Work Stealing through Partial Asynchronous Delegation

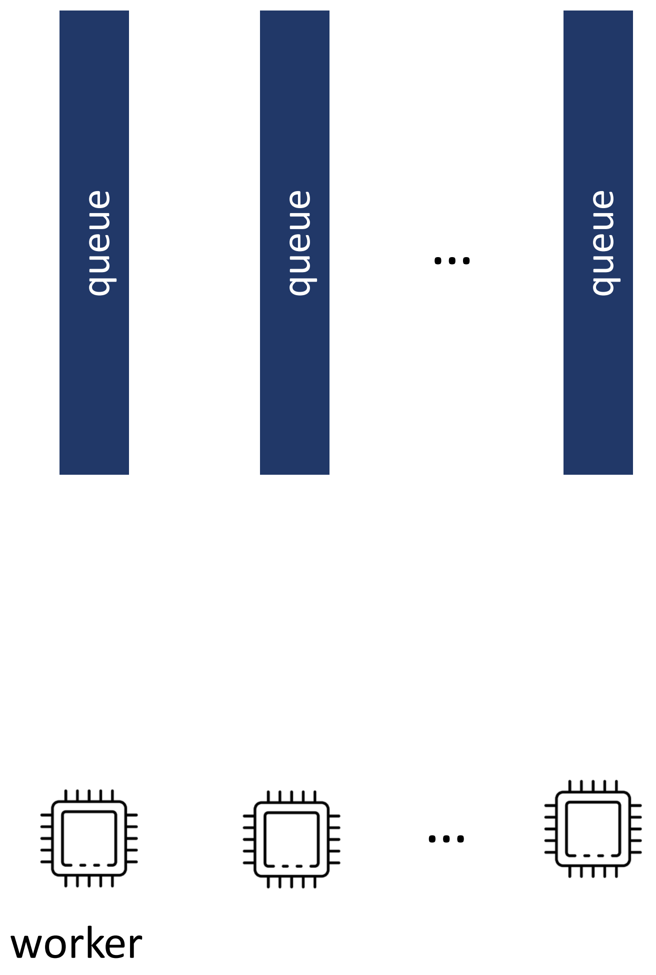
Jiawei Wang[1 2], Yutao Liu[1], Ming Fu[1], Hermann Härtig[2], and Haibo Chen[1 3]

[1] HUAWEI

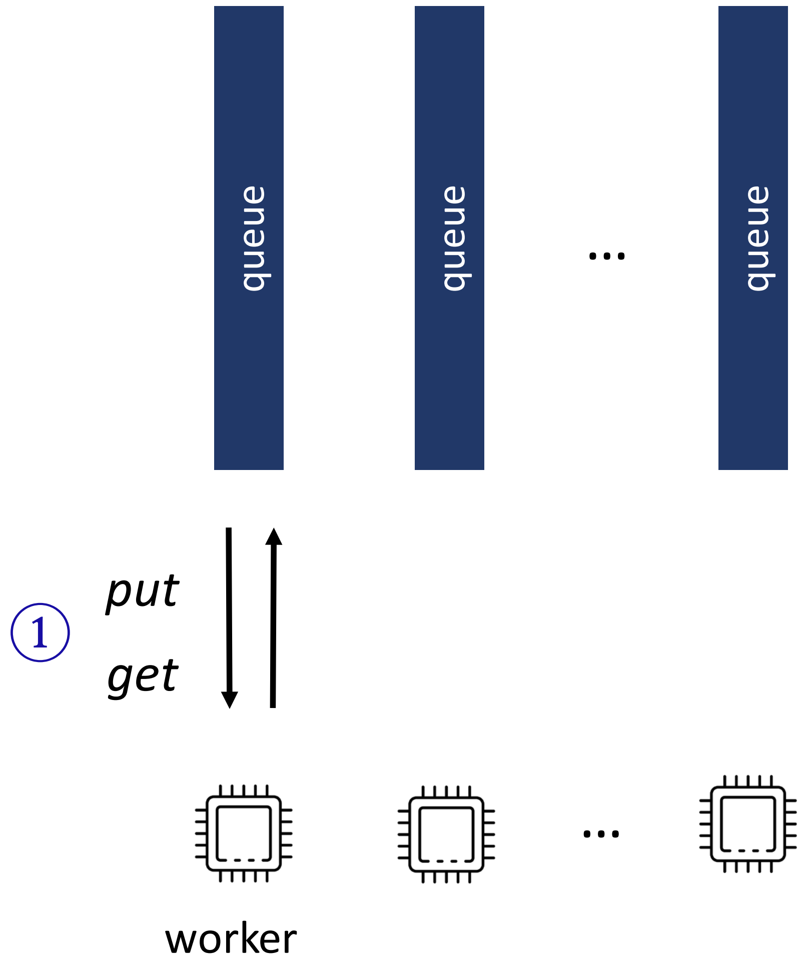
[2] TECHNISCHE UNIVERSITÄT DRESDEN

[3] SHANGHAI JIAO TONG UNIVERSITY 1896

Work Stealing -- Load Balancing for Multi-core Task Processing

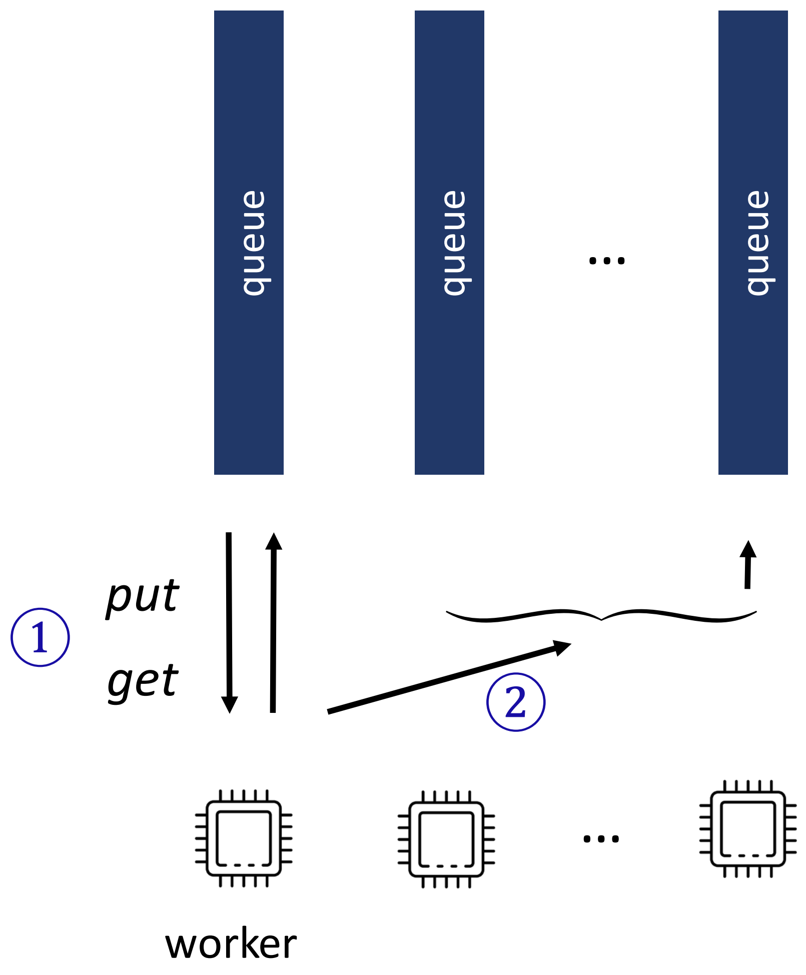


Work Stealing -- Load Balancing for Multi-core Task Processing



① A worker (core) puts on / gets from its queue

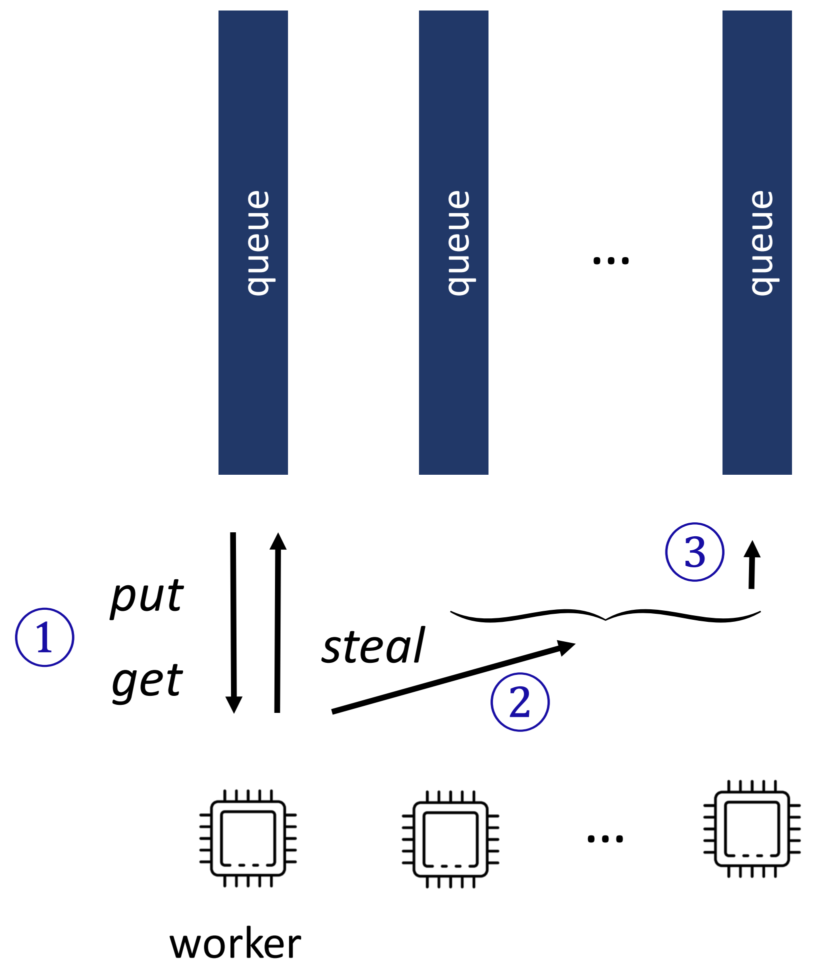
Work Stealing -- Load Balancing for Multi-core Task Processing



① A worker (core) puts on / gets from its queue

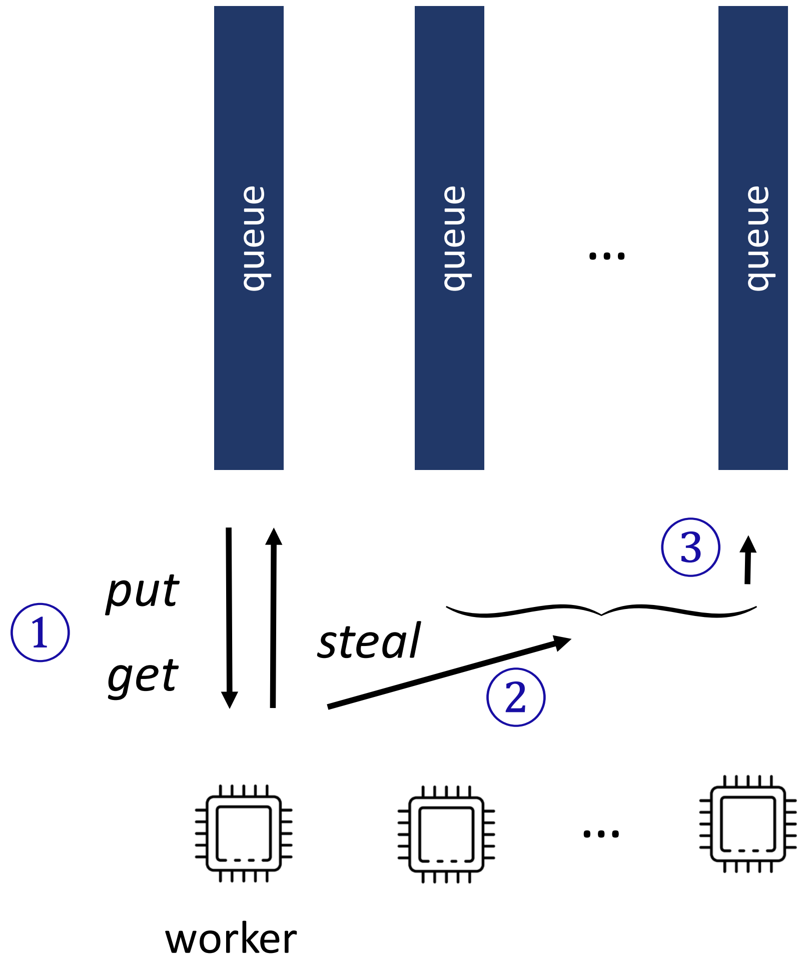
② When its queue is empty, it selects another queue

Work Stealing -- Load Balancing for Multi-core Task Processing



1. A worker (core) puts on / gets from its queue
2. When its queue is empty, it selects another queue
3. and try to steal from it.

Work Stealing -- Load Balancing for Multi-core Task Processing



① A worker (core) puts on / gets from its queue

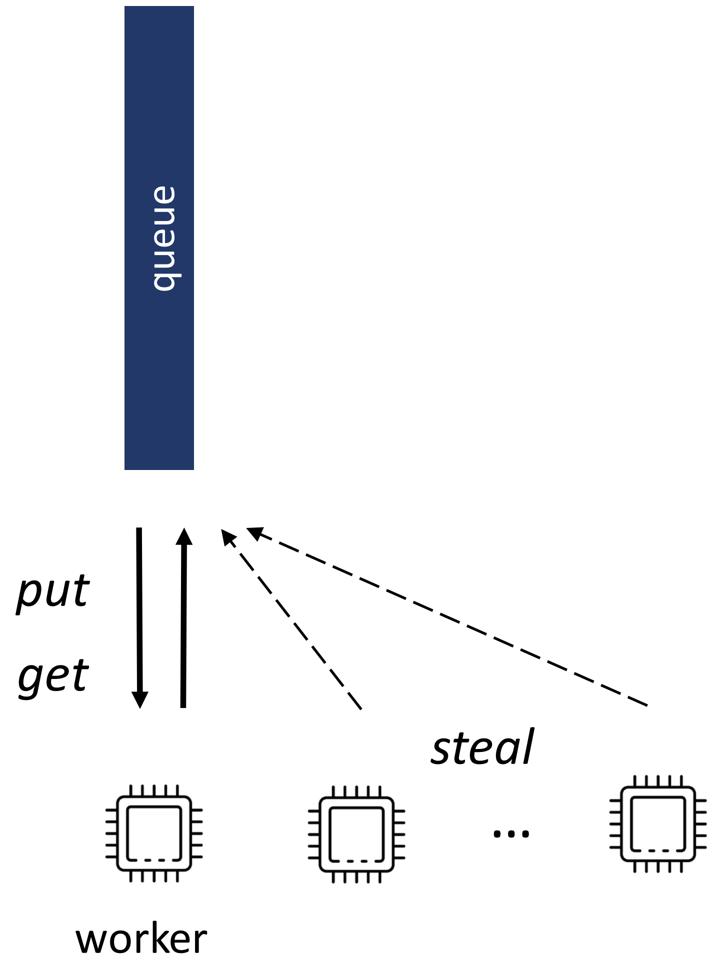
② When its queue is empty, it selects another queue

③ and try to steal from it.

Work Stealing Scenarios

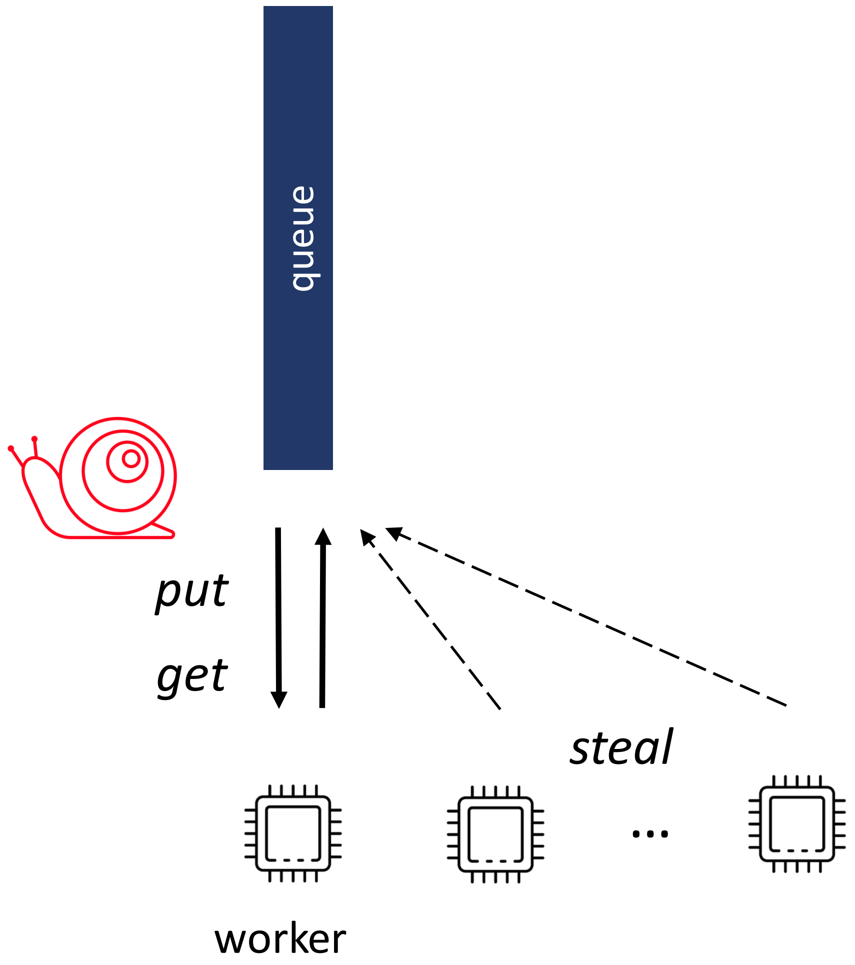
Java, Rust, Go, Julia, Kotlin, JS, OpenCilk, Intel® TBB runtime

Existing Works and Their Limitations



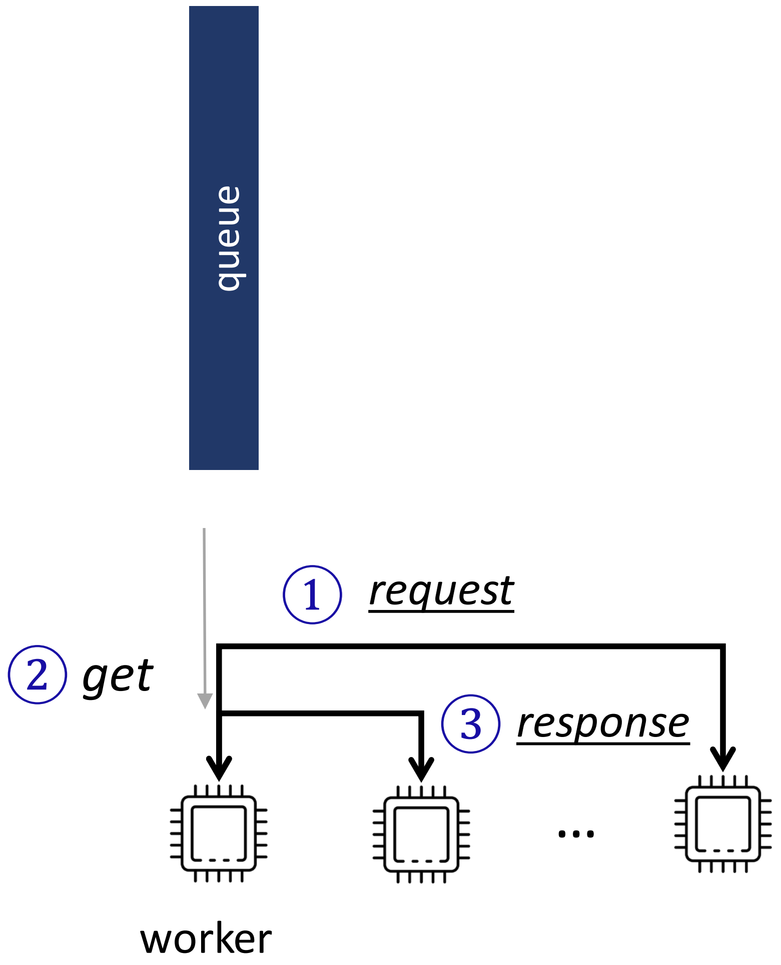
ABP [SPAA’98]:

Existing Works and Their Limitations



ABP [SPAA'98]:
- Costly synchronization primitives for every pus/get

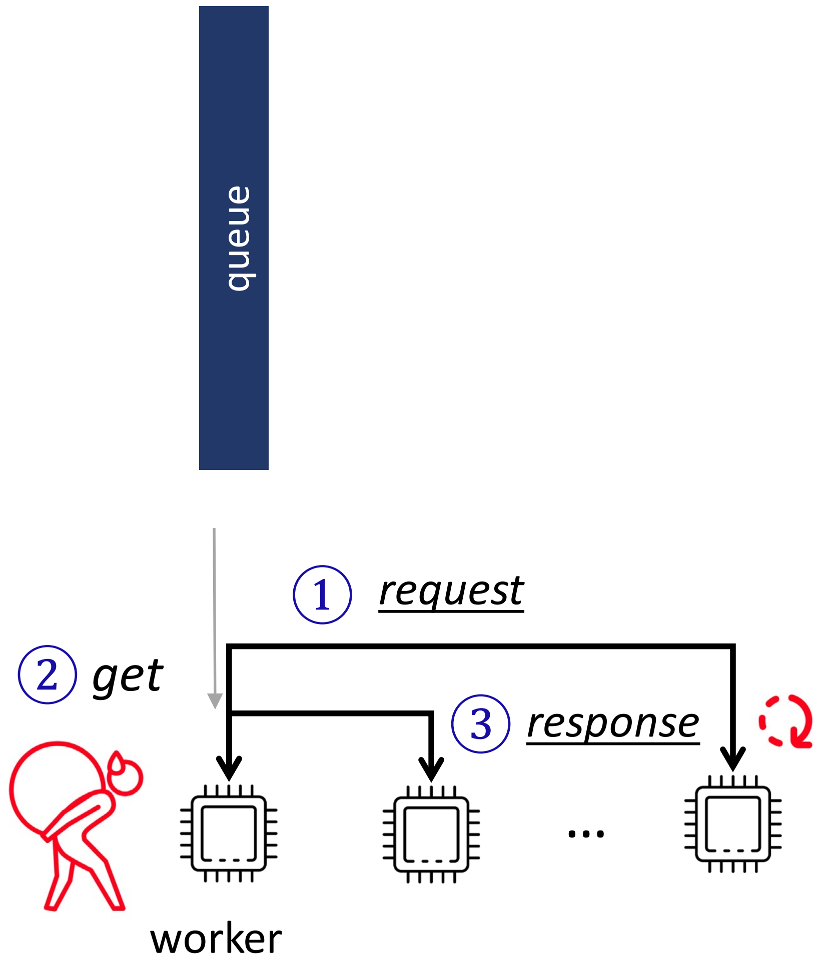
Existing Works and Their Limitations



ABP [SPAA'98]:
 - Costly synchronization primitives for every pus/get

Delegation [PPoPP'13]:

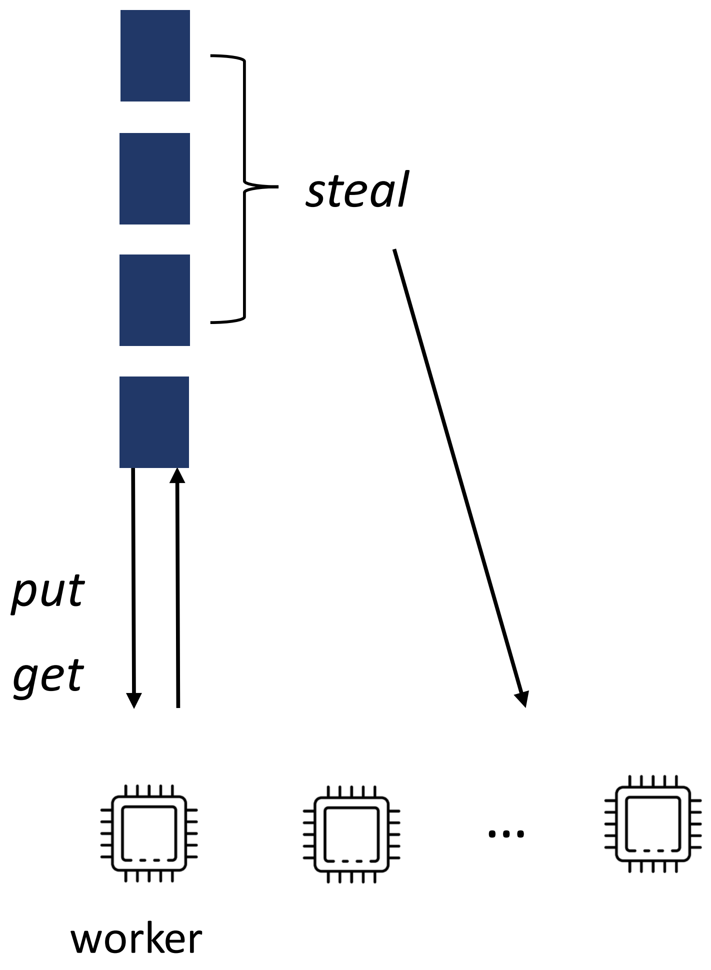
Existing Works and Their Limitations



ABP [SPAA'98]:
- Costly synchronization primitives for every pus/get

Delegation [PPoPP'13]:
- Spinning on the thief side, waiting for a response
- The owner is burdened with delegated workloads
- The owner and thieves frequently access the same communication variables (contention)

Existing Works and Their Limitations

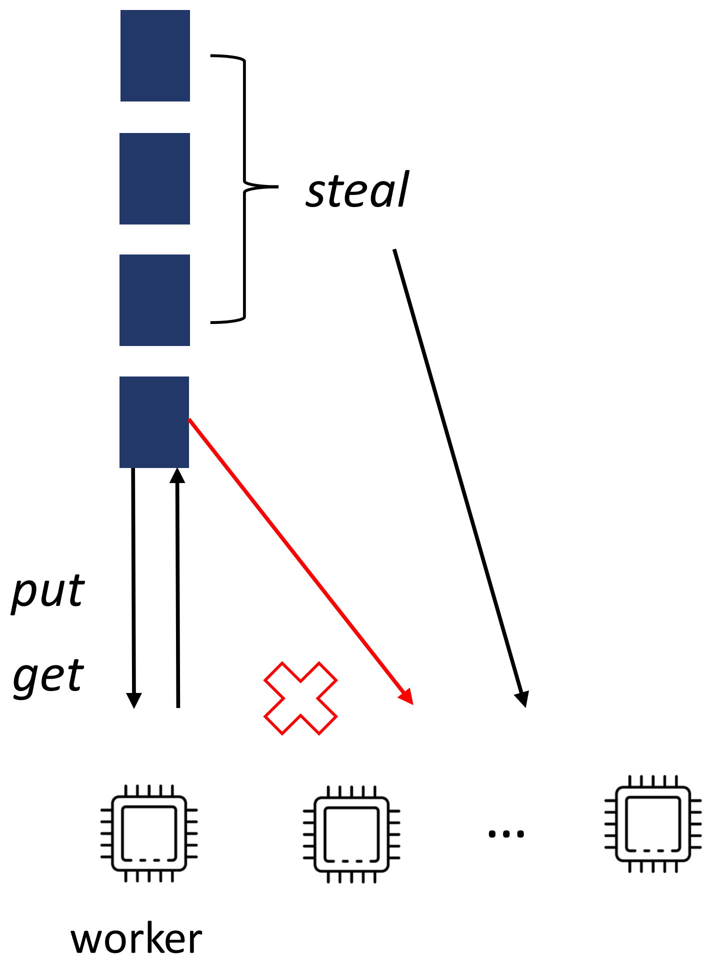


ABP [SPAA'98]:
 - Costly synchronization primitives for every pus/get

Delegation [PPoPP'13]:
 - Spinning on the thief side, waiting for a response
 - The owner is burdened with delegated workloads
 - The owner and thieves frequently access the same communication variables (contention)

BWoS [OSDI'23]:

Existing Works and Their Limitations

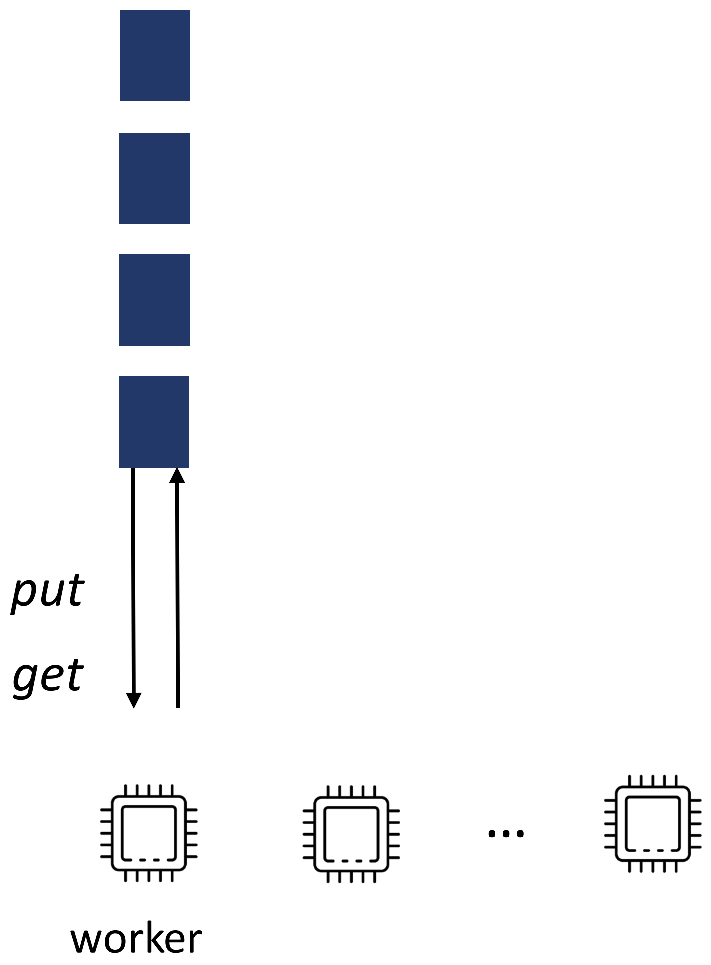


ABP [SPAA’98]:
- Costly synchronization primitives for every pus/get

Delegation [PPoPP’13]:
- Spinning on the thief side, waiting for a response
- The owner is burdened with delegated workloads
- The owner and thieves frequently access the same communication variables (contention)

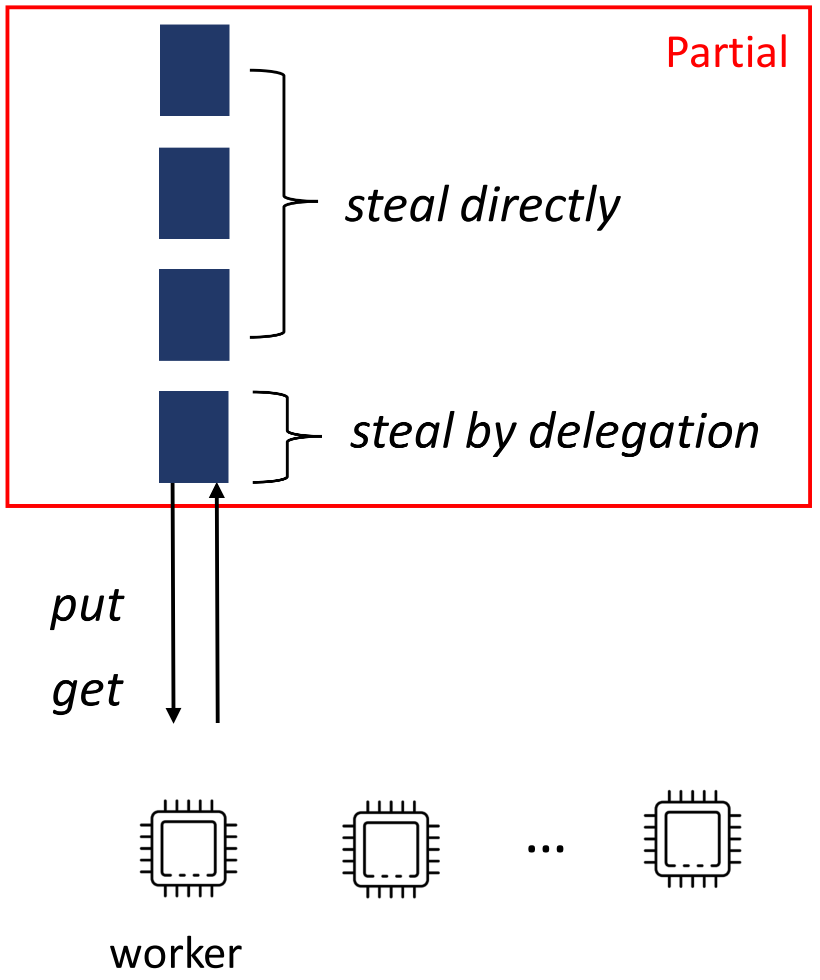
BWoS [OSDI’23]:
- Thieves can’t steal from the block where the owner is
  (Bad performance in specific scenarios)

Our Solution: Partial Asynchronous Delegation



Using the block-based design [ATC'22, OSDI'23] to avoid contention

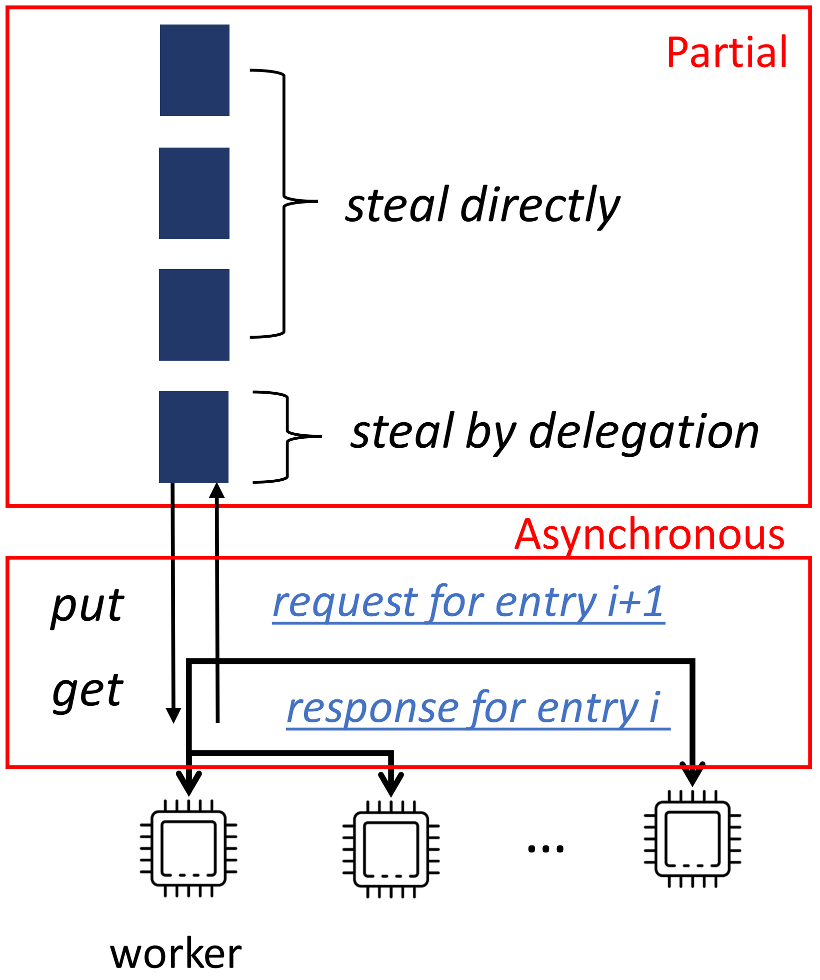
Our Solution: Partial Asynchronous Delegation



Using the block-based design [ATC'22, OSDI'23] to avoid contention

Partial:
- Delegation is enabled only for the block where the owner is present
- When the owner advances to the next block, the delegation of owner's current block is closed, and the next one is opened
- Allows for stealing from the owner's block compared to BWoS

Our Solution: Partial Asynchronous Delegation



Using the block-based design [ATC'22, OSDI'23] to avoid contention

Partial:
- Delegation is enabled only for the block where the owner is present
- When the owner advances to the next block, the delegation of owner's current block is closed, and the next one is opened
- Allows for stealing from the owner's block compared to BWoS

Asynchronous:
- A steal operation requests for the next steal op.
- A thief requests entry i+1 and obtains entry i that is requested by the previous steal operation (if available) without waiting

Thanks!