SCALABLE COMPUTING
SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY

Harmonia: An Interference-Aware Dynamic I/O Scheduler for Shared Non-Volatile Burst Buffers

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun, and Jay Lofstead*
Illinois Institute of Technology, *Sandia National Laboratories
akougkas@hawk.iit.edu

Cluster'18
Belfast, UK
September 12th, 2018

In collaboration with Sandia National Laboratories

Highlights of this work

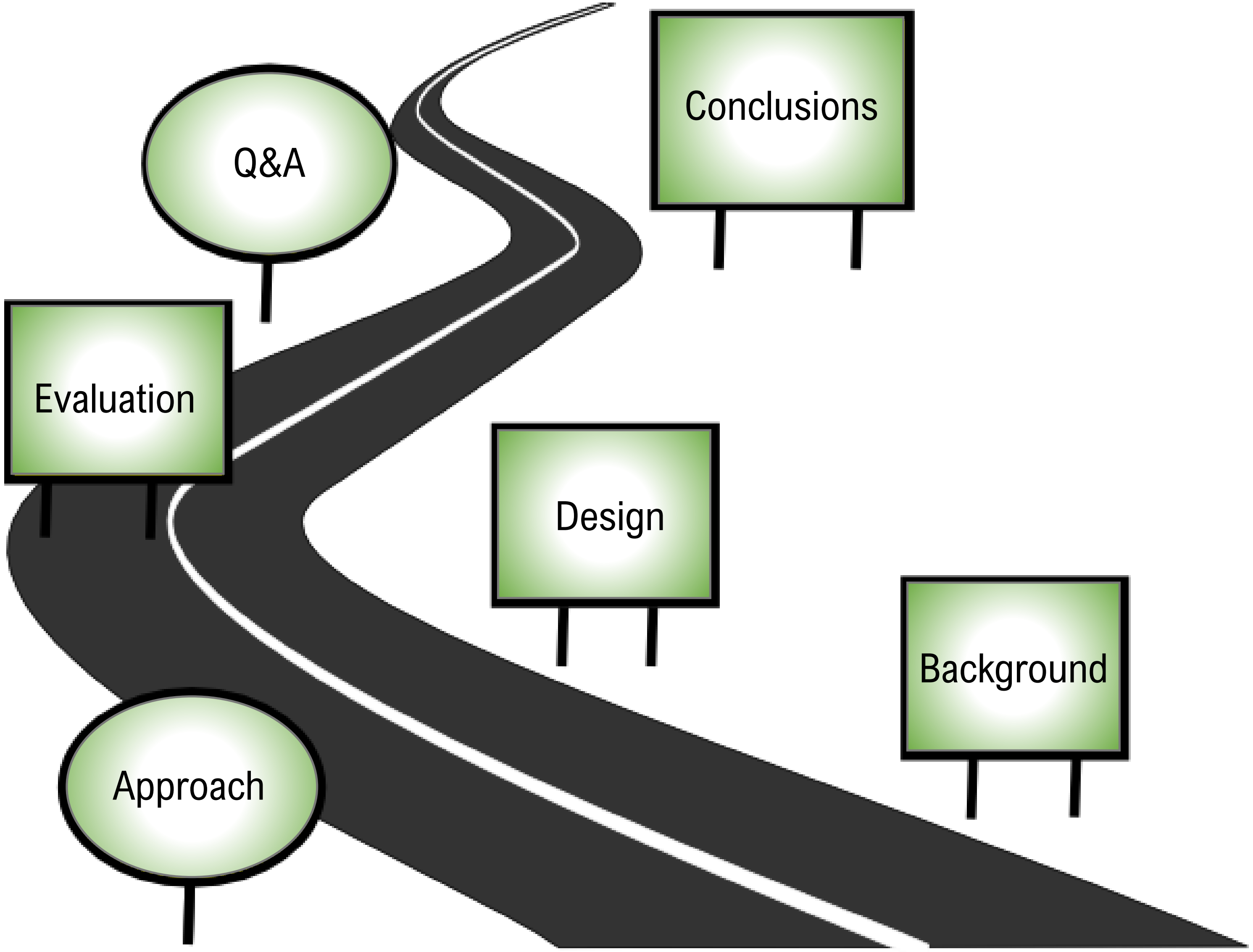
2-way adaptive burst buffer scheduler
- Interference Aware
- Dynamic in nature

5 new I/O scheduling policies
- Minimize cross application interference
- Maximize resource utilization

Harmonia can offer
- **3x** performance improvements
- Optimize the efficiency of buffers

Anthony Kougkas, akougkas@hawk.iit.edu

Agenda

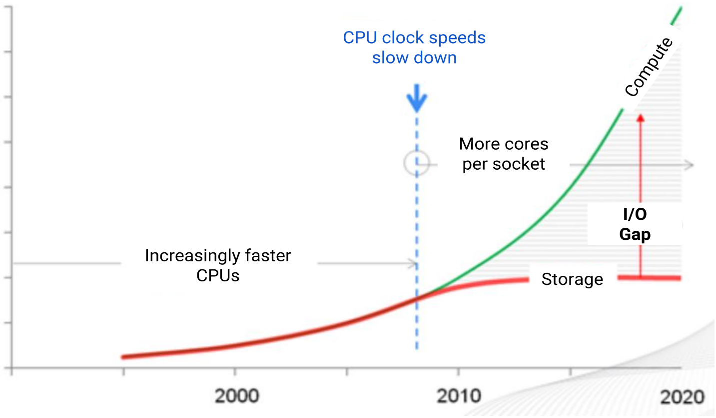


Talk roadmap

Anthony Kougkas, akougkas@hawk.iit.edu
ILLINOIS INSTITUTE OF TECHNOLOGY

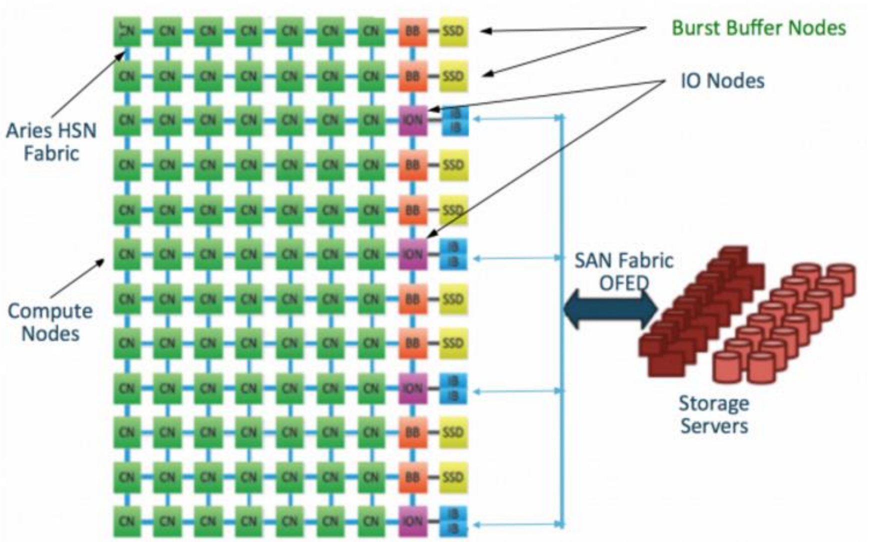
Storage in HPC

- Parallel File Systems (PFS):
  - Peak performance: ~2000GiB/s
  - Capacity: >70PiB
- Interfaces:
  - POSIX, MPI-IO, HDF5, etc.,
- Limitations:
  - Scalability, complexity, metadata services
  - Small file access, data synchronization, etc.,

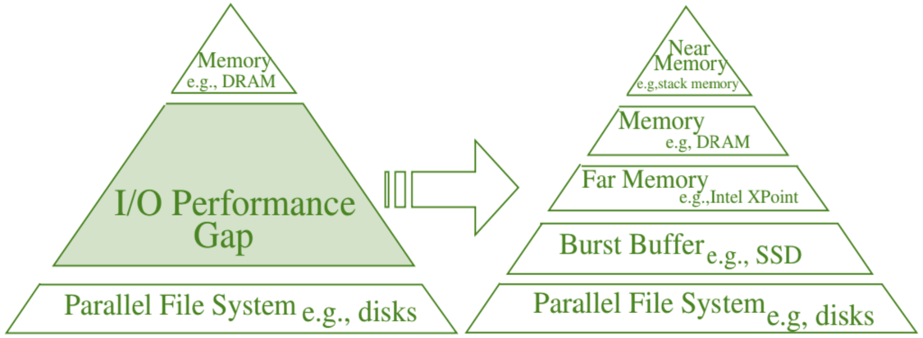


Burst Buffers

- Shared I/O buffering nodes, called Burst Buffers (BB)
- Flash storage deployments
  - Cost of SSD decreases – 2.2x price premium by 2021
- Low-latency with faster networks
- Several HPC sites have already deployed BBs:
  - NERSC's Cori
  - KAUST's ShaheenII
  - JCAHPC's Oakforest-PACS
  - LANL's Trinity
  - ORNL's Summit
  - …and more to come
- Use cases:
  - as a cache on top of the PFS
  - as a fast temporary storage for out-of-core applications
  - for intermediate results (data may not be persisted), and
  - as an in-situ/in-transit visualization and analysis



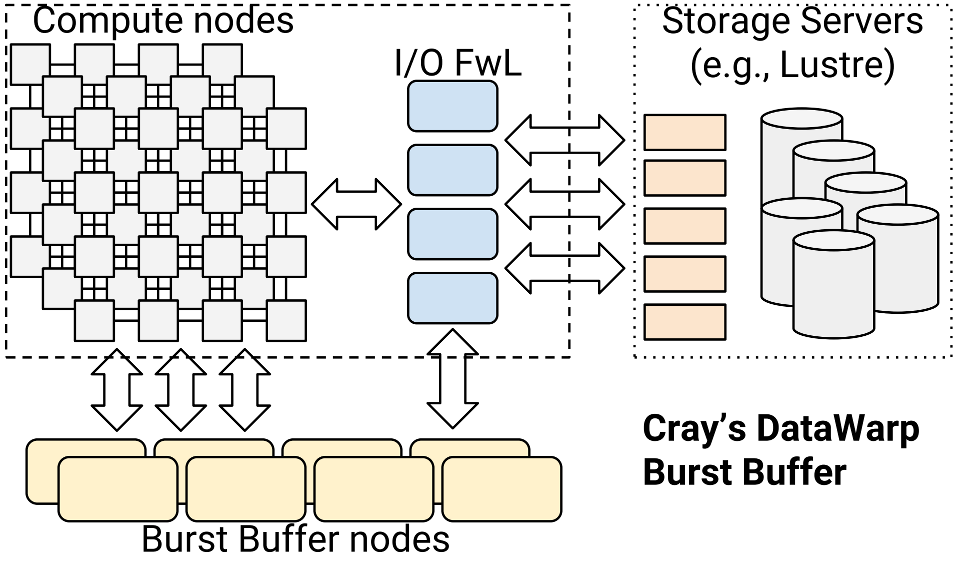
Cori, a Cray XC40 system at NERSC uses Cray's DataWarp BB technology



BB example: Cray's DataWarp in Cori

- Each buffer node is equipped with 2 PCIe x8 SSDs of 3.2TB capacity each
- Access to buffers via batch scheduler
- BB reservation lifetime same as application
- Data flushing at the end of the job
- Two levels of granularity – degree of striping data:
  - Small pool at 20GB per buffer
  - Large pool at 200GB per buffer (default)
- Distribution in round robin fashion
- Two types of allocation – visibility of data
  - Per-job instance
  - Persistent instance

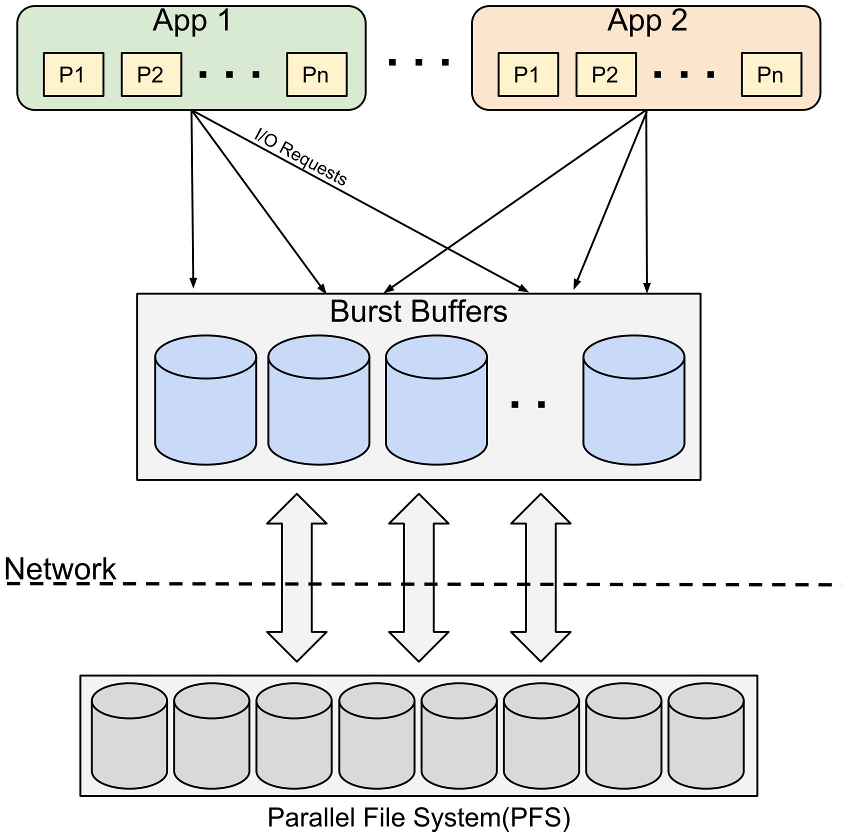
```
#!/bin/bash
#SBATCH -q debug
#SBATCH -N 1
#SBATCH -C haswell
#SBATCH -t 00:15:00
#DW jobdw capacity=10GB access_mode=striped type=scratch
srun a.out INSERT_YOU_CODE_OPTIONS_HERE
```



Cray's DataWarp Burst Buffer

Cross-Application I/O Interference

- Millions of cores
- Capacity vs Capability Computing
- Multiple applications share access to BBs
- Resource contention:
  - Compute and network are managed by the global scheduler
  - Storage resources deal with I/O contention leading to severe performance degradation



I/O Scheduling in BB vs. PFS

1. Lower latency and higher BW changes the way applications access the buffers
2. Different storage mediums (i.e., flash-based vs spinning) dictates different access concurrency
3. Different use cases for BBs lead to different access patterns and semantics
4. Access to BBs is reserved through the global scheduler (i.e., Slurm) and not via a mount point

Anthony Kougkas, akougkas@hawk.iit.edu

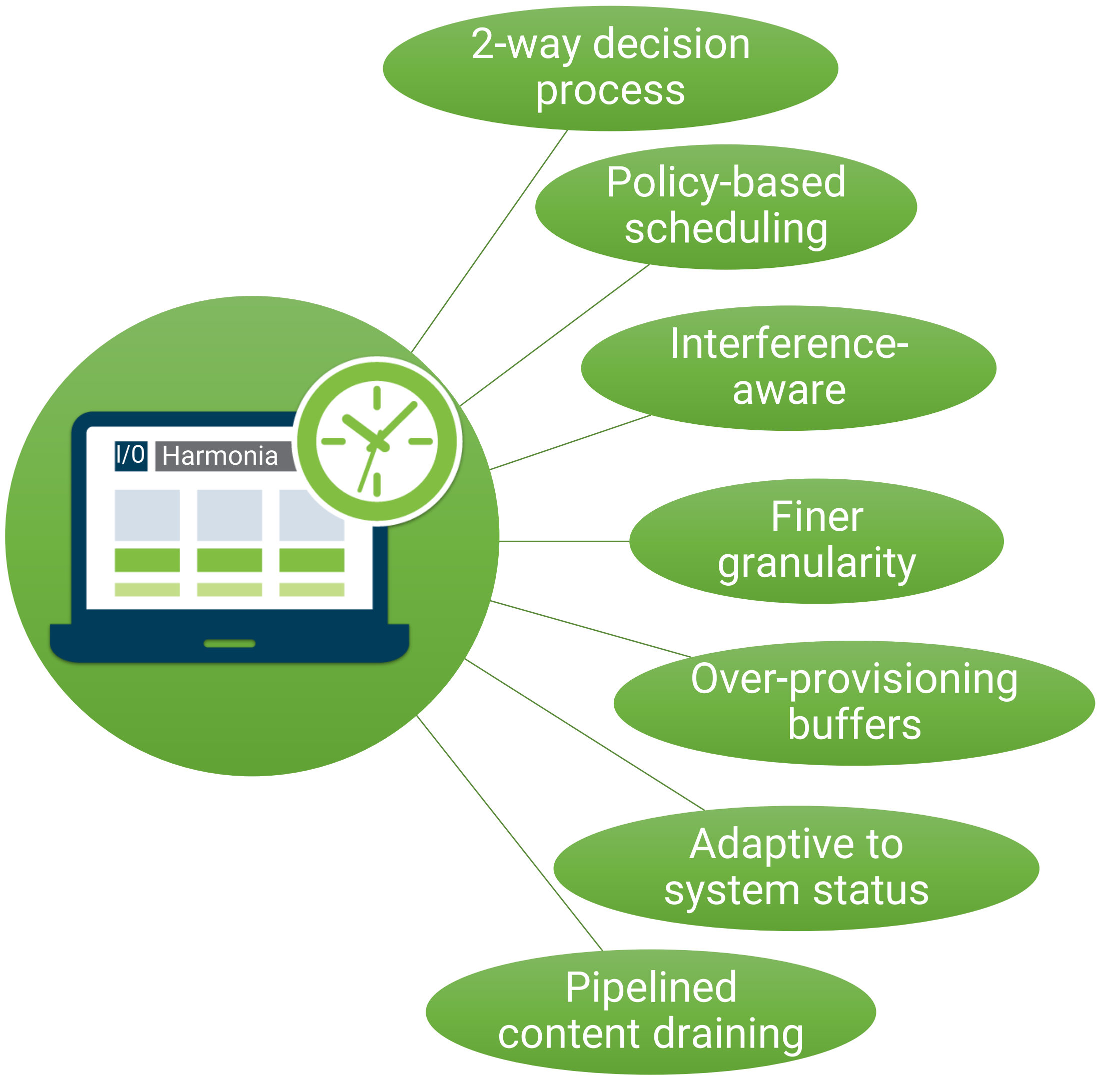
Challenges of I/O Scheduling in BB

- Concurrent access and interference
- Allocation exclusivity
- Reservation mechanisms
- Capacity constraints
- Buffer content draining

Harmonia: An Interference-Aware Dynamic I/O Scheduler for Shared Non-Volatile Burst Buffers

- Schedule I/O phases instead of entire application
- Overprovision buffer nodes
- 5 scheduling policies to fit a variety of workloads

Harmonia Highlights



Approach

1

Interference Sensitivity

- Medium Sensitivity to Concurrent Access (MSCA)

2

I/O Phase Detection

- User-defined
- Source code analysis
- Binary

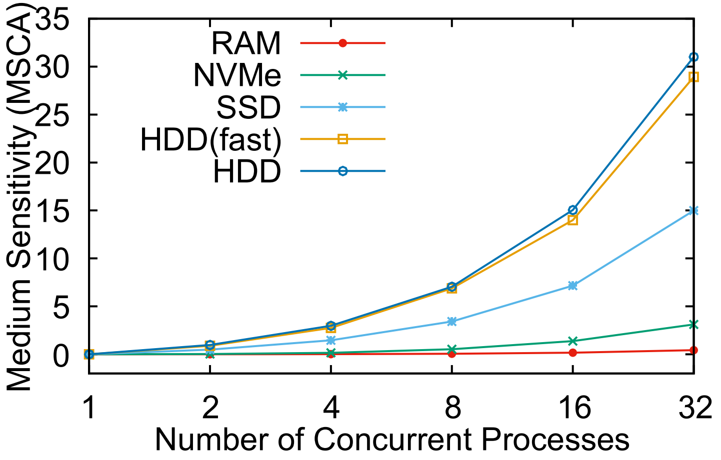
3

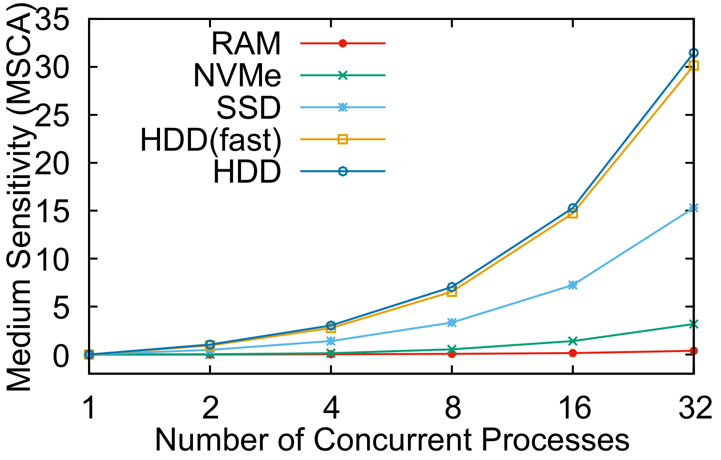
I/O Scheduling Policies

- Optimize scheduling metrics
- Adapt to system and workload

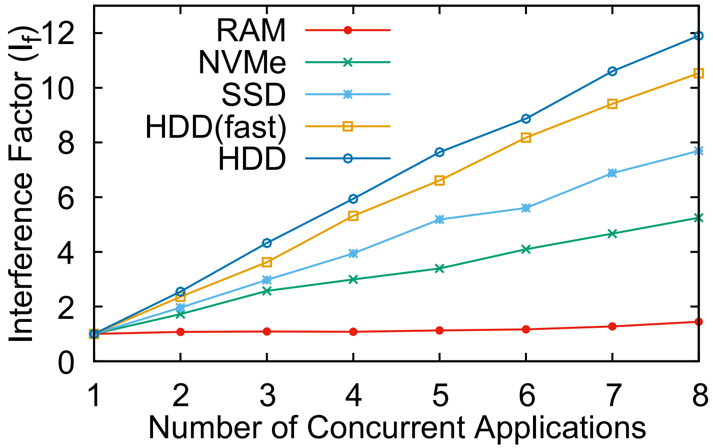
Medium Sensitivity to Concurrent Access

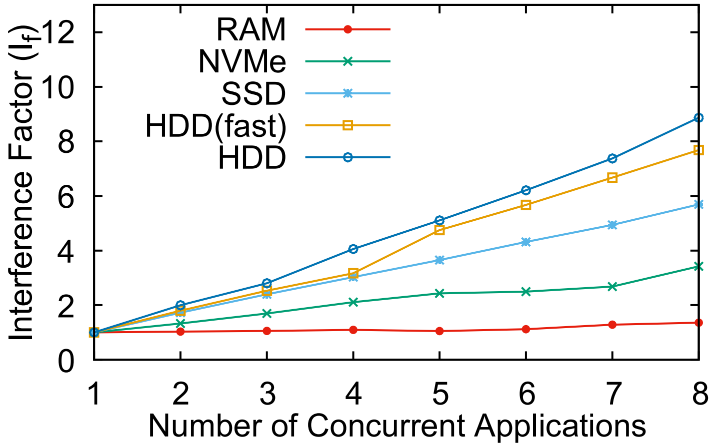
- Different buffering devices handle concurrent access in different ways
- Each controller offers different concurrency:
  - RAM has multiple memory lanes and banks
  - NVMe has multiple PCIe lanes
  - SSDs offer internal parallelism
  - SATA HDDs have only one lane
- A new metric used by Harmonia scheduler:
  - $MSCA = \frac{Application\_Num\_Concurrent\_Accesses}{I/O\_Device\_Concurrency} \times \frac{MaxBW - RealBW}{MaxBW}$
- Higher MSCA values means that the medium is more sensitive to concurrent accesses
- Scheduler takes MSCA into account to minimize interference stemming from the hardware





- Cross-application I/O interference
  - Multiple applications competing for access to I/O resources
  - Leads to severe performance degradation and variability
- Interference Factor:
  - Describes the slowdown applications experience due to resource contention
  - $I_f = \frac{Execution\ Time\ \mathbf{with}\ Interference}{Execution\ Time\ \mathbf{without}\ Interference}$
- ***I_f*** provides an absolute reference for a non-interfering system when ***I_f = 1***
- Context-dependent metric:
  - Allows comparison of applications with different I/O size or I/O requirement





Slowdown due to I/O Interference

User-defined

- User declares start - end of each I/O phase by injecting:
  - #pragma harmonia_io_start(…))
- **Pros:**
  - Flexibility and control to the user
  - Higher accuracy of I/O phase detection
- **Cons:**
  - Requires good understanding of the application's I/O behavior
  - Might lead to malpractice or exploitation of pragmas

Source code analysis

- Parse code identifying start - end of each I/O phase and inject auto-generated pragmas
- **Pros:**
  - Automatic I/O phase detection
  - Less user involvement
- **Cons:**
  - Incorrect classification (i.e., false positives-negatives)
  - User might not be allowed to submit source code due to security

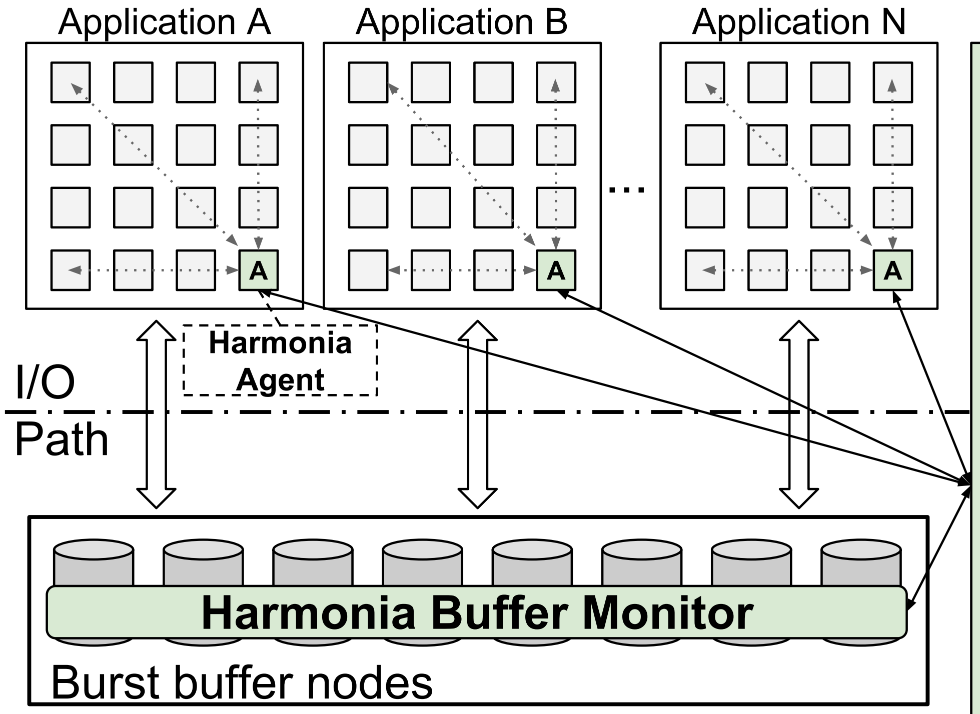
Binary (executable)

- Dynamically intercept I/O calls marking start - end of each I/O phase
- **Pros:**
  - No need for user input or source code submission
  - Dynamic detection during linking
- **Cons:**
  - Lower accuracy
  - Some overhead

I/O Phase Detection

Harmonia Architecture

- Middleware library in C++
- Applications link or re-compile
- All communications via MPI one sided
- Main components:
  - Agent
  - Buffer Monitor
  - Scheduler



Harmonia Scheduling Queue

JobQueue (Distributed Queue)	RunningJobs (List)	ScheduledJobs (Distributed Queue)	CompletedJobs (Distributed Map)
7	5	6	21
2	8	5	24

Harmonia Scheduler

Applications

App	Phase	Size
A	1	128
B	0	0
C	1	64

System Metrics

Dev	MSCA	I_f
Ram	1.8	1.58
SSD	11.2	7.8
HDD	29.5	12

Buffer nodes

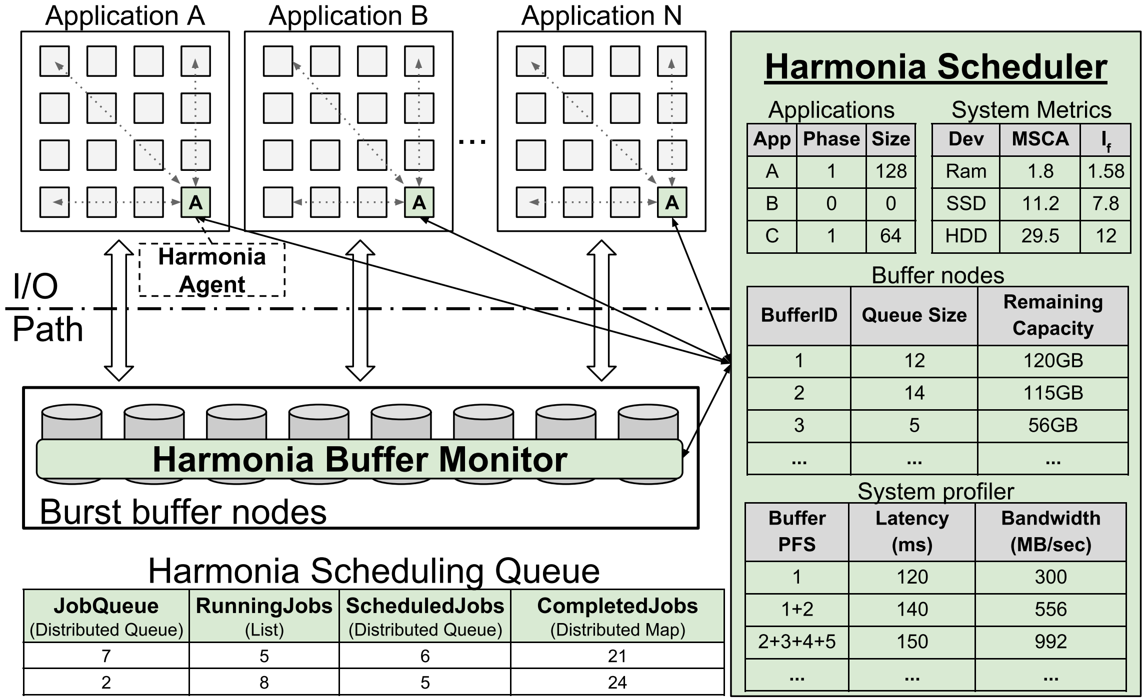
BufferID	Queue Size	Remaining Capacity
1	12	120GB
2	14	115GB
3	5	56GB
...	...	...

System profiler

Buffer PFS	Latency (ms)	Bandwidth (MB/sec)
1	120	300
1+2	140	556
2+3+4+5	150	992
...	...	...

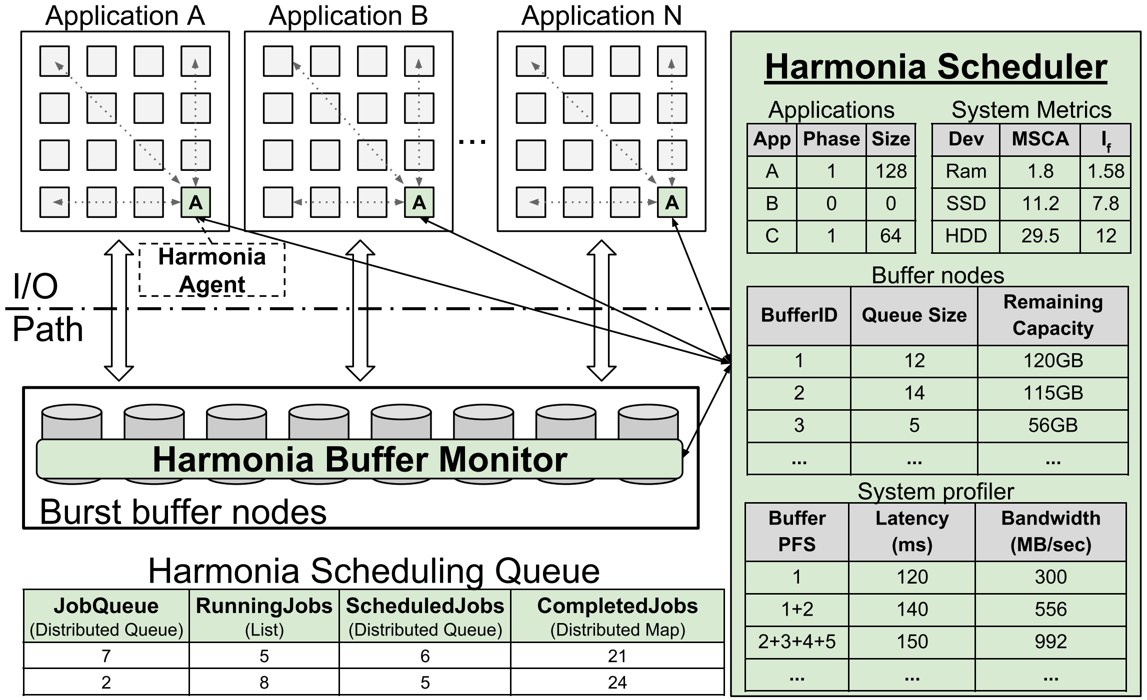
Anthony Kougkas, akougkas@hawk.iit.edu
ILLINOIS INSTITUTE OF TECHNOLOGY

- Represents the application
- The number of Agents per application depends on the size of the job
(e.g., every 64 application cores -> 1 Agent)
- Upon initialization (i.e., `MPI_Init()`), Agent registers several information to the Scheduler:
    - Application name
    - Job size
    - Group that owns the application, etc.,
- Responsible for:
    - Executing the I/O Phase Detection
    - Communicating the intention to do I/O
    - Accepting messages from Scheduler



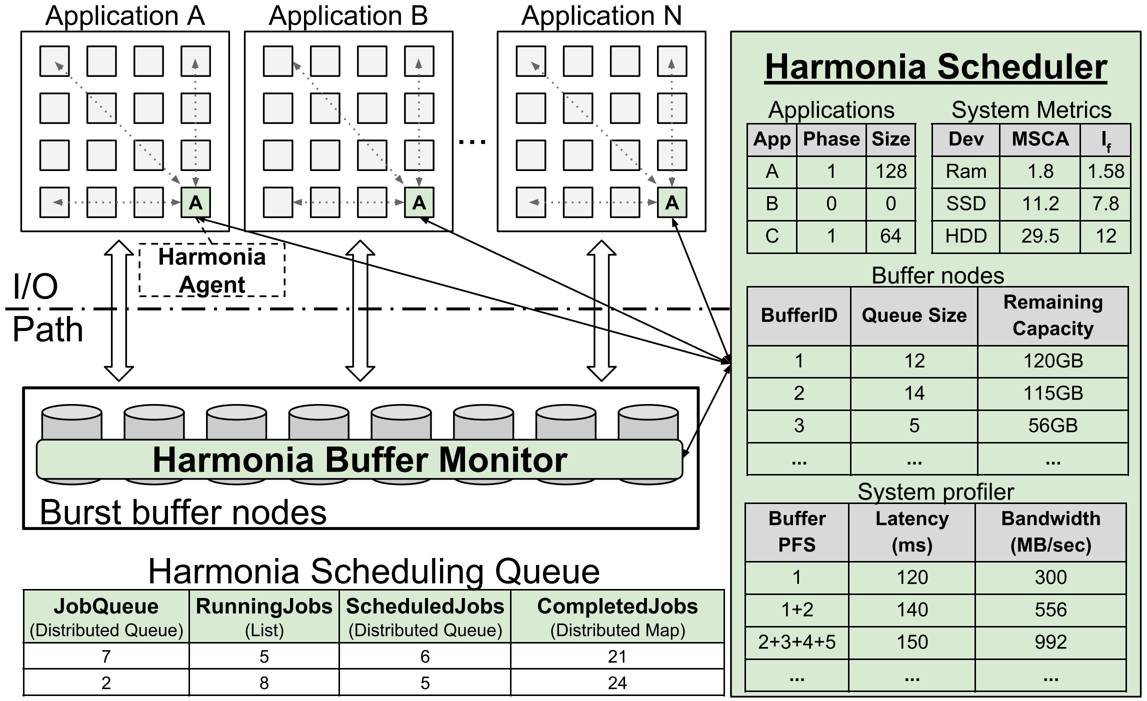
Harmonia Agent

- Lightweight background process running on buffer nodes
- Uses MPI one sided communications
- Collects information from buffer nodes
  - Remaining capacity
  - Node state: I/O queue size (i.e., `iostat`)
  - Node availability: busy or free (i.e., during flush)
- Maintains a small percentage of buffer nodes as backup or *flusher* nodes to handle
  - Overflowing data or
  - Data ready to be flushed



Harmonia Buffer Monitor

- Global multithreaded entity running on one or more separate nodes
- 2-way scheduling process collecting info from both the applications and the buffer nodes
- Structures:
  - Application registry
  - Buffer status table
  - System profiler and metrics
  - Scheduling queue
- Dynamic Programming approach



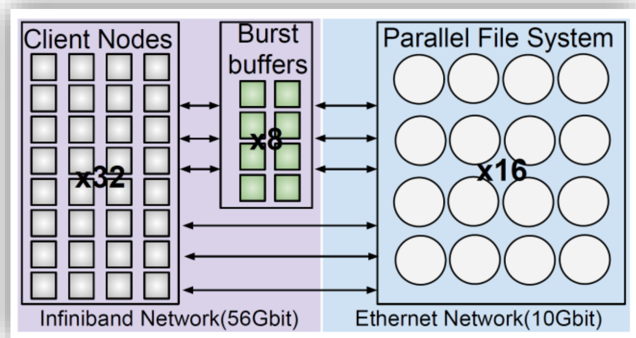
$$OPT(n,m) = \left\{ \begin{array}{ll} \infty & , if\ n=0, \\ 0 & , if\ \frac{n}{m} \geq X\ or\ m < Y, \\ \max \begin{pmatrix} C^{ij} + OPT(n^{-i}, m^{-j}), \\ OPT(n^{-i}, m), \\ OPT(n, m^{-j}), \\ OPT(n^{-i}, m^{-j}) \end{pmatrix} & , otherwise \end{array} \right\}$$

- *OPT(n,m)* is the optimal solution *n* phases to *m* buffers at time *t*
- C^{ij} the cost to schedule the i*th* phase to the j*th* buffer
- X is the maximum number of collocated apps (based on MSCA and I_f) and Y min number of buffers for this phase

Harmonia Scheduler

Harmonia Scheduling Policies

Maximum Buffer System Efficiency	•Maximizing buffer utilization – minimize idleness •Hides flushing behind computation to keep buffers busy
Application Priority	•Prioritize specific applications' buffer access, interference-free •Sort all I/O phases based on priority and schedule accordingly
Maximum Application Bandwidth	•Maximizing bandwidth each application experiences •Schedule I/O phases exclusively, interference-free, possibly wait
Application Fairness	•Equal division of buffer resources to applications (or random) •Same waiting and execution times for all apps (until threshold)
Minimum Application Stall Time	•Minimize the average wait time by sacrificing bandwidth •Minimize slowdown time due to interference



Device	RAM	NVMe	SSD	HDD fast	HDD
Model	M386A4G40DM0	Intel DC P3700	Intel DC S3610	Seagate ST600MP0005	Seagate ST9250610NS
Connection	DDR4 2133Mhz	PCIe Gen3 x8	SATA 6Gb/s	12Gb/s SAS	SATA 6Gb/s
Capacity	512 GB(32GBx16)	1 TB	1.6 TB	600 GB	2.4 TB
Latency	13.5 ns	20 μs	55-66 μs	2 ms	4.16 ms
RPM	-	-	-	15000	7200
Device Concurrency	8	4	2	1	1
Max Read BW	65000 MB/s	2800 MB/s	550 MB/s	215 MB/s	115 MB/s
Max Write BW	59000 MB/s	1900 MB/s	500 MB/s	185 MB/s	95 MB/s

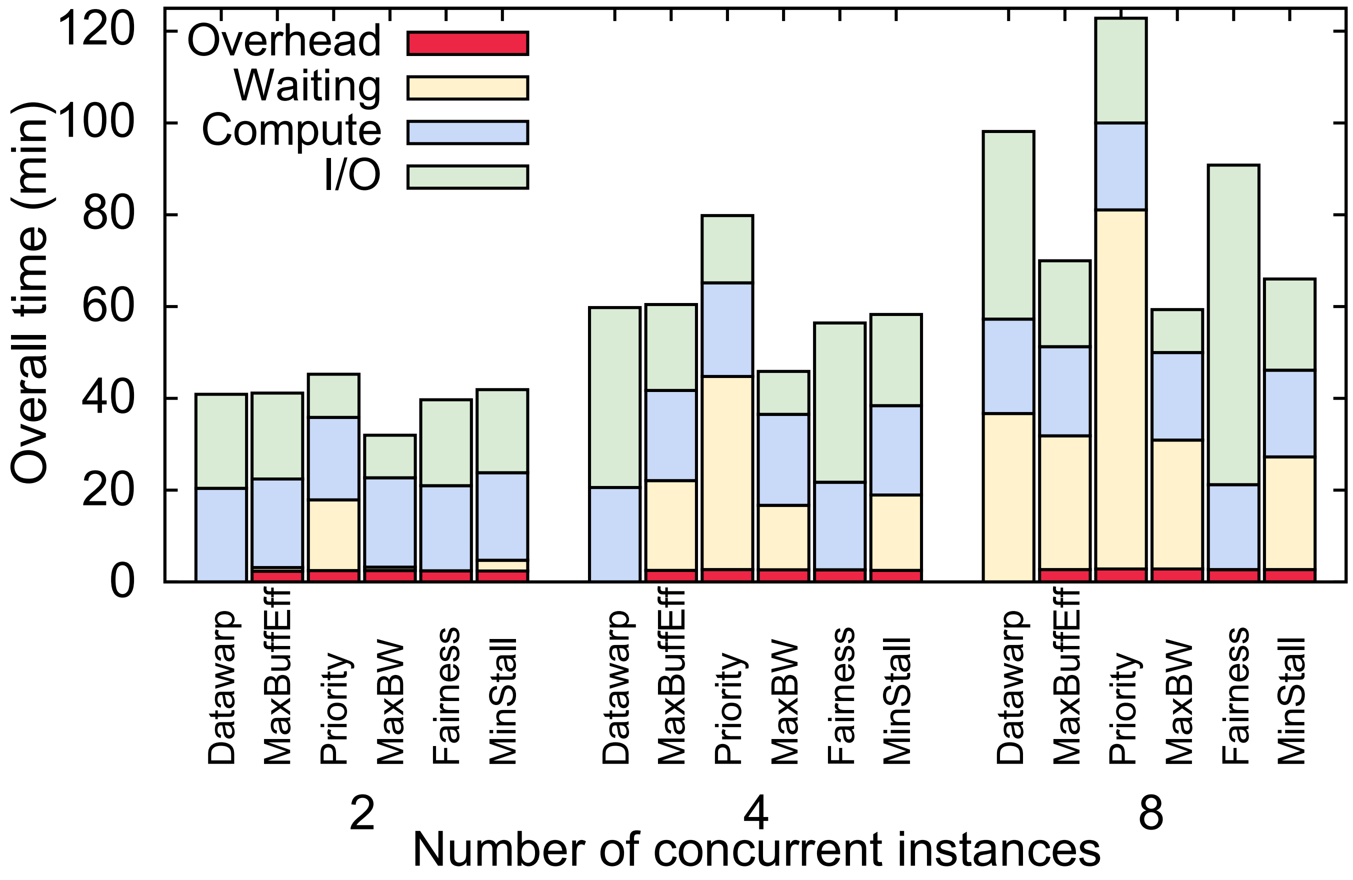
- Testbed: Chameleon System, Appliance: Bare Metal
- OS: Centos 7.1, Storage: OrangeFS 2.9.6, MPI: OpenMPI
- Programs:
    - Synthetic benchmark alternating computing and I/O phases
        - Workloads:
            - Compute-intensive
            - Balanced
            - I/O intensive
            - Only I/O
    - VPIC: particle simulation
    - HACC-IO: cosmological simulation I/O kernel
- Buffer aggregate capacity: 800GB
- Total dataset size for 8 instances: 1.6TB

Anthony Kougkas, akougkas@hawk.iit.edu

Testbed and Configurations

Performance and overheads

- Synthetic benchmark
  - Balanced workload (compute-I/O)
- Average completion time
  - Wait to be scheduled
  - Computation time
  - I/O time
  - Overheads
- Concurrent execution scaling
  - 2-8 instances
  - Buffer can hold data up to 4 instances before they flush
- Compared to DataWarp scheduling

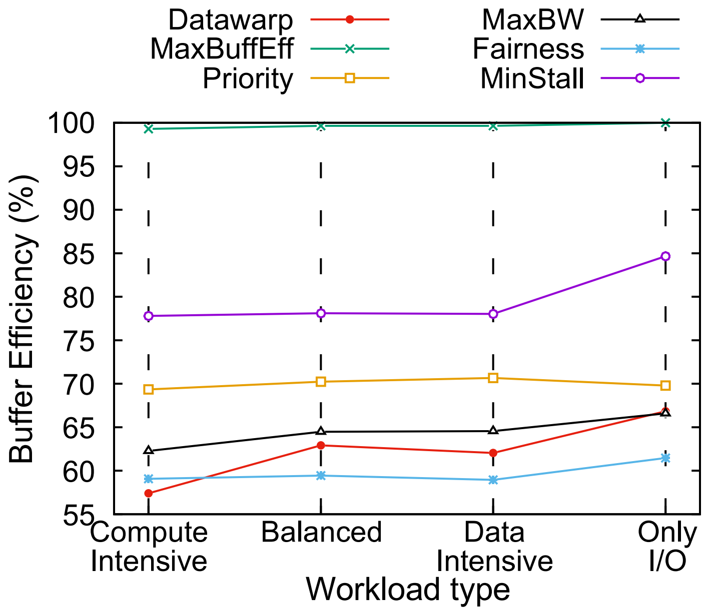


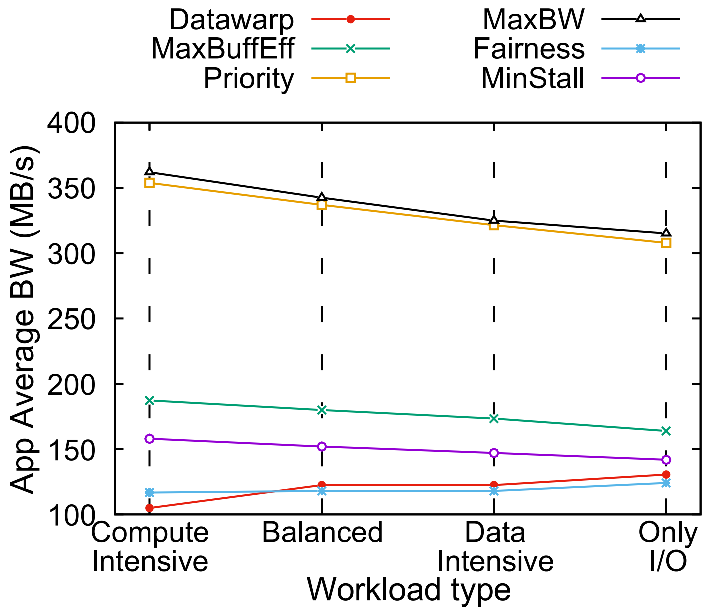
- 40% faster execution than DataWarp for 8 concurrent instances
- 4% overhead on average to perform I/O phase detection offline
- MaxBW offers the best I/O time whereas Fairness the slowest I/O
- Harmonia's scheduling policies offer greater flexibility to the system

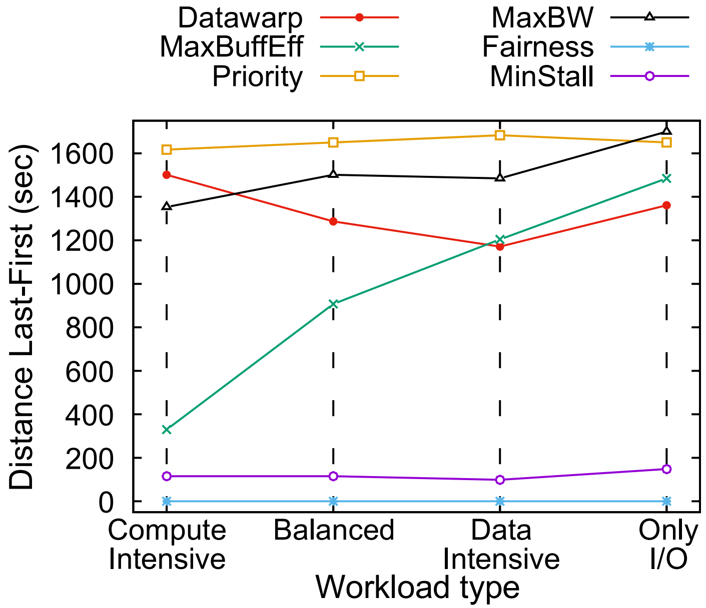
Anthony Kougkas, akougkas@hawk.iit.edu

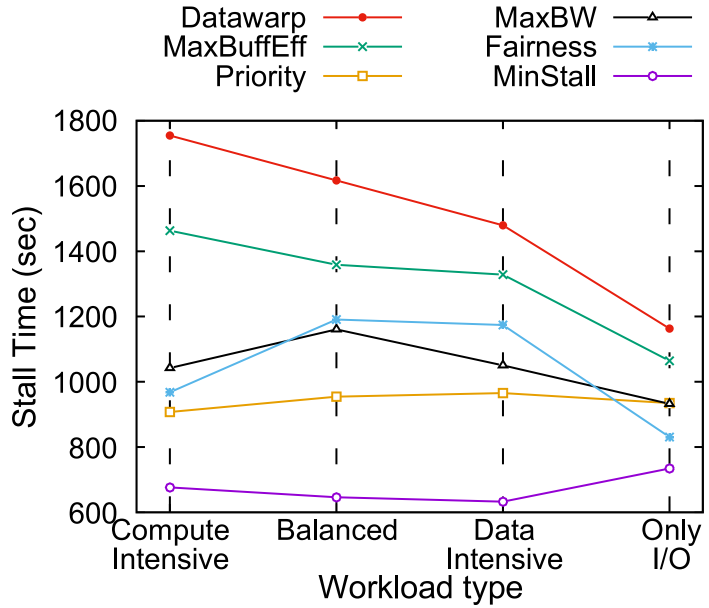
Scheduling Metrics

- Max Buffer Efficiency:
  - Harmonia can be **2x** more efficient
- Max Bandwidth:
  - Harmonia can offer **3x** higher average bandwidth
- Fairness:
  - Harmonia can achieve **10x** higher fairness
- Min Stall Time:
  - Harmonia can minimize stall time for application by **3x**





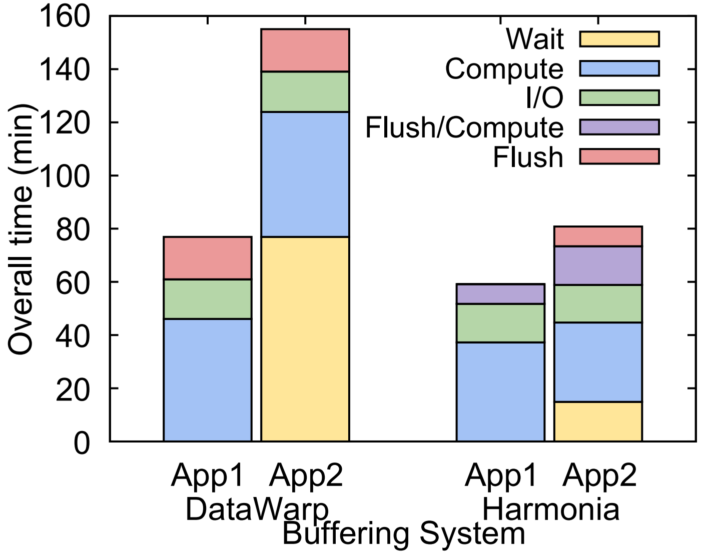




- Harmonia's policies can better adapt to workloads than other buffering systems

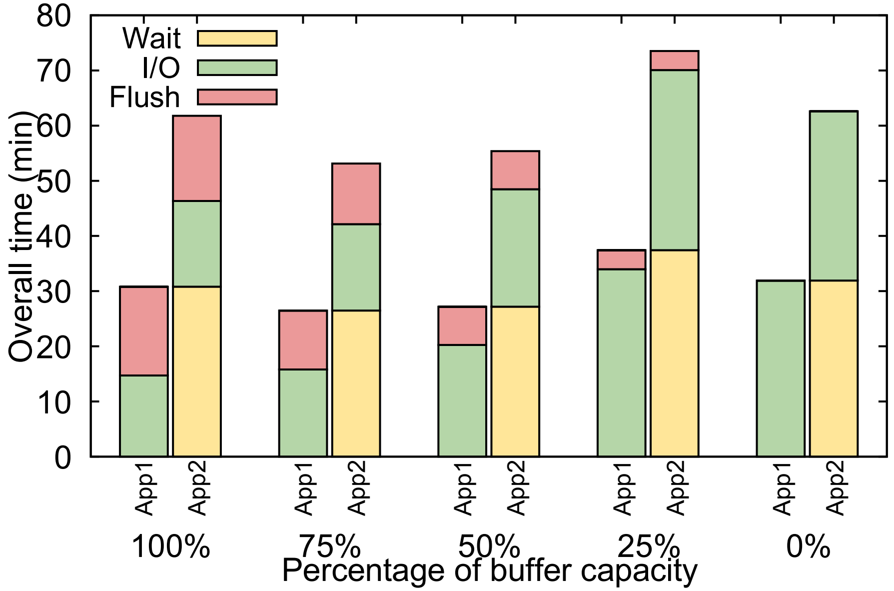
Buffer draining (data flushing)

- Buffer draining: flushing of data from buffers to the persistent layer (i.e.,PFS)
- 2 instances of VPIC with 16 steps:
  - Buffer can hold data only for 1 instance
  - In each step:
    - Computation phase
    - Writing data to buffers
  - Harmonia leverages computation phases to drain the buffers
- **2x** better performance than DataWarp
- Flushing threshold initiates flushing:
  - 100% case same behavior as DataWarp
  - 0% case incoming I/O conflicts with flush
  - 50-75% threshold offers the best overlapping of incoming I/O and flushing



Description	DataWarp	Harmonia
Min Completion Time	76.9	59.2
Max Completion Time	155.1	80.8
Average Completion Time	116	70
Average Wait Time	38.45	7.45

Test Metrics in Minutes



- Harmonia leverages computation to "hide" flushing

Discussion

Q: How does Harmonia handle read-after-write (RAW) workloads?

A: Harmonia employs a hinting system where an I/O phase is marked as ”cached” or ”flushable” during the I/O Phase Detection. It uses those hints to drive its scheduling decisions.

Q: How about asynchronous I/O calls?

A: Harmonia can utilize the traffic service classes implemented in InfiniBand networks (i.e., traffic class field TClass in Mellanox) to handle both I/O and compute traffic.

Anthony Kougkas, akougkas@hawk.iit.edu

In summary

- **Cross-application I/O interference** is a source of performance degradation that I/O schedulers need to be aware of.
- Buffering mediums (i.e., storage devices) handle **concurrency** differently which can be effectively used by the scheduler to minimize interference.
- **Policy-based scheduling** works better for diverse I/O workloads.
- **Harmonia,** a new, dynamic, interference-aware I/O scheduler
    - schedules individual I/O phases for finer granularity.
- By **overlapping computation and I/O phases** and calculating I/O interference into its decision making process, Harmonia can be **3x faster** than state of the art buffering systems leading to **better resource utilization**.

Harmonia

An Interference-Aware Dynamic I/O Scheduler for Shared Non-Volatile Burst Buffers

Thank you.

Anthony Kougkas

akougkas@hawk.iit.edu

https://www.akougkas.com

This work was supported by the National Science Foundation under grants no. CCF-1744317, CNS-1526887, and CNS-0751200.

Special thanks to Dr. Jay Jofstead for his invaluable help and insight.

In collaboration with

Sandia National Laboratories

SCALABLE COMPUTING SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY