Pacific Northwest
NATIONAL LABORATORY

Locality Aware Scheduling For Scalable Heterogeneous Environments

ROSS 2020

Alok Kamatar, Ryan Friese, Roberto Gioiosa

U.S. DEPARTMENT OF ENERGY BATTELLE

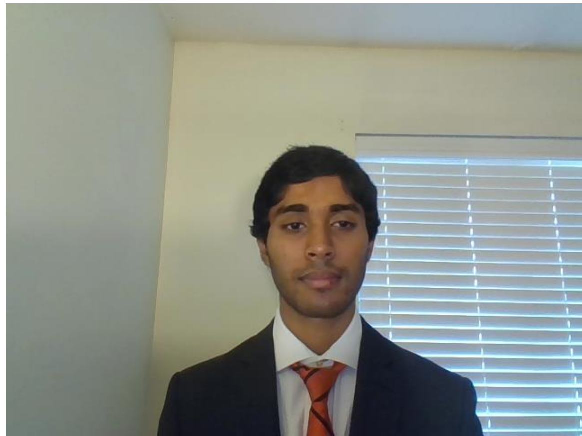
PNNL is operated by Battelle for the U.S. Department of Energy

Outline

- Motivation
- The Minos Computing Library
- Architecture Details
  - Resident Memory
  - Scheduling Algorithm
- Experimental Results
  - SHOC Benchmarks
  - Hyper-parameter evaluation
- Conclusions and Future Work

Exploiting Scalable, Heterogeneous Systems

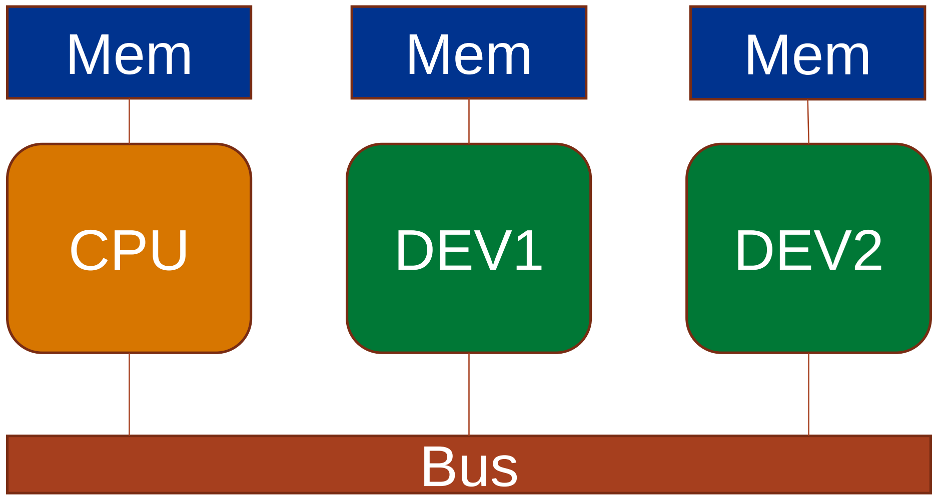
- Increasing levels of hardware specialization
  - i.e. GPUs, Deep Learning Accelerators, DSPs, etc.
- Higher complexity to program new applications
- Potential pitfalls when scaling to other architectures

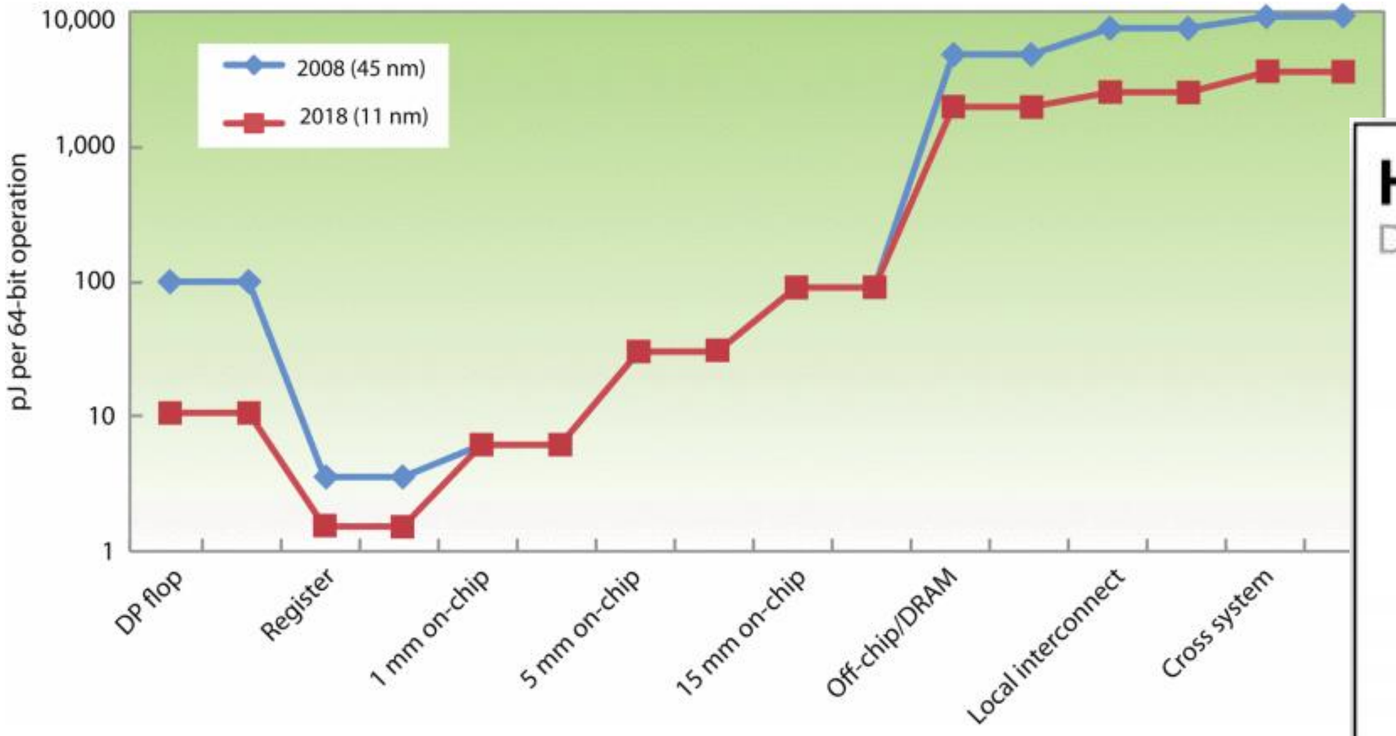


Motivation

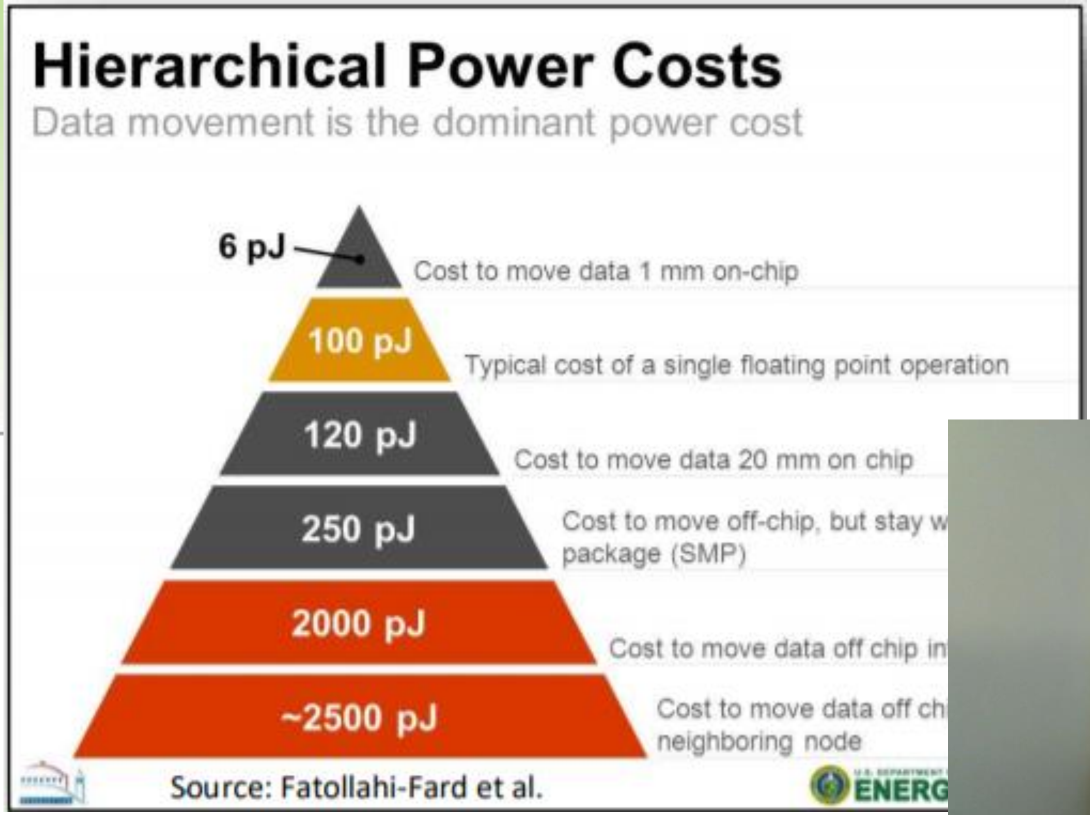
- Data movement is now the critical aspect of performance

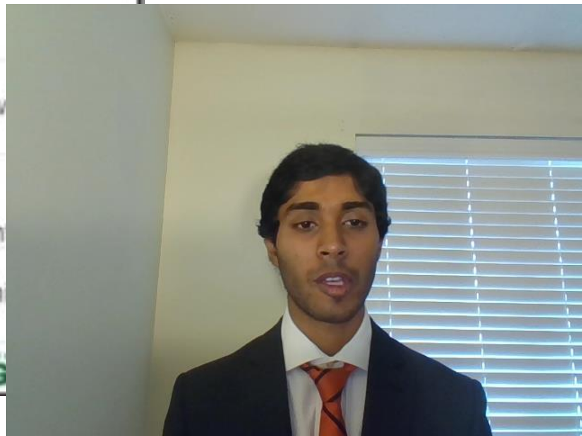
Multiple Memory Nodes





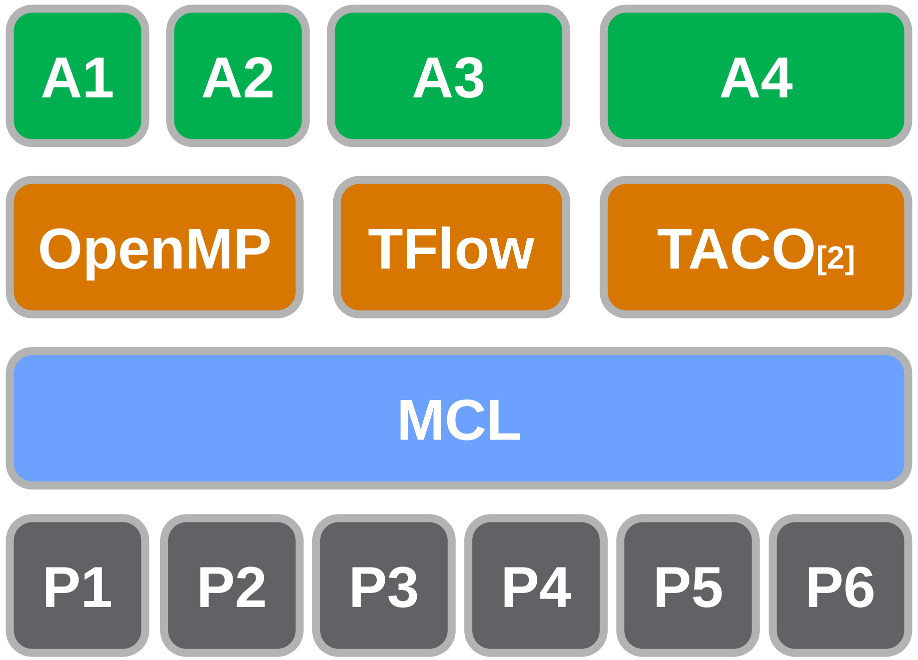
Source: Kogge, Shalf 2013



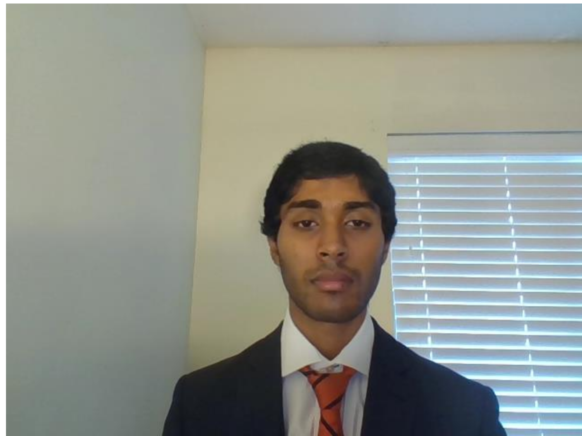


The Minos Computing Library (MCL) [1]

- Framework for programming heterogeneous systems
- Features
  - Dynamic task scheduling onto available resources
  - Co-schedule independent applications
  - Internal profiling and tracing capabilities
- Flexibility
  - Automatic scaling to available resources
  - Works/can be integrated with commonly used technologies
  - Independent scheduler framework



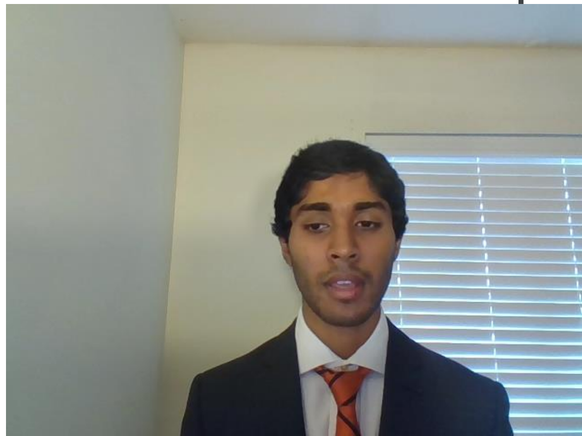
[1] R. Gioiosa, B. Mutlu, S. Lee, J. Vetter, G. Picierro, M. Cesati. 2020. The Minos Computing Library: Efficient Parallel Programming for Extremely Heterogeneous Systems. In General Purpose Processing Using GPU (GPGPU '20), February 23, 2020, San Diego, CA, USA
[2] G. Kestor, R. Gioiosa, M. Raugas. Towards Performance Portability through an Integrated Programming Eco-System for Tensor Algebra. In Performance, Portability, and Productivity in HPC Forum, to be held online, September 2020



MCL Program Example

- Same piece of code will exploit all available resources
- OpenCL kernels are directly usable by MCL

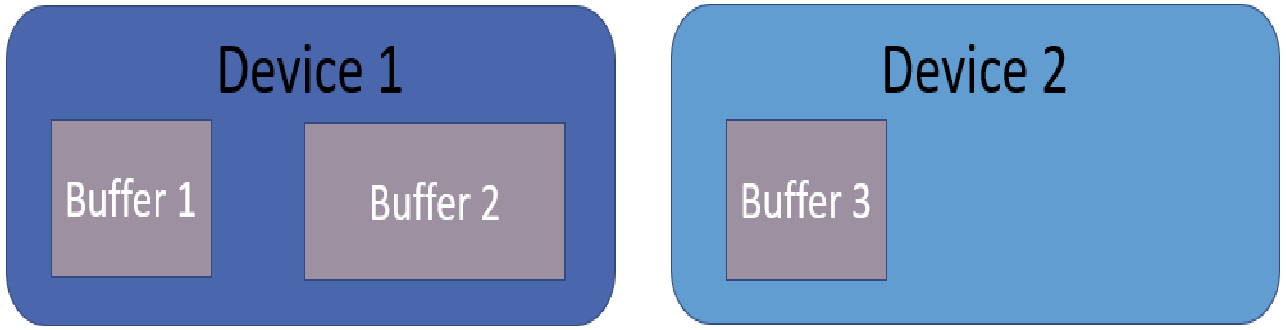
```c
#include <minos.h>

int main(void){
    mcl_init(NWorkers, <flags>);
....

    for(i=0; i< NIter; i++){
        hdl[i] = mcl_task_create(<filename>, <kernel>, <flags>);
        mcl_task_set_arg(hdl[i], 0, <addr>, <size>, <flags>);
        ...
        mcl_task_set_art(hdl[i], k, <addr>, <size>, <flags>);
        mcl_exec(hdl[i], <pes>, NULL, MCL_TASK_GPU);
    }
...
    mcl_wait_all();
...
    mcl_finit();

    return 0;
}
```



MCL Resident Memory Module

- Allows persistent data to remain in device memory across tasks
- Coordinates data movement so correct data is transferred to the correct device
- Supports read-only (i.e. multiple copies) and read-write data (exclusive copies)

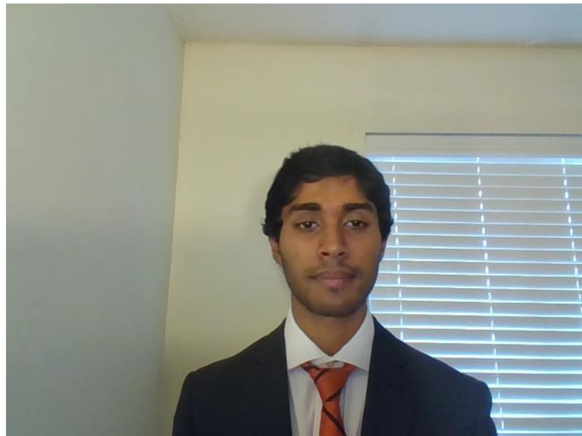
Pid	Memory	Concurrent Uses	Size	Device 1	Device 2
1	1	1	10 MB	1	0
1	2	1	20 MB	1	0
1	3	1	10MB	0	1
…	…	…	…	…	…



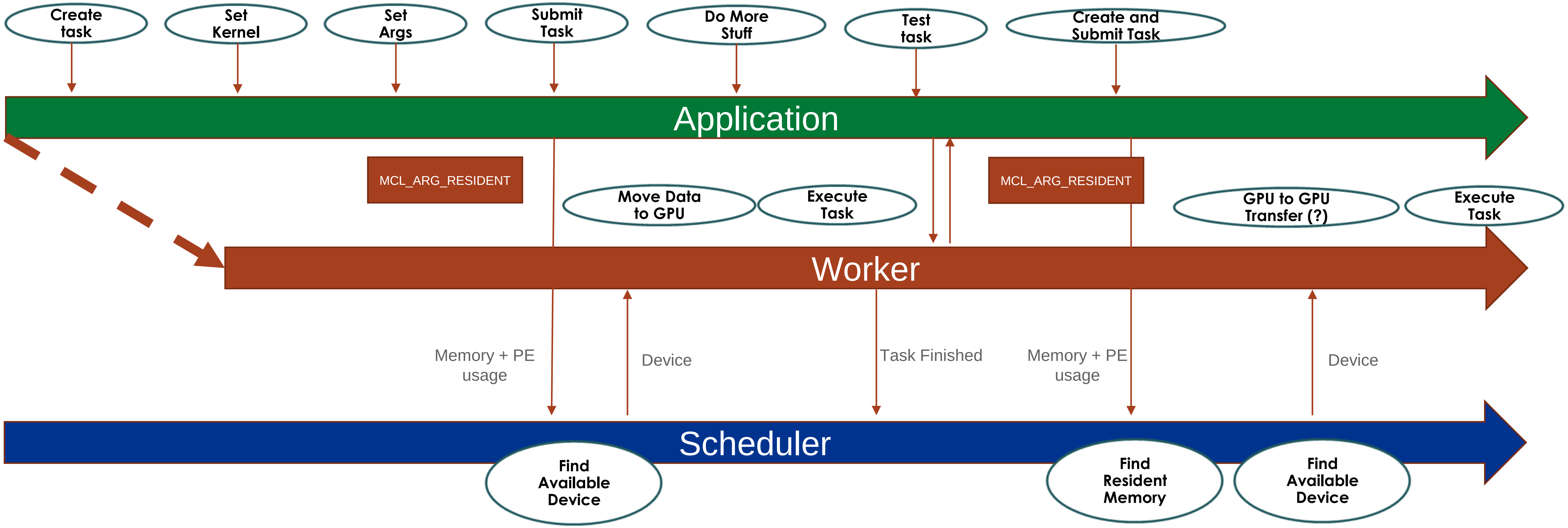
MCL_ARG_RESIDENT

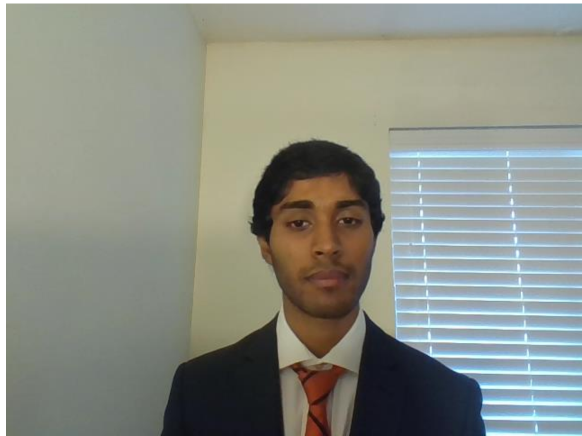
MCL_ARG_INVALID

MCL_ARG_DONE



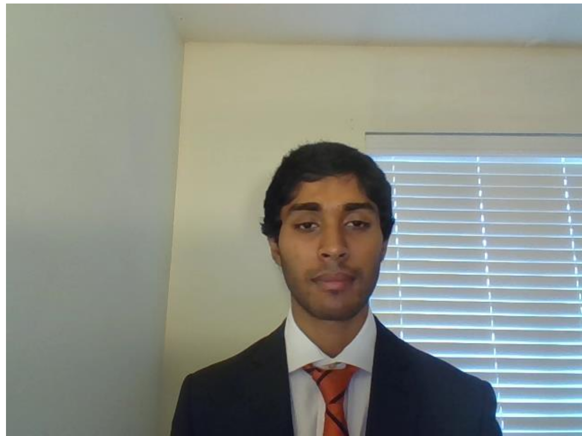
MCL Trace with Resident Memory





MCL Schedulers: Round-Robin Scheduler

- Schedules tasks in a first-in first out manner to the next available device
- Typically achieves high resource utilization
- Problems
  - Suffers from head-of-line blocking
  - Makes poor scheduling decisions when there is data reuse between kernels
  - **Unnecessary data transfers**



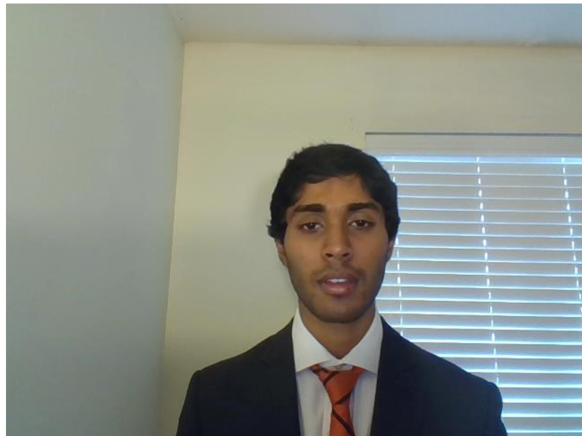
Pacific Northwest NATIONAL LABORATORY

MCL Schedulers: Delay Scheduler

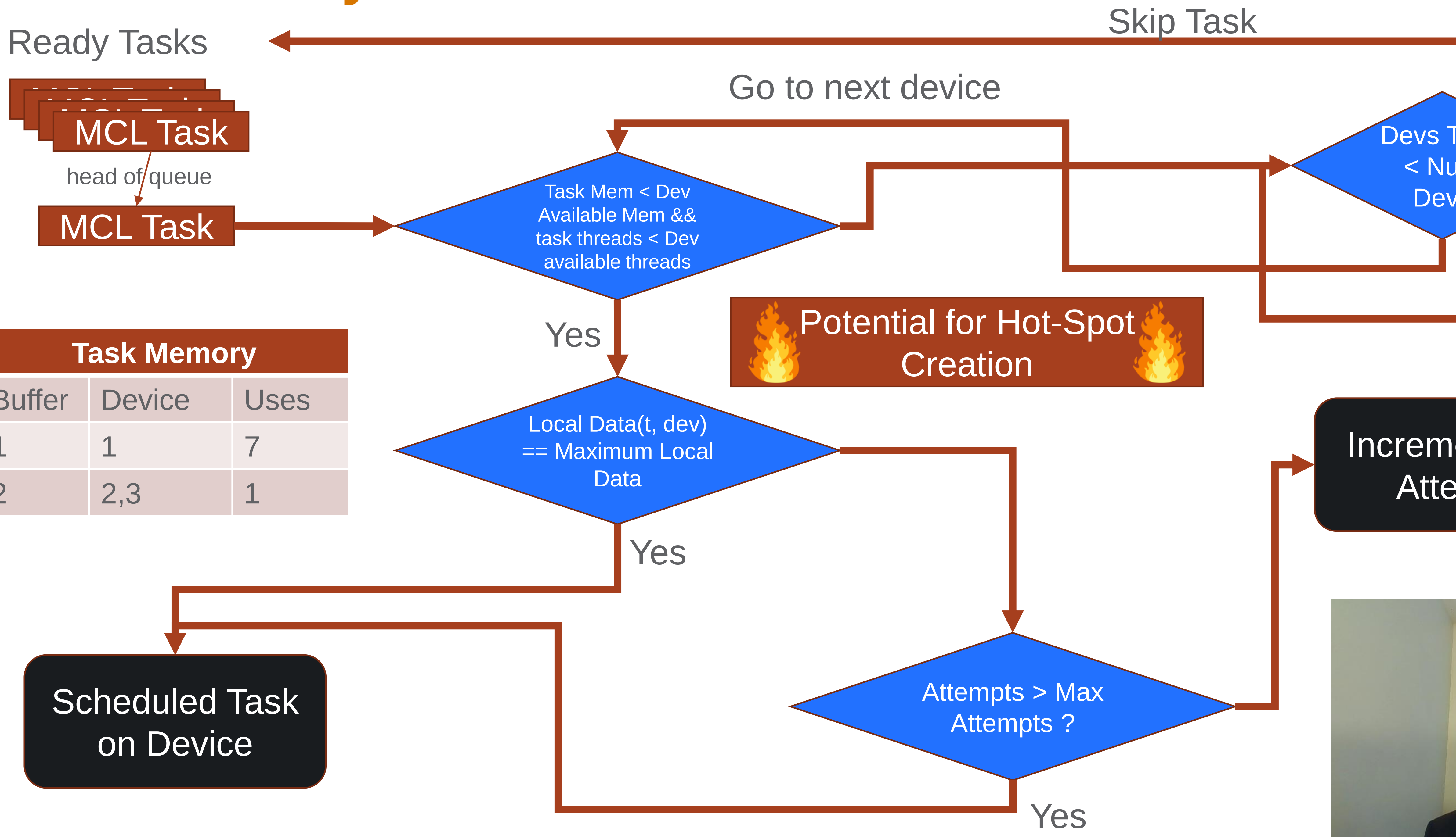
- Delays kernels from running on devices without device local data to minimize data transfers
- Skips devices that do not have device local data
- Skips tasks when waiting for busy devices
- Limits the number of times a task can be delayed to prevent a task from blocking too long

Algorithm 1 Device Local Data

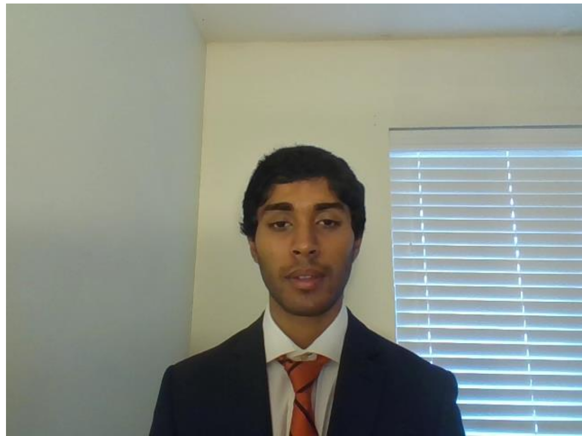
LocalData(Device δ**, Task** t**):**
$bytes \leftarrow 0$
for all β in $t.buffers$ **do**
    if β in $\delta.data$ **then**
        $bytes \leftarrow bytes + \beta.size$
    end if
end for
return($bytes$)
end LocalData



Delay Scheduler Illustration



Task Memory		
Buffer	Device	Uses
1	1	7
2	2,3	1

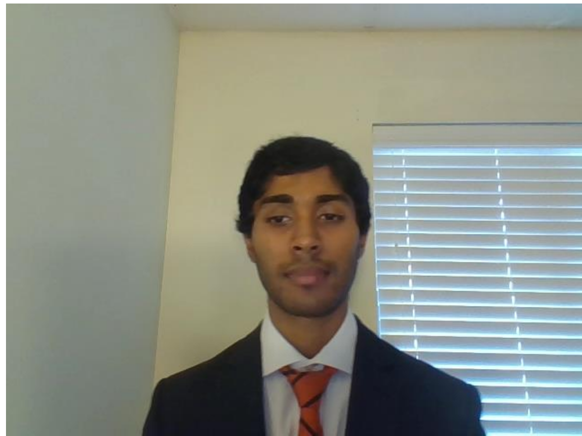


MCL Schedulers: Mixed Scheduler

- Attempts to balance system utilization with data locality concerns
- Detects popular pieces of data to create replicas
- The best predictor of how popular data will be is how popular it was in the past

Algorithm 3 Device Local Data With Copy Factor

LocalDataCopyFactor(Device δ**, Task** t**, CopyFactor** γ**):**
$bytes \leftarrow 0$
for all β in $t.buffers$ **do**
  if β in $\delta.data$ and DEVICES(β)$\geq$ $\log_\gamma$ TASKS(β) **then**
    $bytes \leftarrow bytes + \beta.size$
  end if
end for
return($bytes$)
end LocalDataCopyFactor

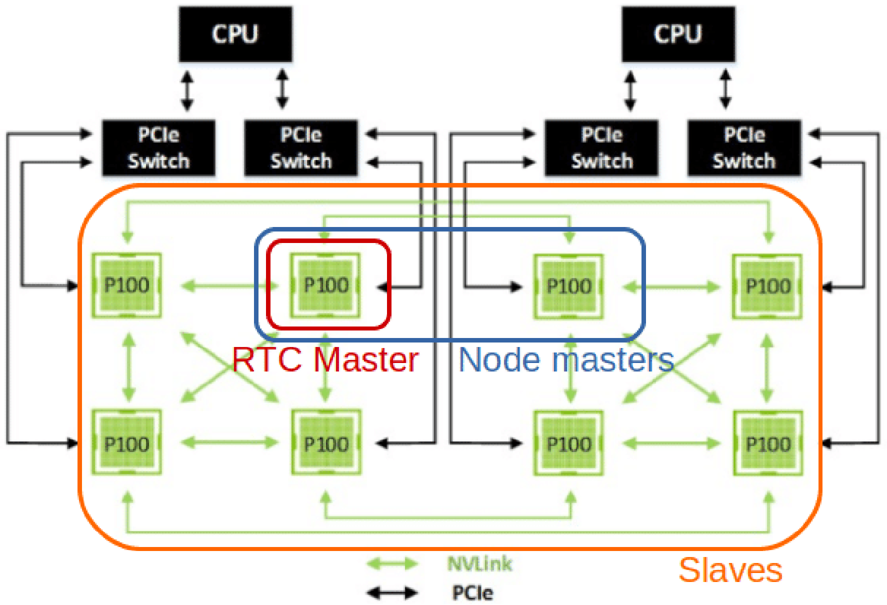


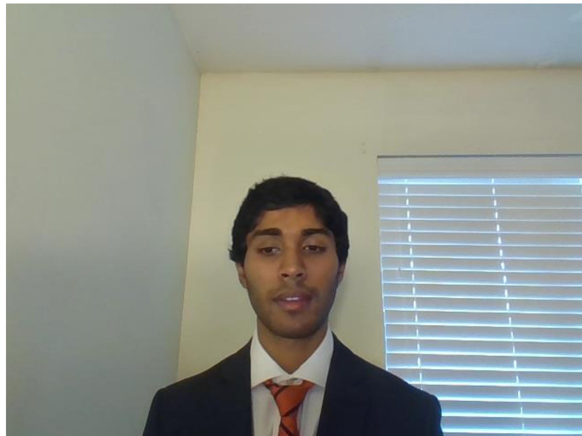
Experimental Setup

- Benchmarks (from SHOC)
  - BFS*
  - Sort*
  - FFT*
  - DGEMM
  - SPMV
  - MD5Hash

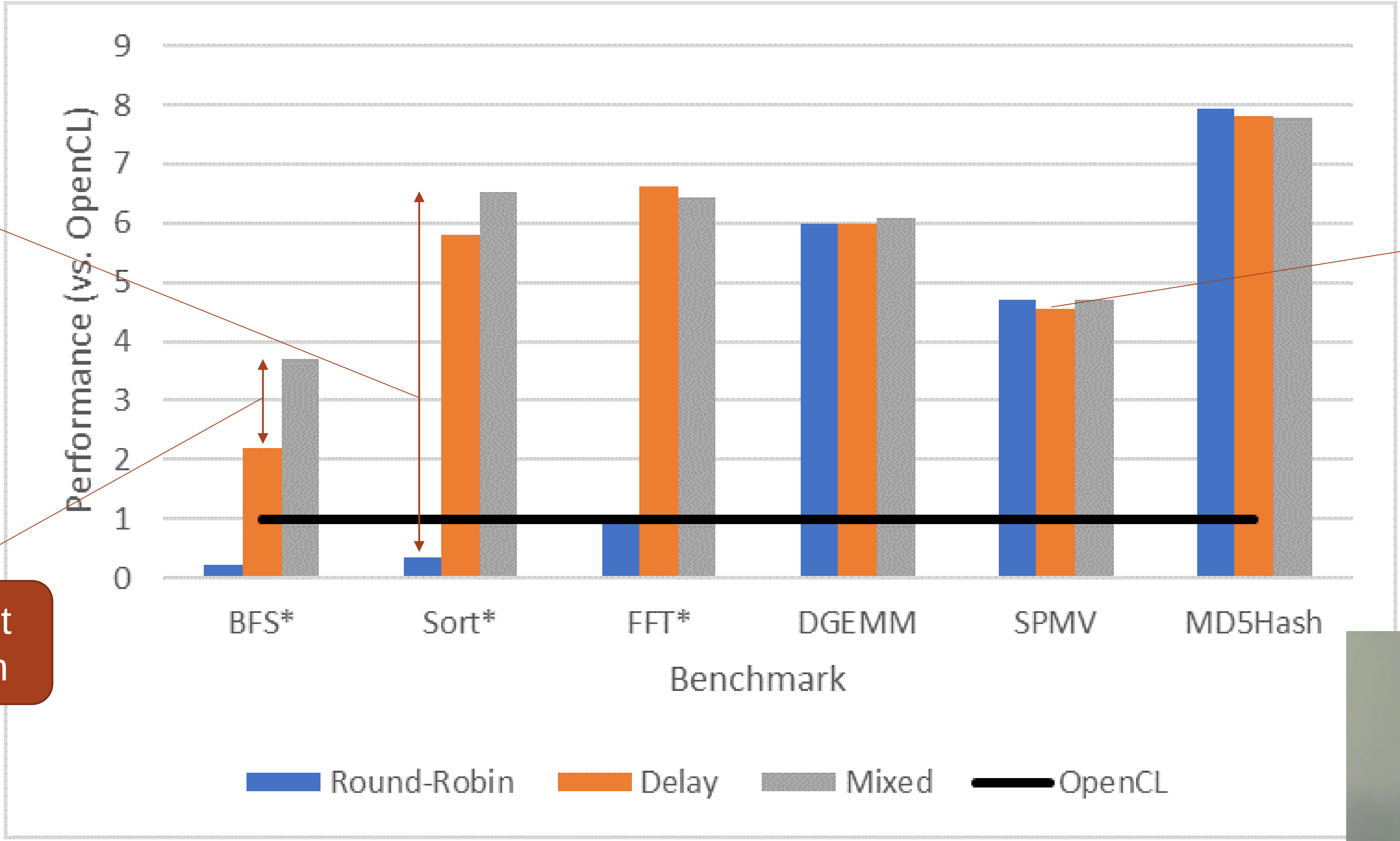
* = Exploits Data Locality

DGX-1 P100
- 2x 20 Core Intel Xeon CPU
- 8x Nvidia P100 GPU, NVLINK
- 256 GB RAM + 16 GPU





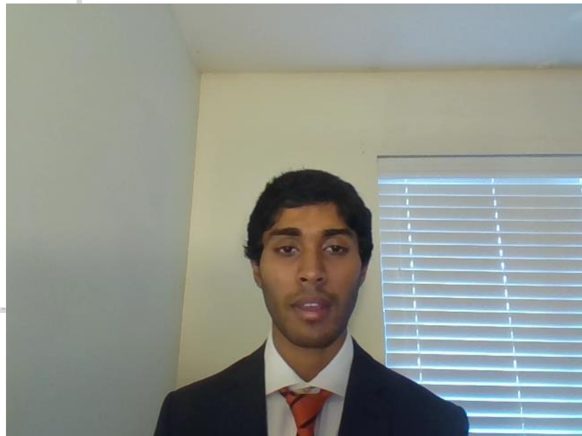
Scheduling Evaluation



Benefit of Locality

Hot-Spot Creation

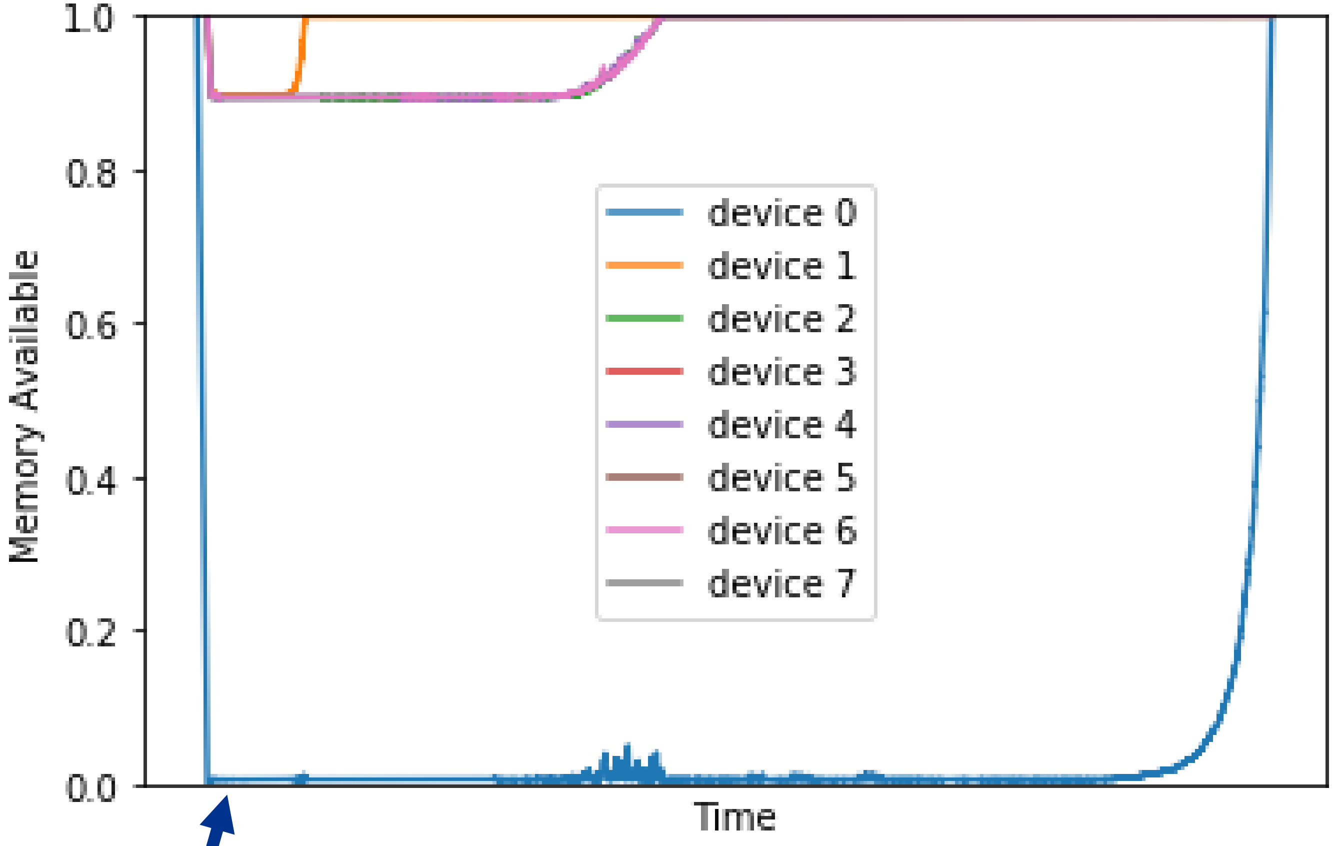
Low Additional Overhead



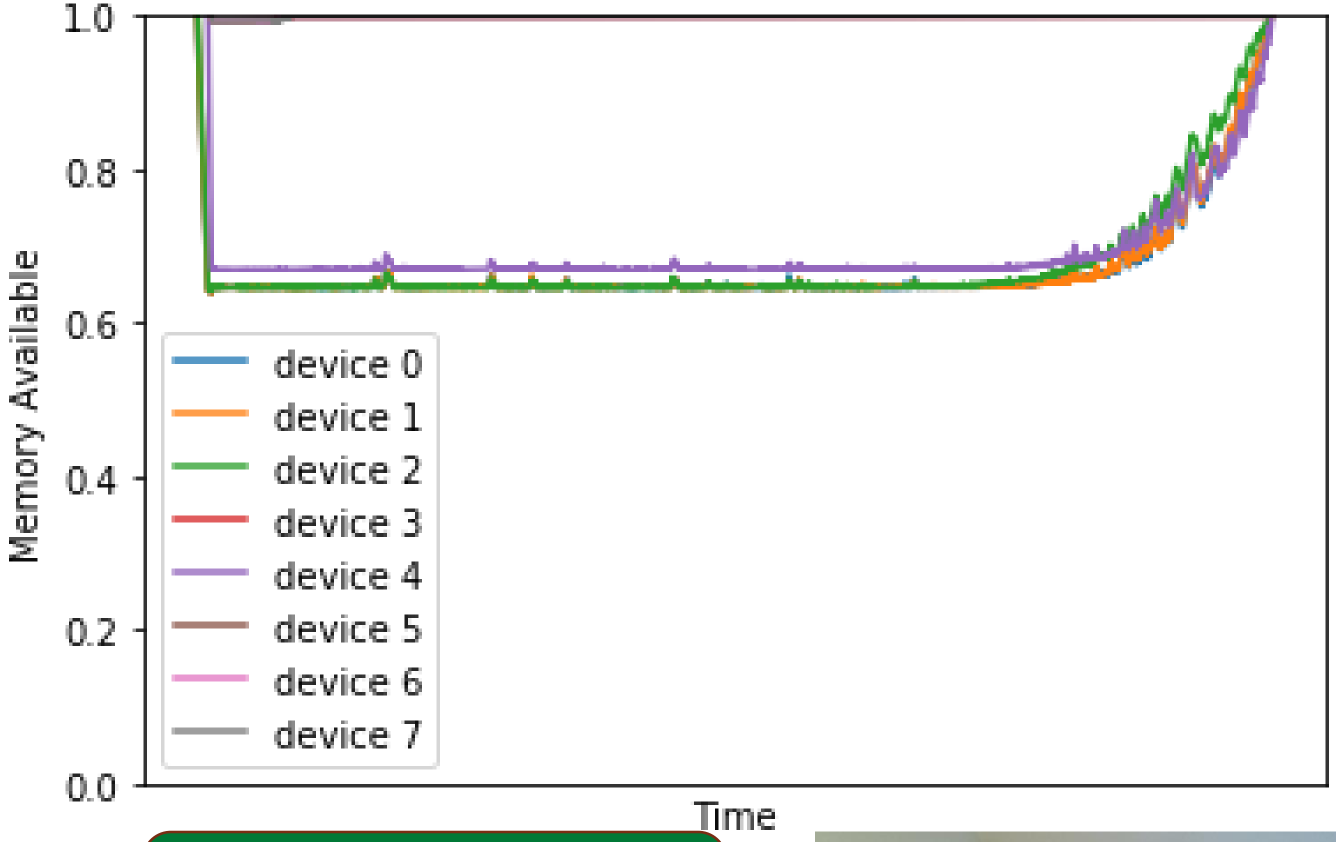
Pacific Northwest
NATIONAL LABORATORY

BFS Memory Usage Comparison

Delay Scheduler Memory Trace



Mixed Scheduler Memory Trace

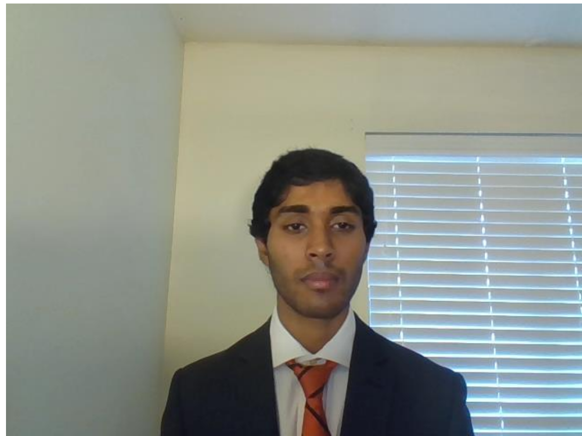


GPU 1

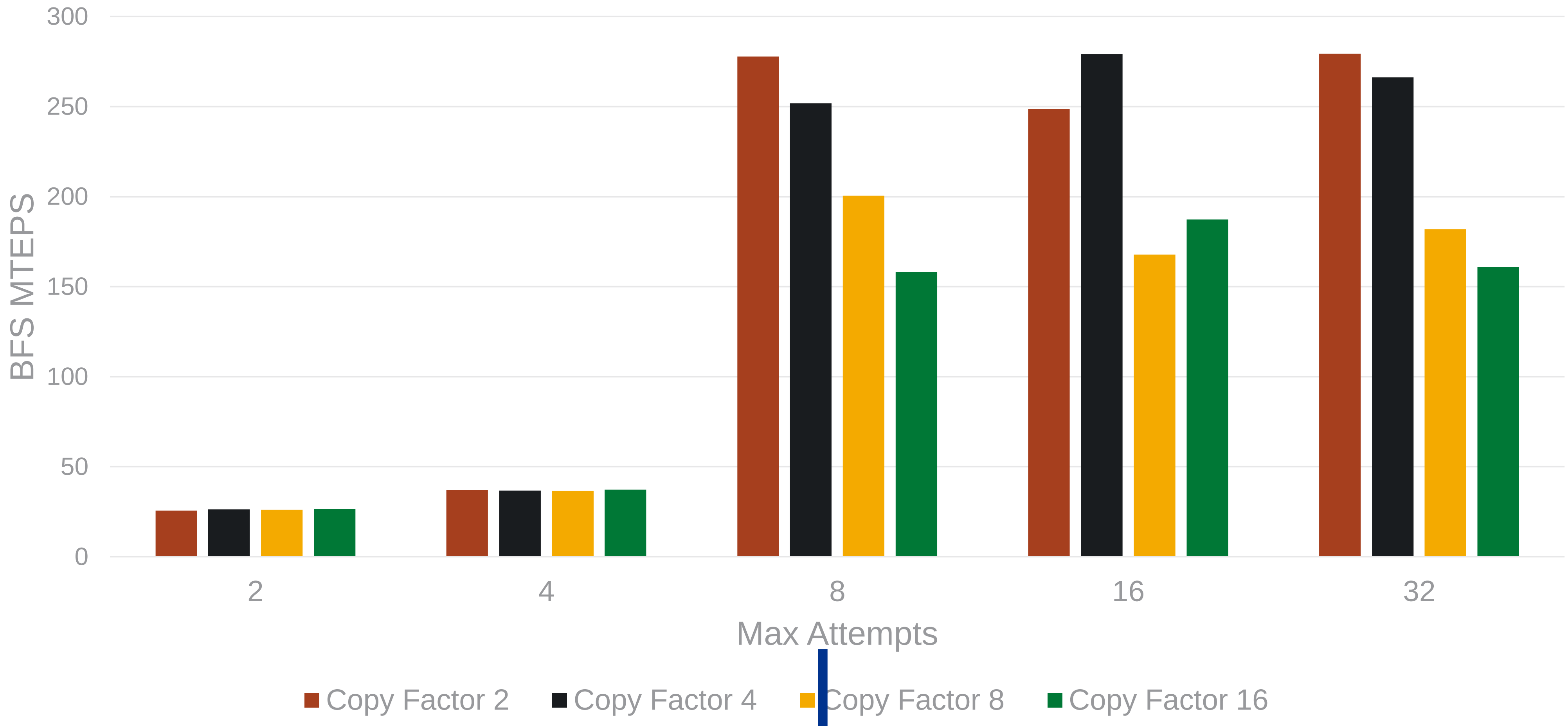
Hot Spot creation on device 1

Adaptive replication of popular data

Memory use from MCL internal trace capabilities



Effect of Hyperparameters on Performance

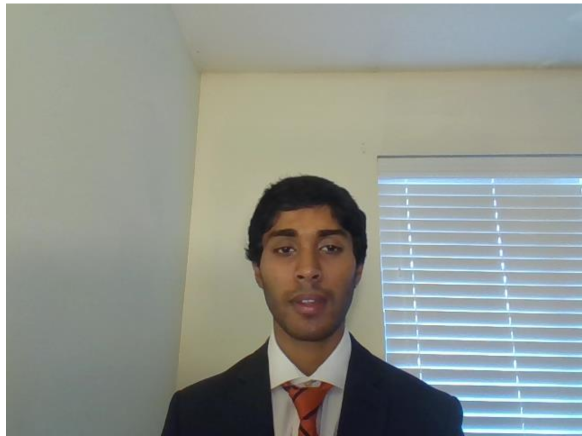


BFS Benchmark
- 1,000,000 vertices
- 4096 Tasks

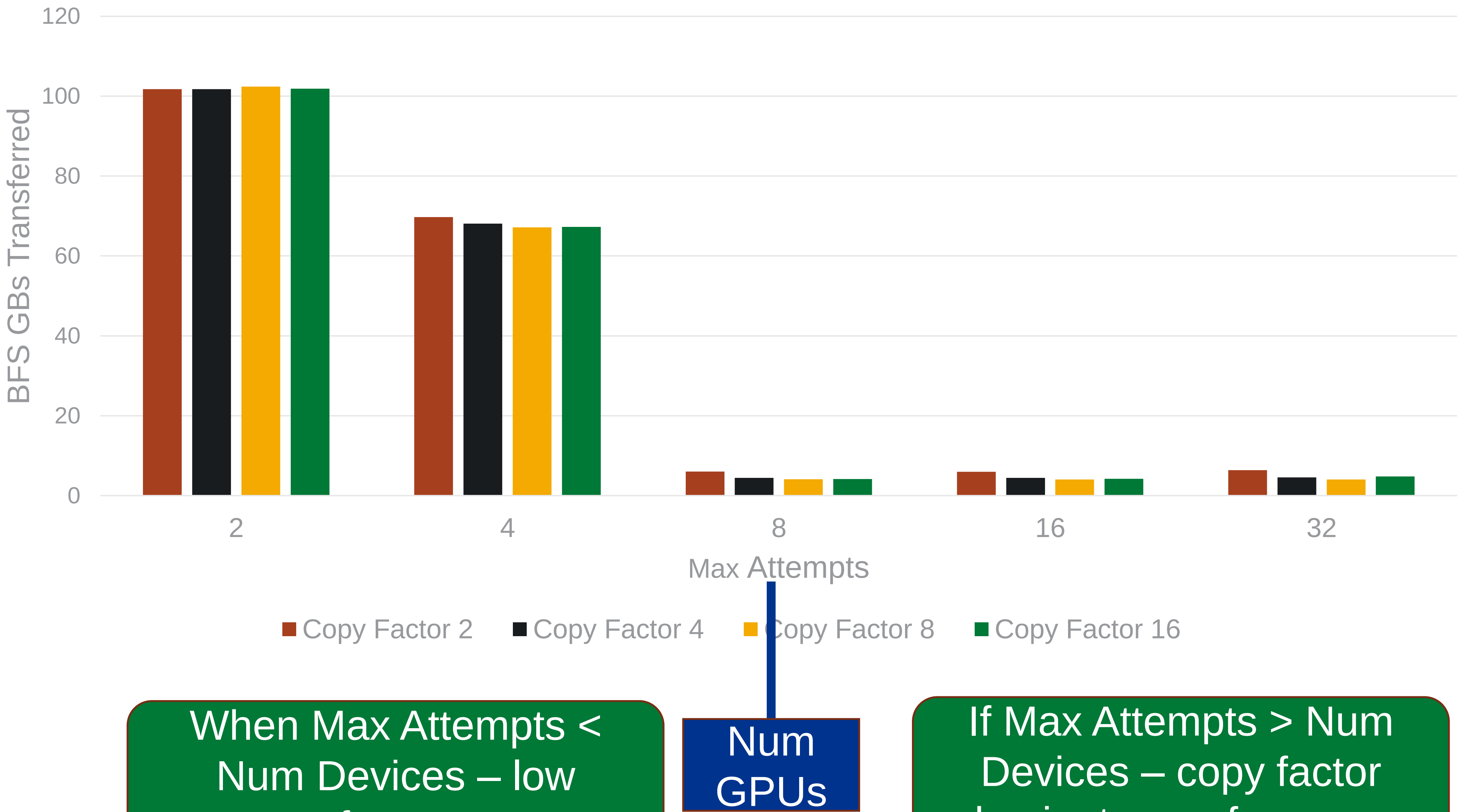
Num GPUs

When Max Attempts < Num Devices – low performance

If Max Attempts > Num Devices – copy factor dominates performance



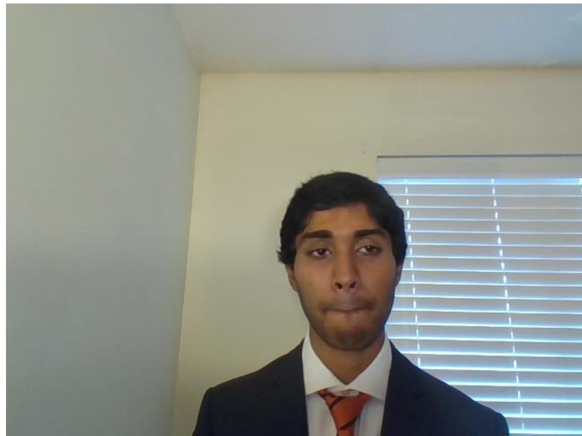
Effect of Hyperparameters on GBs Transferred



BFS Benchmark
- 1,000,000 vertices
- 4096 Tasks

When Max Attempts < Num Devices – low performance

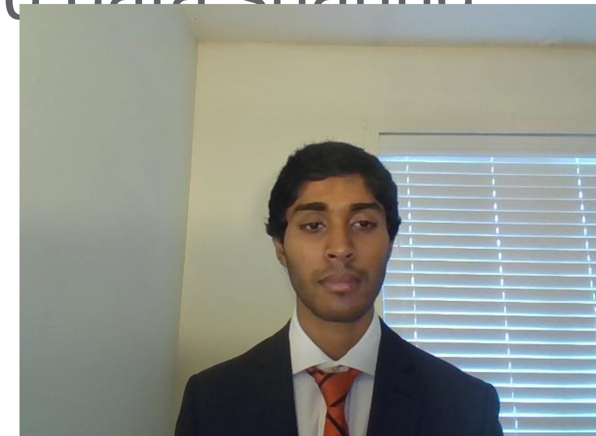
If Max Attempts > Num Devices – copy factor dominates performance



Pacific Northwest
NATIONAL LABORATORY

Conclusions + Future Work

- One of the primary challenges of using multiple devices is managing memory and coordinating data movements
- We introduce MCL Resident Memory to seamlessly manage device memory and coordinate data transfers
- Results demonstrate the importance of locality in achieving performance speedups
- Improves performance against other baselines
- Improve scheduler by dynamically determining hyperparameters
- In the future we want to expand this work to allow scheduling and data sharing among multiple applications



Pacific Northwest
NATIONAL LABORATORY

Thank you

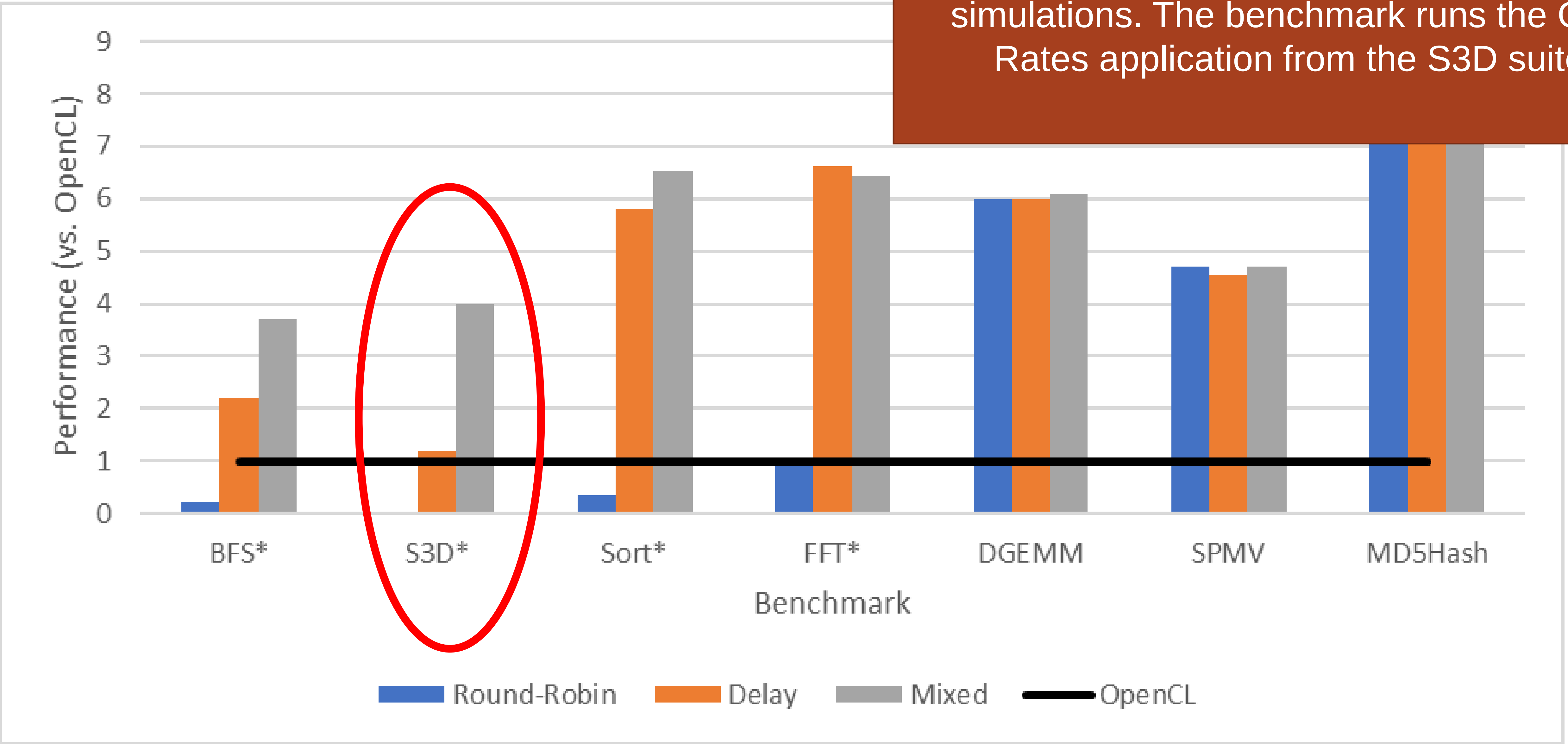
Alok Kamatar
Alok.Kamatar@pnnl.gov

Roberto Gioiosa
Roberto.Gioiosa@pnnl.gov

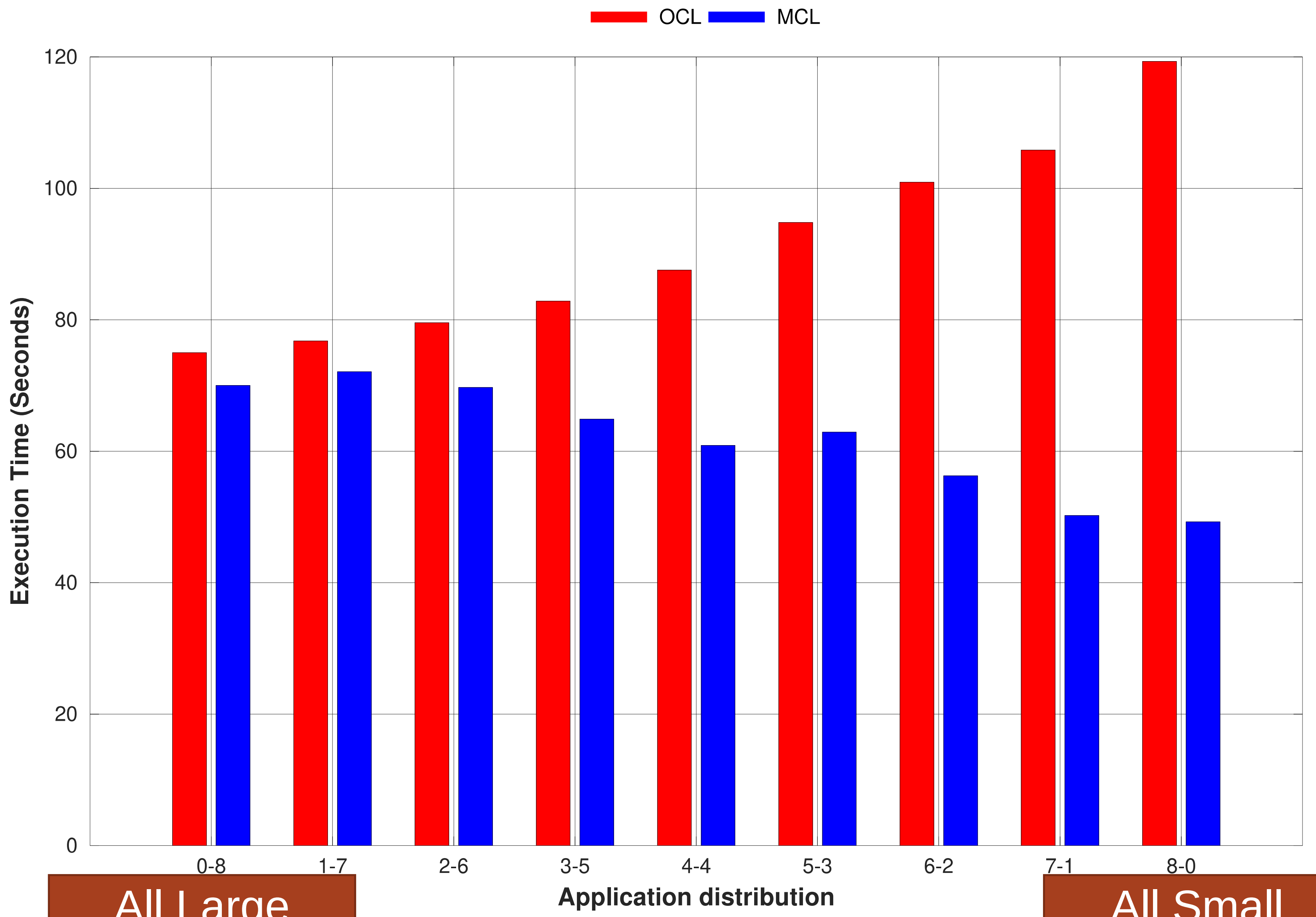
Additional Slides

Results With S3D

S3D is a standard direct numerical solver (DNS) used for combustion and other chemical simulations. The benchmark runs the Get-Rates application from the S3D suite



Application Composition



All Large Tasks

All Small Tasks

Experiment Details:

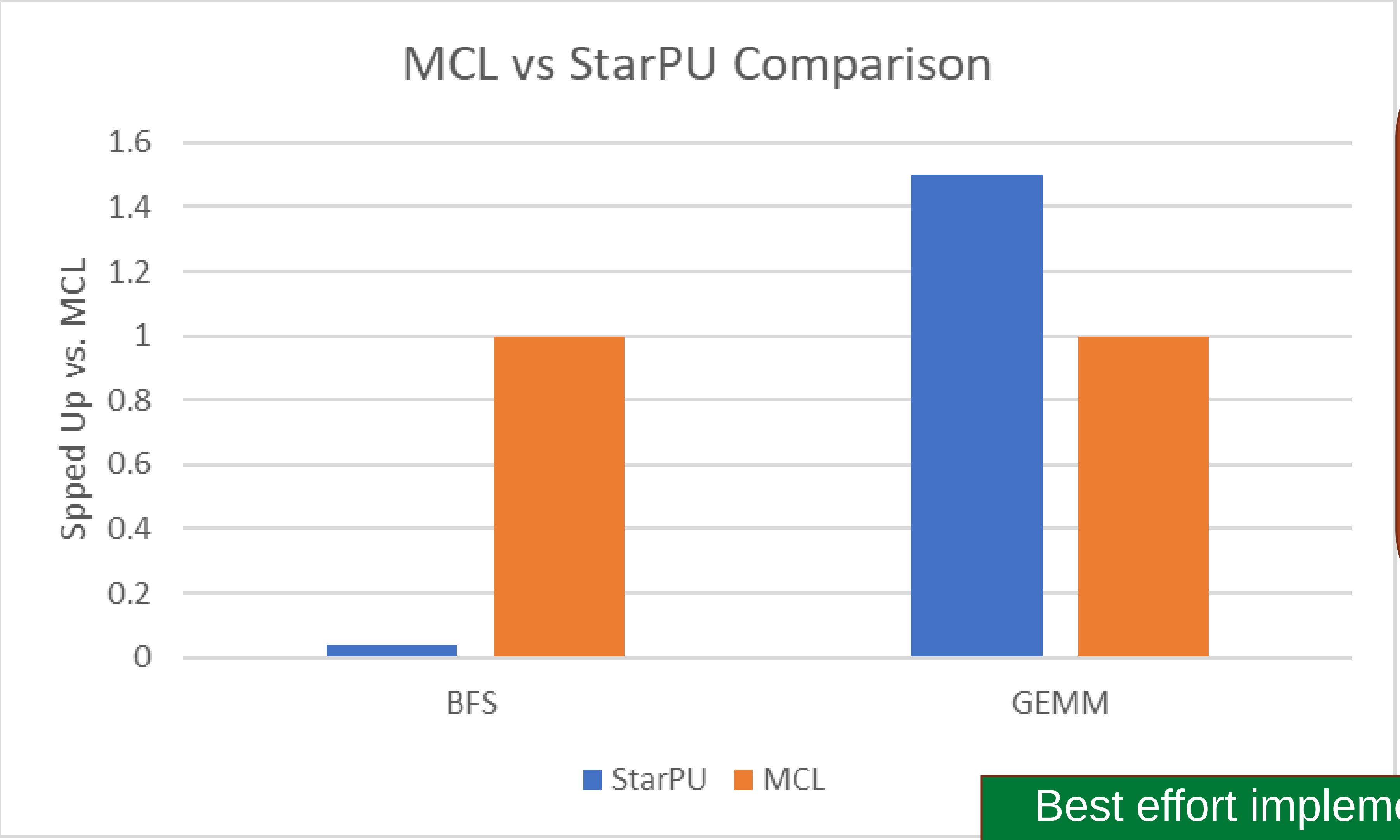
- OpenCL application partitioned among 8 devices
- A variable mix of small (64x64) and large (1024x1024) GEMMS

Eviction/Checkpointing

- Memory Usage is a limited resource that is under demand in a HPC system
- MCL supports flexible eviction policies that can be combined with scheduler policies
- When applications are unable to be run because no device has enough available memory, resident data can be evicted back to main memory
- To the user, MCL still behaves the same
- Currently supports a LRU policy

Pacific Northwest
NATIONAL LABORATORY

Comparison Against StarPU



StarPU Scheduling LWS:

- Task is automatically scheduled on worker that released it
- Idle workers use data transfer performance estimates to determine weather to move data and “steal” work

Best effort implementation of StarPU – still investigating performance discrepancy