INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

ICPP/2020/EDMONTON/CANADA

AUGUST 17-20, 2020

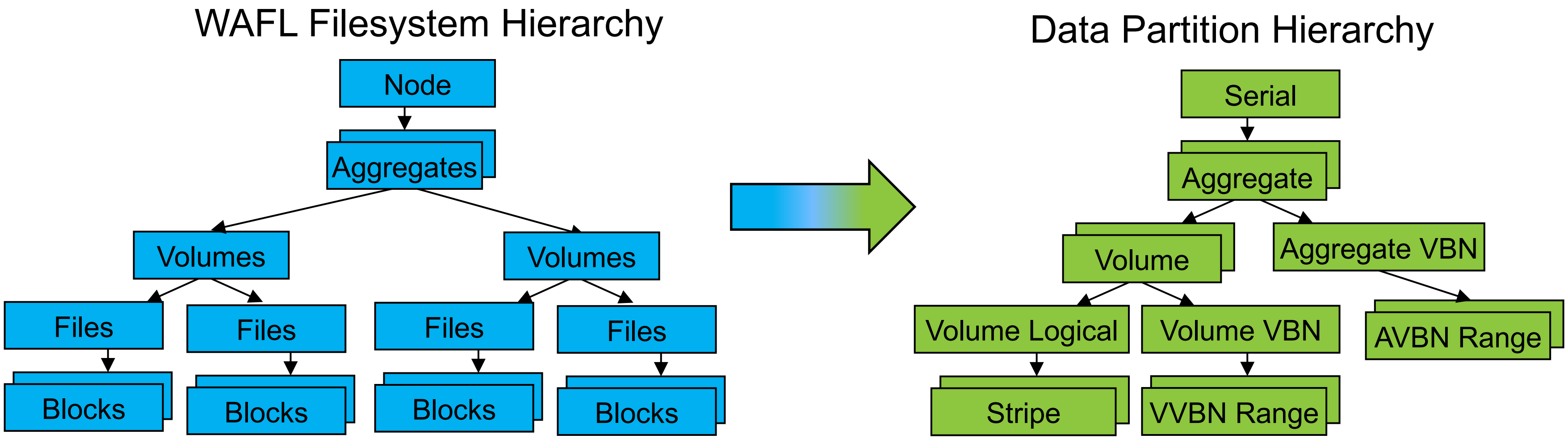
Scalable Coordination of Hierarchical Parallelism

Vinay Devadas, PhD vdevadas@netapp.com

Matthew Curtis-Maury, PhD mcm@netapp.com

NetApp

Hierarchical Parallelism in the WAFL File System



WAFL Filesystem Hierarchy

Node

Aggregates

Volumes

Volumes

Files

Files

Files

Files

Blocks

Blocks

Blocks

Blocks

Data Partition Hierarchy

Serial

Aggregate

Volume

Aggregate VBN

Volume Logical

Volume VBN

AVBN Range

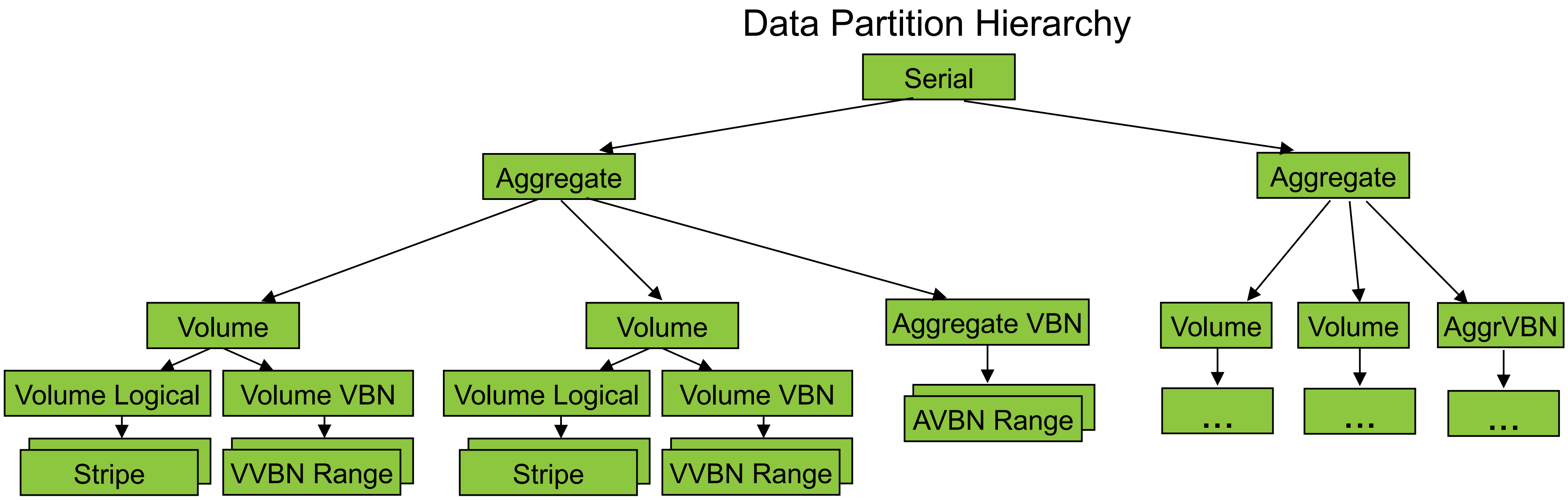
Stripe

VVBN Range

- WAFL is a high-performance commercial filesystem
- Hierarchical data partitioning to match hierarchical data
- File system work is mapped each partition
- Scheduler picks partitions that can run safely together

“To Waffinity and Beyond” – OSDI 2016

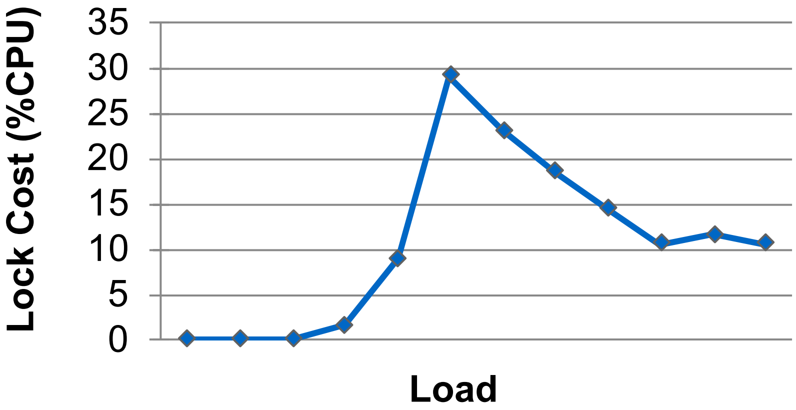
Scheduling Work with Hierarchical Parallelism



- An executing partition prevents the execution of its parents and children
- Analogous to a tree of Reader-Writer locks
  - Take Writer lock on target partition and Reader lock on all parents
  - Such systems exist and can benefit from our techniques
- Volume Logical and Volume VBN can run concurrently
- Volume Logical and Volume cannot run concurrently

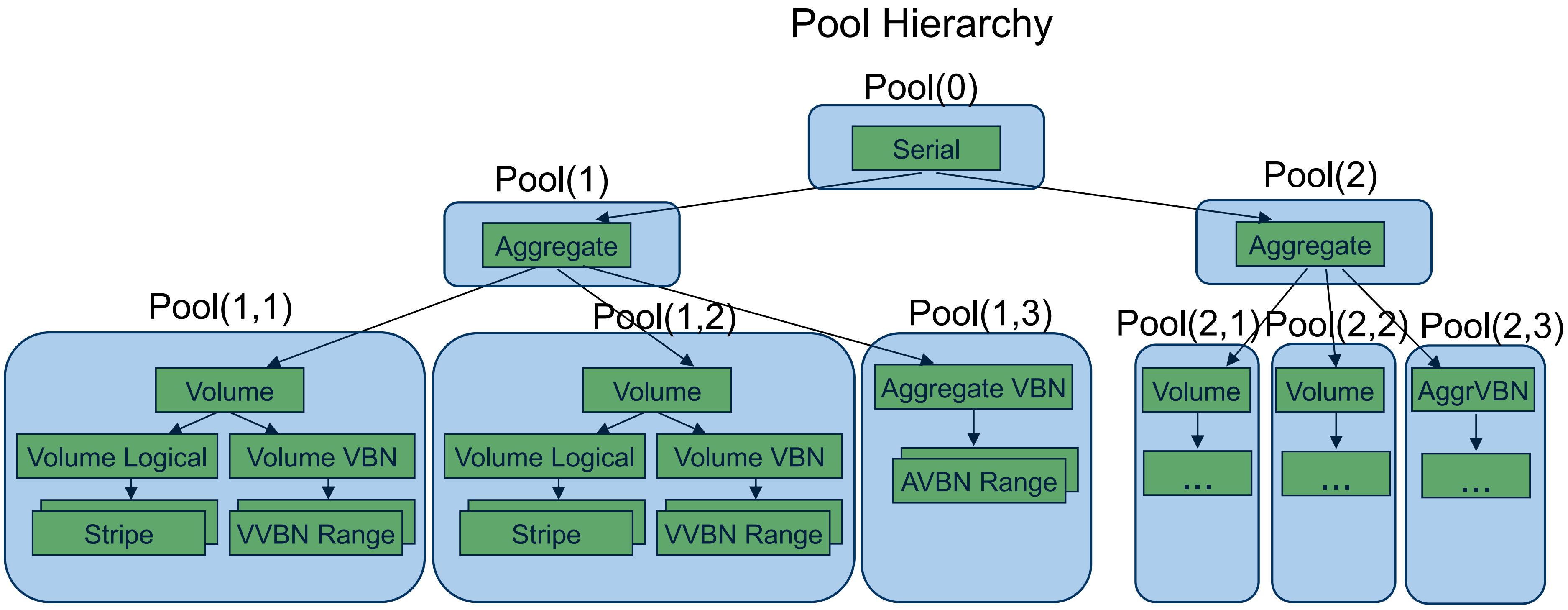
Problem 1: Scheduler Lock Contention

- Global knowledge required to enforce the hierarchy
  - We have a single global spinlock taken whenever scheduling occurs
- Under high load, enough work in each partition to reduce the amount of switching



SFS2008 benchmark running on 36-core system.

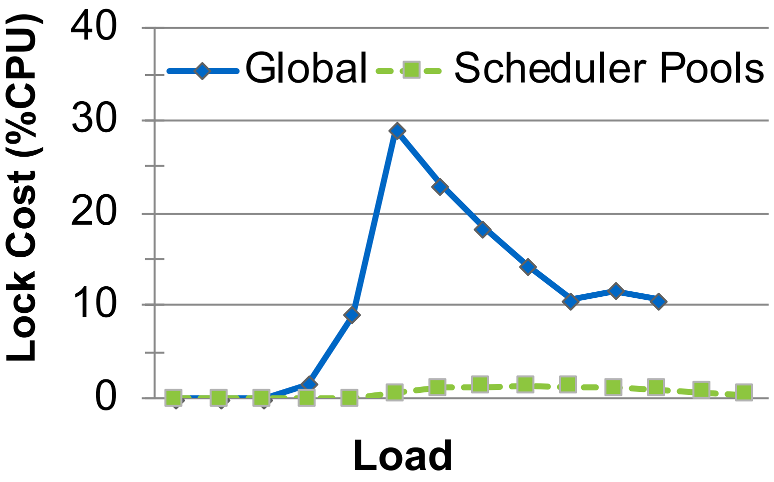
Scheduler Pools

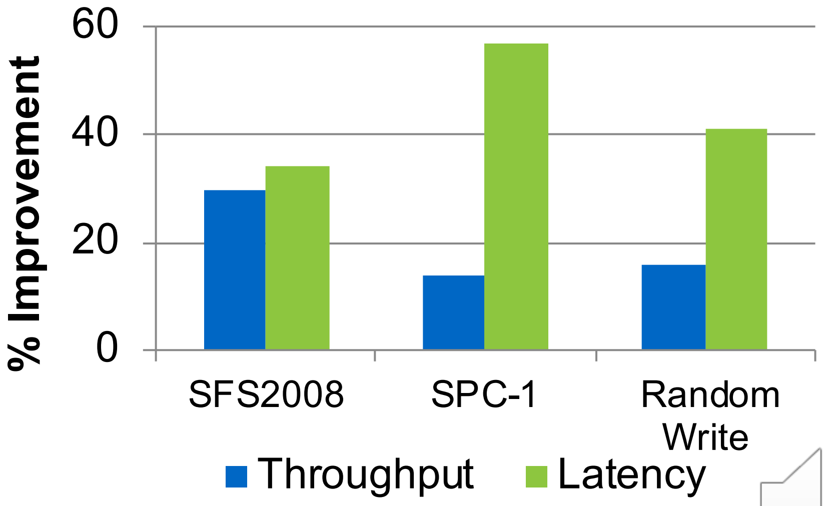


- Break the hierarchy into pieces, each with independent scheduler
- Now must correctly schedule the Pools
  - Can be done without global synchronization in nearly all cases
  - Then each scheduler can run independently to enforce internal Nodes

Scheduler Pools Performance Evaluation

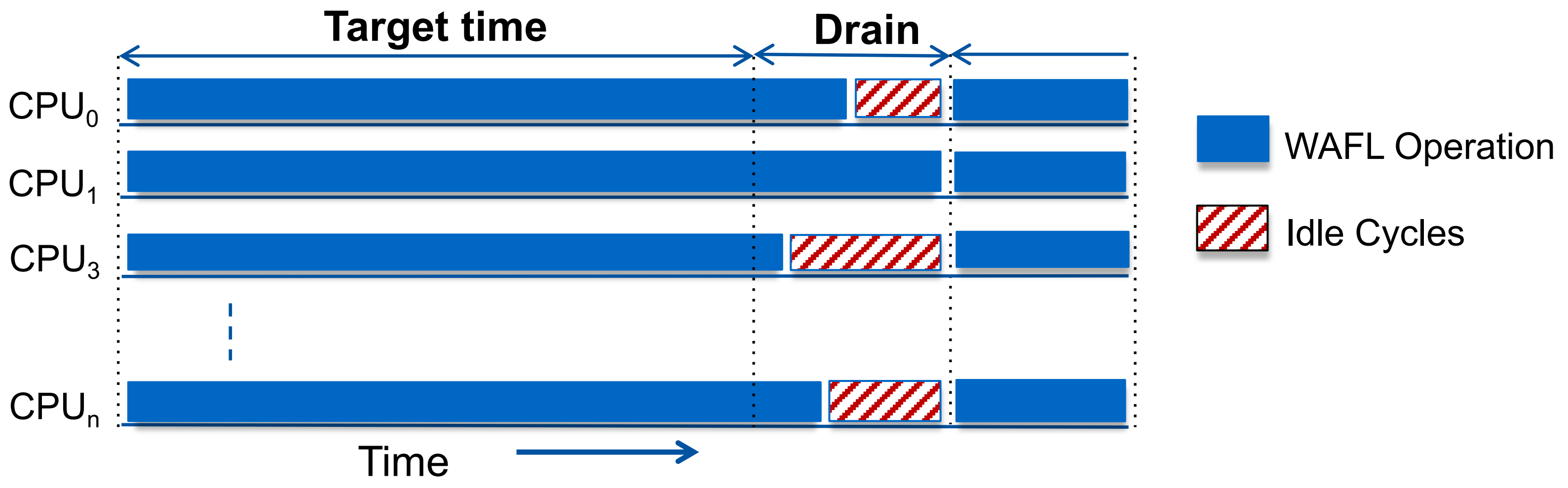
- Lock contention goes way down, nearly negligible
  - Same SFS2008 on 36 cores as earlier
  - Flexible to more pools as needed if it manifests again
- Across 3 key benchmarks, contention was very high
  - Significant improvements in throughput and latency with Scheduler Pools
- Deployed in 2017 with Data ONTAP 9.2 release





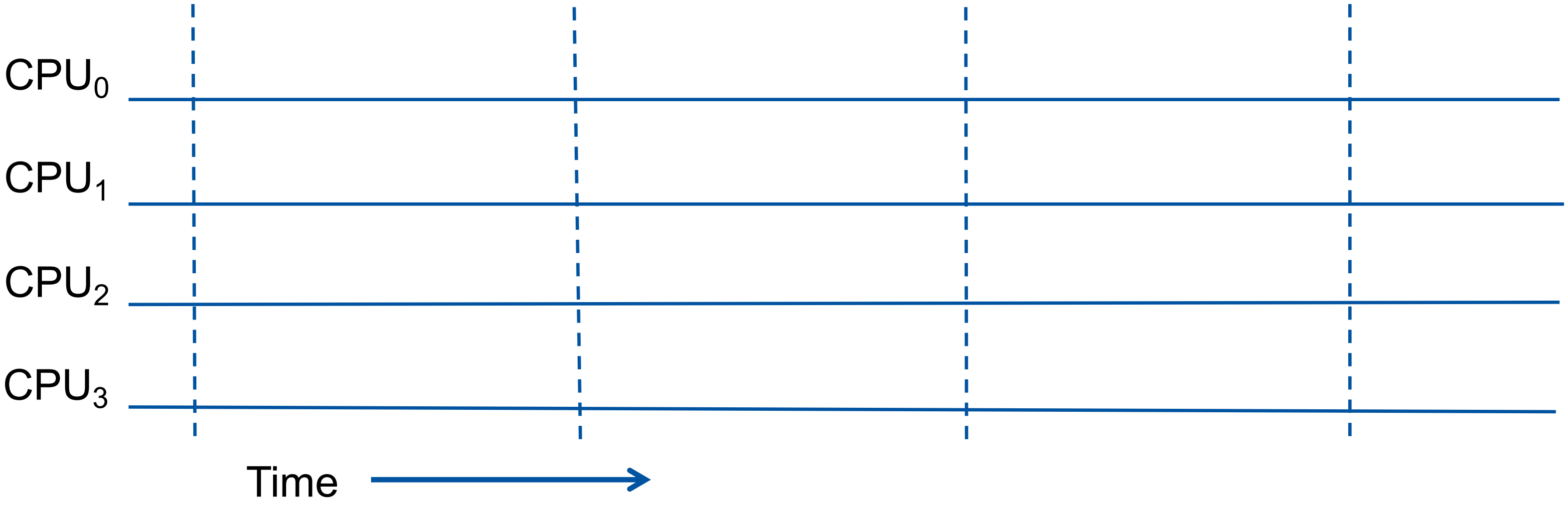
Problem 2: Inefficient Rescheduling

- To schedule a partition, must stop running all conflicting partitions
  - Analogous to scheduling a Writer on a R/W lock
  - They will not all stop at the same time
- Existing policy: Drain everything periodically
  - Provides flexibility to subsequently schedule ANY pool



Hierarchy-Aware Draining

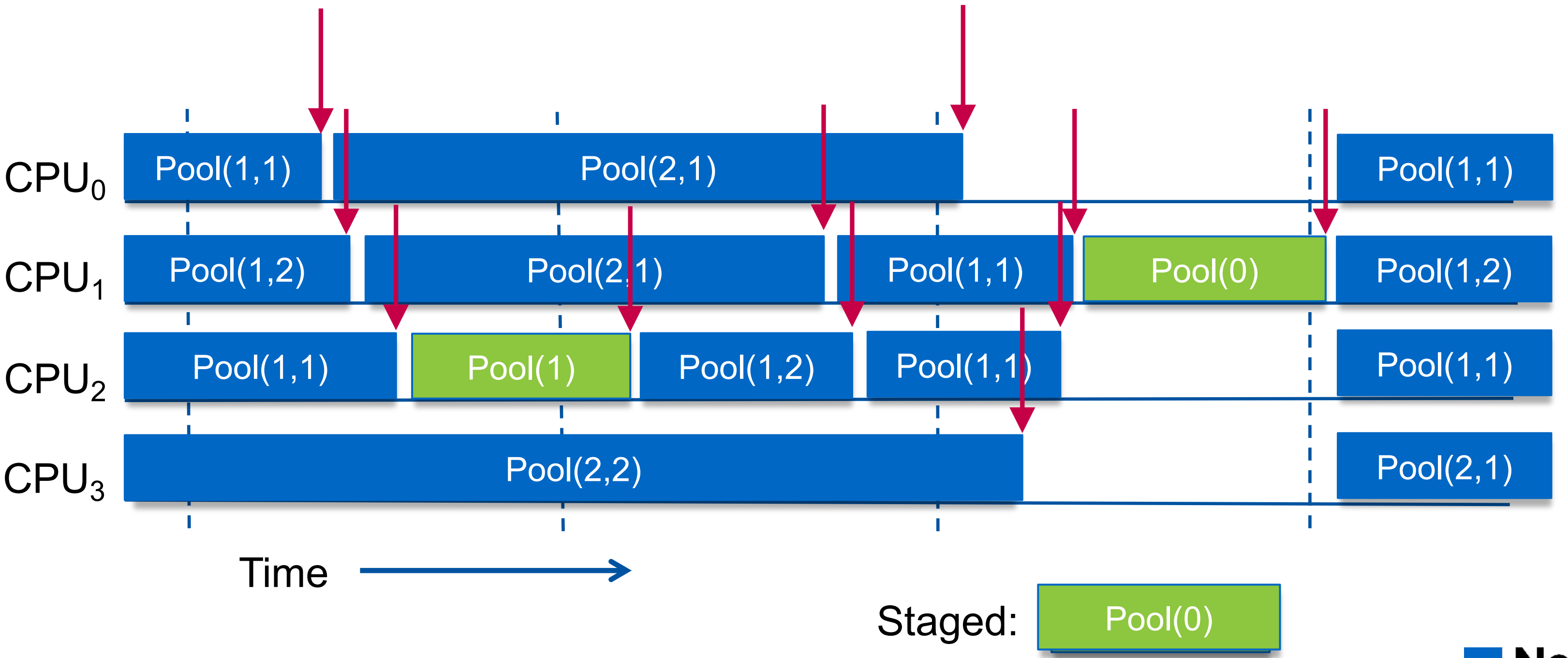
- Most of the time, simply try to maximize parallelism
- Periodically "stage" the next desired Pool/partition
  - Mechanism for forcing the scheduling of certain partition
- Leverages knowledge of hierarchy to make productive use of CPUs
  - Prevent scheduling of any conflicting partition
  - Allow scheduling of any non-conflicting partition



Staged:

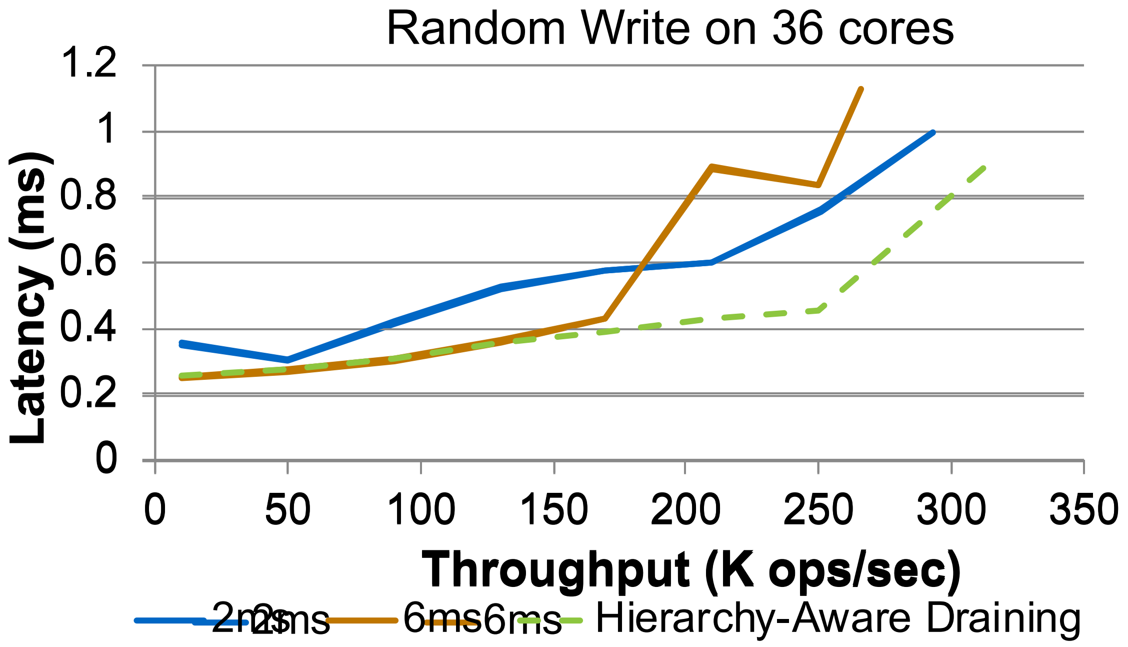
Hierarchy-Aware Draining

- Most of the time, simply try to maximize parallelism
- Periodically “stage” the next desired Pool/partition
  - Mechanism for forcing the scheduling of certain partition
- Leverages knowledge of hierarchy to make productive use of CPUs
  - Prevent scheduling of any conflicting partition
  - Allow scheduling of any non-conflicting partition



Hierarchy-Aware Draining Performance Evaluation

- Increasing the target window improves efficiency at low load
  - Leads to starvation and poor performance at higher load
- HAD provides higher efficiency across all levels of load
- Deployed in 2018 with Data ONTAP 9.3 release



Conclusion

- Scheduler Pools
  - Partition the hierarchy into *mostly* independent schedulers
- Hierarchy-Aware Draining
  - Allow continued processing while draining for target (staged) Pools
- Both apply to other systems with hierarchical parallelism

Thank you.

mcm@netapp.com