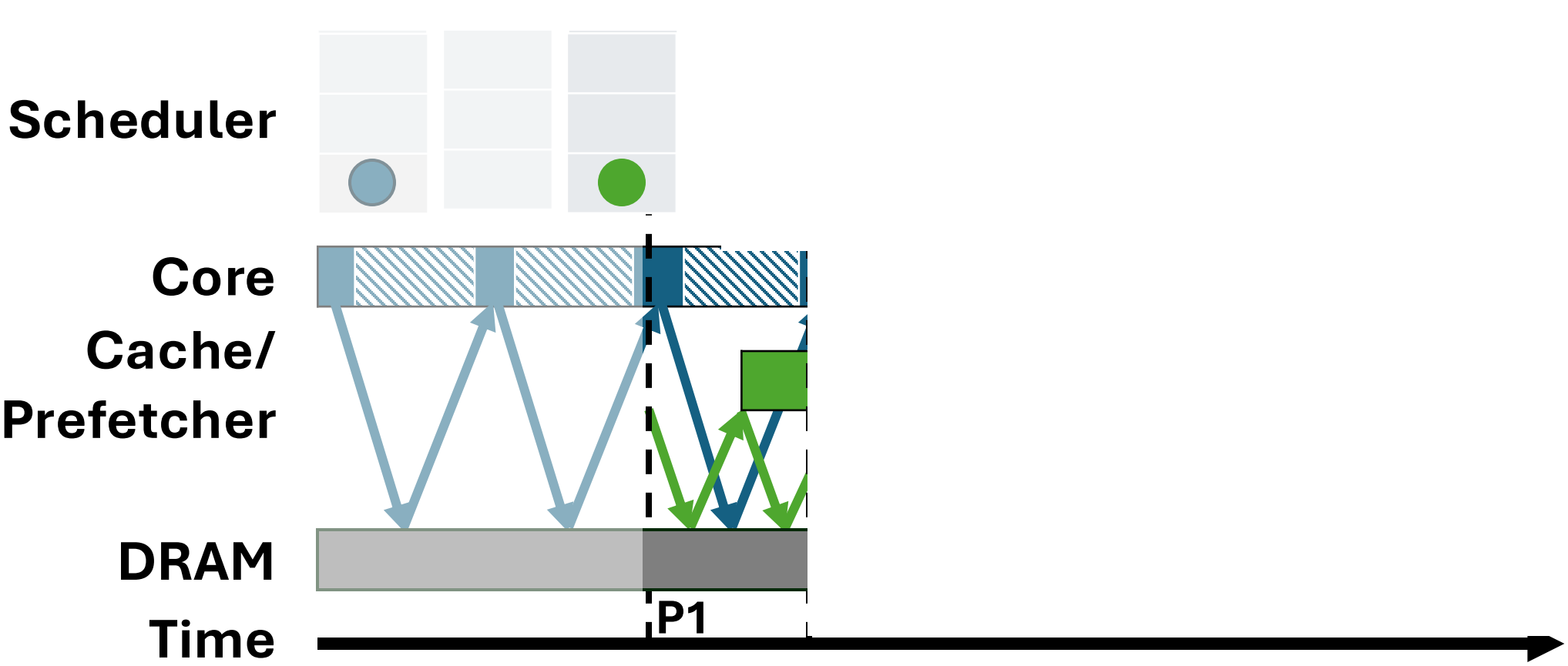

```
void SSSPTask(int dist, int node){
 if (dist >= dists[node]) return;
 dists[node] = dist
 for (int nbr = nbrs[node]; nbr < nbrs[node+1]; nbr++){
  runtime::enqueue(dsts[nbr], dist + weights[nbr])
}}
```

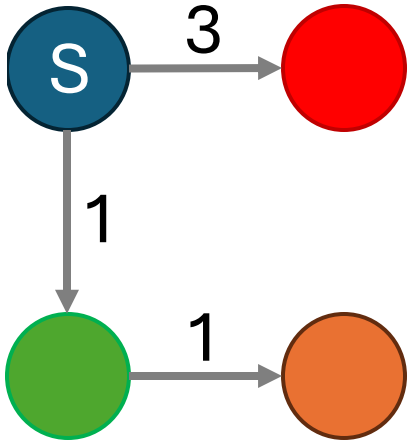
TSP Enables Prefetching Task Data



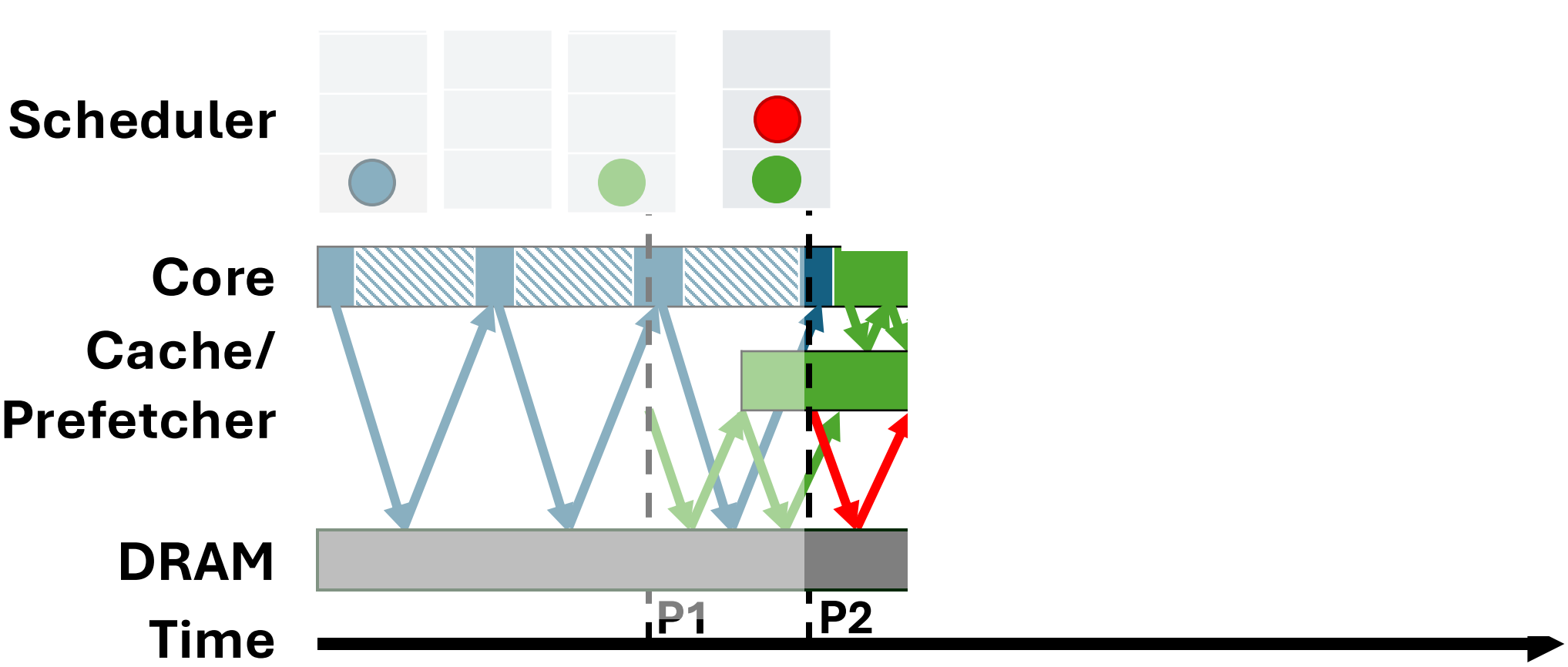


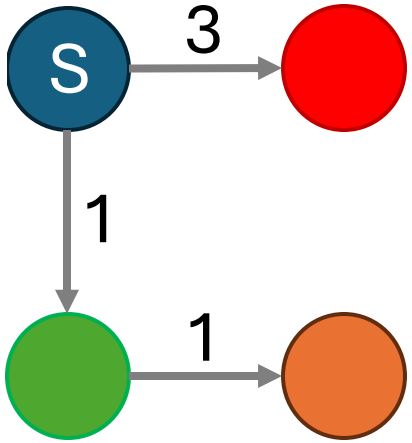
TSP Enables Prefetching Task Data



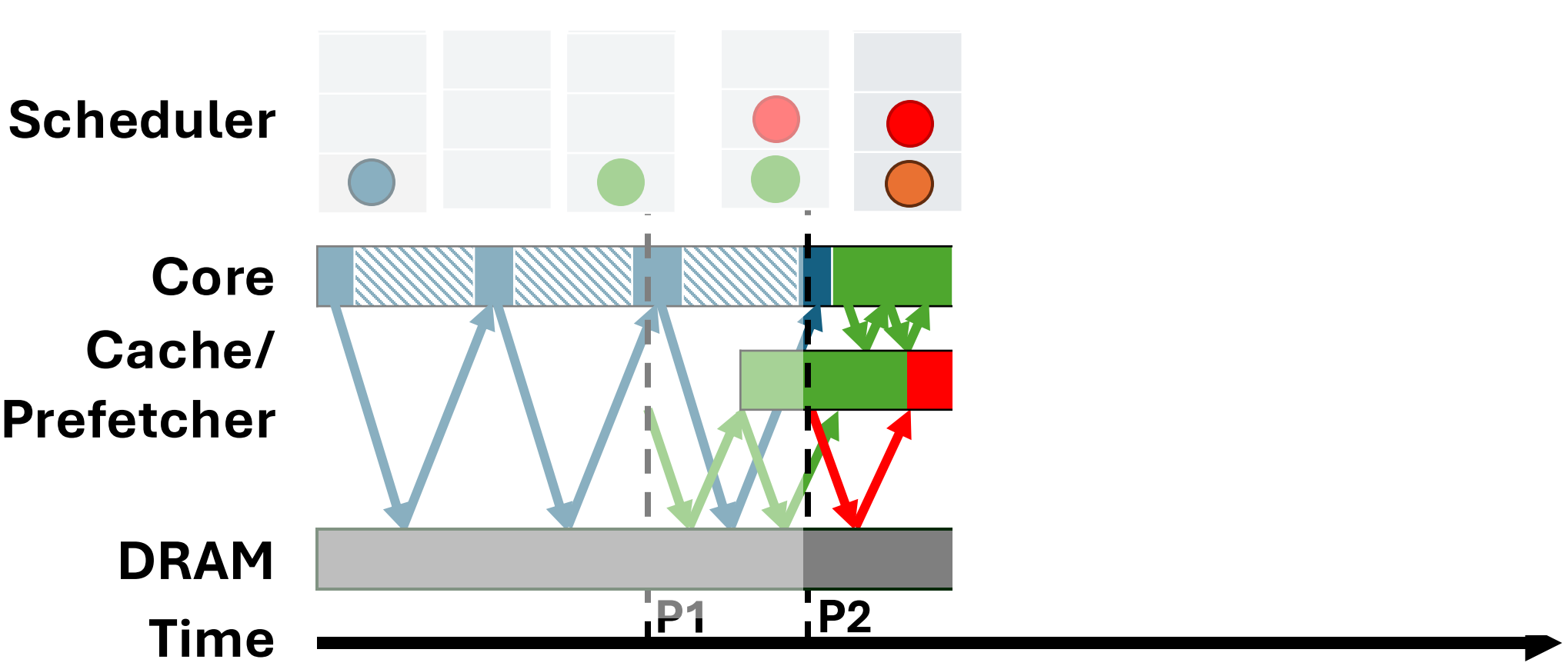


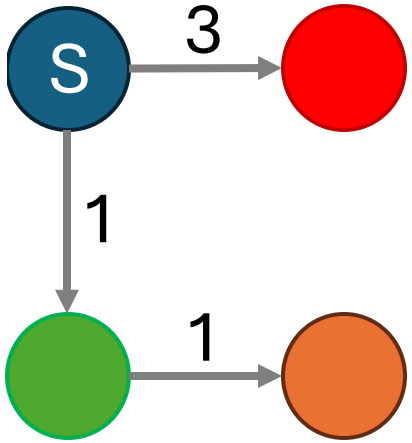
TSP Enables Prefetching Task Data



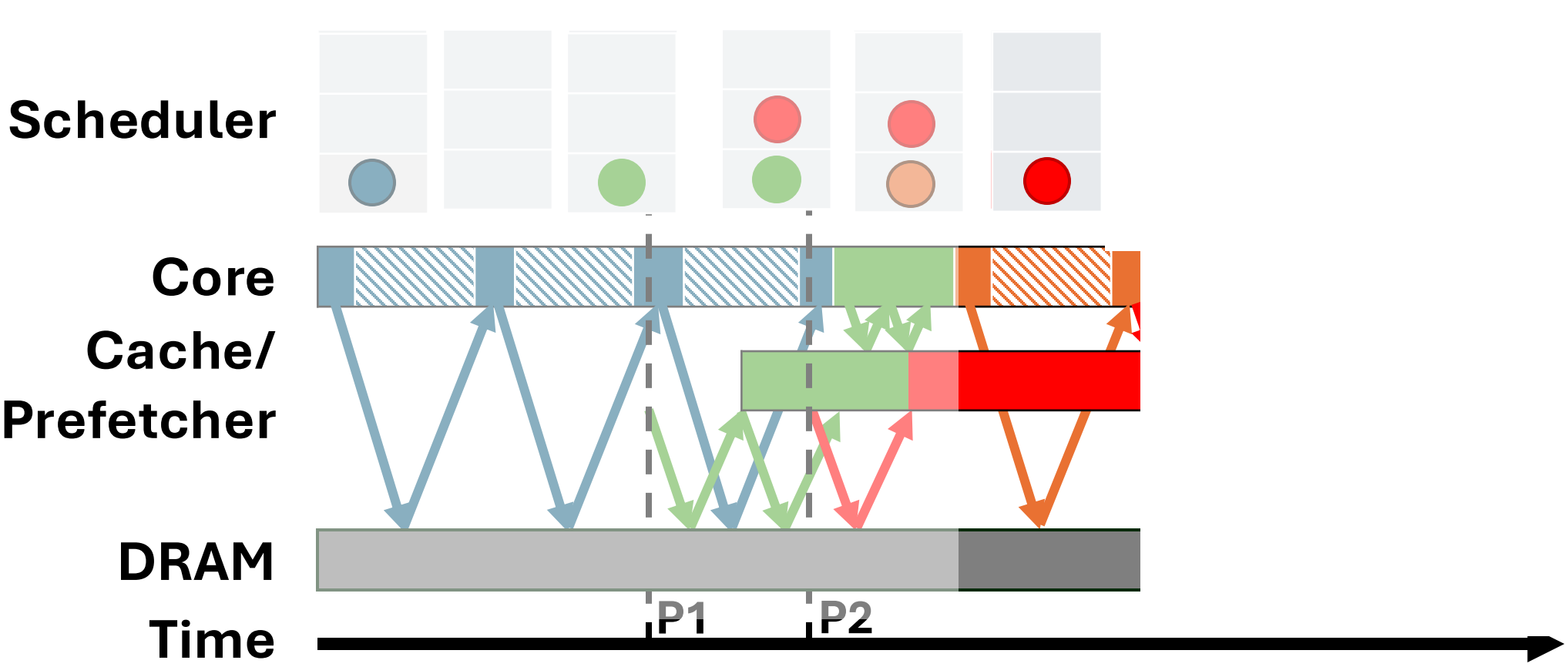


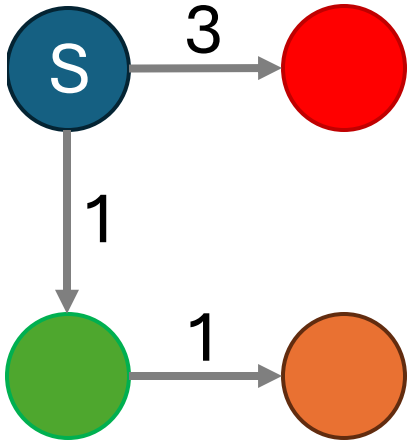
TSP Enables Prefetching Task Data



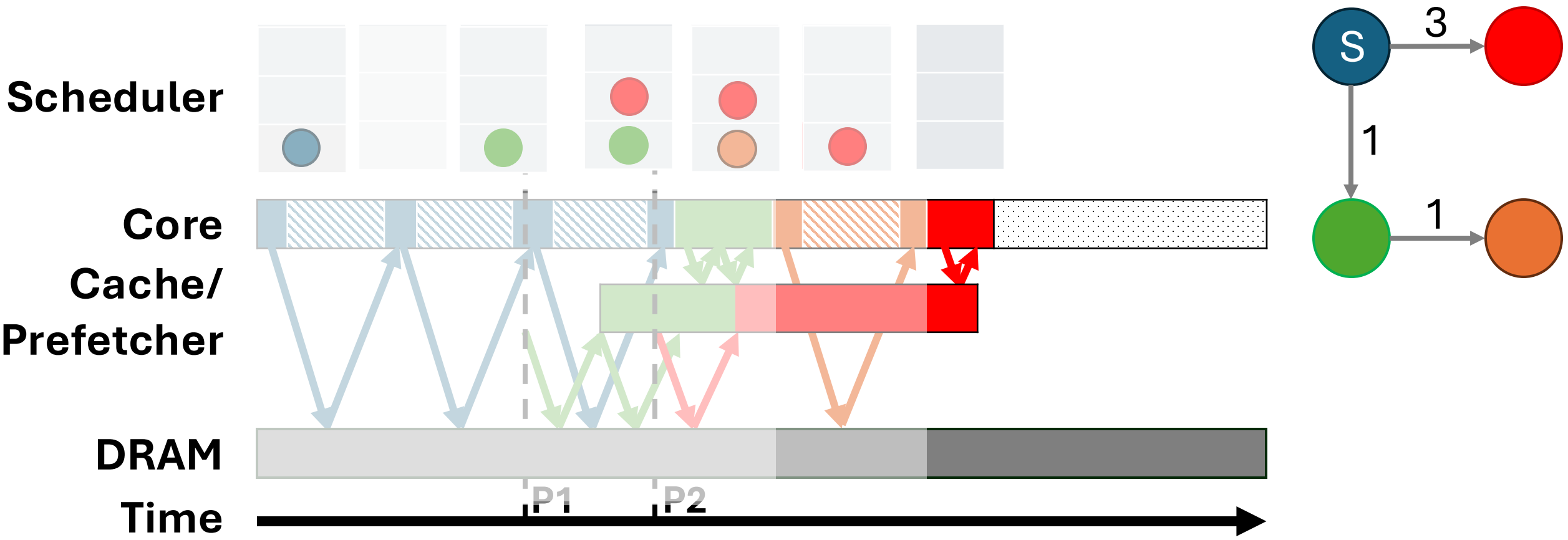


TSP Enables Prefetching Task Data

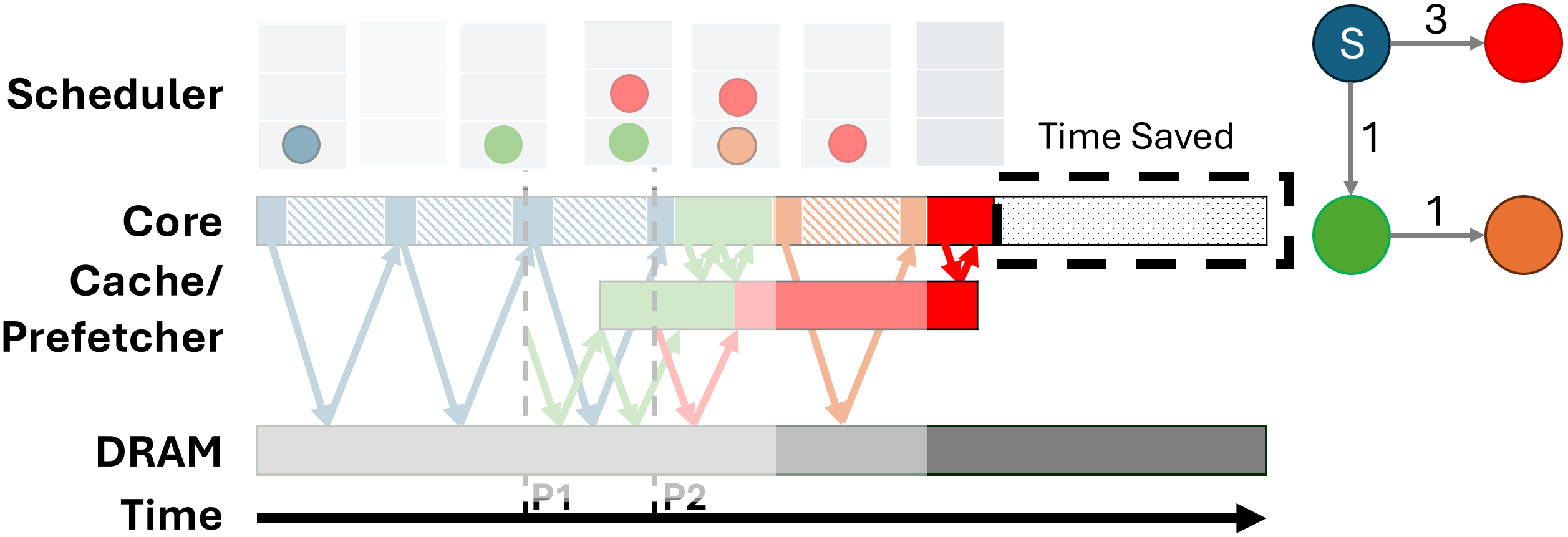




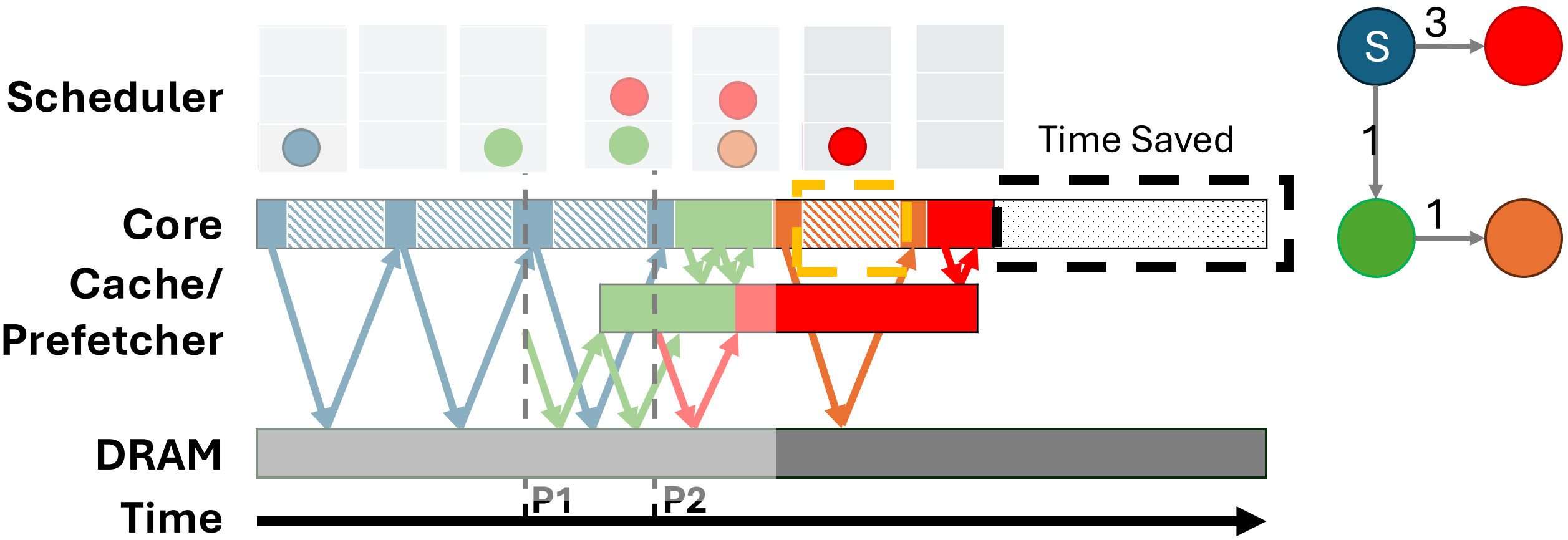
TSP Enables Prefetching Task Data



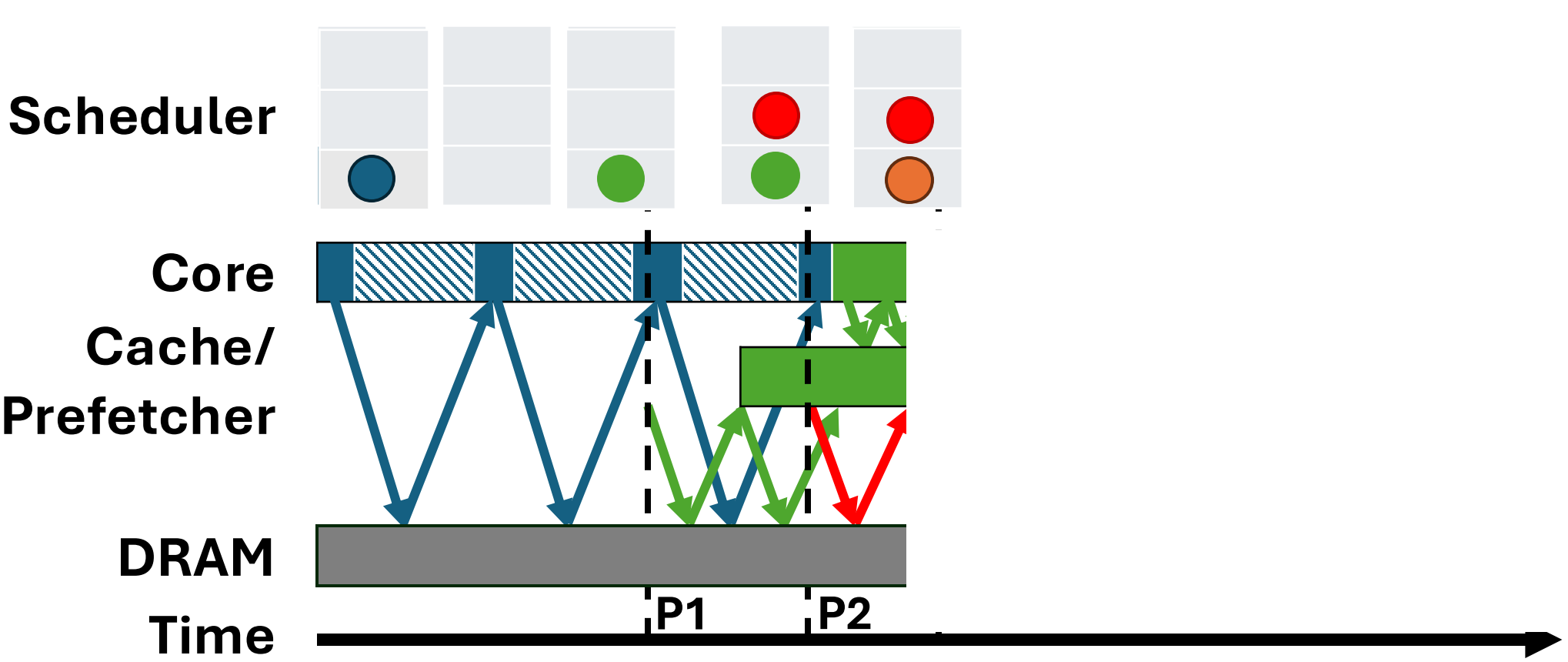
TSP Enables Prefetching Task Data

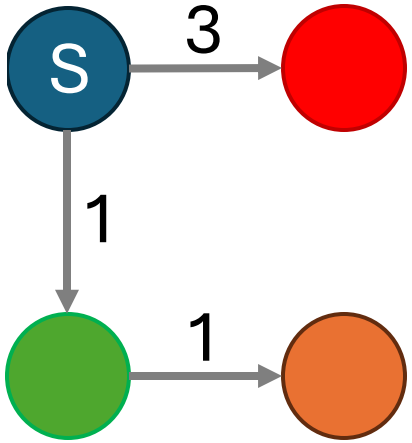


TSP Enables Prefetching Task Data

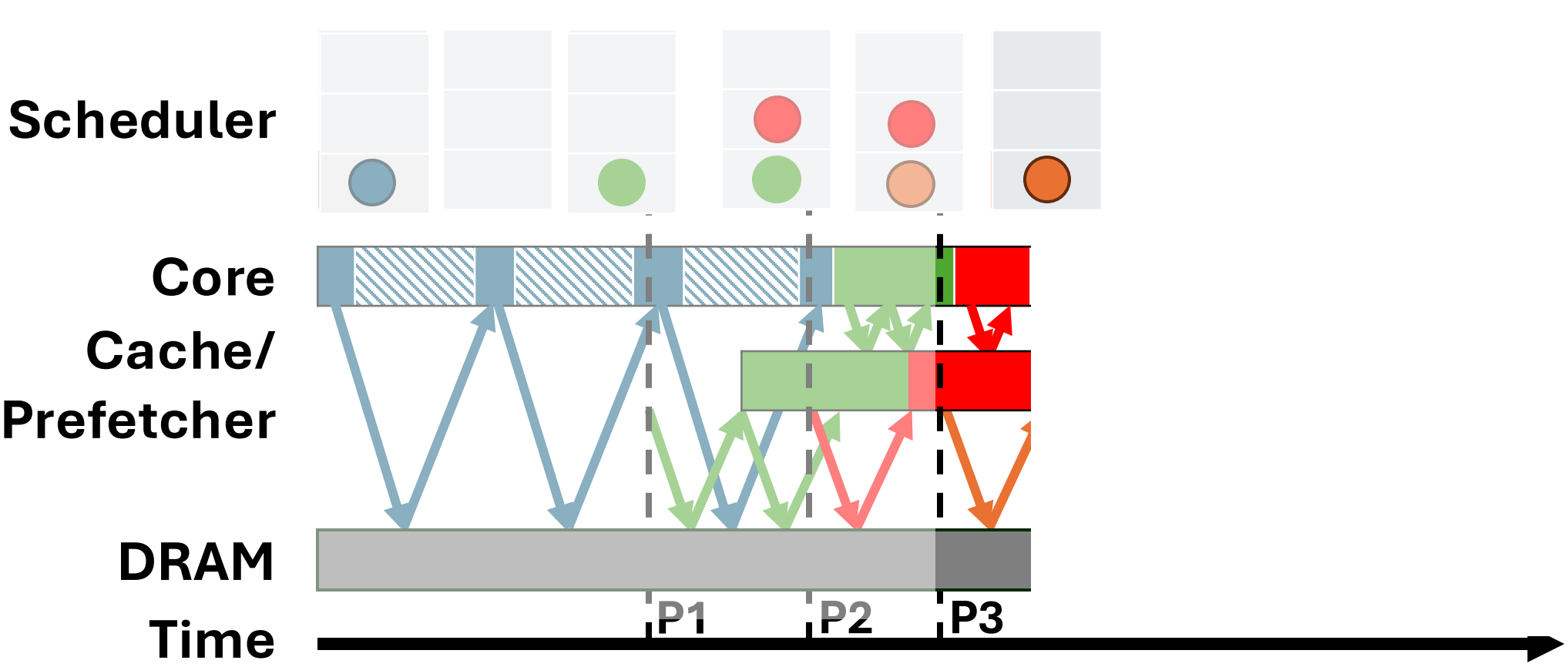


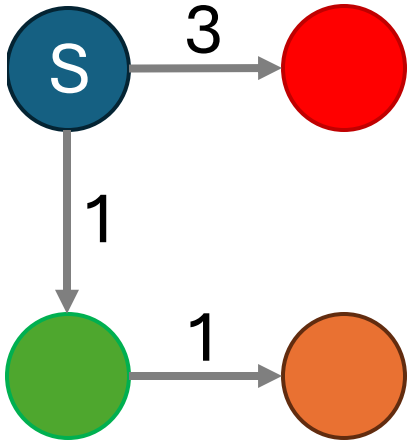
MRS adjusts the task schedule for cache hits



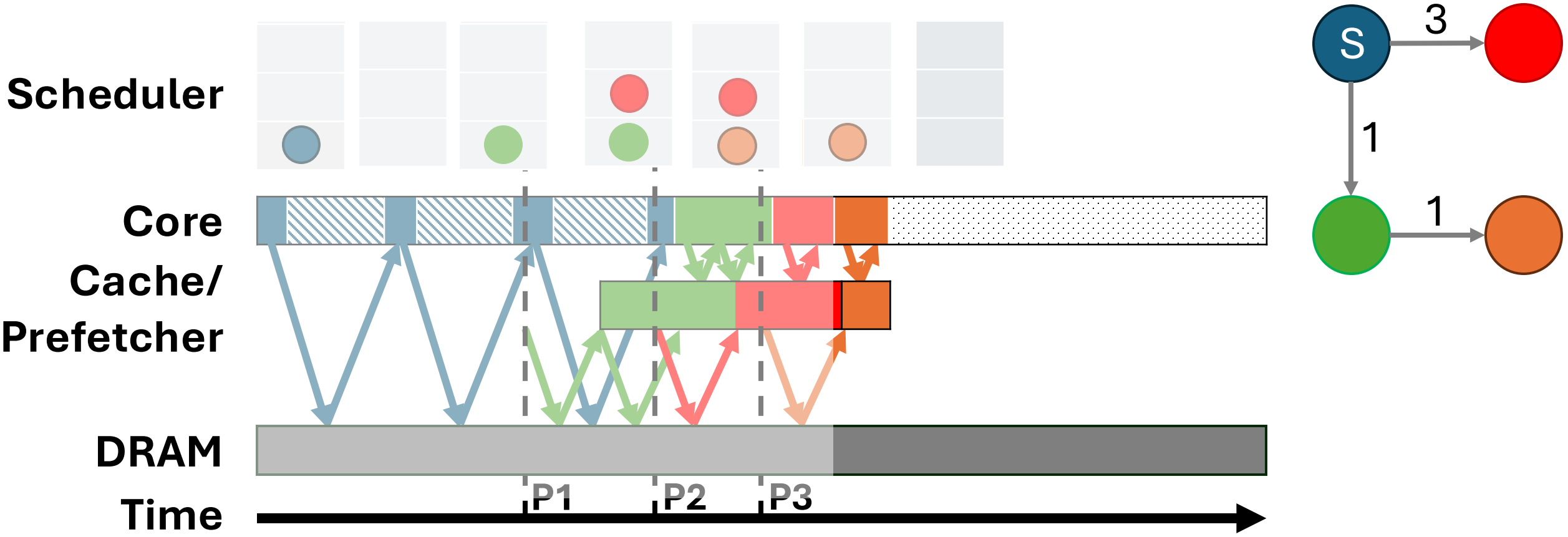


MRS adjusts the task schedule for cache hits

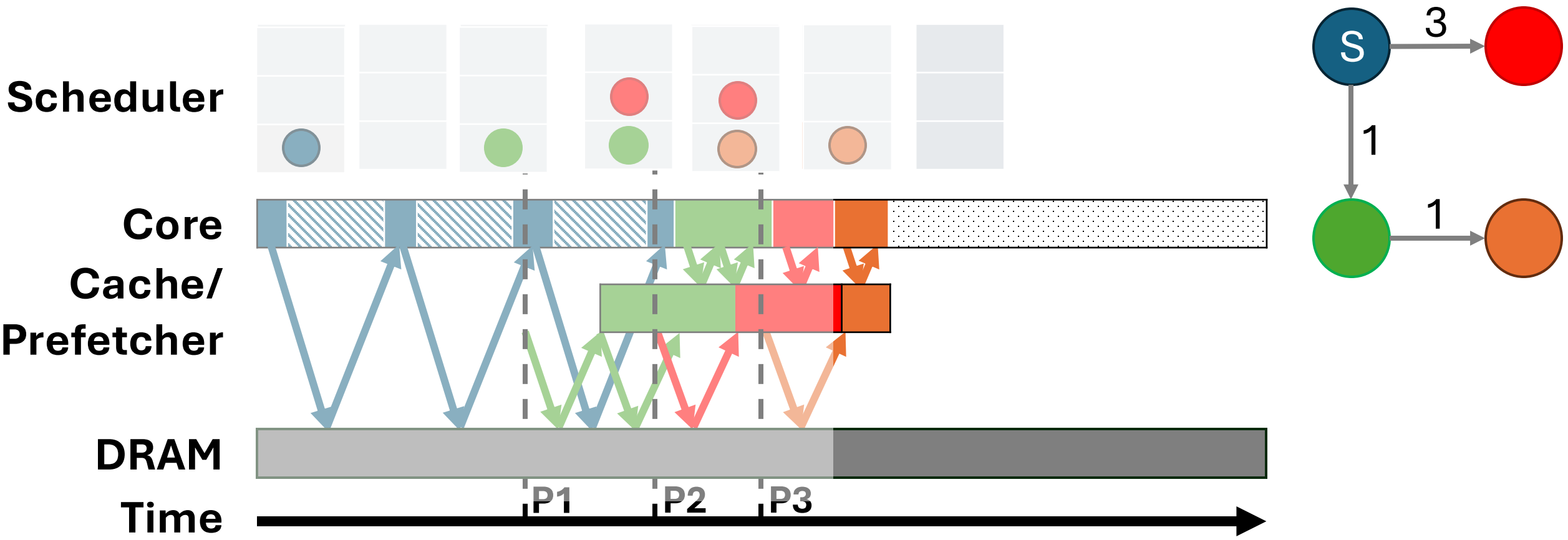




MRS adjusts the task schedule for cache hits

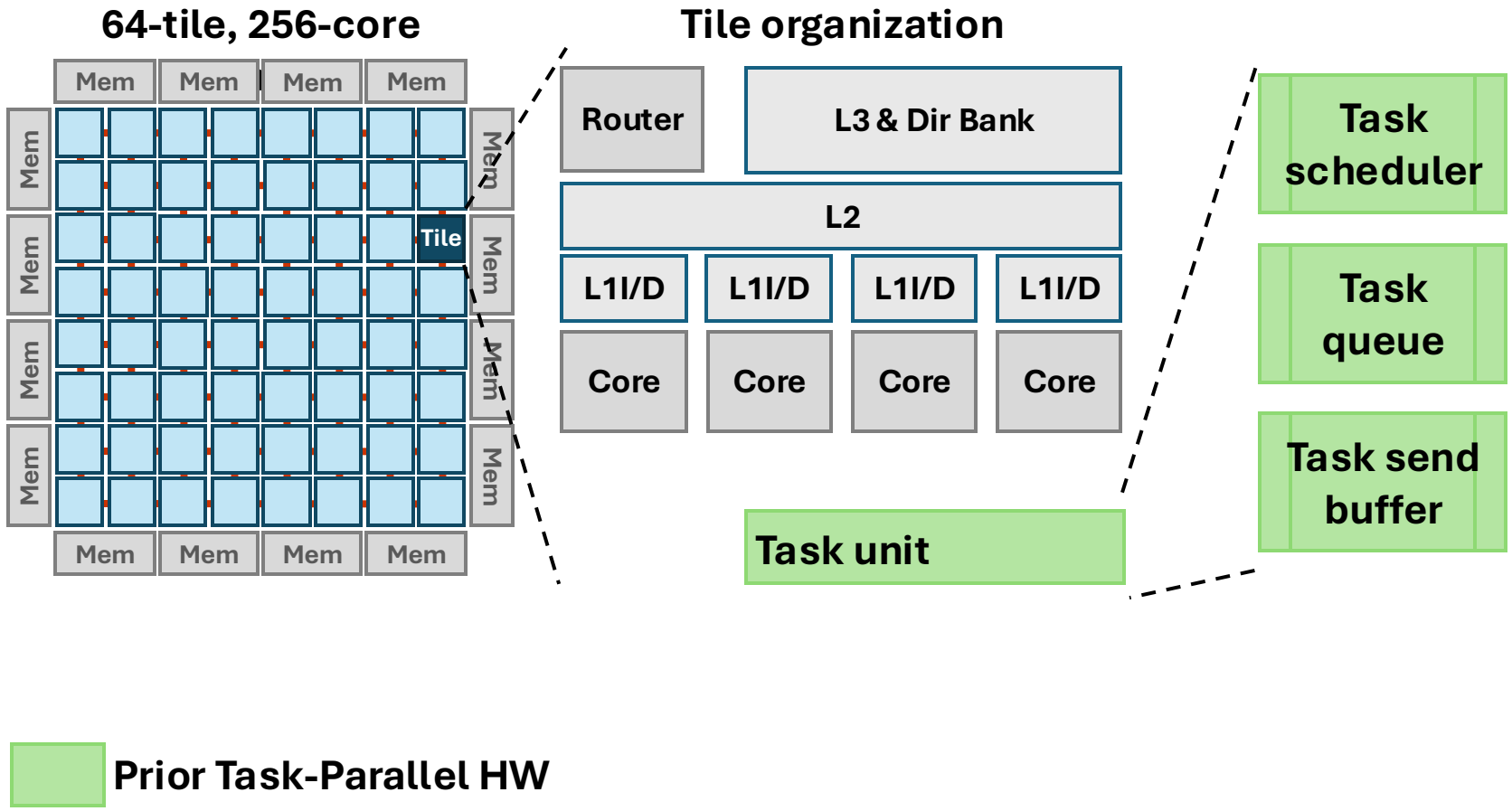


MRS adjusts the task schedule for cache hits

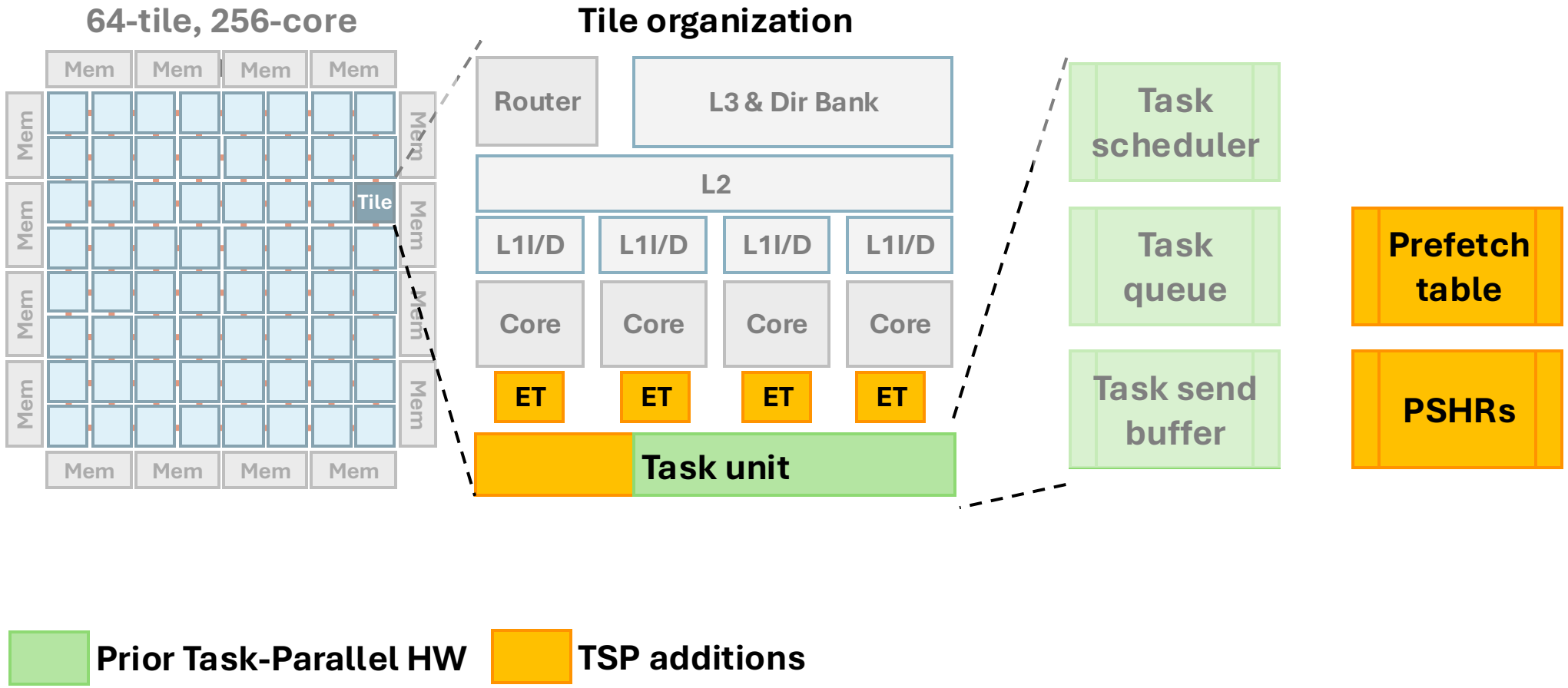


The prefetcher can tell the scheduler which tasks are ready

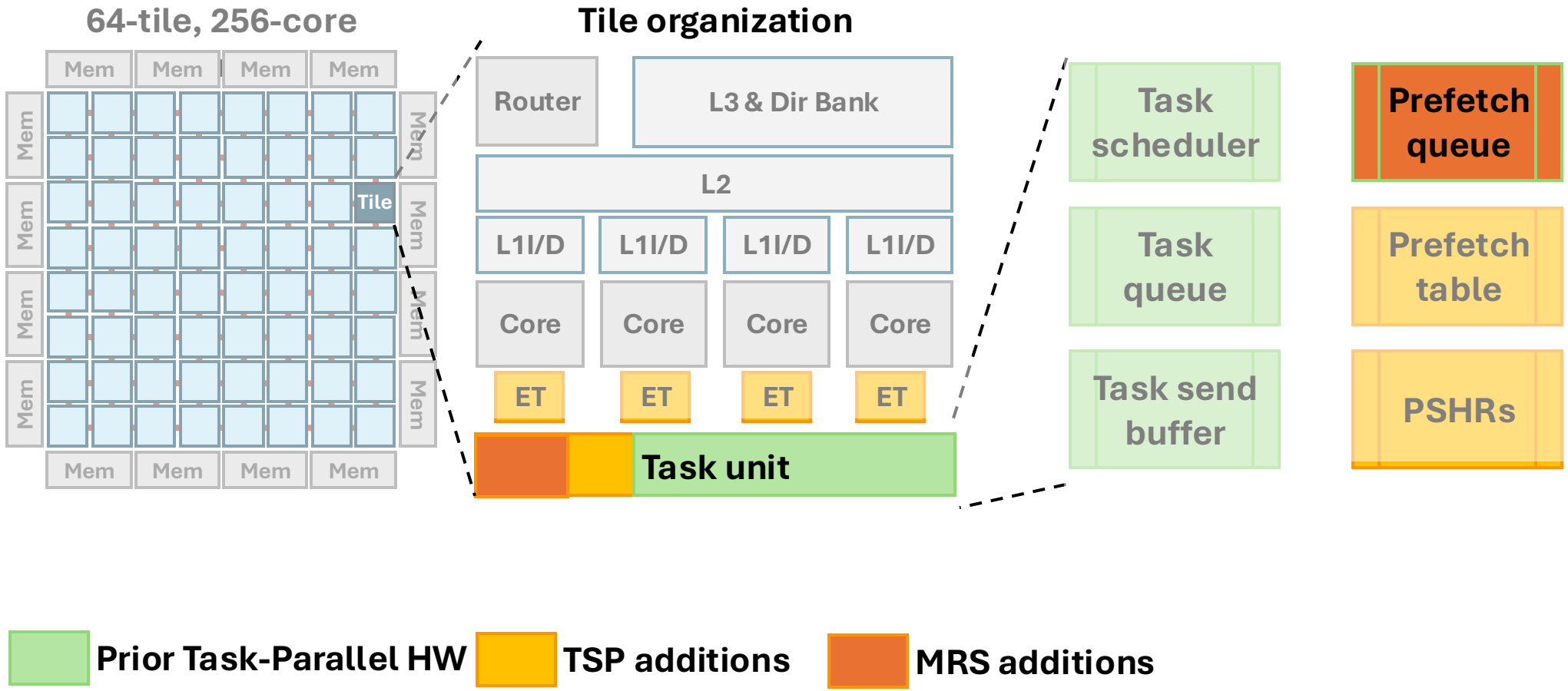
The Task Seeded Prefetcher (TSP)



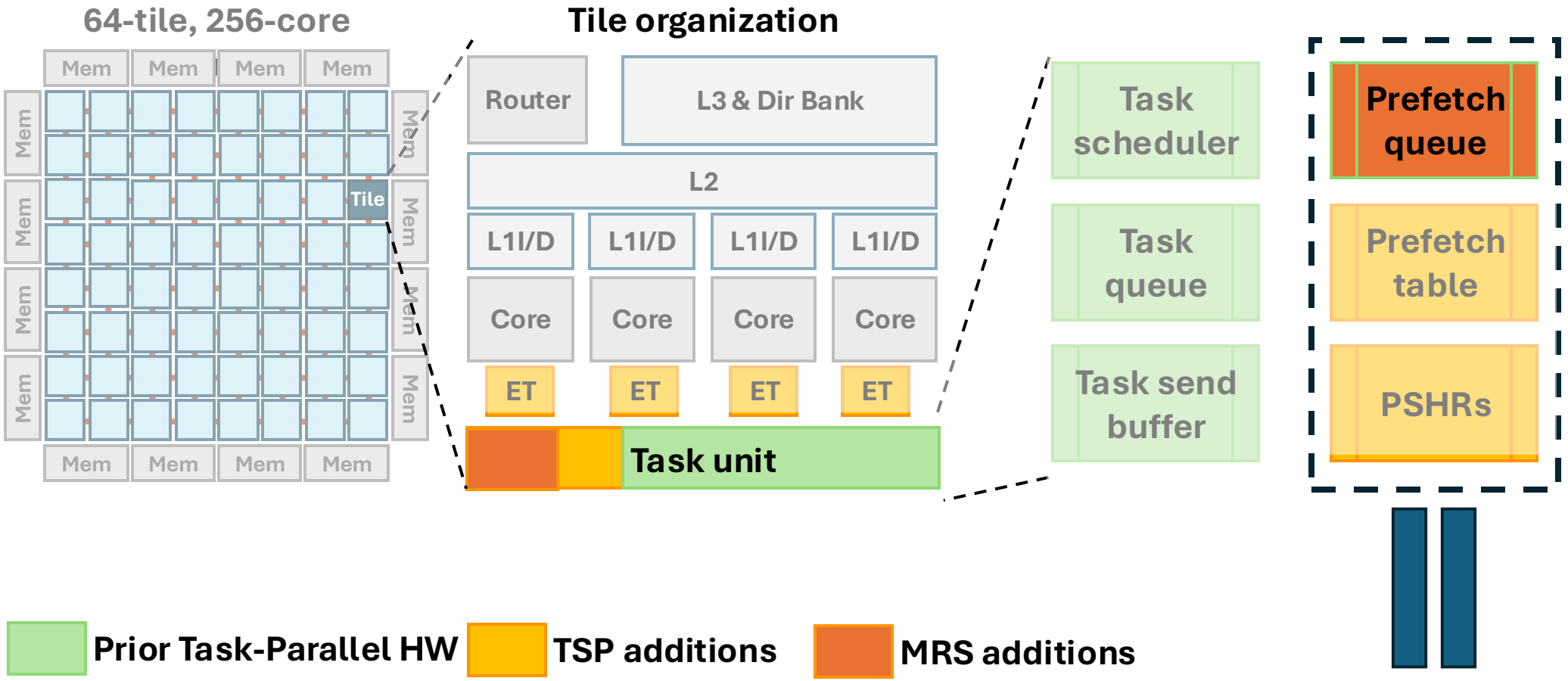
The Task Seeded Prefetcher (TSP)



The Memory Response Scheduler (MRS)



The Memory Response Scheduler (MRS)



~0.1X area of the 1MB L2 cache

See the paper for…

- TSP table layouts
- Recursive indirect prefetching
  - `dsts[nbrs[node]]`
- Inner loop prefetching
  - `for (int nbr=nbrs[node]; nbr<nbrs[node+1]; nbr++)`
- Training TSP
- Energy and Power

Evaluation

Methodology

- Open-Source Swarm simulator[1]
- 13 Task Parallel Benchmarks
  - 22 benchmark-input pairs
  - Graphs, Optimization, Machine Learning, Statistical Inference
- 4 Schedulers X 2 Cache sizes

1: https://github.com/SwarmArch/sim

Methodology

- Open-Source Swarm simulator[1]
- 13 Task Parallel Benchmarks
  - 22 benchmark-input pairs
  - Graphs, Optimization, Machine Learning, Statistical Inference
- 4 Schedulers X 2 Cache sizes

	Non-Speculative	Speculative
Random	RELD [Yesil et al. SC'19]	Swarm [Jeffrey et al. MICRO'15]
Spatial	Dalorex [Orenes-Vera et al. HPCA`23]	Spatial Hints [Jeffrey et al. MICRO'16]

1: https://github.com/SwarmArch/sim

Methodology

- Open-Source Swarm simulator[1]
- 13 Task Parallel Benchmarks
  - 22 benchmark-input pairs
  - Graphs, Optimization, Machine Learning, Statistical Inference
- 4 Schedulers X 2 Cache sizes

	Non-Speculative	Speculative
Random	RELD [Yesil et al. SC'19]	Swarm [Jeffrey et al. MICRO'15]
Spatial	Dalorex [Orenes-Vera et al. HPCA`23]	Spatial Hints [Jeffrey et al. MICRO'16]

This Talk

1: https://github.com/SwarmArch/sim

Methodology

- Open-Source Swarm simulator[1]
- 13 Task Parallel Benchmarks
  - 22 benchmark-input pairs
  - Graphs, Optimization, Machine Learning, Statistical Inference
- 4 Schedulers X 2 Cache sizes

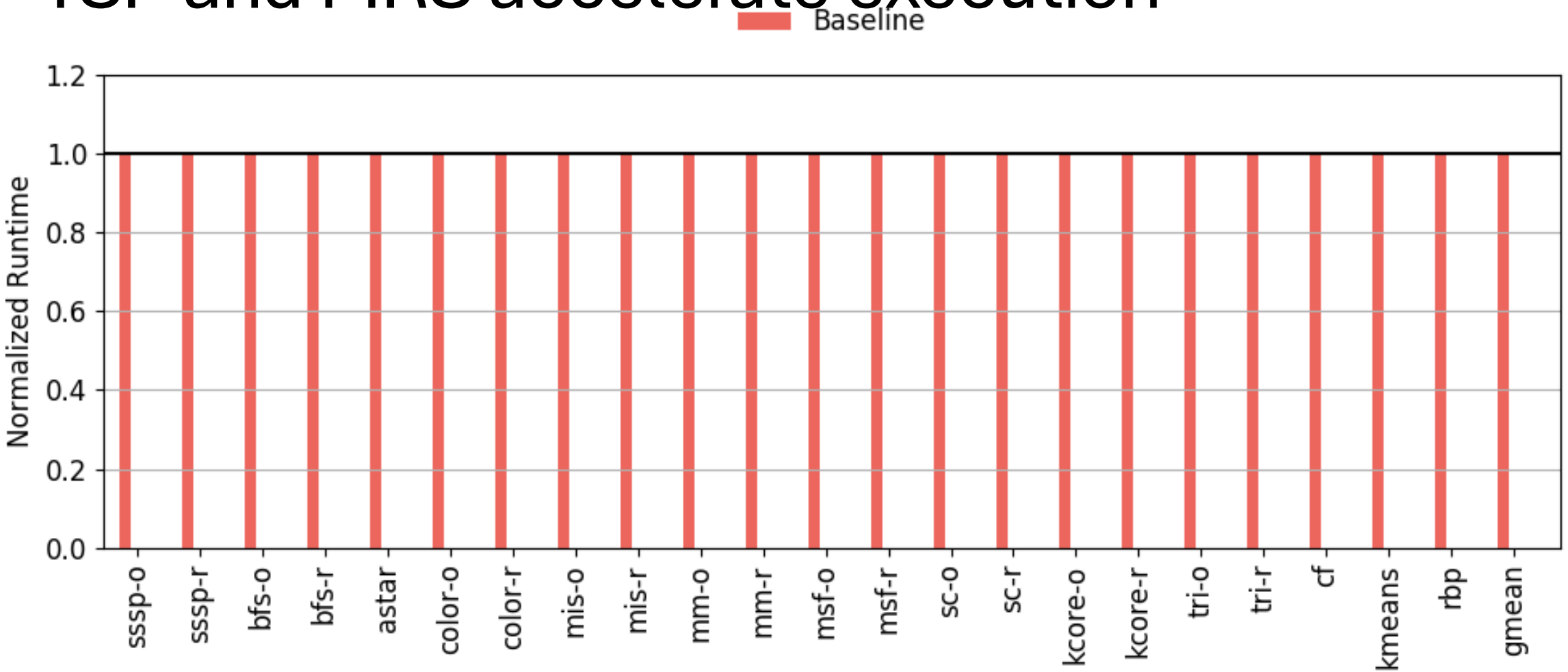
	Non-Speculative	Speculative
Random	RELD [Yesil et al. SC'19]	Swarm [Jeffrey et al. MICRO'15]
Spatial	Dalorex [Orenes-Vera et al. HPCA`23]	Spatial Hints [Jeffrey et al. MICRO'16]

See the paper

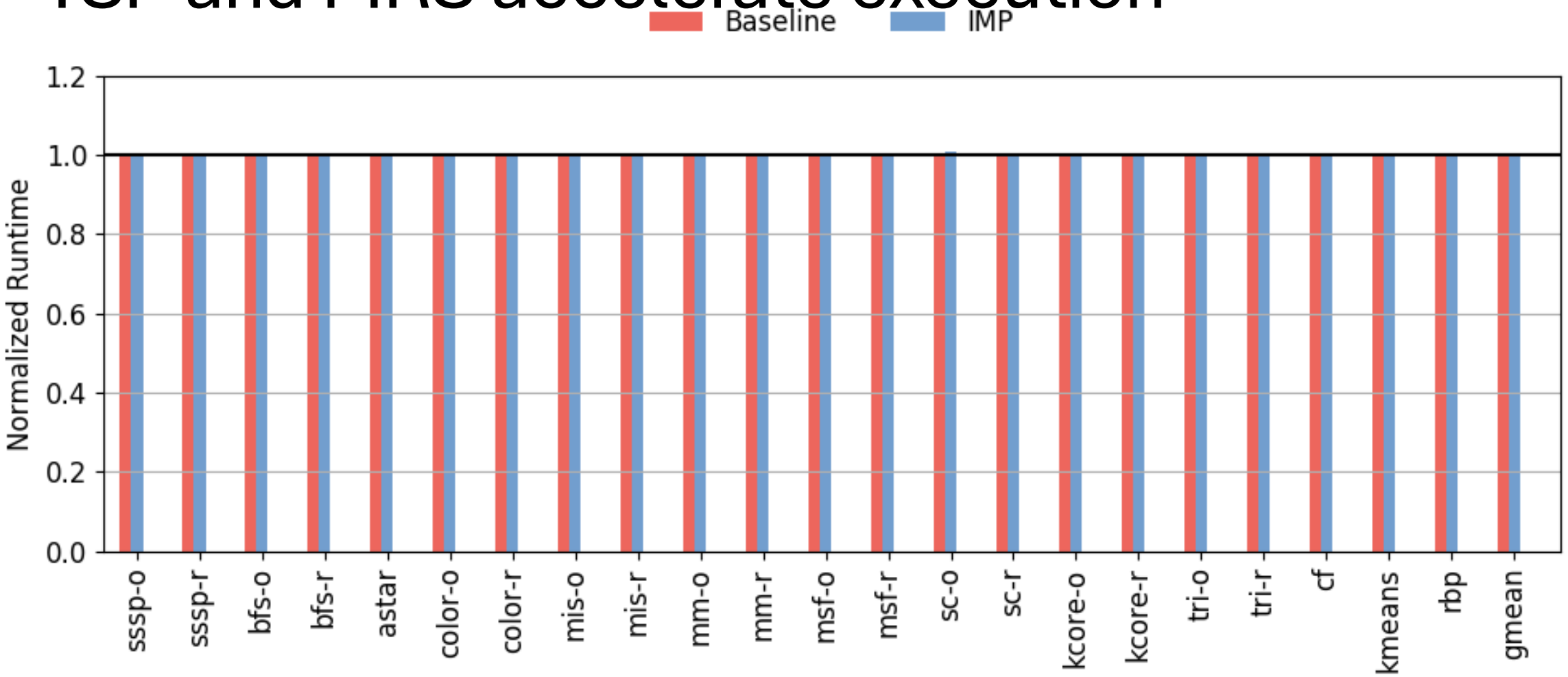
This Talk

1: https://github.com/SwarmArch/sim

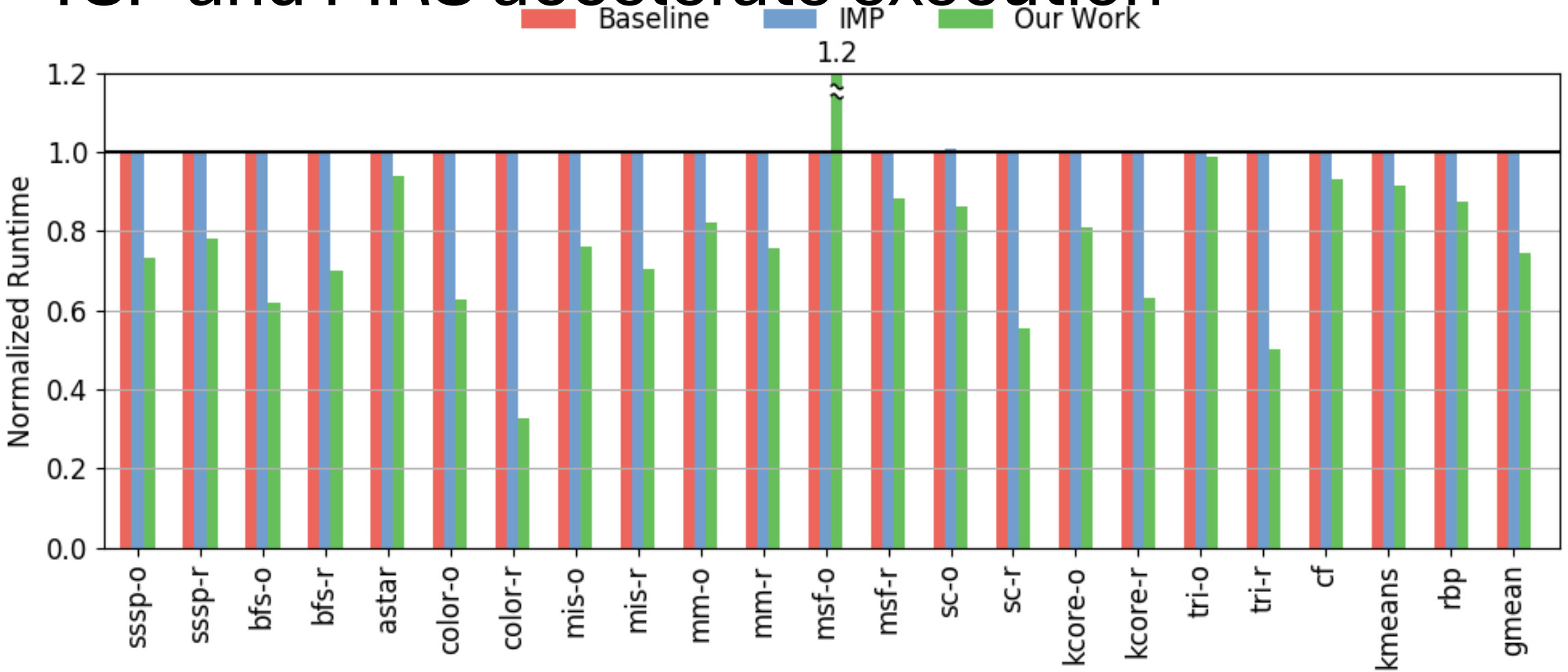
TSP and MRS accelerate execution



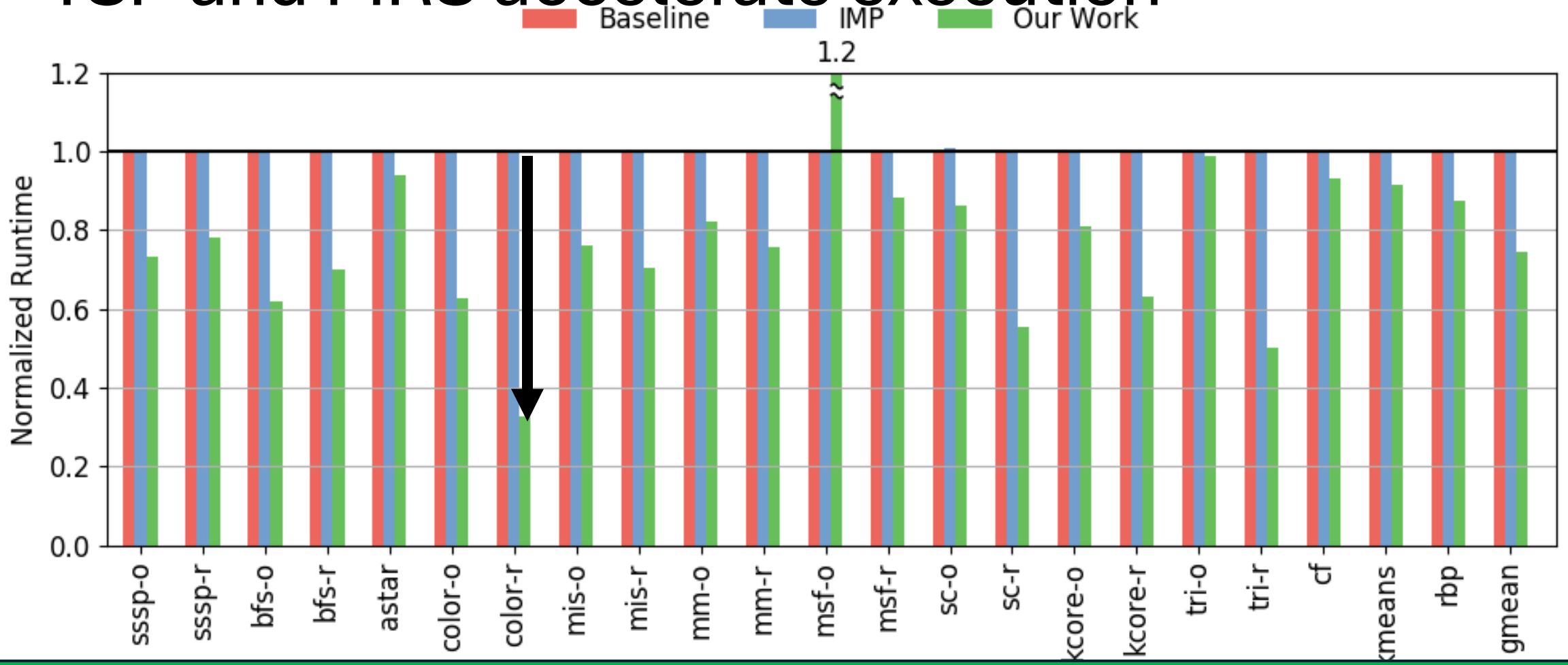
TSP and MRS accelerate execution



TSP and MRS accelerate execution

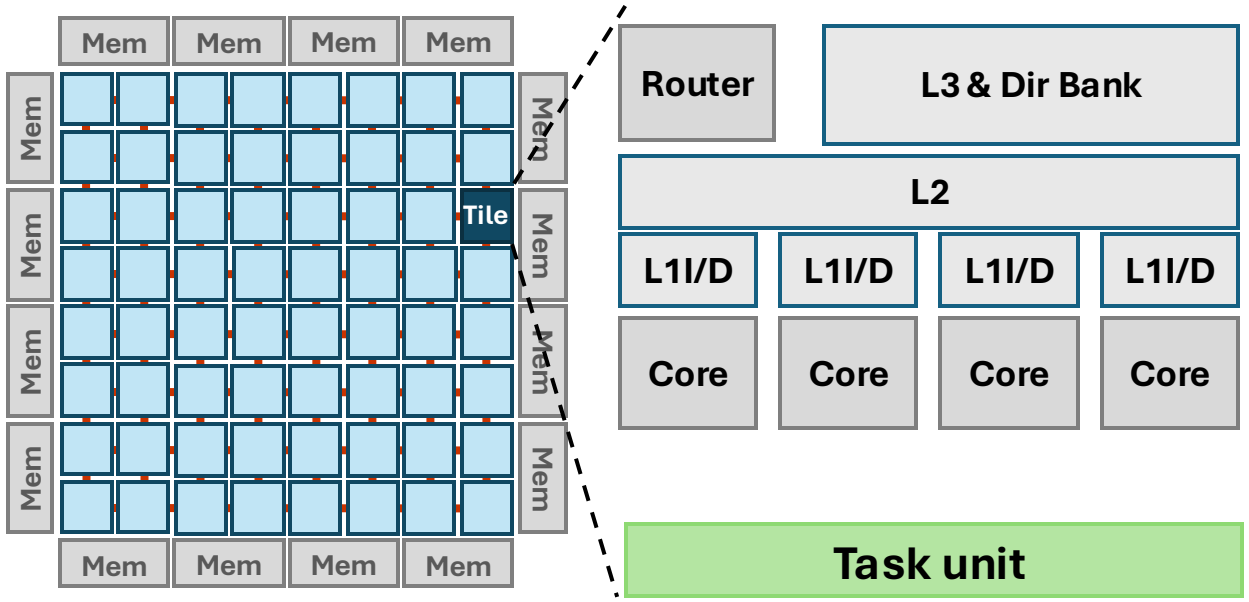


TSP and MRS accelerate execution

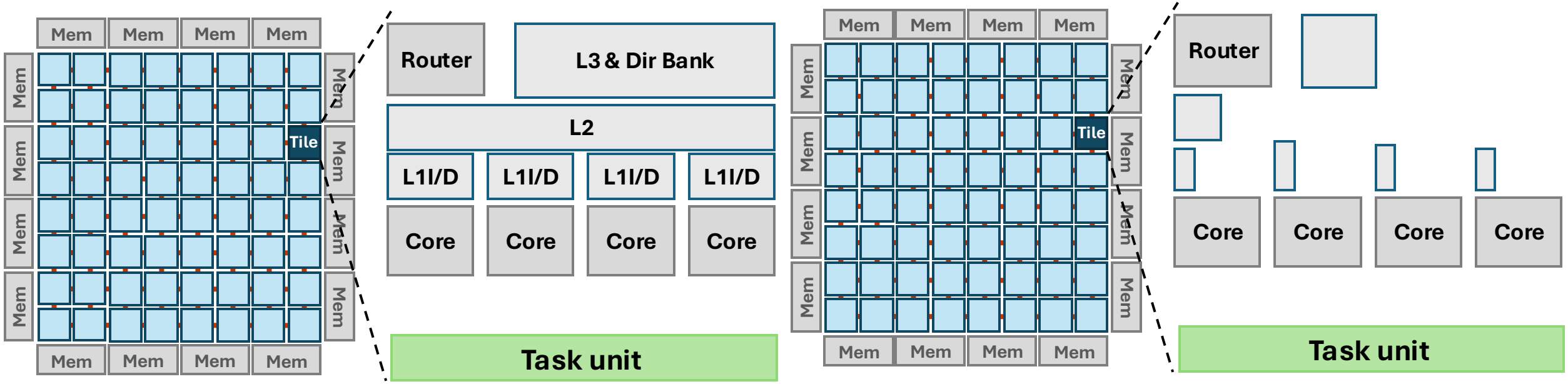


TSP+MRS Provide speedups up to 3.1X

Cutting down the cache sizes

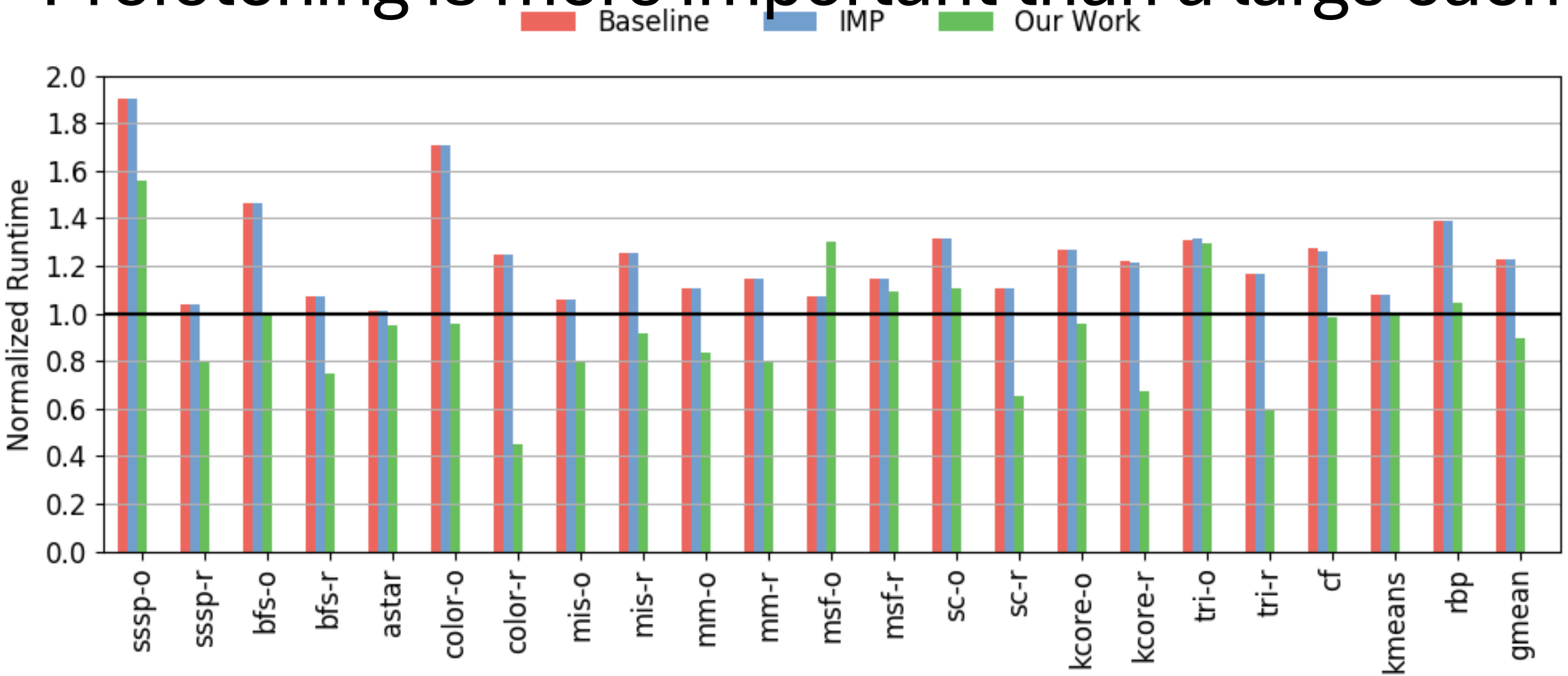


Cutting down the cache sizes

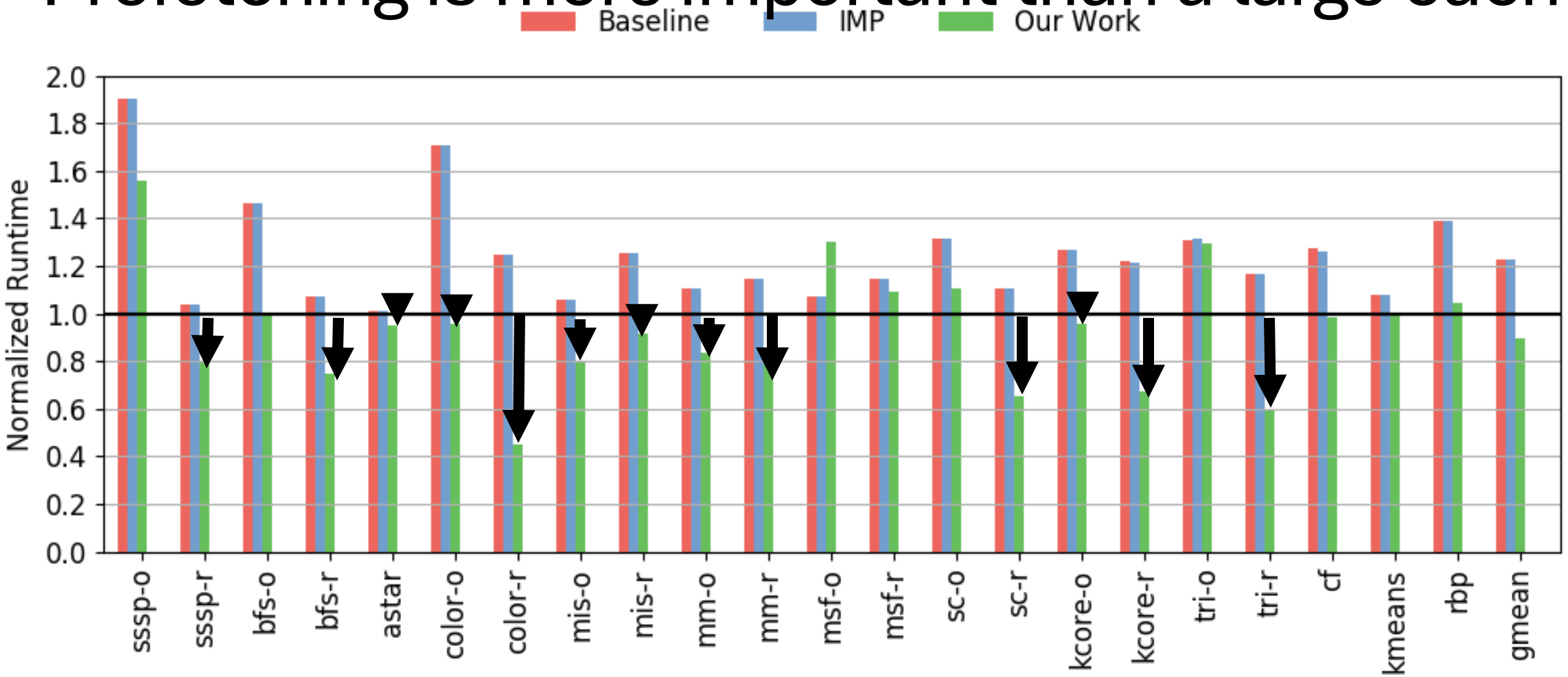


1MB->64KB

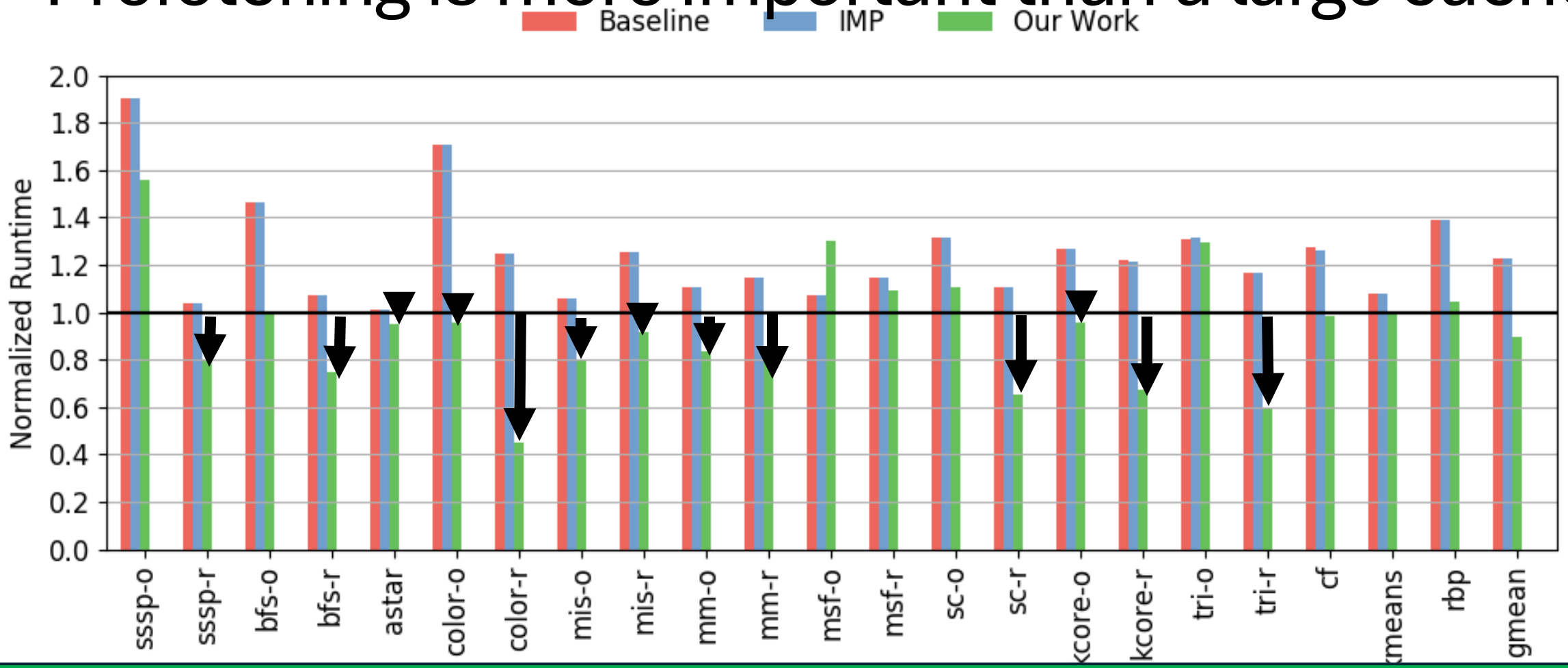
Prefetching is more important than a large cache



Prefetching is more important than a large cache

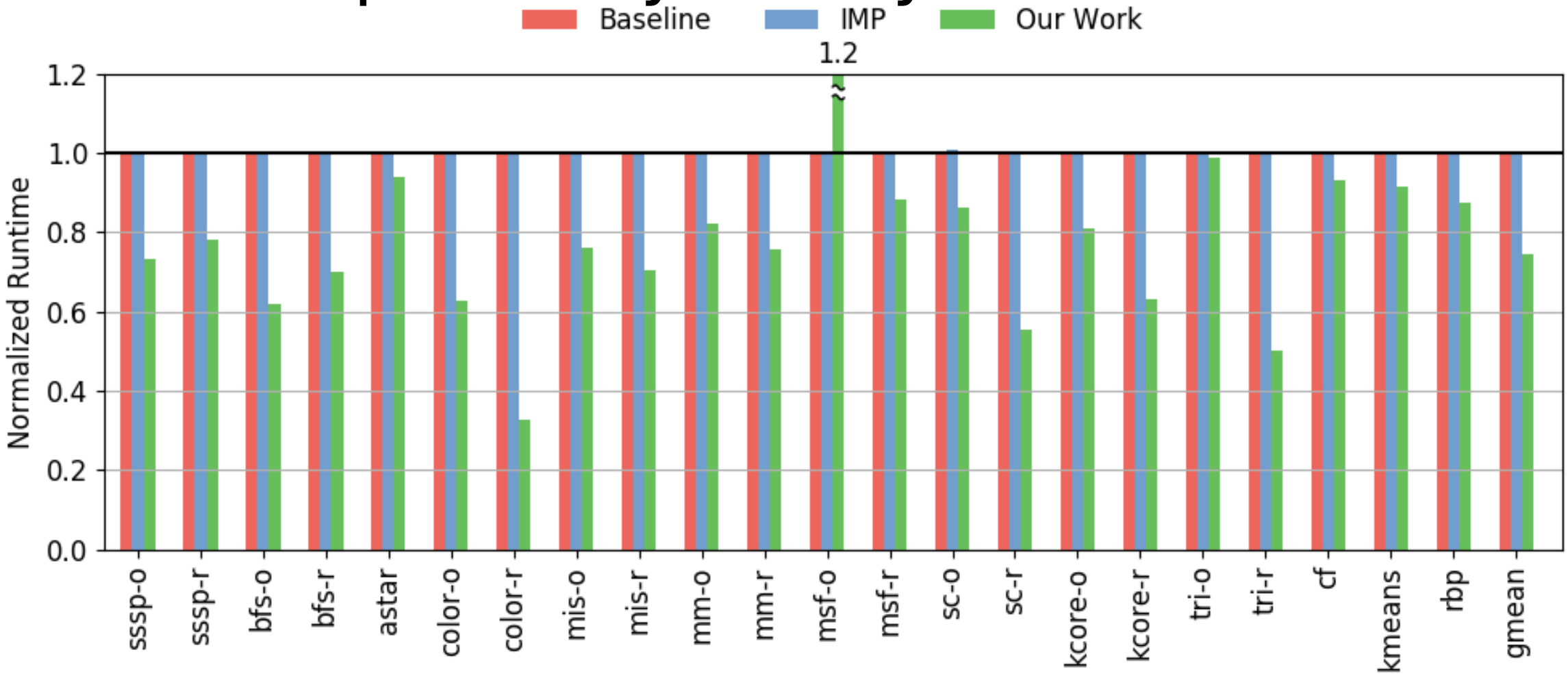


Prefetching is more important than a large cache

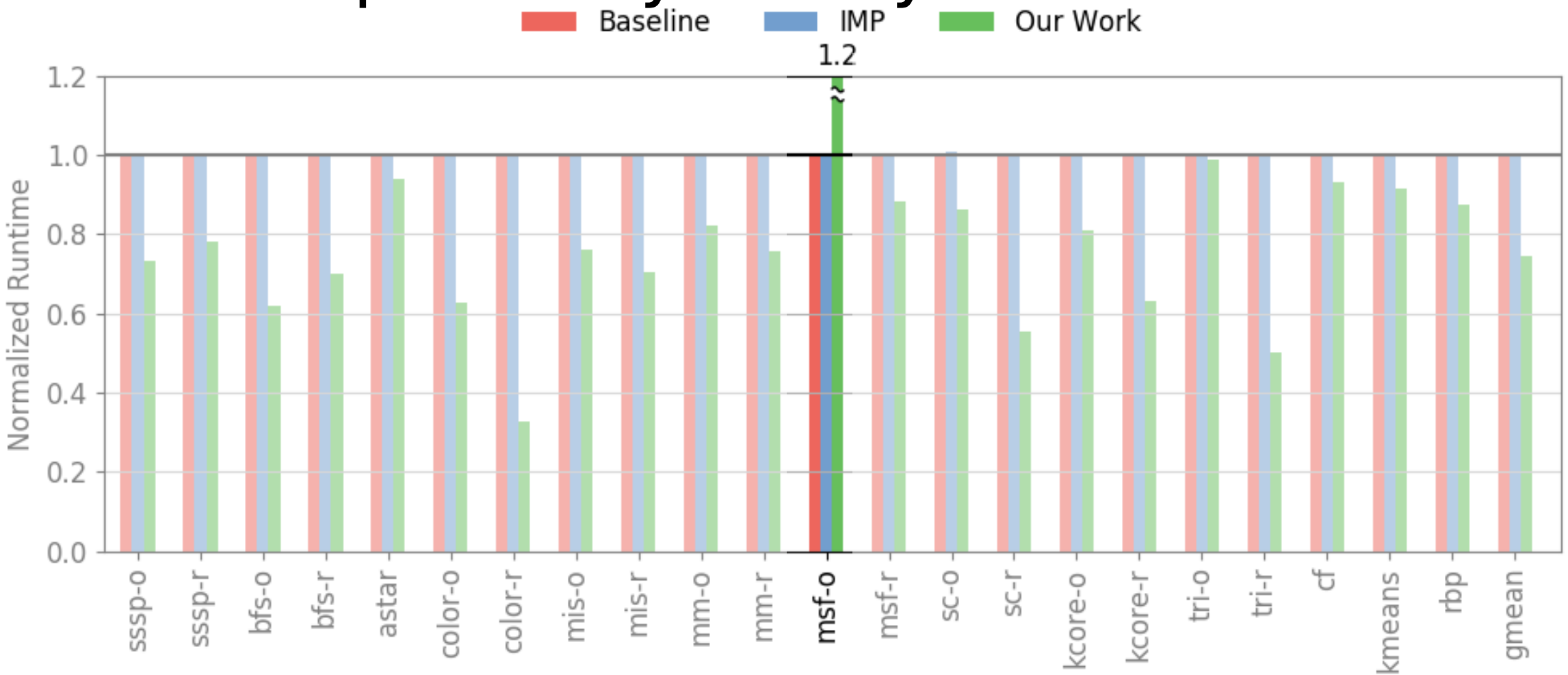


We can trade >90% of the cache for prefetching and come out ahead

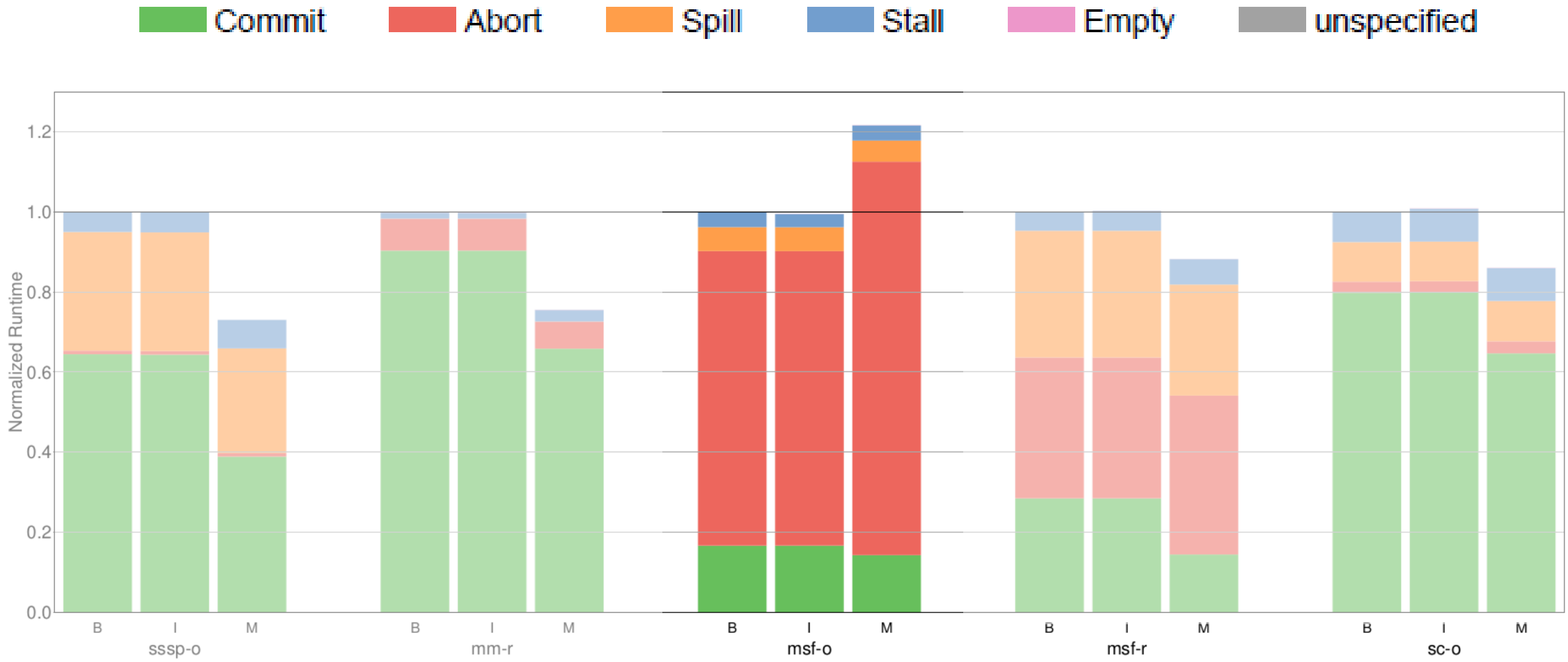
You can’t prefetch your way out of a bad schedule



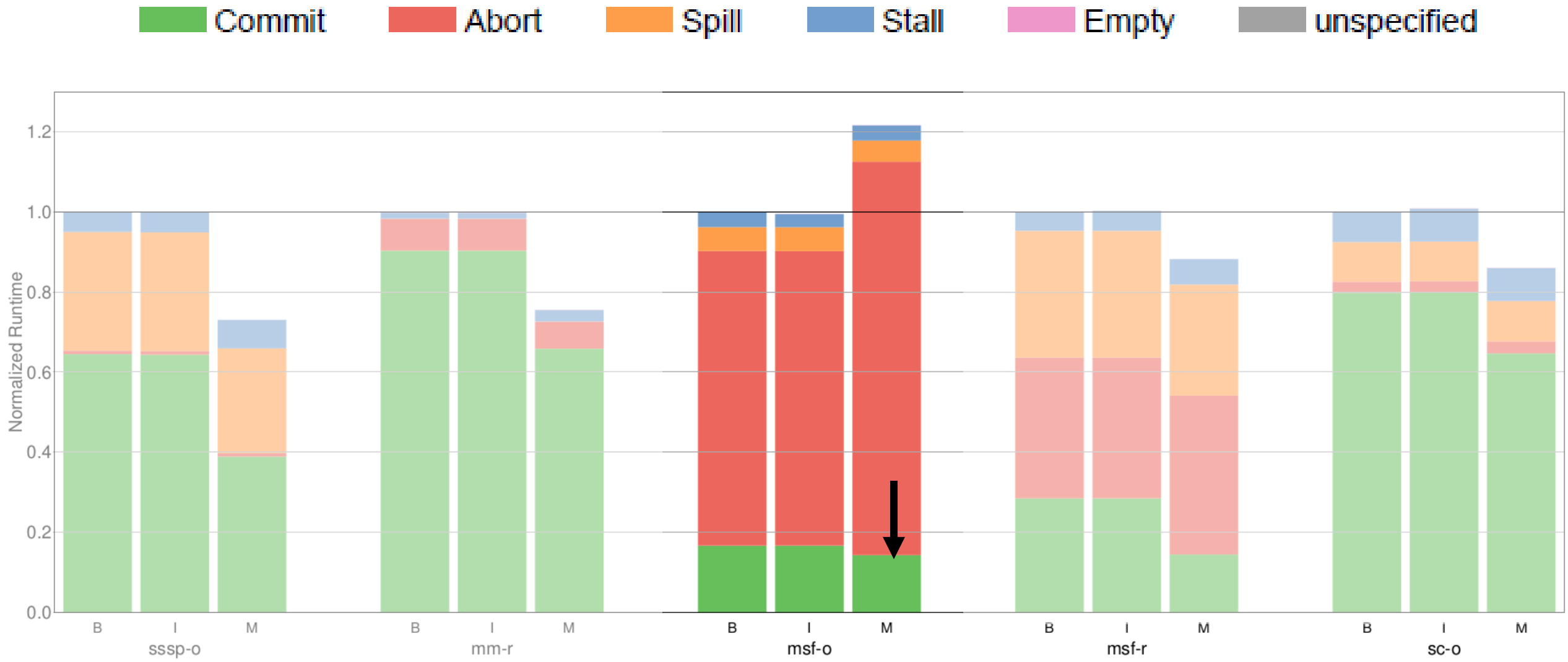
You can’t prefetch your way out of a bad schedule



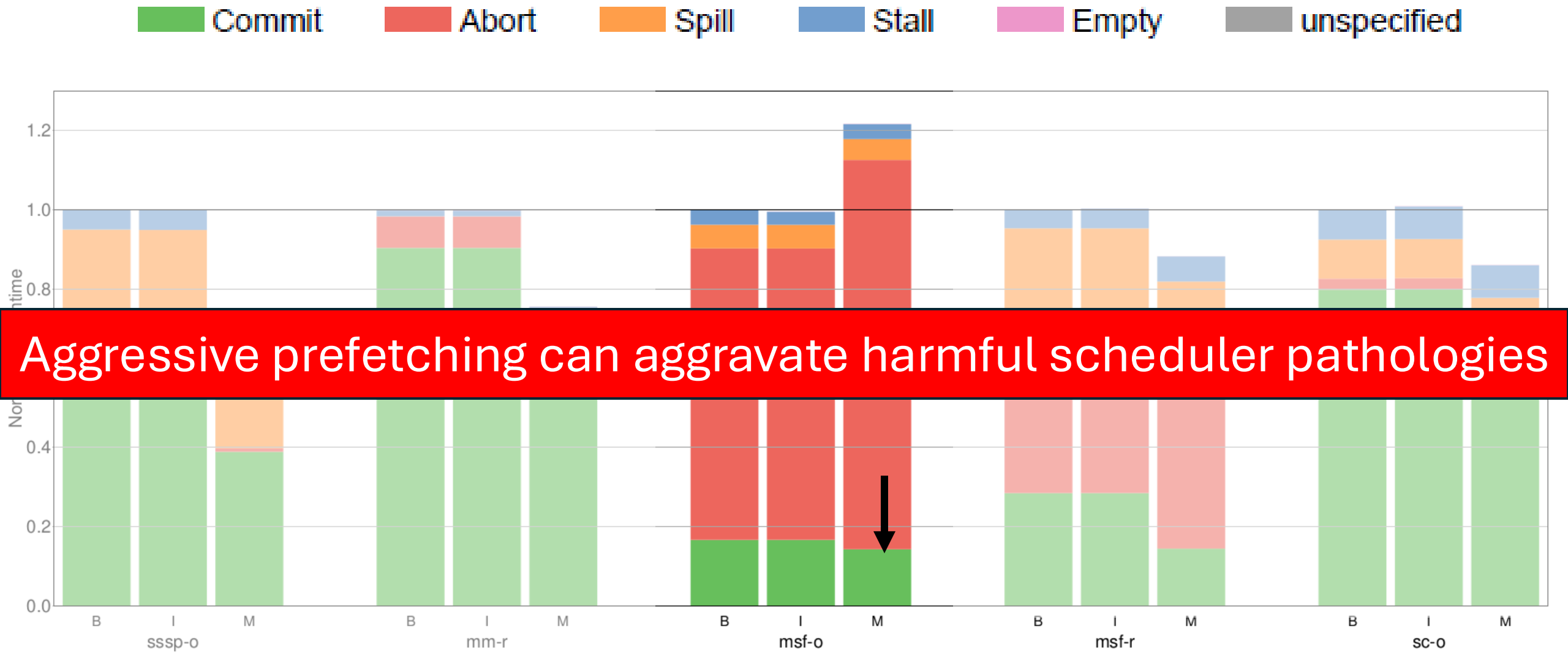
You can’t prefetch your way out of a bad schedule



You can’t prefetch your way out of a bad schedule



You can’t prefetch your way out of a bad schedule



Aggressive prefetching can aggravate harmful scheduler pathologies

Conclusions

- HW task schedulers enable 100X speedups over SW parallelism
- Prior HW task schedulers interfere with prior prefetchers
- The Task-Seeded Prefetcher uses task arguments to prefetch
- The Memory Response Scheduler adjusts the task schedule
- By working together, TSP and MRS achieve greater performance than either could manage independently

Gilead Posluns Mark C. Jeffrey

UNIVERSITY OF TORONTO