Reuse, don't Recycle

Transforming Algorithms that Throw Away Descriptors

Maya Arbel-Raviv
Technion

Trevor Brown
University of Toronto

Lock-Free Algorithms

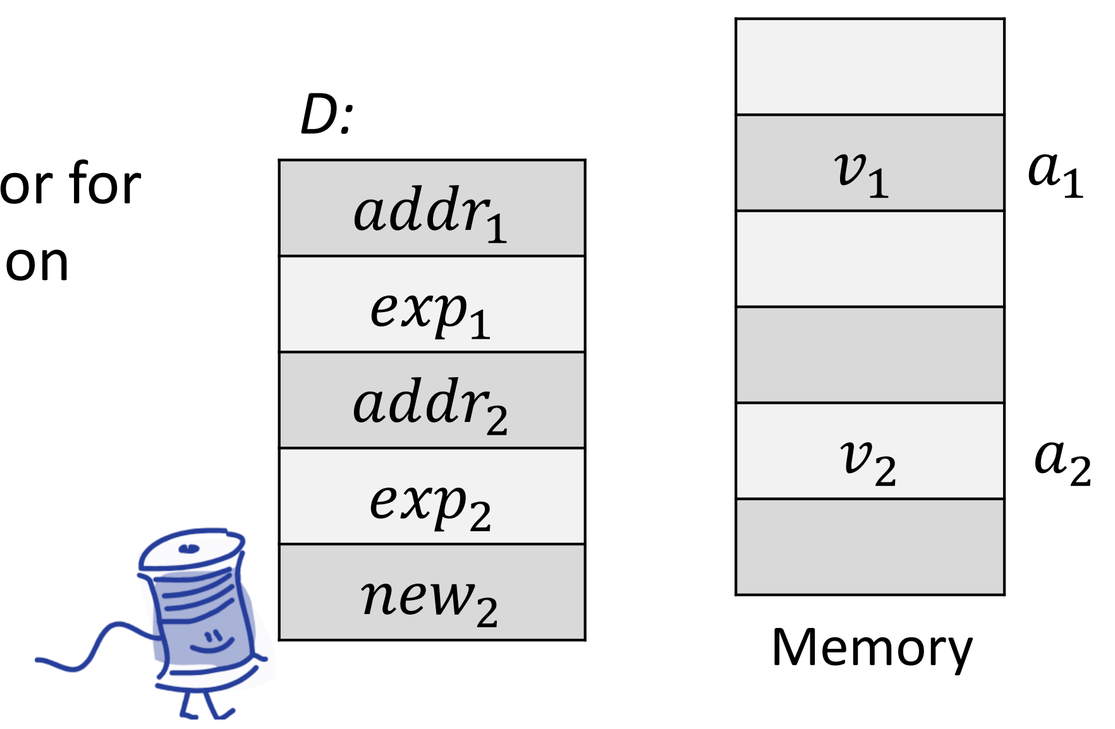
Lock-freedom: allows threads to starve, but requires that the system as a whole always makes progress. Usually guaranteed using helping, mutual exclusion for *operations* instead of mutual exclusion for *threads*.

Throwaway Descriptors

Example: double-compare single-swap (DCSS)
[Harris et al. 2002]

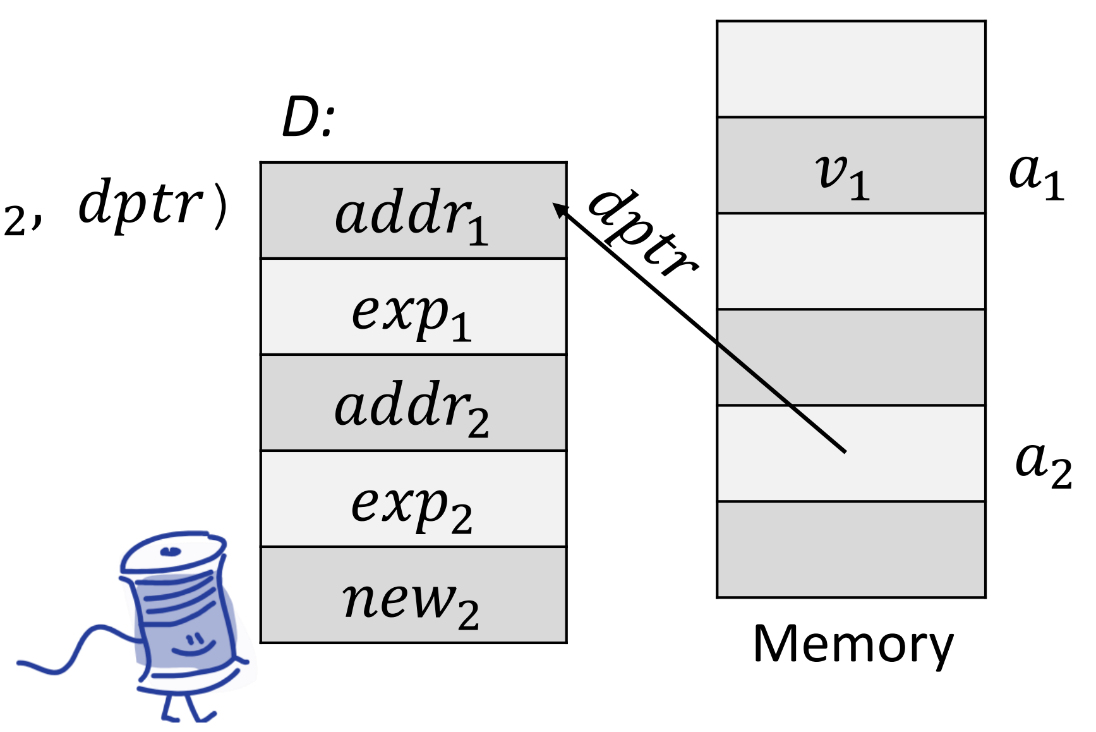
Allocation

New descriptor for **every** operation



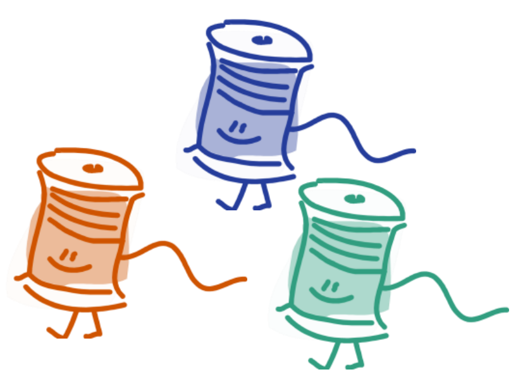
Publish

CAS(a_2, exp_2, $dptr$)



Help

```
if ($exp_1 = v_1$)
    CAS($a_2, dptr, new_2$)
else
    CAS($a_2, dptr, exp_2$)
```



Reclaim

Since a descriptor can be accessed by **many** threads, and **at any time**, it can only be freed once no thread has a pointer to it

Reusable Descriptors

A **single** multi versioned reusable descriptor *per thread*

Key idea

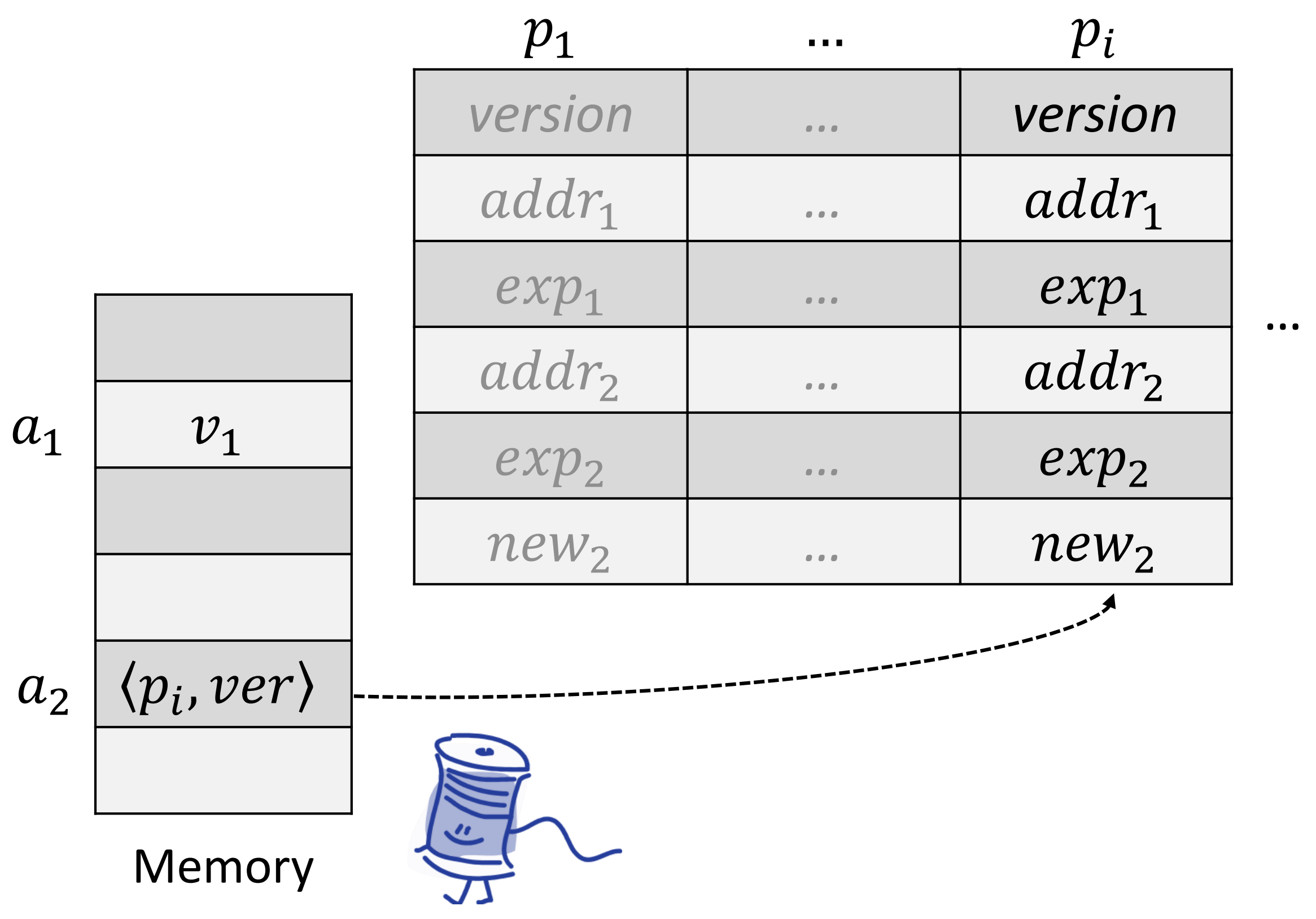
Descriptors can be reused **before** it is safe to free them

Instead of Allocation

Simply increment descriptor's version

Publish

CAS(a_2, exp_2, $\langle p_i, ver \rangle$)



p_1	…	p_i
version	…	version
$addr_1$	…	$addr_1$
exp_1	…	exp_1
$addr_2$	…	$addr_2$
exp_2	…	exp_2
new_2	…	new_2

Help

What if a thread wants to access a descriptor that has already been reused?

- When accessing a descriptor make sure its version hasn't changed.
- If the version changed, then the thread that owns that descriptor finished the operation that it describes. Thus, it no longer needs to be helped.

No need to reclaim descriptors

Experiments

We implemented the k-CAS algorithm of Harris et al. with reusable descriptors and throwaway descriptors, using several memory reclamation schemes:

- Hazard pointers (HP) [Michael 2004]
- DEBRA [Brown 2015]
- Read-copy-update (RCU) [McKenney et al. 1998]

Methodology

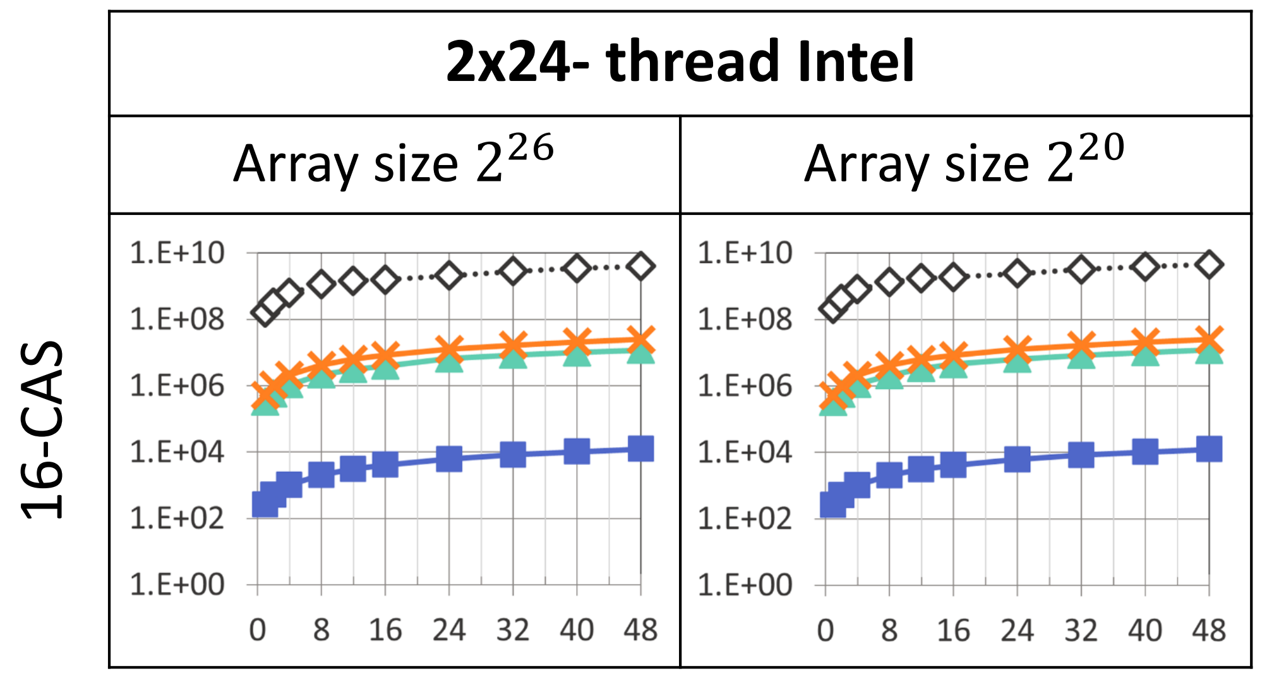
Array based microbenchmark. In each timed trial, n threads run for one second and repeatedly increment k array locations using a k-CAS.

Hardware

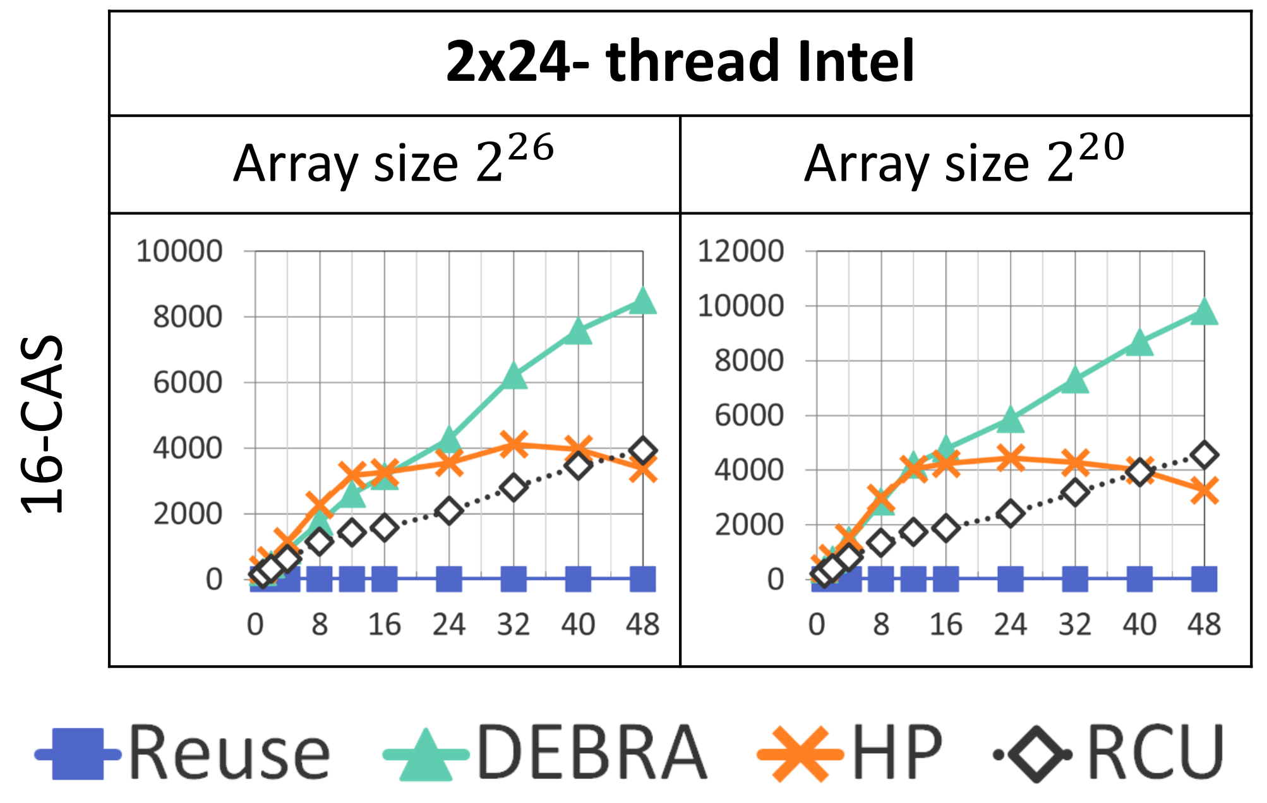
- 2-socket Intel E7-4830 v3, 48 threads
- 4-socket AMD Opteron 6380, 64 threads

Memory Usage Results

Descriptor footprint (in bytes, log scale):

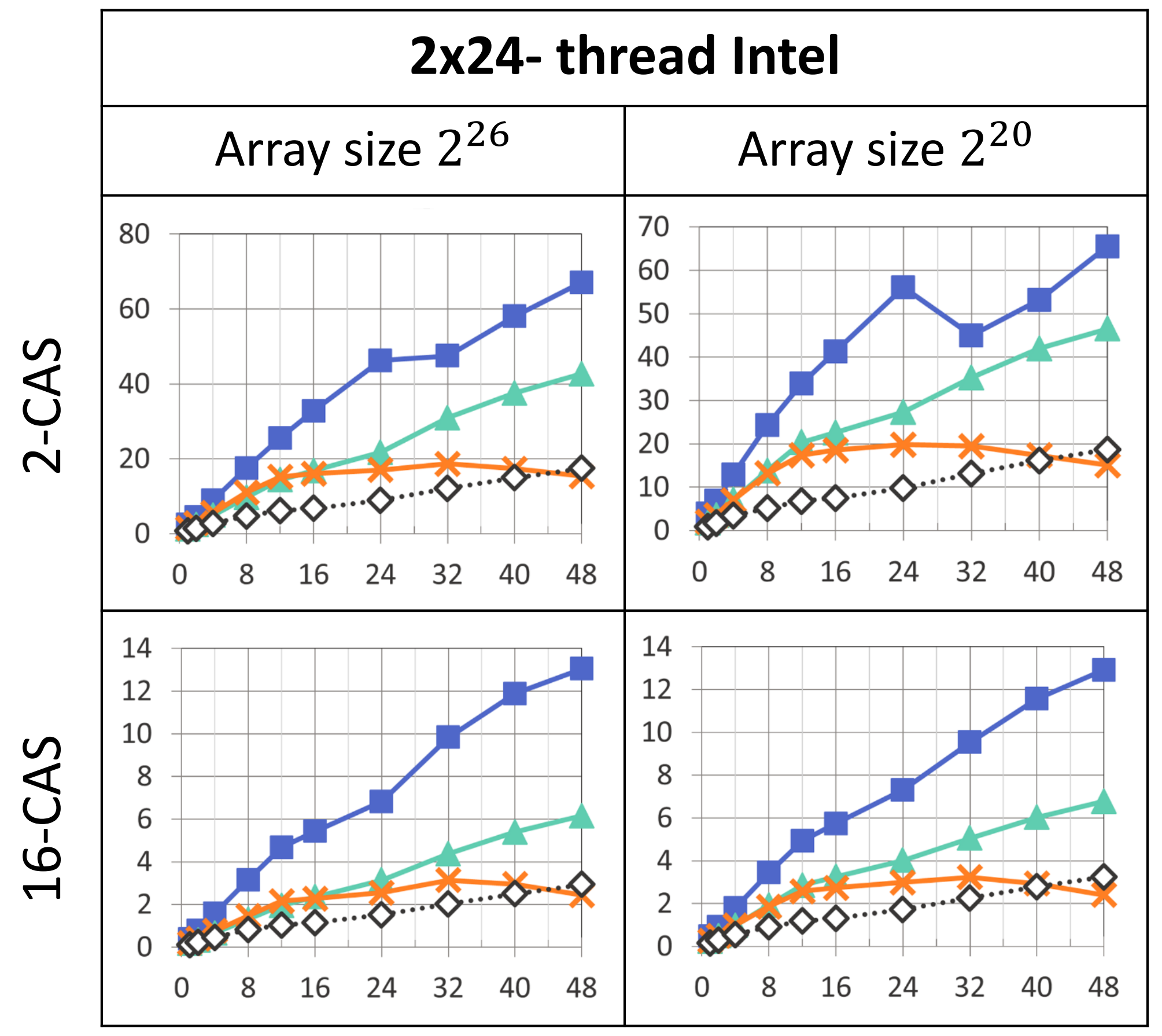


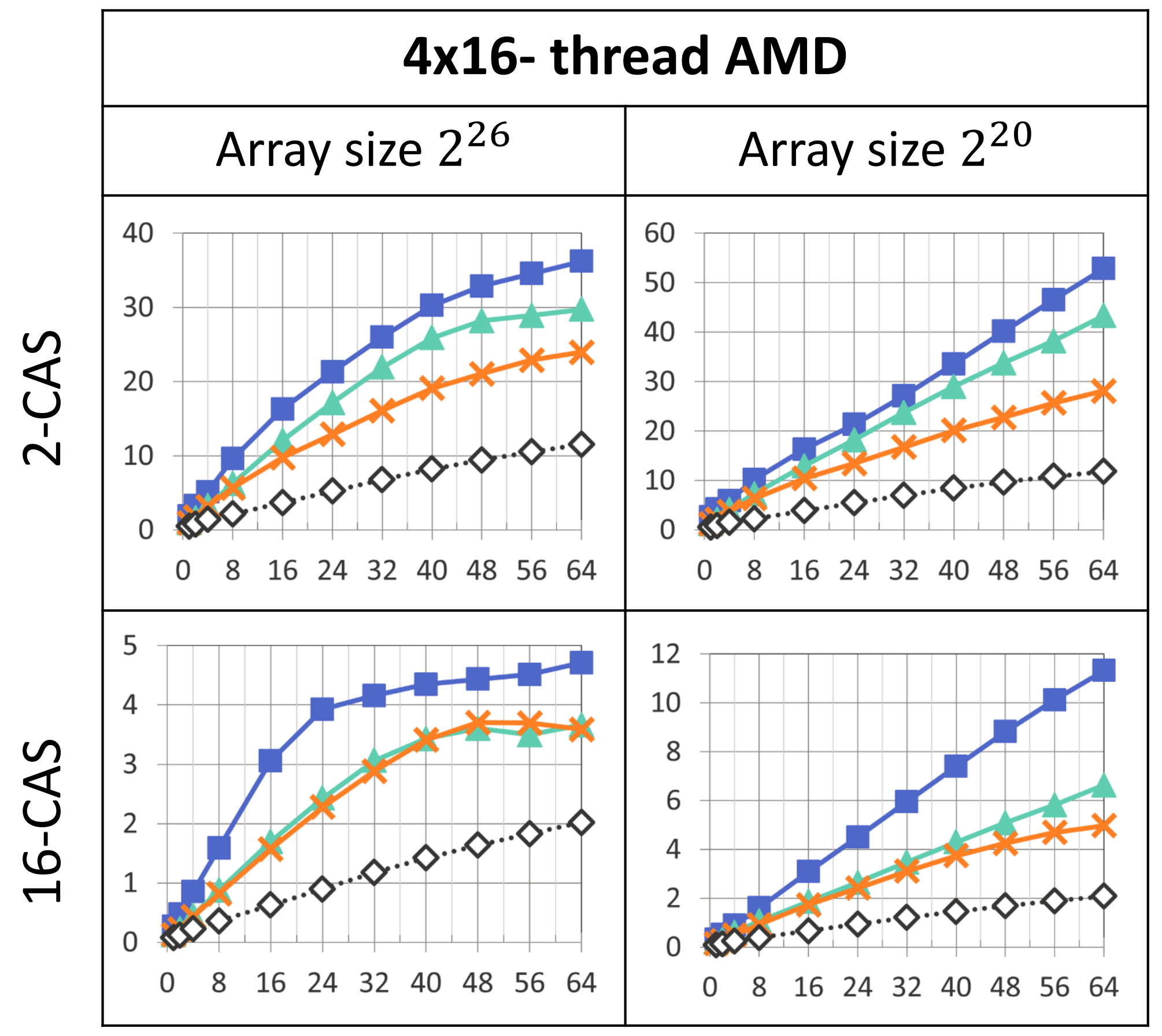
Descriptor allocations (in MB):



Throughput Results

Operations per microsecond:





Acknowledgments

HPI
Mitacs