Pacific Northwest
NATIONAL LABORATORY

MCL Multi-GPU Support + Schedulers

PPoPP '21

Alok Kamatar, Ryan Friese, Roberto Gioiosa

U.S. DEPARTMENT OF ENERGY BATTELLE

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

Pacific Northwest NATIONAL LABORATORY

Use Cases

- Translating OpenCL code from one device to multiple:

```
For dev in devs:
    clCreateContext ...
    clCreateProgramWithSrc ...
    clCreateCommandQueueWithProps ..
    clCreateKernel ...
    //Create a buffer for each array
    //etc.
```

- Lots of complexity overhead for each device
- Coordinate which devices are busy and idle
- Coordinate memory between devices

Pacific Northwest NATIONAL LABORATORY

Use Cases

- Translating MCL Code from single device to multiple devices

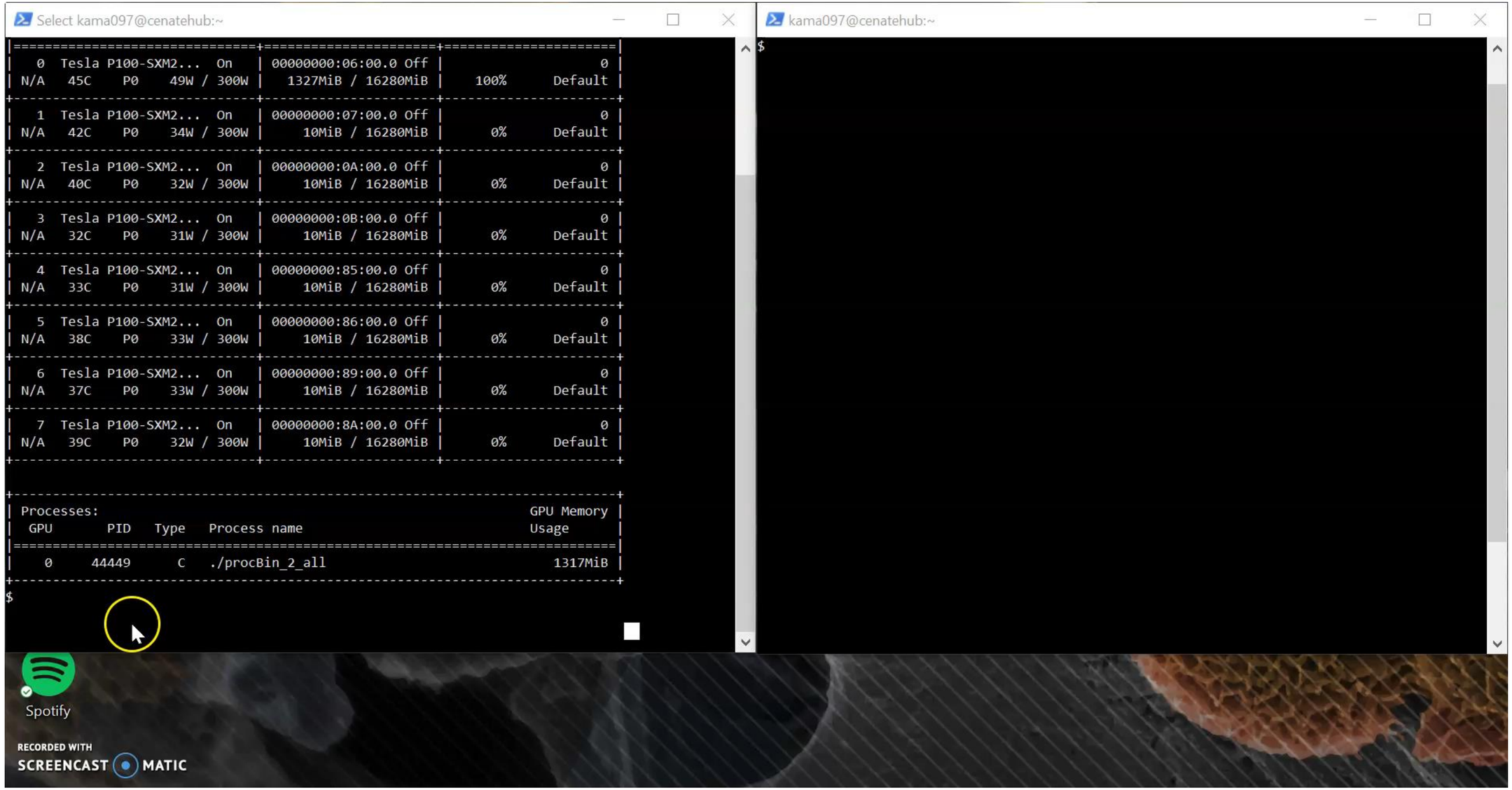
```
mcl_exec(task, pes, block_size, MCL_TASK_GPU);
```

```
mcl_exec(task, pes, block_size, MCL_TASK_GPU);
```

- Same code runs on all available resources
- MCL takes care of and hides complexity overhead

Pacific Northwest NATIONAL LABORATORY

Example – Easy Scaling



Pacific Northwest
NATIONAL LABORATORY

Running Commands

- To replicate the results in the video, you can run these commands on your own GPU enabled system

```
>>> mcl_sched -s fffs -p rr &
>>> cd mcl/test
>>> make mcl_gemm
>>> ./mcl_gemm -r 8092 -w 16
-----------------------------------------
>>> export CUDA_VISIBLE_DEVICES=1,2,3,4,5,6,7
>>> mcl_sched -s fffs -p rr &
>>> cd mcl/test
>>> make mcl_gemm
>>> ./mcl_gemm -r 8092 -w 16
```

Commands used for DGX-1 demo 1

Pacific Northwest NATIONAL LABORATORY

MCL Schedulers

- MCL scheduler is a separate process that determines where and when tasks are run
- Flexible – i.e. different scheduler policies divided into too classes
  - First In First Out – The first task in the queue must be scheduled first,
  - FFFS scheduler – Allows the resource scheduler to select and of the currently enqueued tasks
- Different Policies
  - First Fit
  - Round Robbin
  - Delay Scheduler
  - Hybrid Scheduler

```
$ ~/local/bin/mcl_sched -h
Usage: /home/kama097/local/bin/mcl_sched [options]
        -s, --sched-class {fifo|fffs}  Select scheduler class (def = 'fifo')
        -p, --res-policy {ff|rr|delay|hybrid}  Select resource policy (def = class dependant)
        -h, --help                     Show this help
$
```

First Fit Scheduler

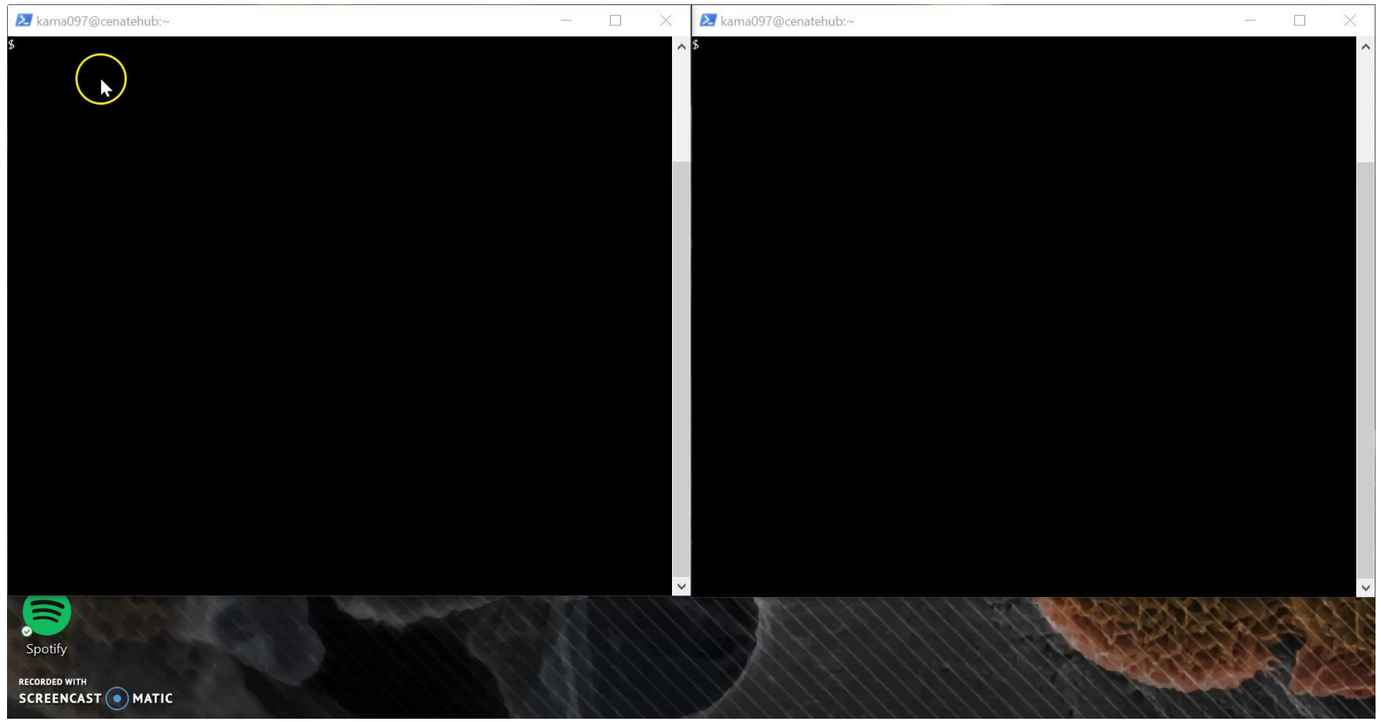
- Schedules the next available task onto the current device if there is space
- Only moves onto the next device if the current device is full
- Space is determined by both memory and available processing threads
- Low Overhead

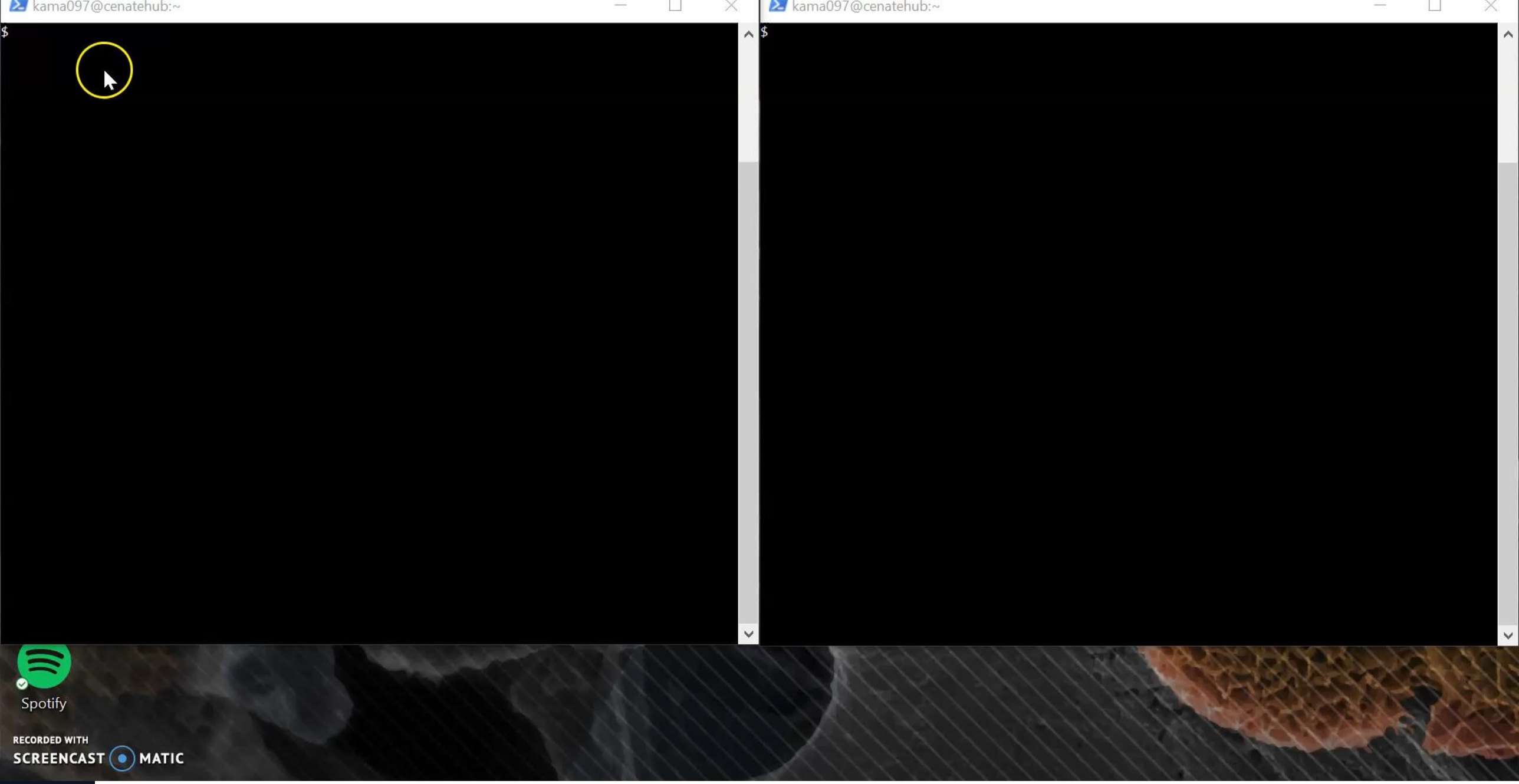
```
>>> mcl_sched -s fffs -p ff &
```

Pacific Northwest
NATIONAL LABORATORY

To Run First Fit Scheduler with Trace

```
>>> make clean
>>> ./configure --enable-trace
>>> make && make install
>>> mcl_sched -s fffs -p ff &
>>> cd test
>>> make mcl_gemm
>>> ./mcl_gemm -r 8092 -w 16
```

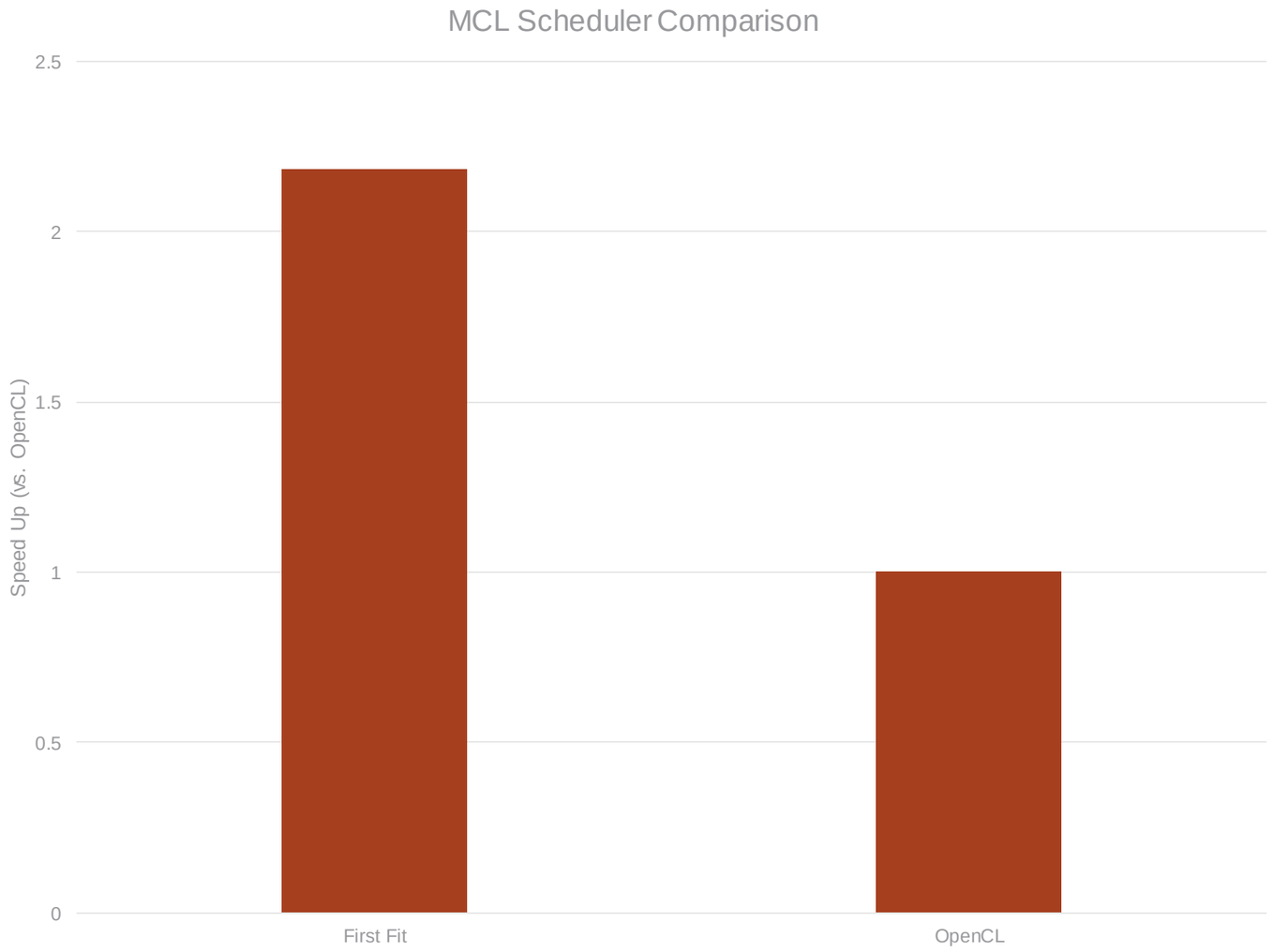



kama097@cenatehub:~
$
kama097@cenatehub:~
$
Spotify
RECORDED WITH
SCREENCAST MATIC

First Fit Scheduler Results

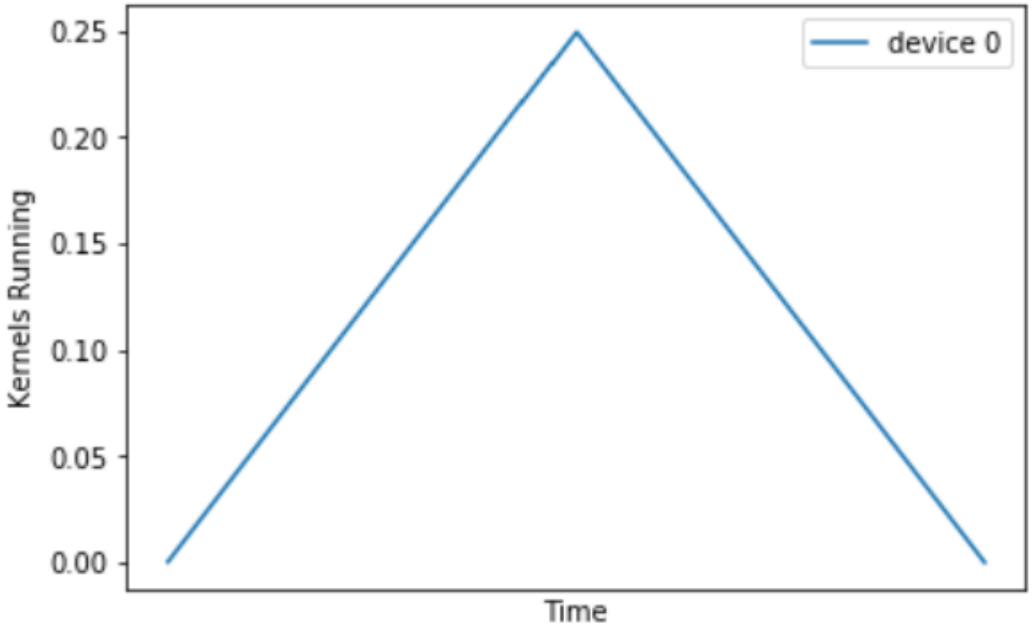
DGEMM Benchmark:
- 64 X 64 matrices
- 1024 tasks

Nvidia DGX-1 System:
- 2 Intel Xeon CPUs
- 8 Nvidia Pascal GPUS
- 256 GB memory + 16 GB per GPU memory



Pacific Northwest
NATIONAL LABORATORY

Scheduler Trace



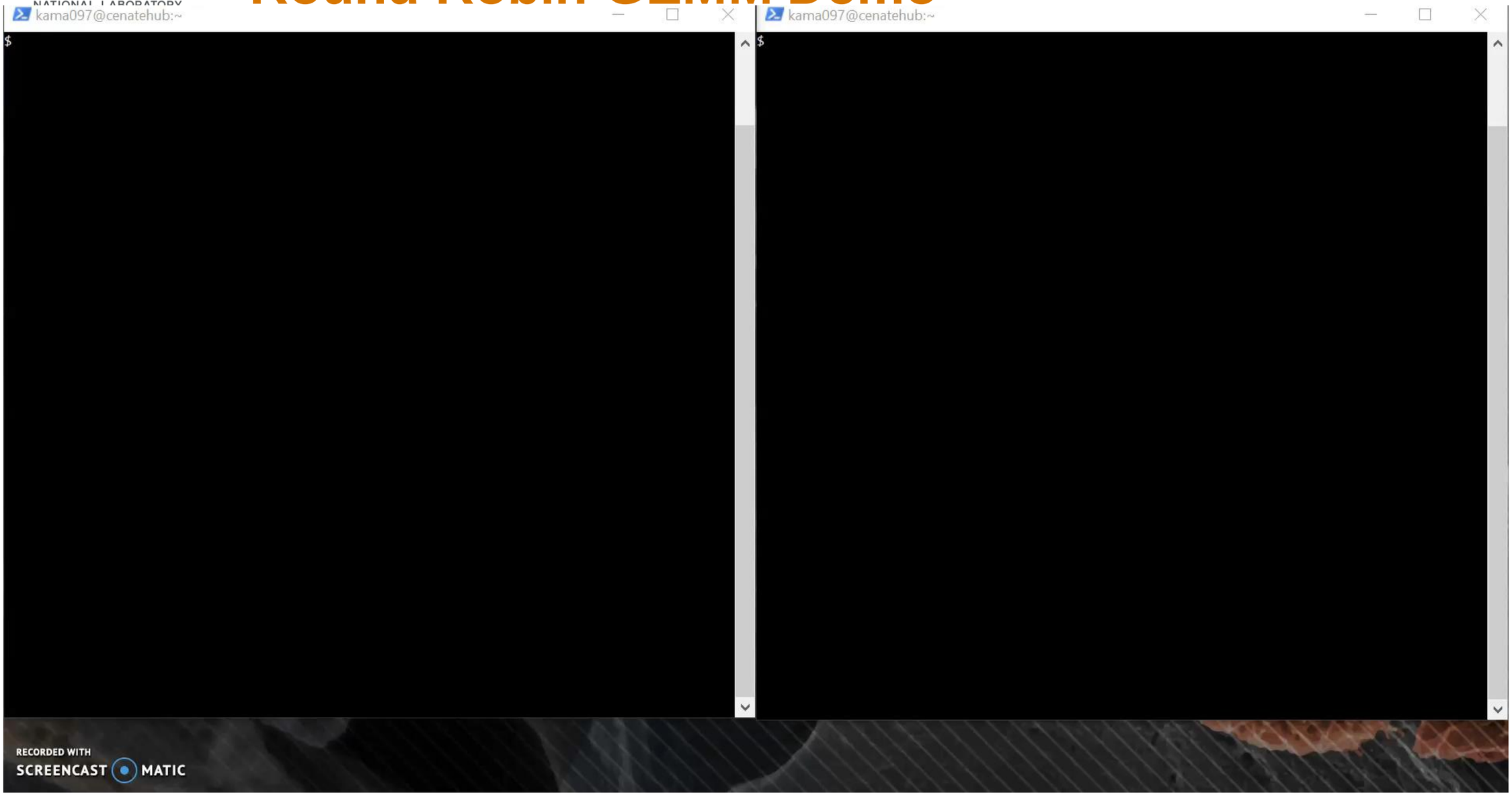
- MCL scheduler trace can be enable by compiling with `—enable-trace`
- Allows us to visualize memory use, processing element use, and number of busy tasks on each device
- Where are the rest of the GPUS?
  - Some GPUs are heavily utilized; rest of the GPUs are under utilized

Round Robin Scheduler

- Schedules tasks to devices in a circular manner
- Maintains a queue of devices.
  - Each incoming task gets assigned to the next device in the queue that is compatible
  - That device is moved to the back of the queue
- Typically achieves good full-system utilization

```
>>> mcl_sched -s fffs -p rr &
```

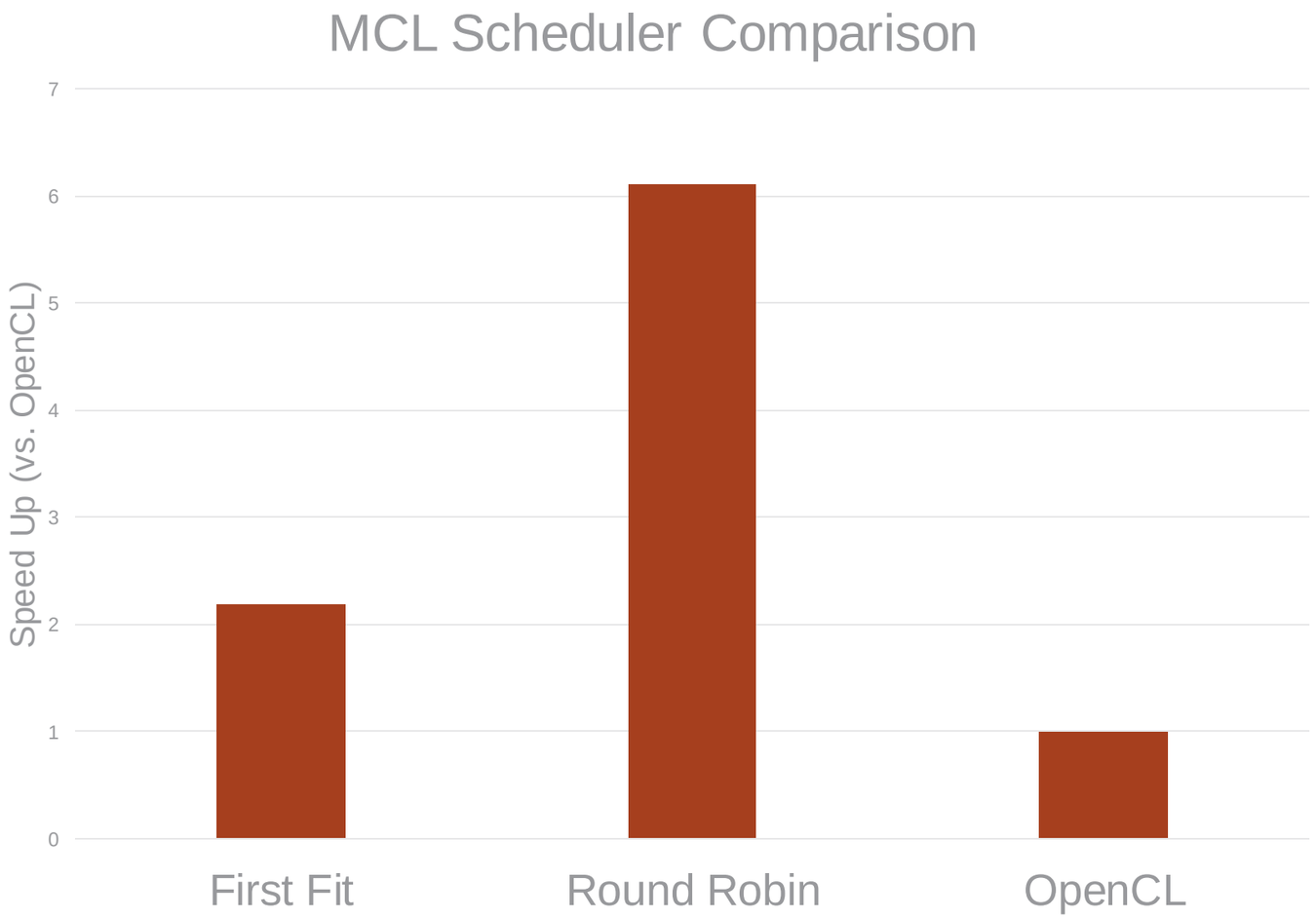
Round Robin GEMM Demo



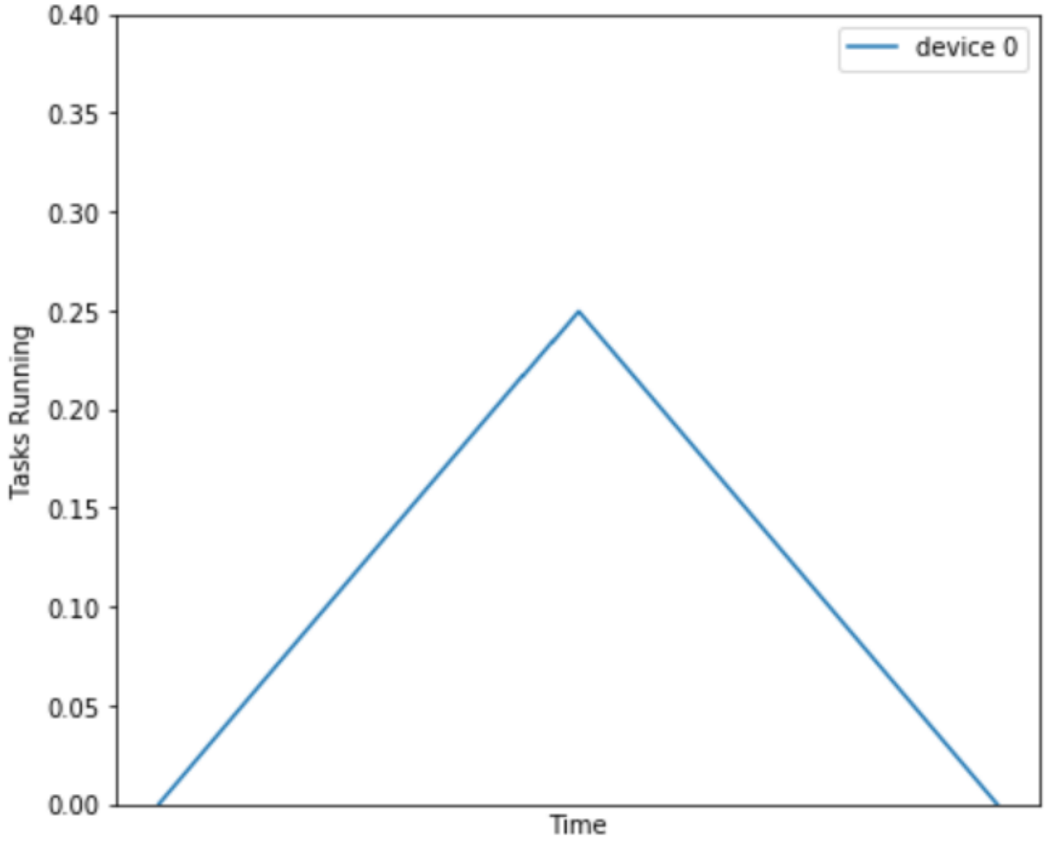
Round Robin Scheduler Results

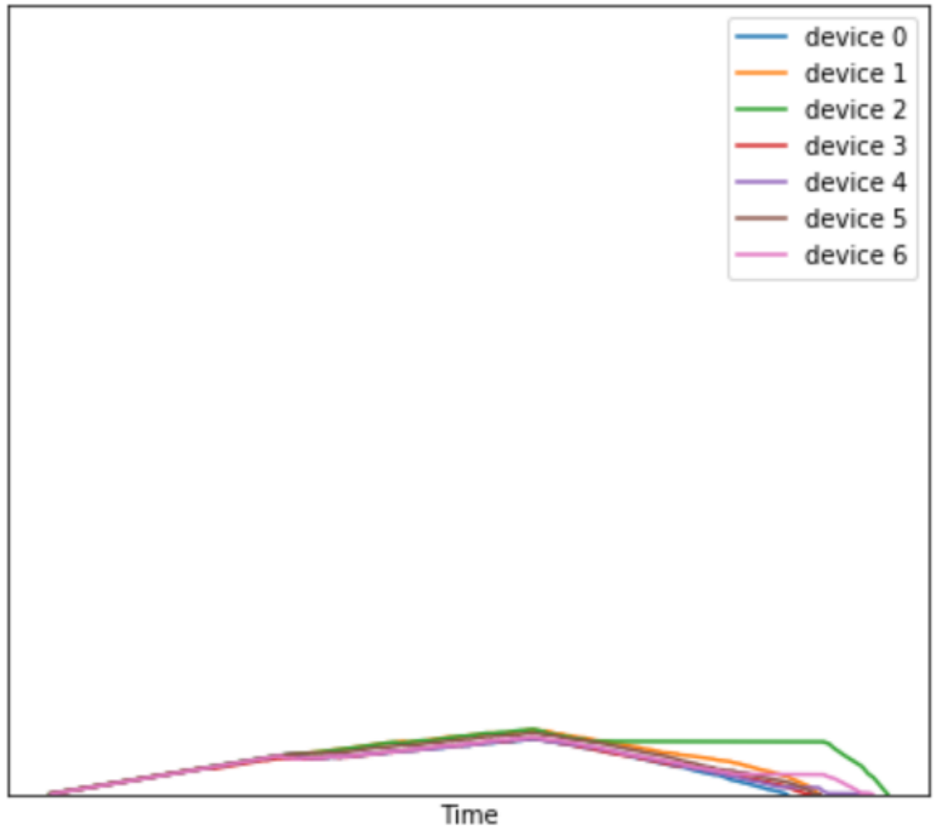
DGEMM Benchmark:
- 64 X 64 matrices
- 1024 tasks

Nvidia DGX-1 System:
- 2 Intel Xeon CPUs
- 8 Nvidia Pascal GPUS
- 256 GB memory + 16 GB per GPU memory



Round Robin vs. First Fit





The load that is just on GPU 0 in the First Fit scheduler is now balanced

Pacific Northwest
NATIONAL LABORATORY

Another Problem

- Breadth First Search
  - Graph is represented in two arrays: one storing an adjacency list, one storing the offsets of each vertex – read only data structure
  - On each iteration i:
    - ✓ The frontier has an array of vertices that distance I from the source
    - ✓ The cost array has best known cost for each vertex
    - ✓ "Explore" indices that are i+1 away from the source using the frontier array
- Each iteration is a new task reusing the same data from the previous tasks
- Memory transfers dominate compared to computation

Pacific Northwest NATIONAL LABORATORY

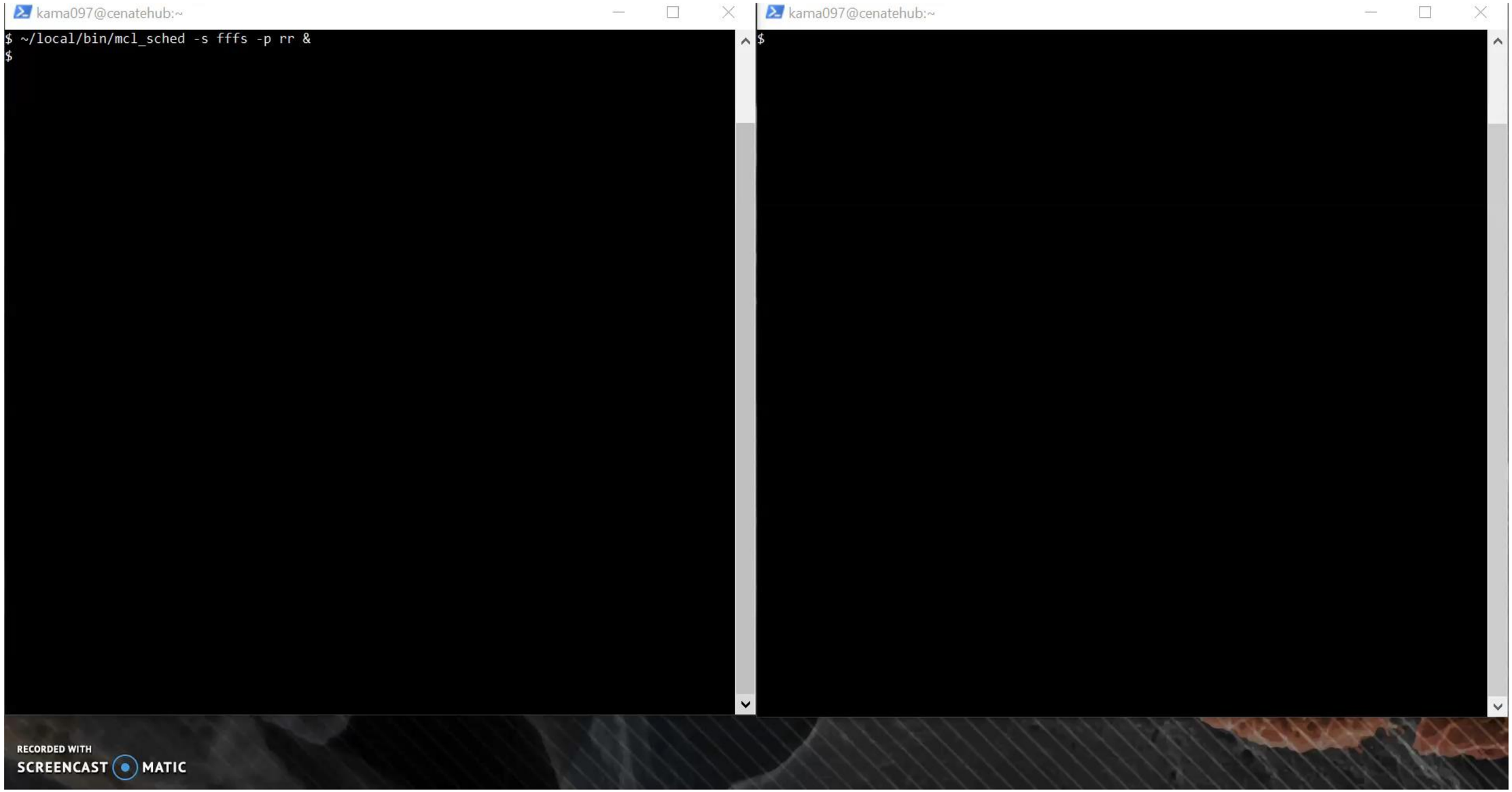
Breadth First Search

- Breadth First Search Code per iterations:

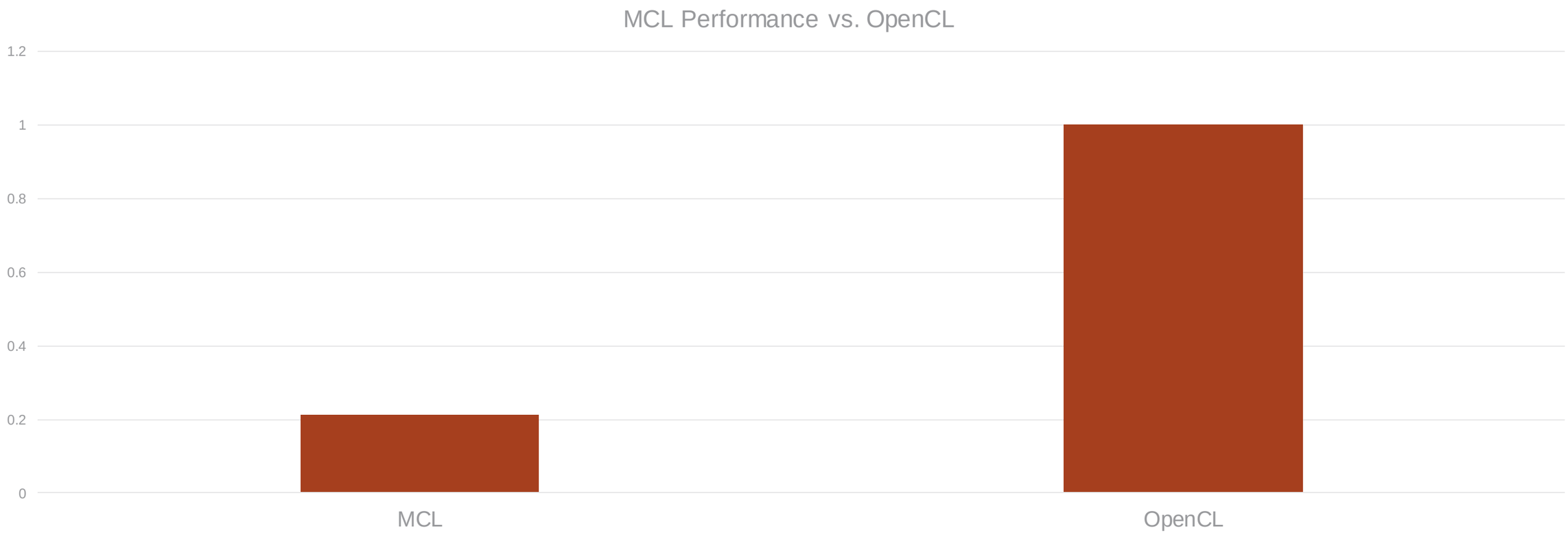
```
mcl_handle* hdl = mcl_task_create();
mcl_task_set_kernel(hdl, kernel_path, "BFS", 8, "", 0x0);
mcl_task_set_arg( hdl, 0, (void*)frontier, ..., flags);
mcl_task_set_arg(hdl, 1, (void*)edge_offsets, ..., flags);
mcl_task_set_arg(hdl, 2, (void*)edge_list, ..., flags);
...
mcl_task_set_arg(hdl, 6, (void*)&iters, sizeof(uint32_t), MCL_ARG_SCALAR);
mcl_exec(hdl, pes, lsize, MCL_TASK_ANY);
```

Pacific Northwest NATIONAL LABORATORY

Running Breadth First Search



Breadth Fist Scheduler v1



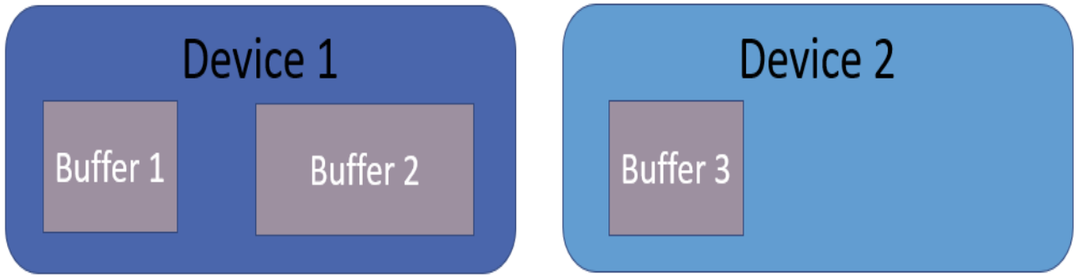
Why so slow?

- Graph is naively transferred for each iteration

MCL Resident Memory Module

- Allows persistent data to remain in device memory across tasks
- Orchestrates 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
…	…	…	…	…	…



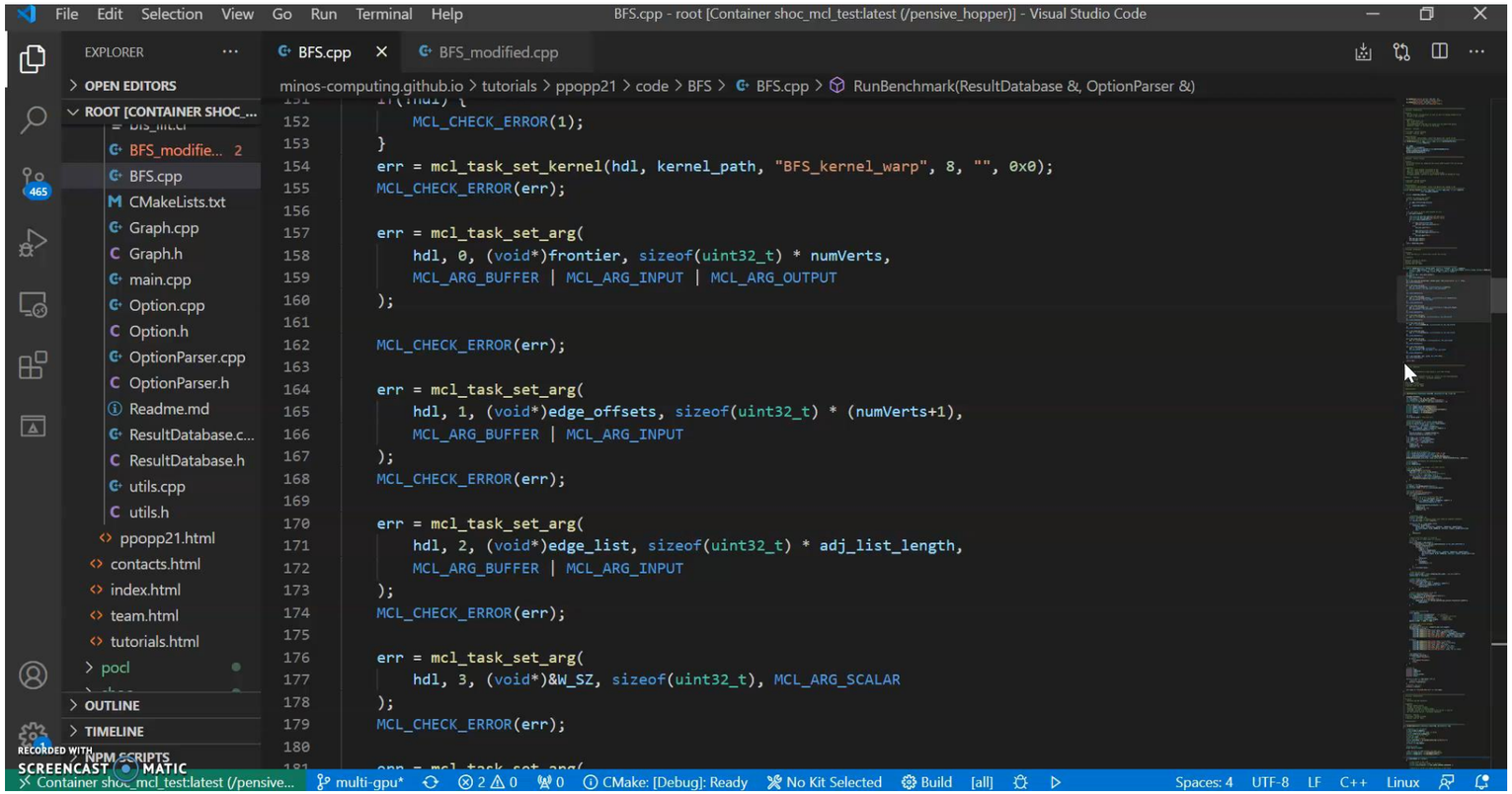
MCL_ARG_RESIDENT

MCL_ARG_INVALID

MCL_ARG_DONE

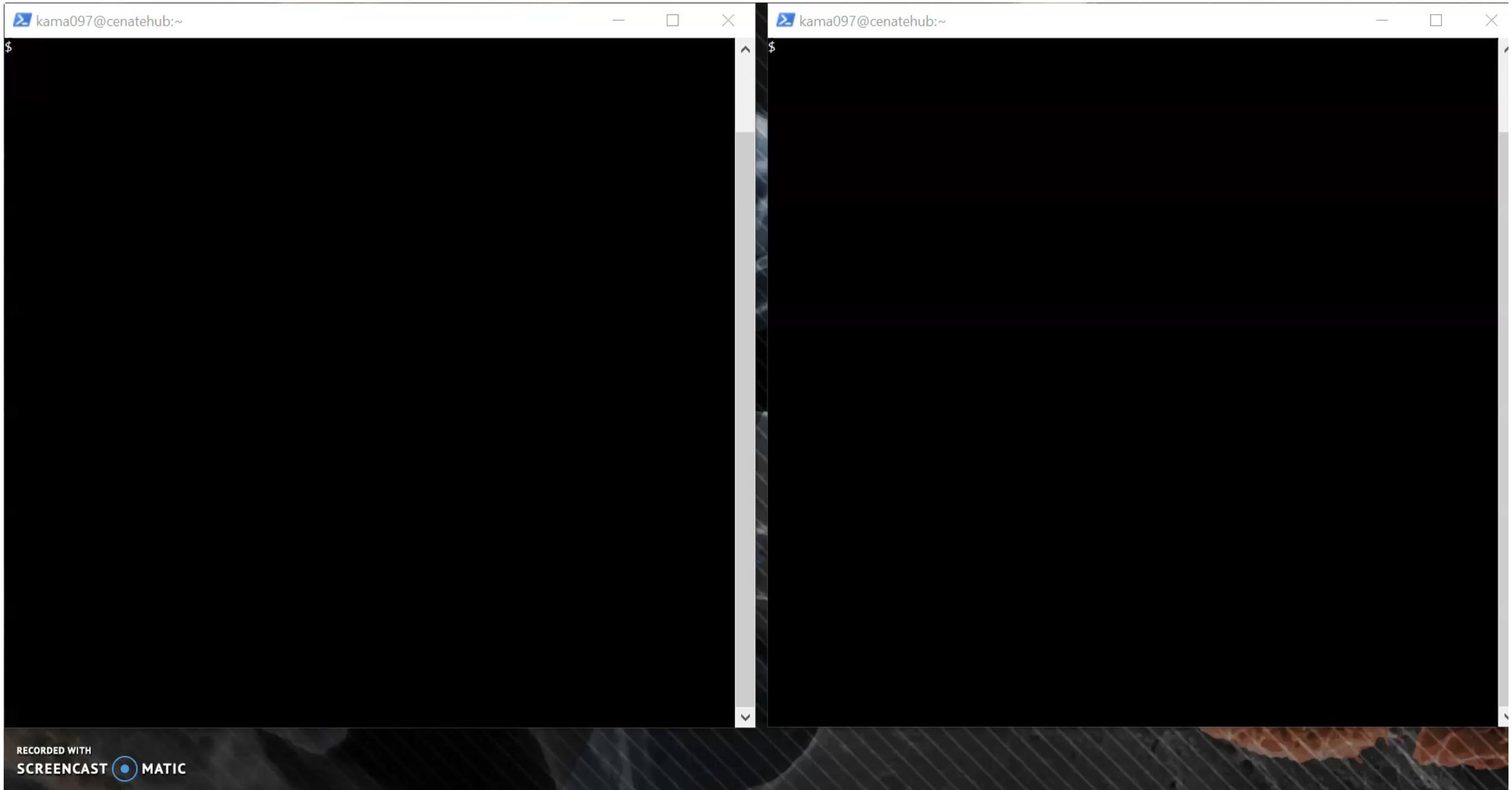
Code Modification Demo

- Show (live?) modifications from non-resident memory to resident memory.
- If you are following along in the code, this is the difference between BFS.cpp and BFS-modified.cpp

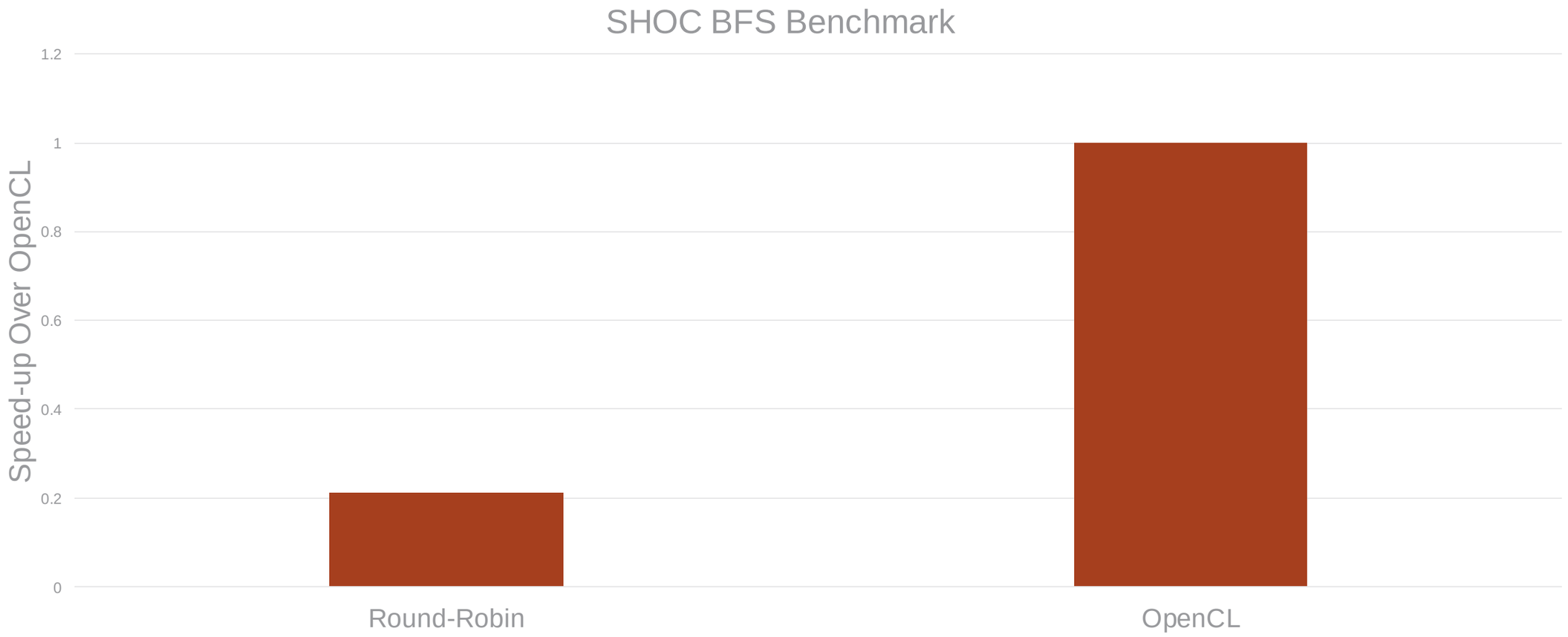


Pacific Northwest
NATIONAL LABORATORY

Running The New Code



Breadth First Search v2



Still so slow?

- Different frontiers must be transferred from other devices

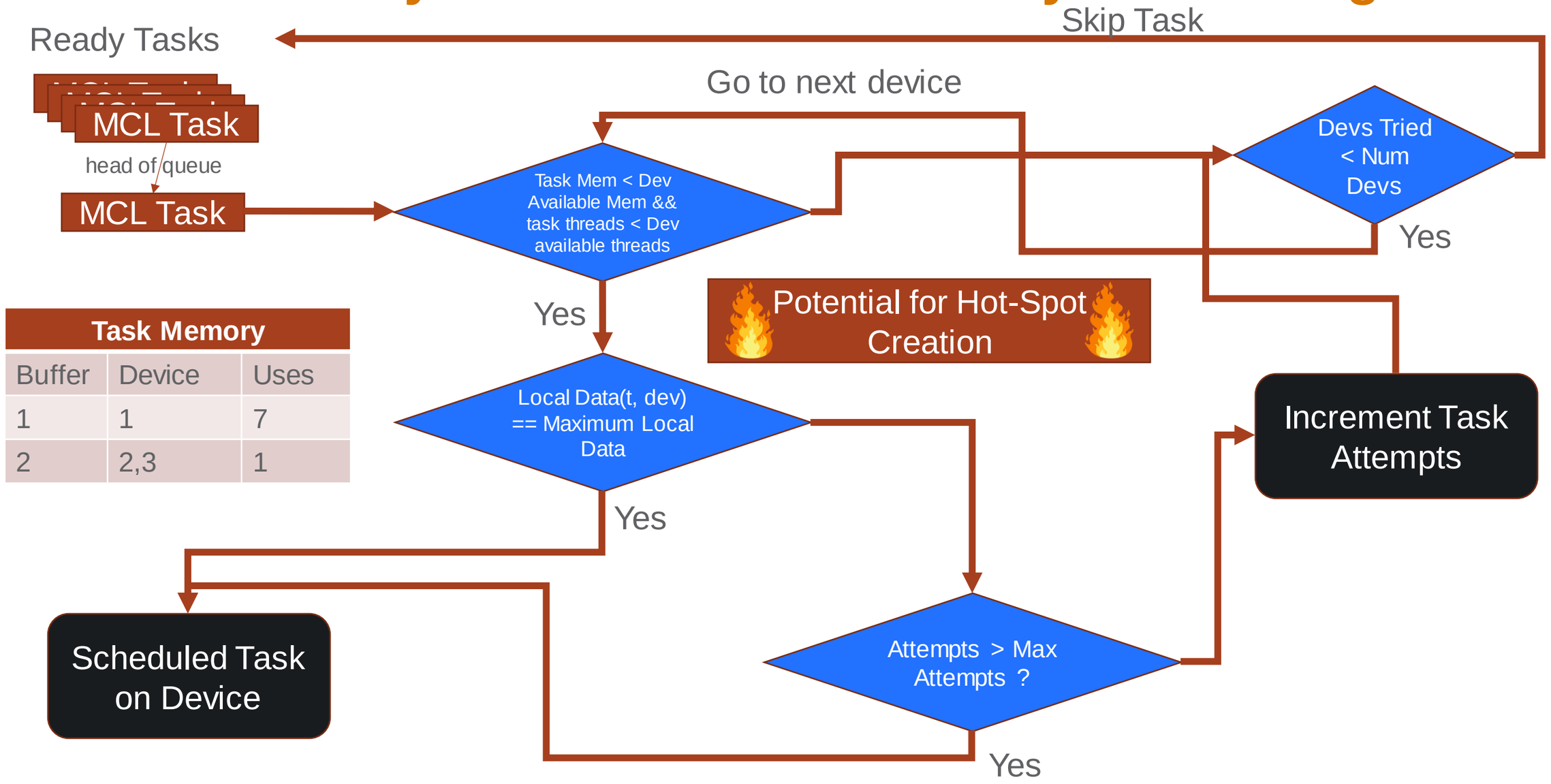
Locality Aware Scheduler: Delay Scheduling

- Scheduler needs to choose the optimal device based off where resident memory is located
- Delay Scheduling:
  - 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
  - "Local data" is required data that is currently on a device

Algorithm 1 Device Local Data

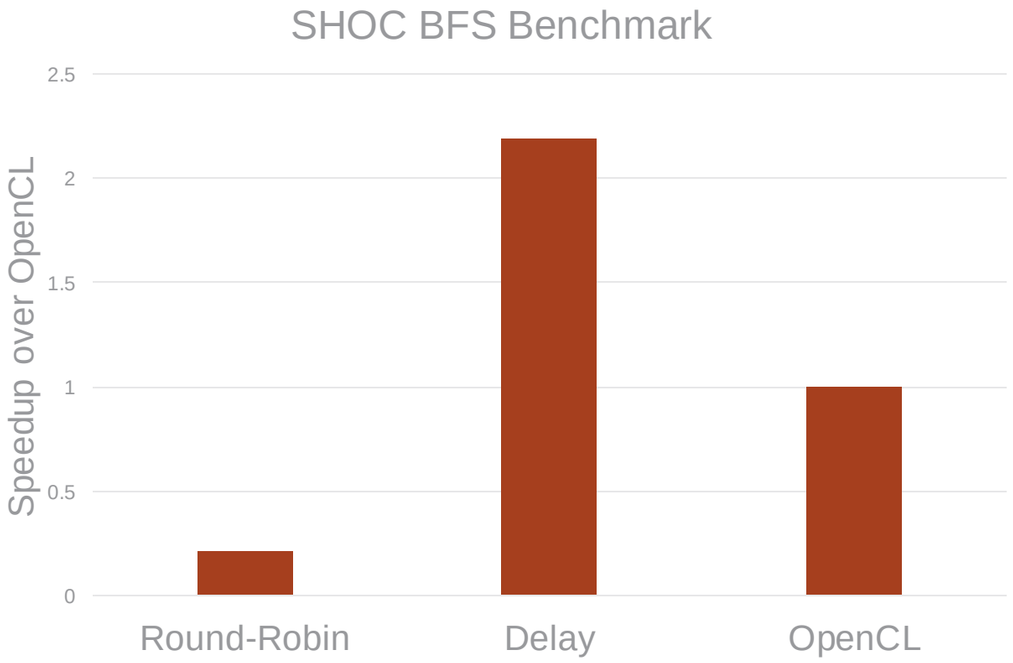
```
1: LocalData(Device δ, Task t):
2: bytes ← 0
3: for all β in t.buffers do
4:     if β in δ.data then
5:         bytes ← bytes + β.size
6:     end if
7: end for
8: return(bytes)
9: end LocalData
```

Locality Aware Scheduler: Delay Scheduling

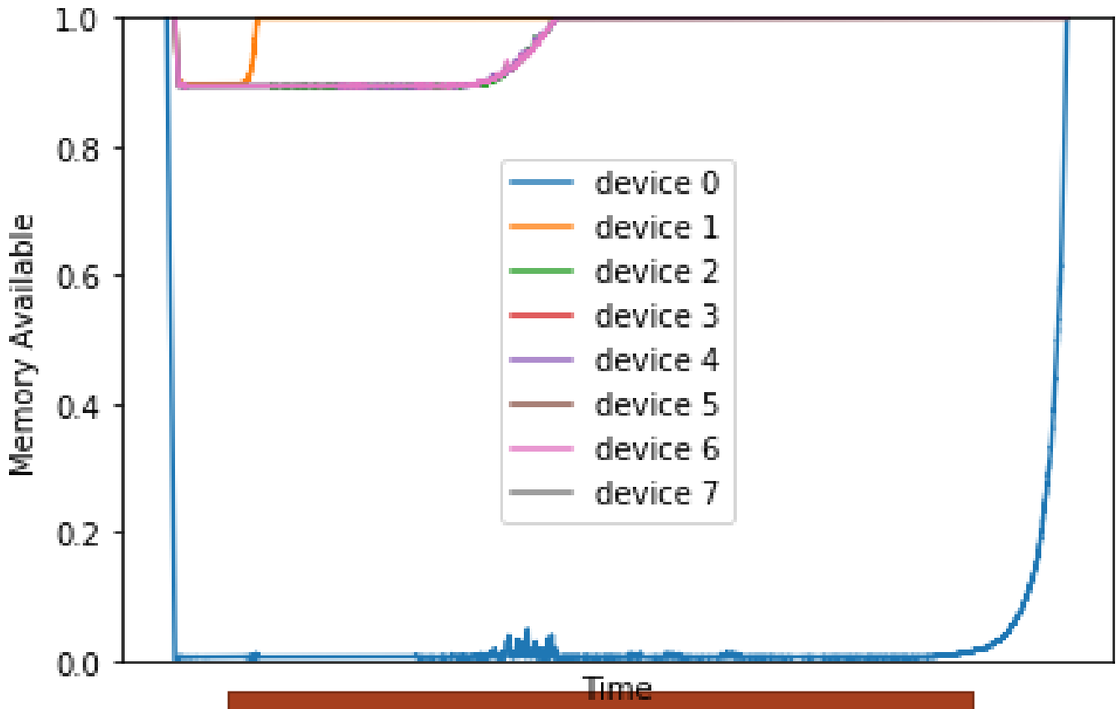


Task Memory		
Buffer	Device	Uses
1	1	7
2	2,3	1

Delay Scheduler Results



Performance Improves! But can we do better...?



MCL scheduler trace reveals that one device is heavily used

Hybrid Scheduler

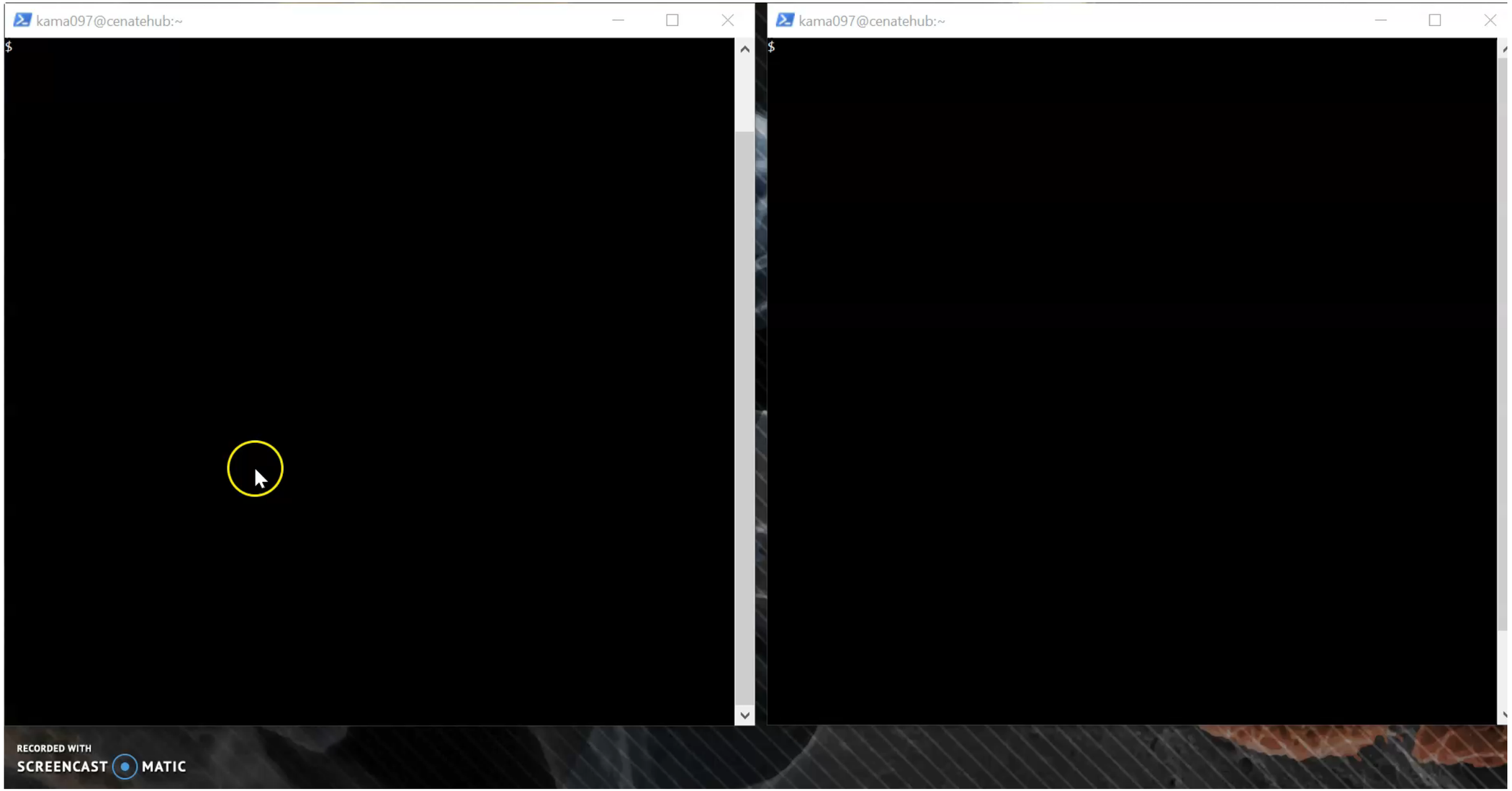
- Needed to balance locality concerns against system utilization=
- Detects popular pieces of data to create replicas – done by changing how we calculate local data
- Hyperparameters are controlled with environment variables: MCL_SCHED_MAX_ATTEMPTS and MCL_SCHED_COPY_FACTOR

Algorithm 3 Device Local Data With Copy Factor

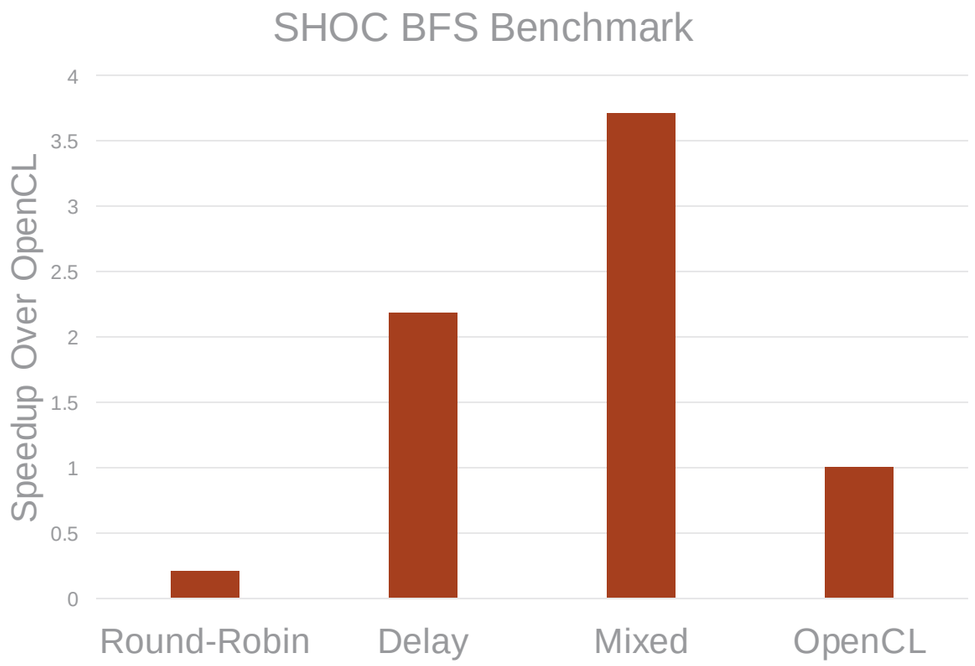
```
1: LocalDataCopyFactor(Device δ, Task t, CopyFactor γ):
2: bytes ← 0
3: for all β in t.buffers do
4:     if β in δ.data and DEVICES(β) ≥ log_γ TASKS(β) then
5:         bytes ← bytes + β.size
6:     end if
7: end for
8: return(bytes)
9: end LocalDataCopyFactor
```

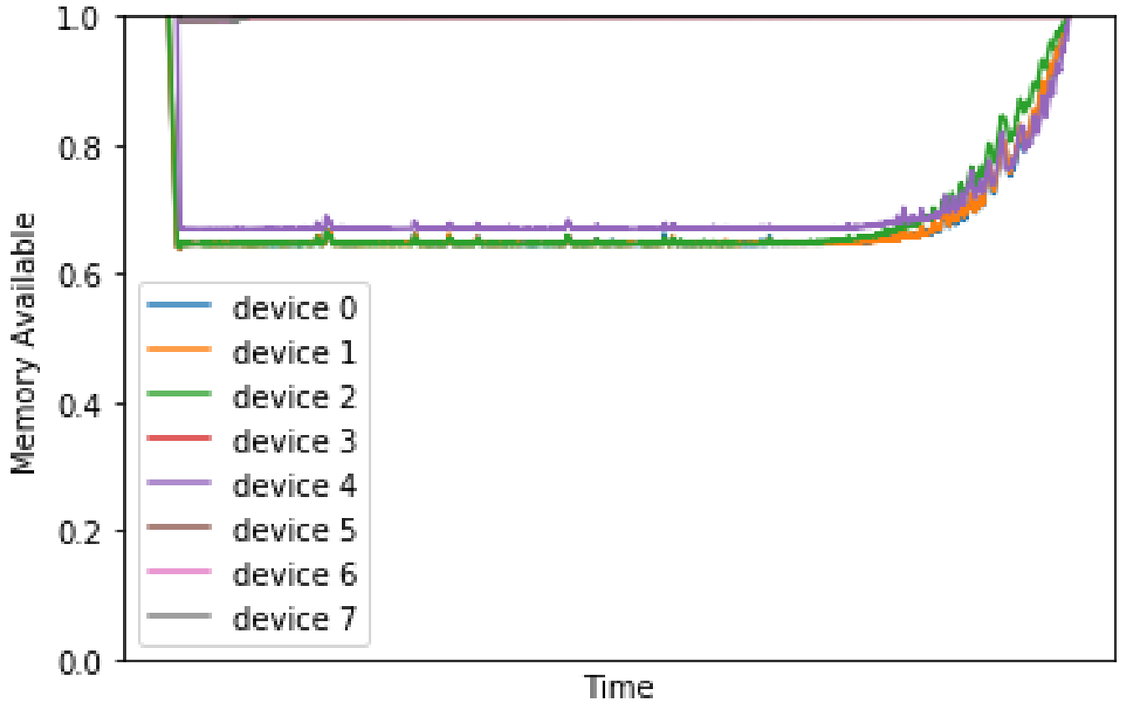
Pacific Northwest
NATIONAL LABORATORY

BFS – Final Version Demo



Breadth First Search Results

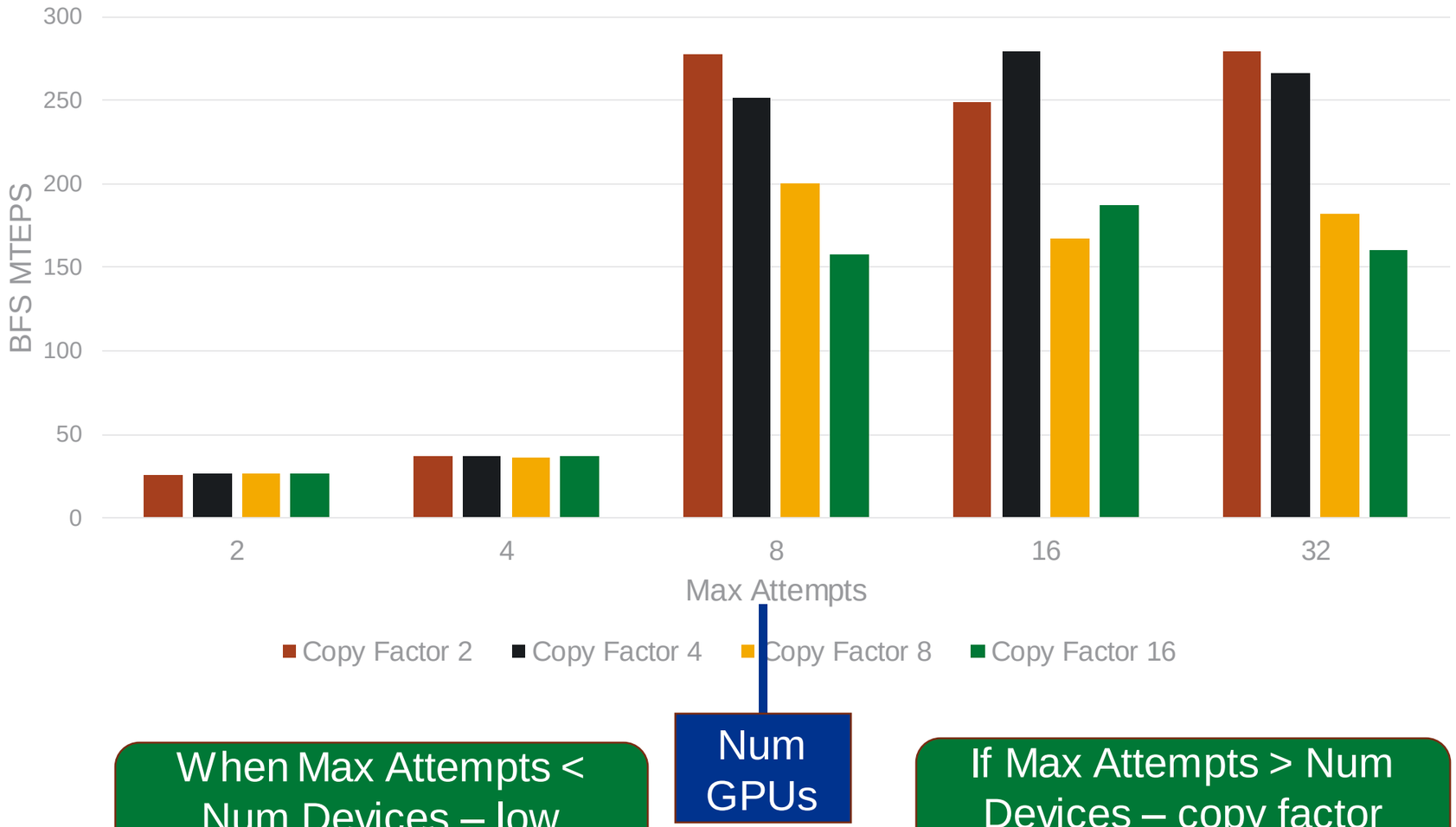




Best performance!

MCL scheduler trace reveals balance among first 4 devices

Effect of Hyperparameters on Performance



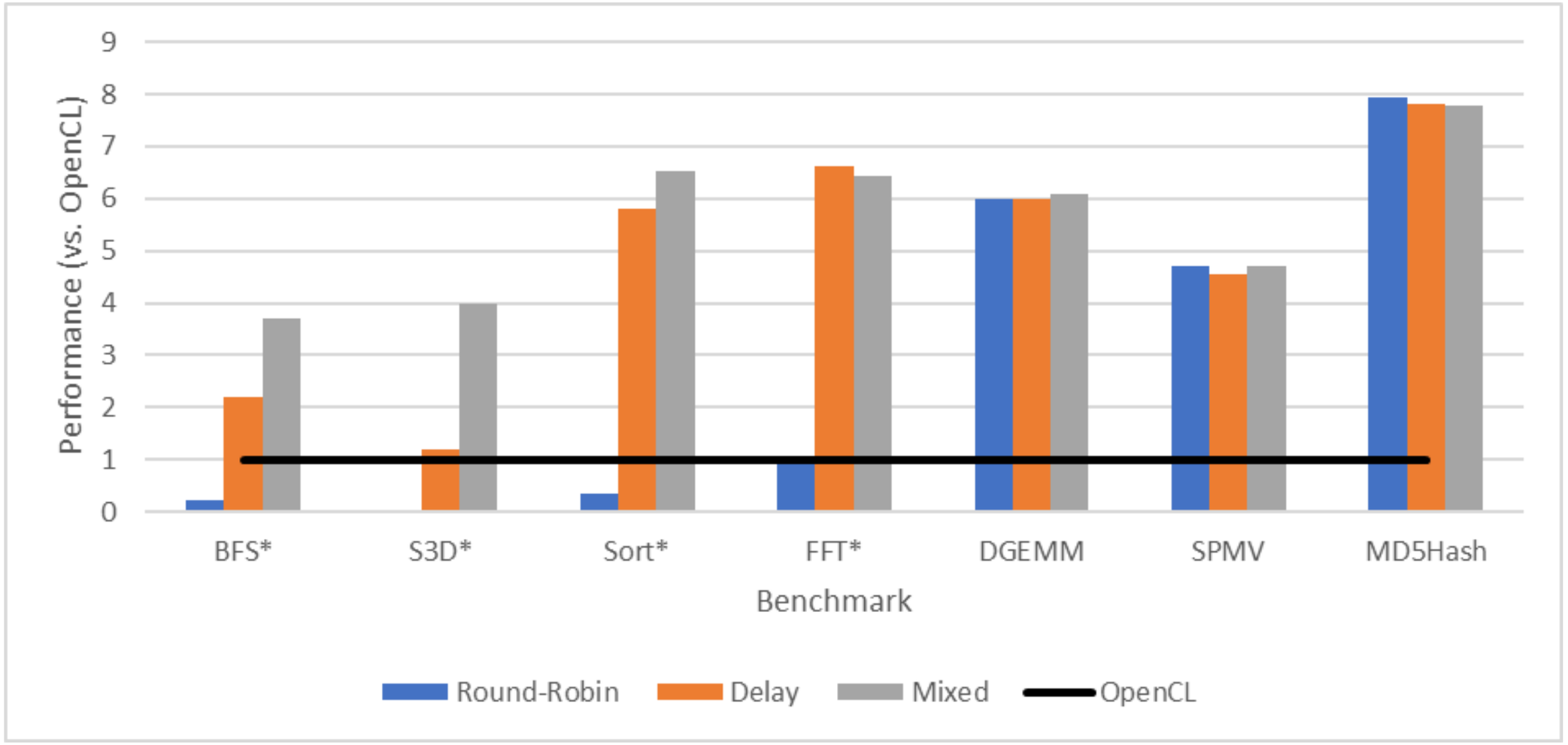
BFS Benchmark
- 1,000,000 vertices
- 4096 Tasks

When Max Attempts < Num Devices – low performance

If Max Attempts > Num Devices – copy factor dominates performance

Pacific Northwest NATIONAL LABORATORY

Full Scheduler Comparison



Writing Your Own Scheduler

- Certain applications require specific requirements from the scheduler
- `init(mcl_resource_t* r, int ndevs)` – initialize representation of resource and any other representation needed
- `find_resource(sched_req_t r)` – Find the device r should run on. Also, set `r->dev` to the assigned device
- `assign_resource (sched_req_t r)` – Allocate resources in resource to the scheduled device
- `assign_resource (sched_req_t r)` – Release resources in resource model from scheduled device

Eviction Policy

- 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

Upcoming Work – Multi-Application Scheduling

- Different applications needs to be run in a pipeline
  - A physics simulation -> a data analysis application to detect events
  - A MD-simulation guided by reinforcement learning
  - Combustion Simulation + Machine Learning
  - Etc.
- Currently in 2 ways:
  - Modify exiting application -> specific to each application, lots of extra work,
  - Leverage Files

Pacific Northwest
NATIONAL LABORATORY

Shared Memory Design

- Leverage POCL to create an additional OpenCL interface that allows buffers to be shared
- Scheduler is aware of where necessary data is even across applications
- The same data does not have to transferred again in different applications
- Patterns Supported:
  - Scratch Pad
  - Producer-Consumer
  - Circular Buffer

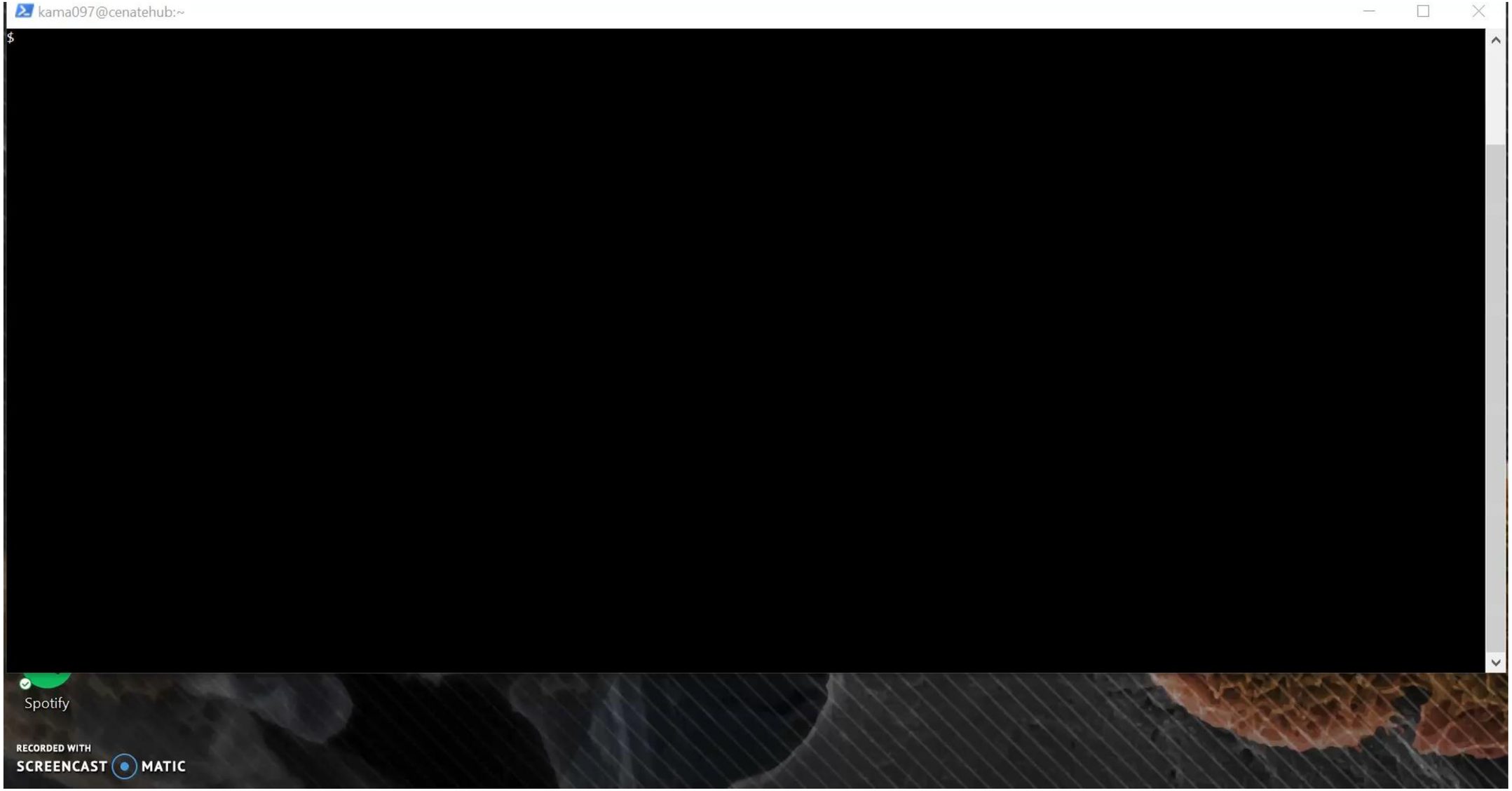
Pacific Northwest
NATIONAL LABORATORY

Pipelining Applications Code

- Experimental Application
  - Producer performs an arbitrary number of floating-point operations on a buffer
  - Consumer reads producers buffer and performs its own operations
- Comparisons:
  - MCL Shared memory
  - OpenCL – File
  - OpenCL – POSIX Shared Mem + Pipe (Statically Partitioned GPUs)

Pacific Northwest
NATIONAL LABORATORY

Pipelined Applications (Example)



Pacific Northwest
NATIONAL LABORATORY

Thank you