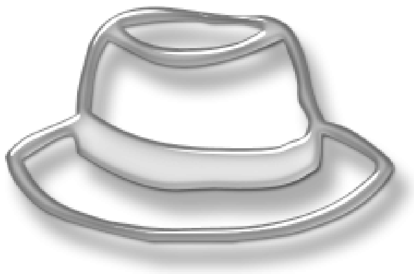
Resource Management in Large Shared Clusters

Konstantinos Karanasos
Microsoft CISL

University of Washington, Database Day

December 2, 2016

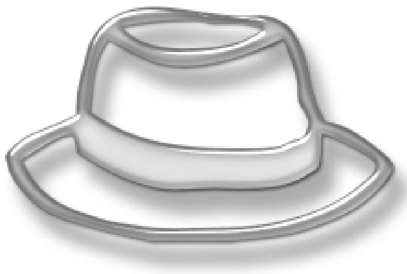
The 3 hats we wear in CISL



The 3 hats we wear in CISL

Applied research group

Systems+database people building prototypes, publishing papers



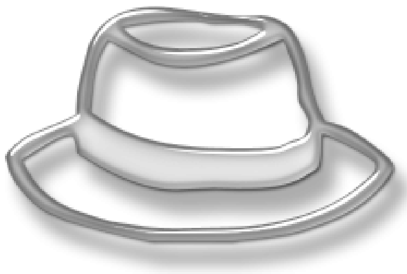
The 3 hats we wear in CISL

Applied research group

Systems+database people building prototypes, publishing papers

Collaborating with Big Data product group at MS

Shipping our code to production



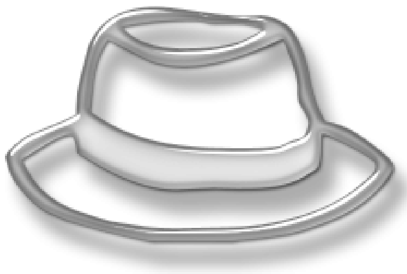
The 3 hats we wear in CISL

Applied research group

Systems+database people building prototypes, publishing papers

Collaborating with Big Data product group at MS

Shipping our code to production



Open-sourcing our code

Apache Hadoop, REEF, Heron

CISL focus

Resource management

Distributed tiered storage

Query optimization

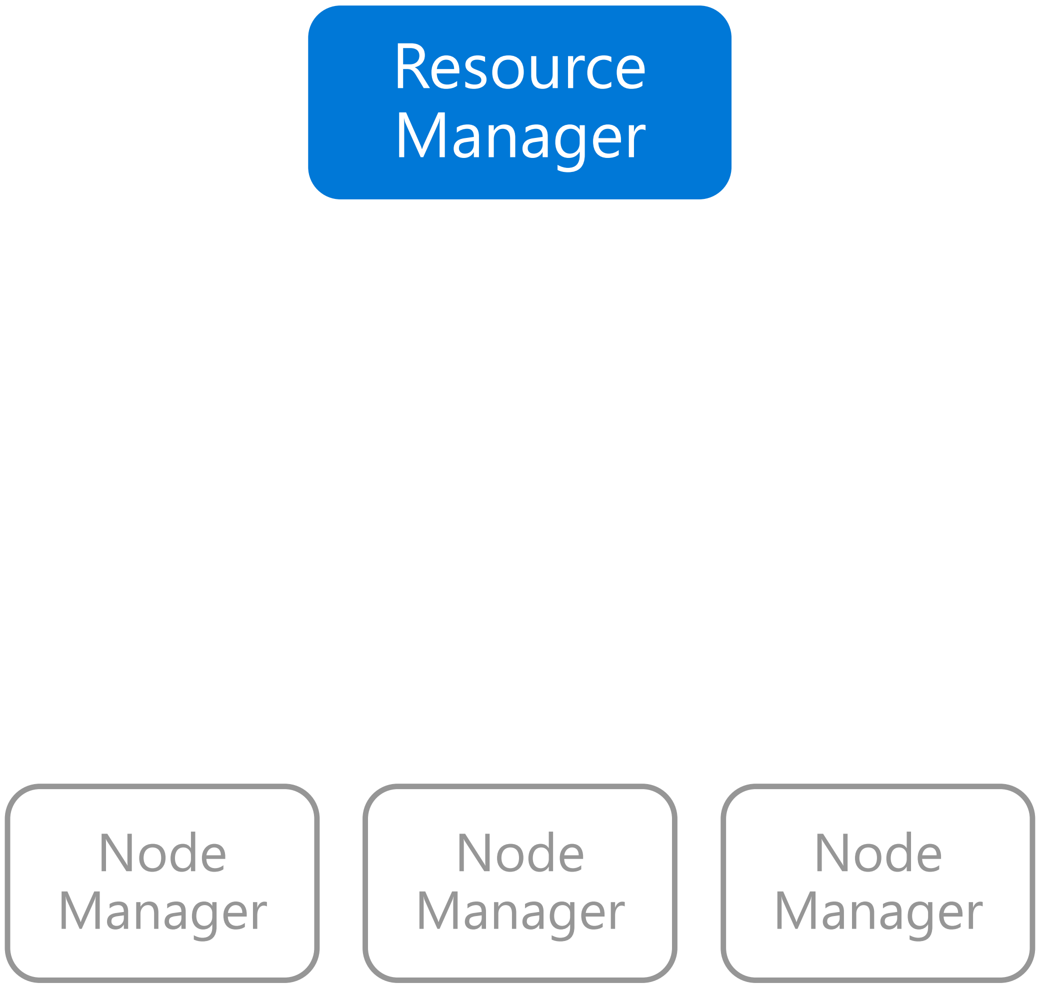
Log analytics

Stream processing

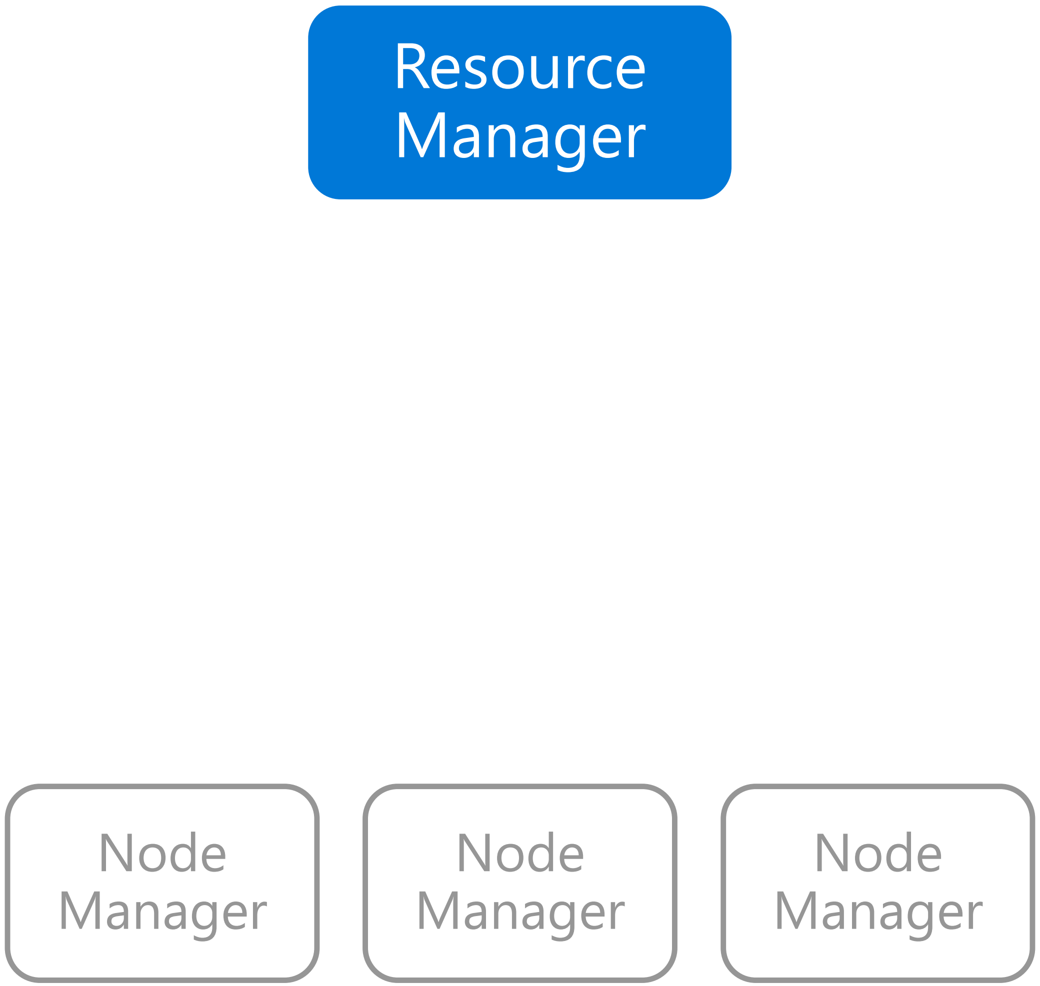
CISL focus

- Resource management
- Distributed tiered storage
- Query optimization
- Log analytics
- Stream processing

What is a Resource Manager?

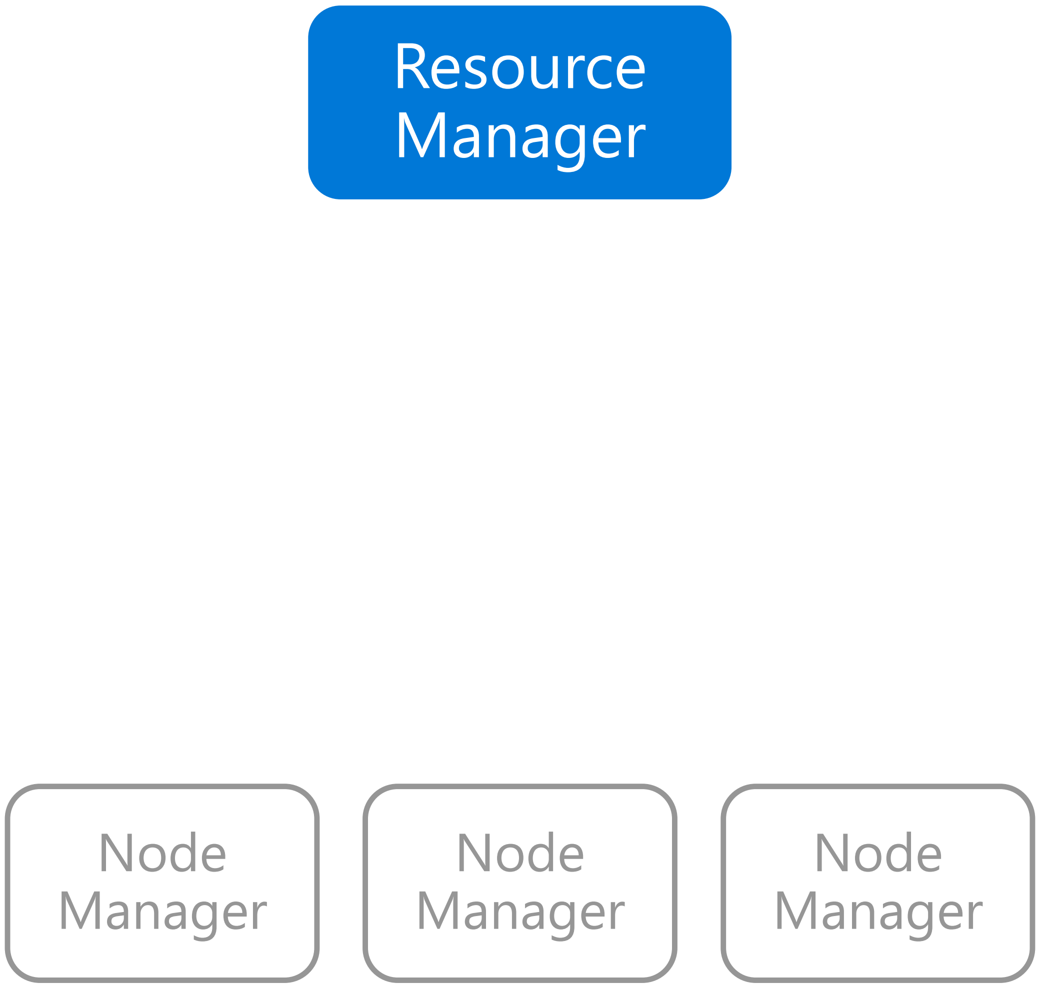


What is a Resource Manager?



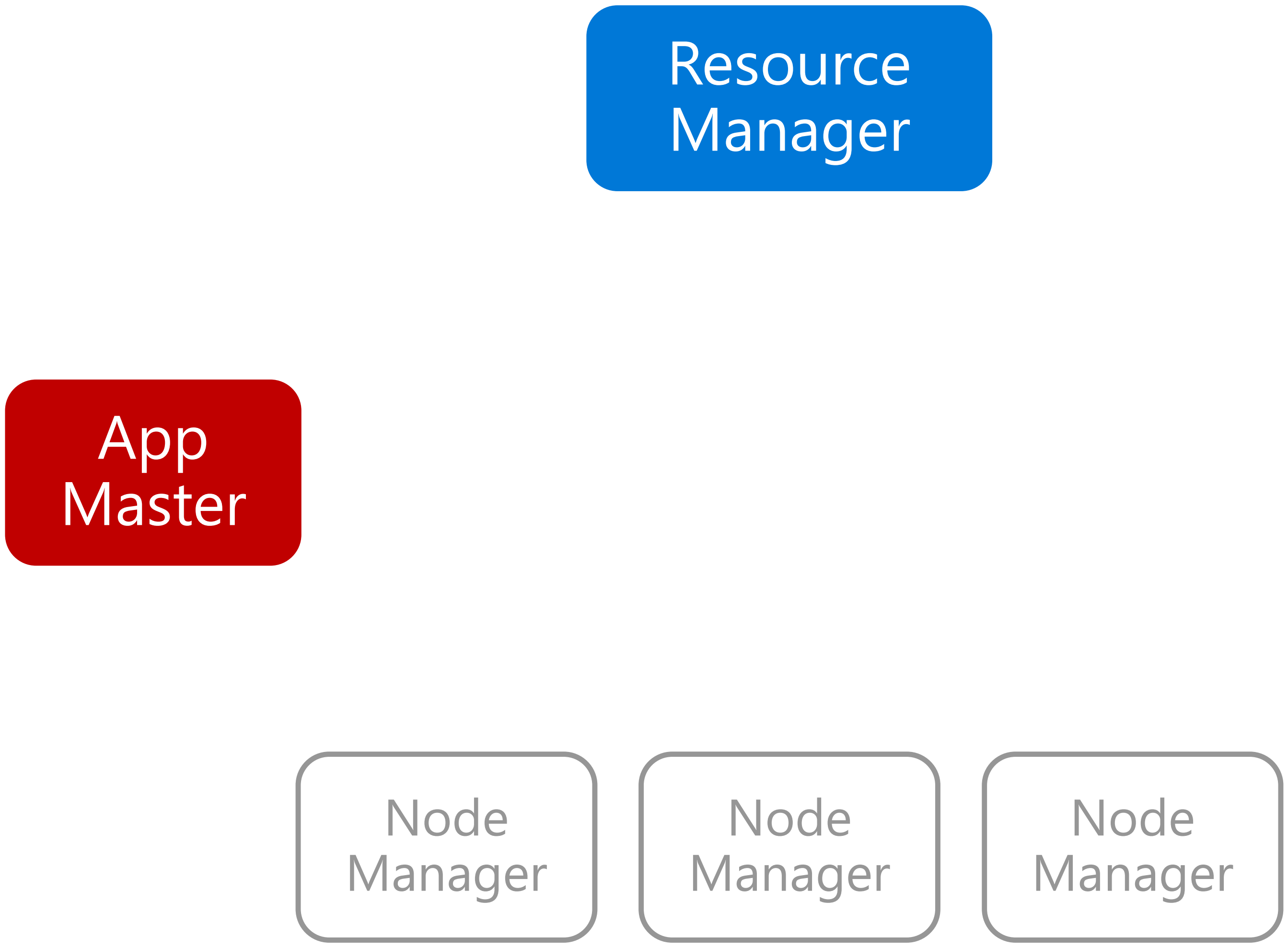
- Jobs consist of tasks

What is a Resource Manager?



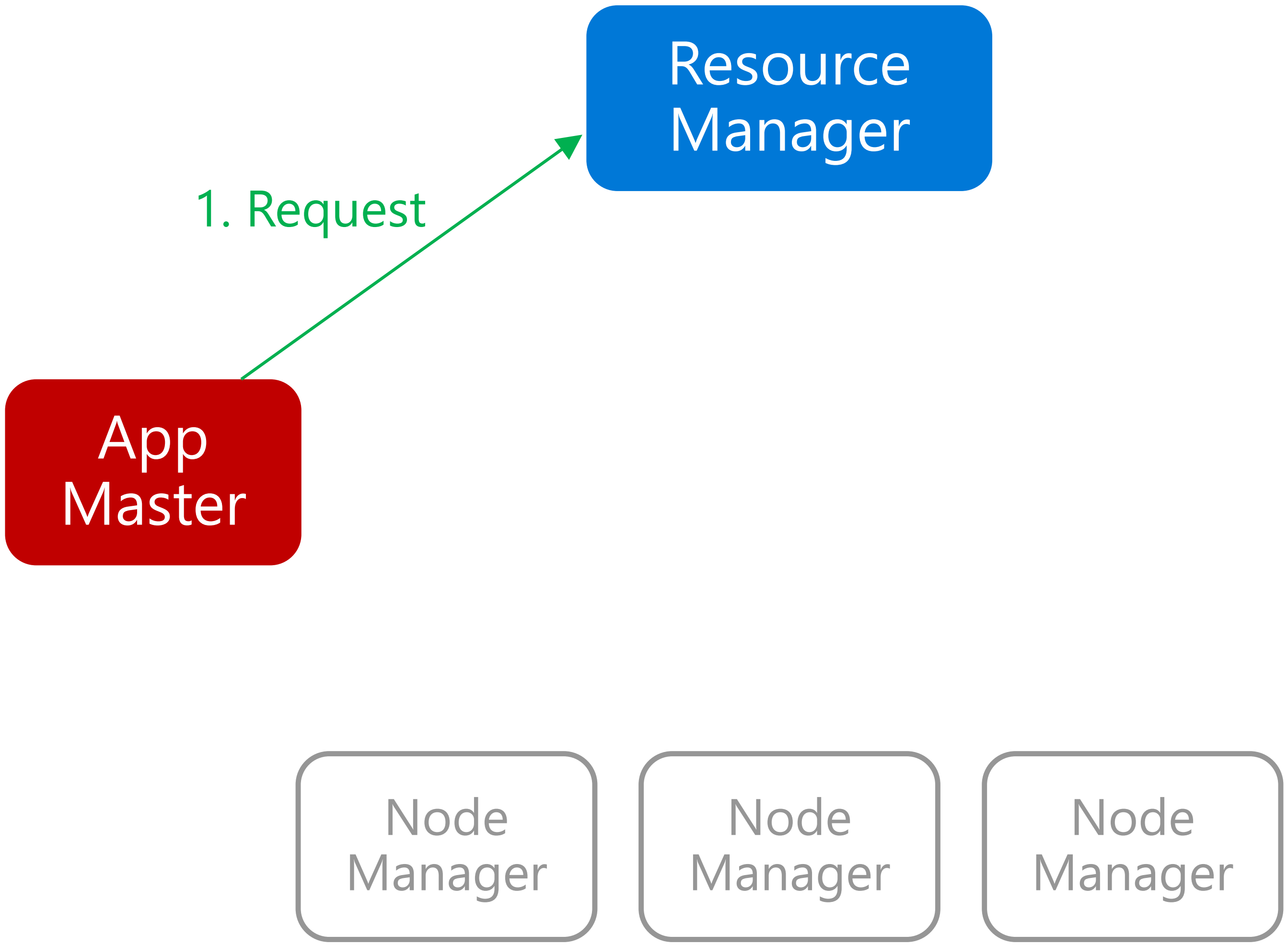
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources

What is a Resource Manager?



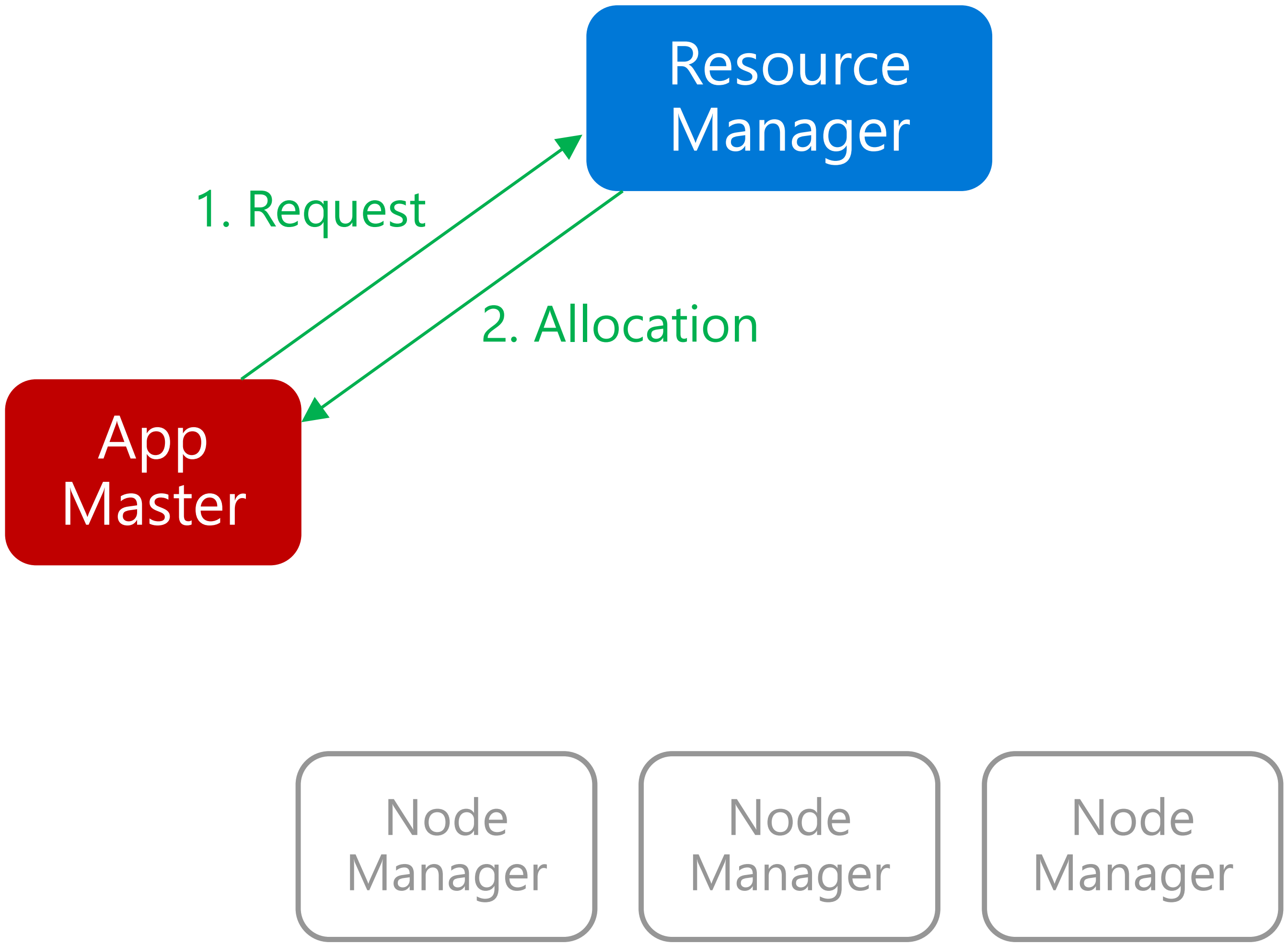
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources

What is a Resource Manager?



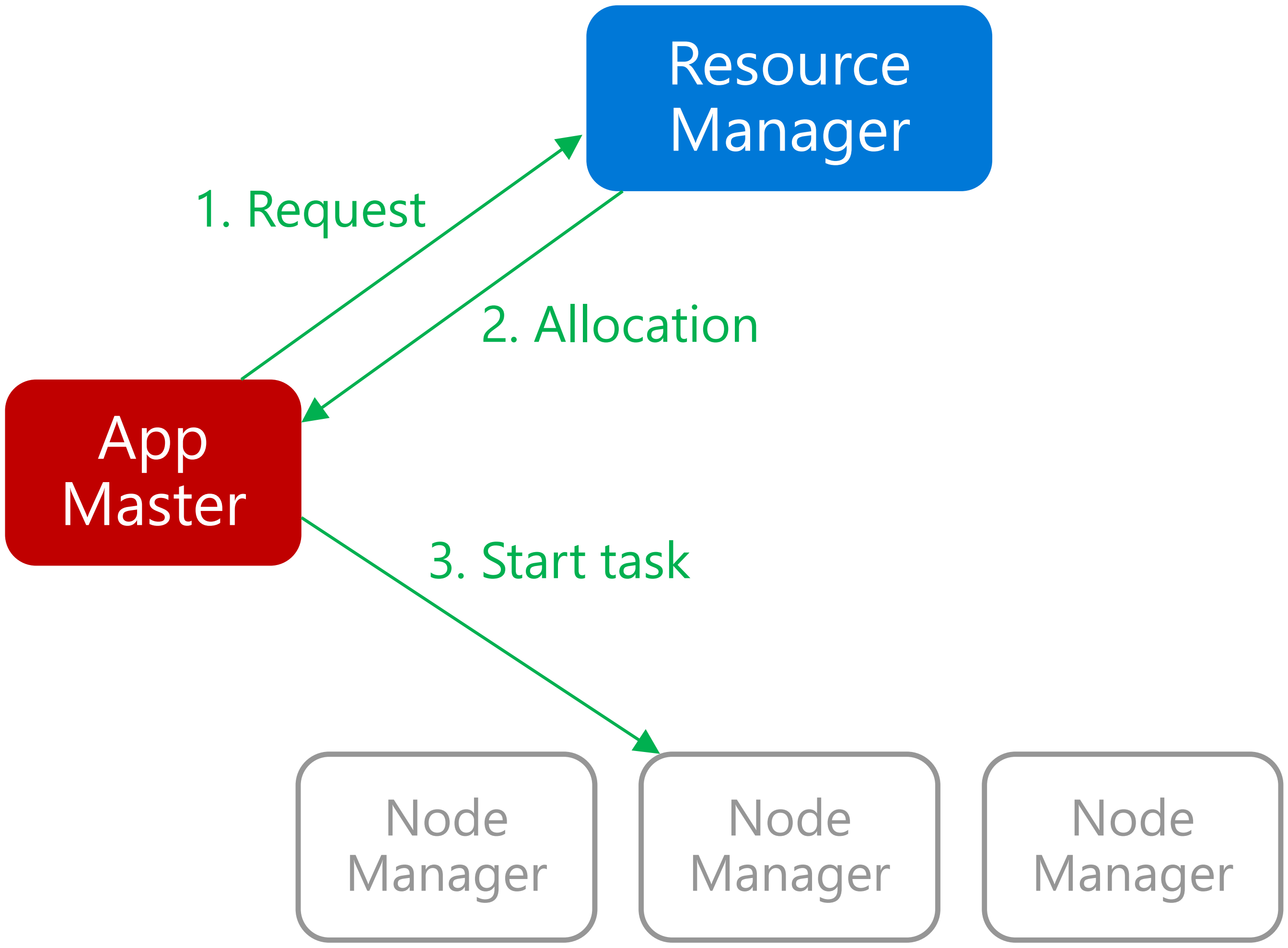
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources

What is a Resource Manager?



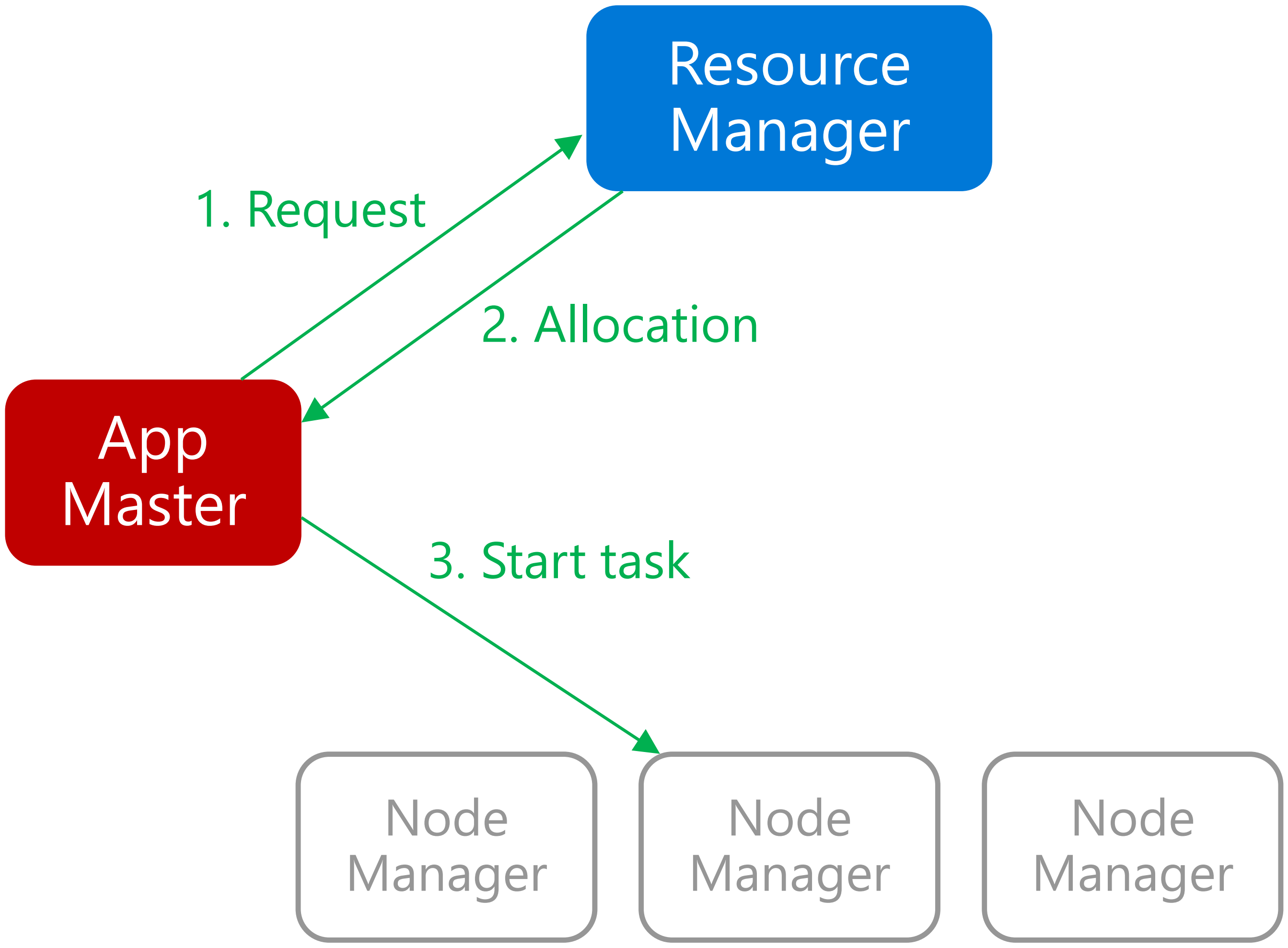
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources

What is a Resource Manager?



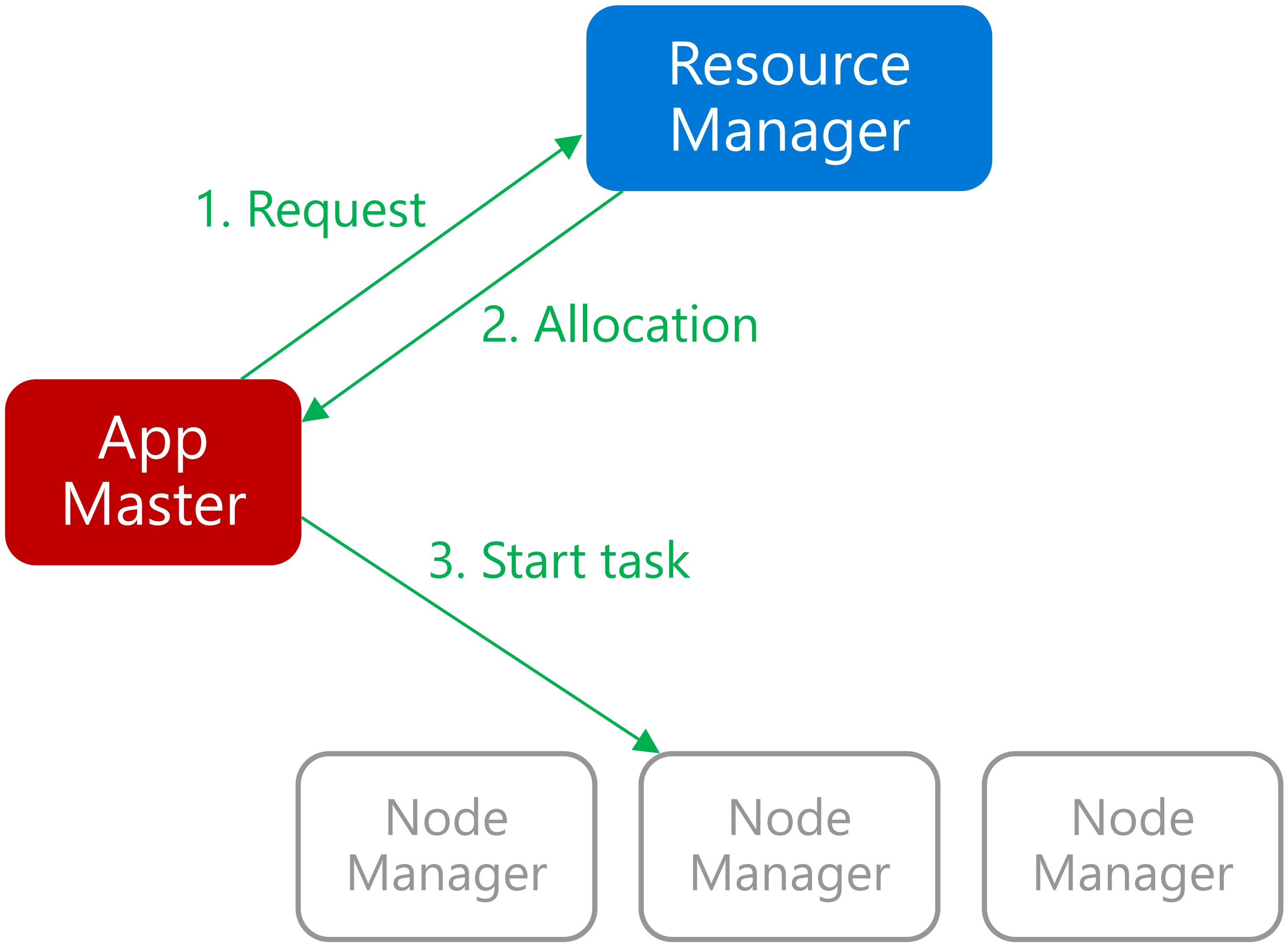
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources

What is a Resource Manager?



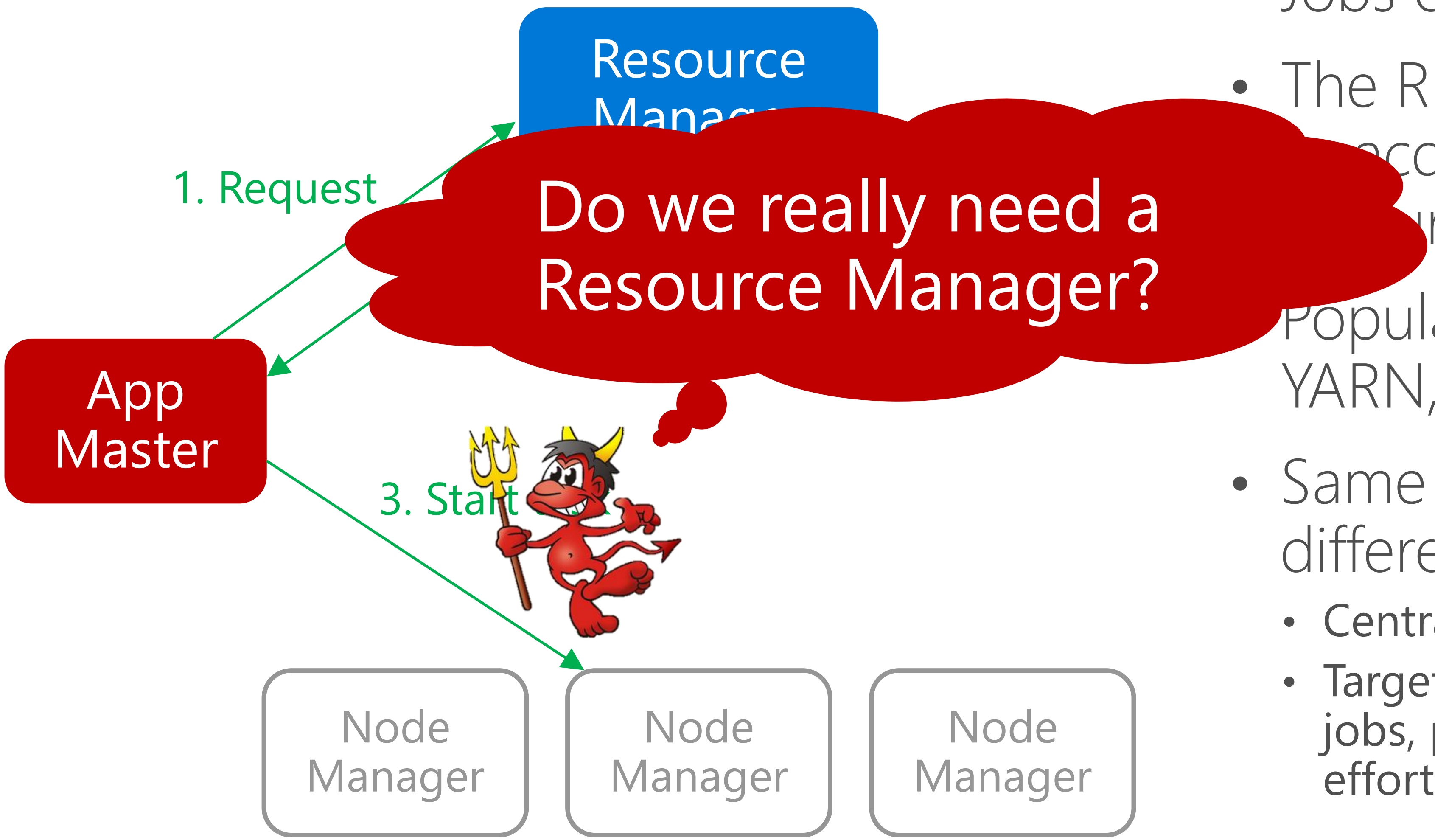
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources
- Popular examples: YARN, Borg, Mesos

What is a Resource Manager?



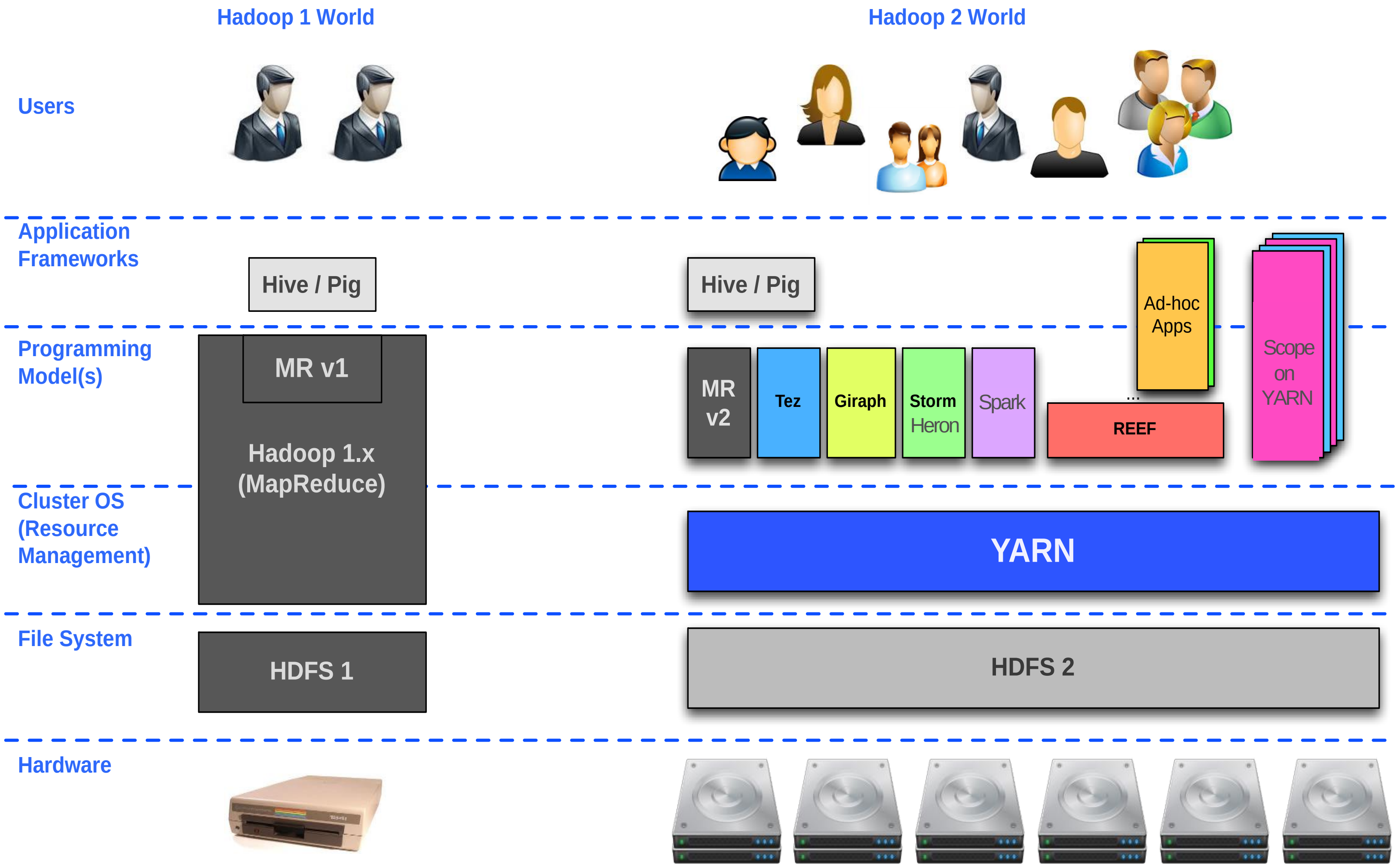
- Jobs consist of tasks
- The RM allows jobs to acquire cluster resources
- Popular examples: YARN, Borg, Mesos
- Same end goal, different designs
  - Centralized/distributed
  - Targeting batch/interactive jobs, production/best-effort jobs, services

What is a Resource Manager?



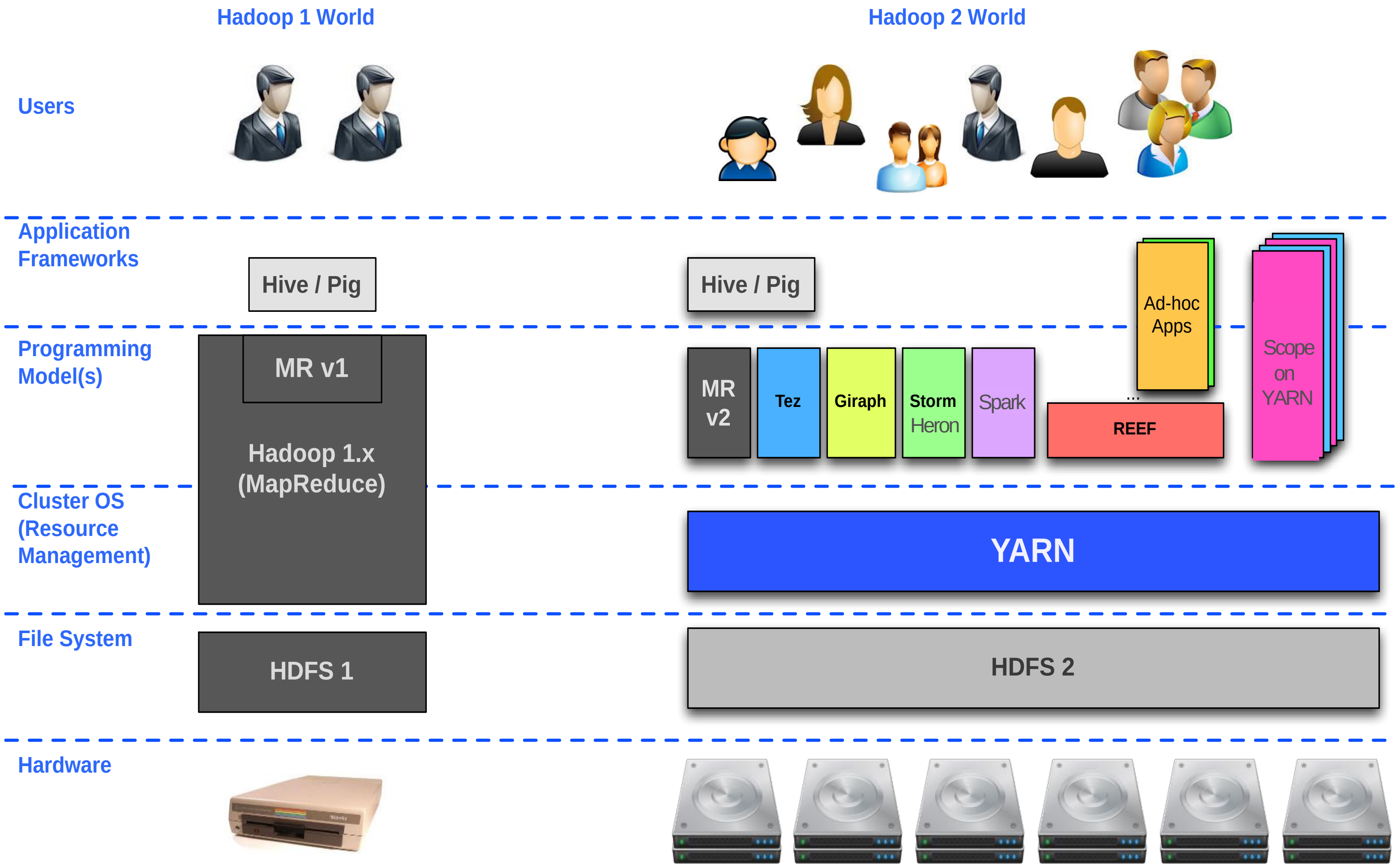
- Jobs consist of tasks
- The RM allows jobs ...cquire cluster ...rces
- ...opular examples: YARN, Borg, Mesos
- Same end goal, different designs
  - Centralized/distributed
  - Targeting batch/interactive jobs, production/best-effort jobs, services

Lessons learned: Abstracting out the RM layer



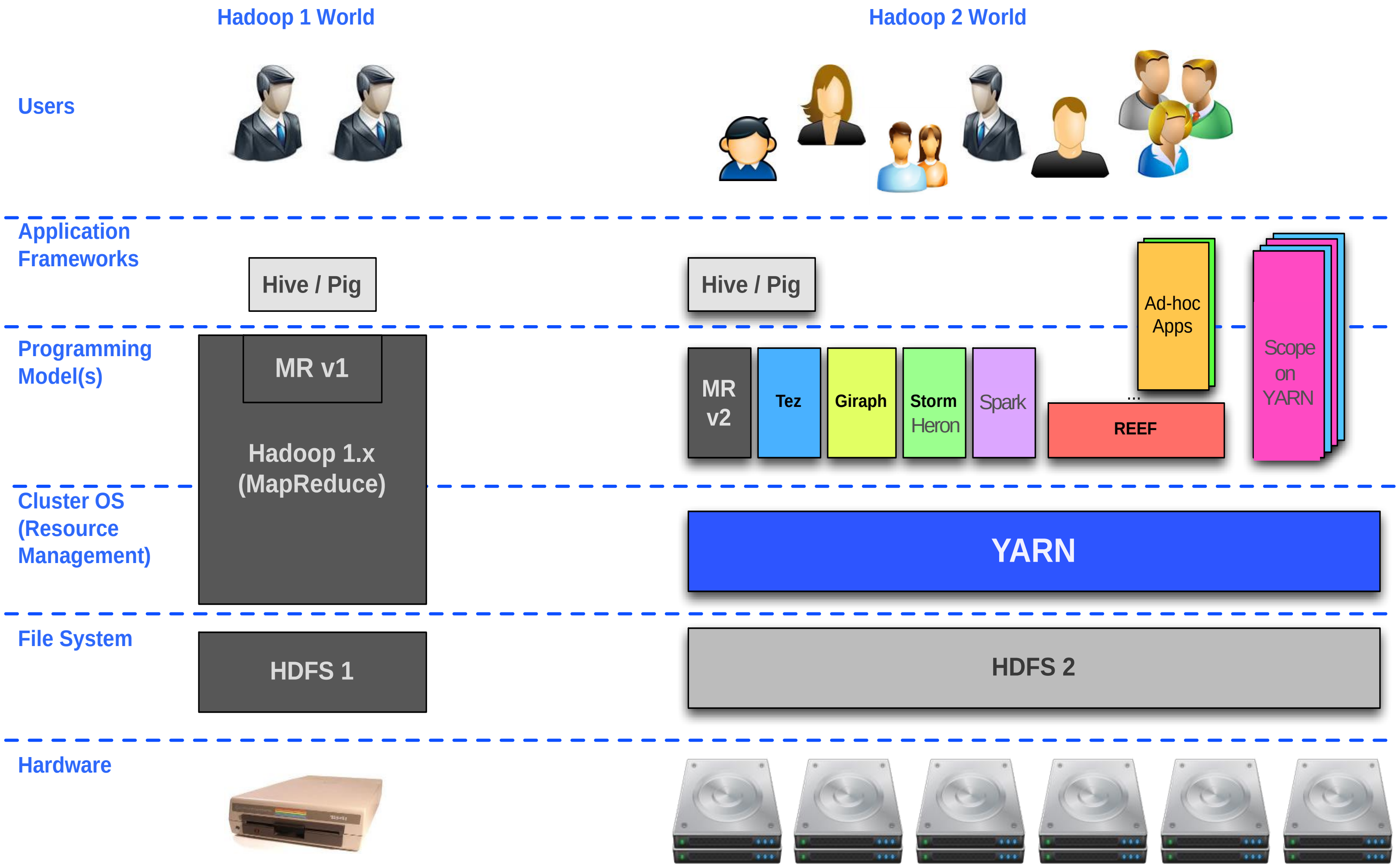
- Initial Big Data systems were monolithic (similar to databases)

Lessons learned: Abstracting out the RM layer



- Initial Big Data systems were monolithic (similar to databases)
- Reuse of RM component by multiple applications

Lessons learned: Abstracting out the RM layer



- Initial Big Data systems were monolithic (similar to databases)
- Reuse of RM component by multiple applications
- We focus on YARN, but most systems follow layering abstractions

Why YARN?

- Centralized scheduler
  - High-quality scheduling decisions
- Initial target: batch analytics jobs
  - Long task durations
- Sharing constraints
  - Fairness/capacity guarantees across users
- Scalability
  - Works well with clusters up to ~5000 nodes
- Mature open-source code base
  - Large community
  - Used by multiple companies (Yahoo!, Twitter, LinkedIn, Hortonworks, Cloudera)

Why YARN?

- Centralized scheduler
  - High-quality scheduling decisions
- Initial target:
  - Long task d
- Sharing constra
  - Fairness/capacity guarantees across users
- Scalability
  - Works well with clusters up to ~5000 nodes
- Mature open-source code base
  - Large community
  - Used by multiple companies (Yahoo!, Twitter, LinkedIn, Hortonworks, Cloudera)

But is all this good *enough* for the Microsoft clusters?

A closer look to our cluster needs

High resource utilization

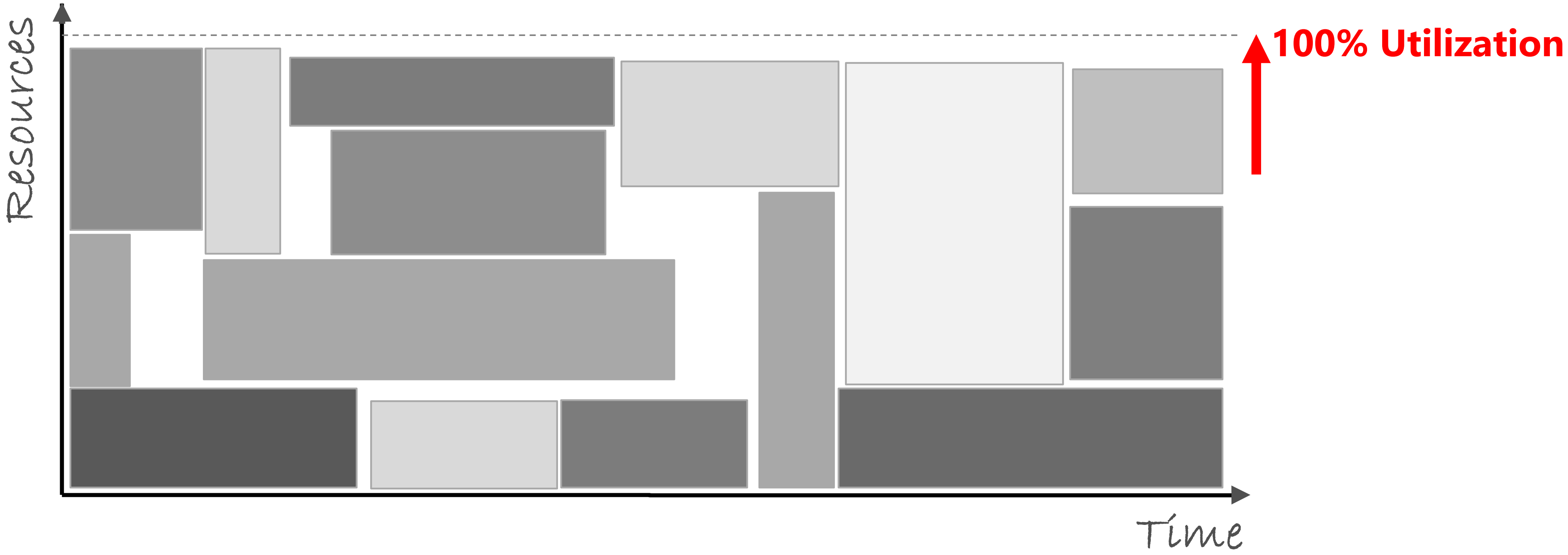
Scalability

Workload heterogeneity

Production jobs and predictability

Resource utilization

- Higher utilization → higher RoI
- Pack as many tasks as possible at each moment



Scalability

Scale to 5000   nodes

Scalability

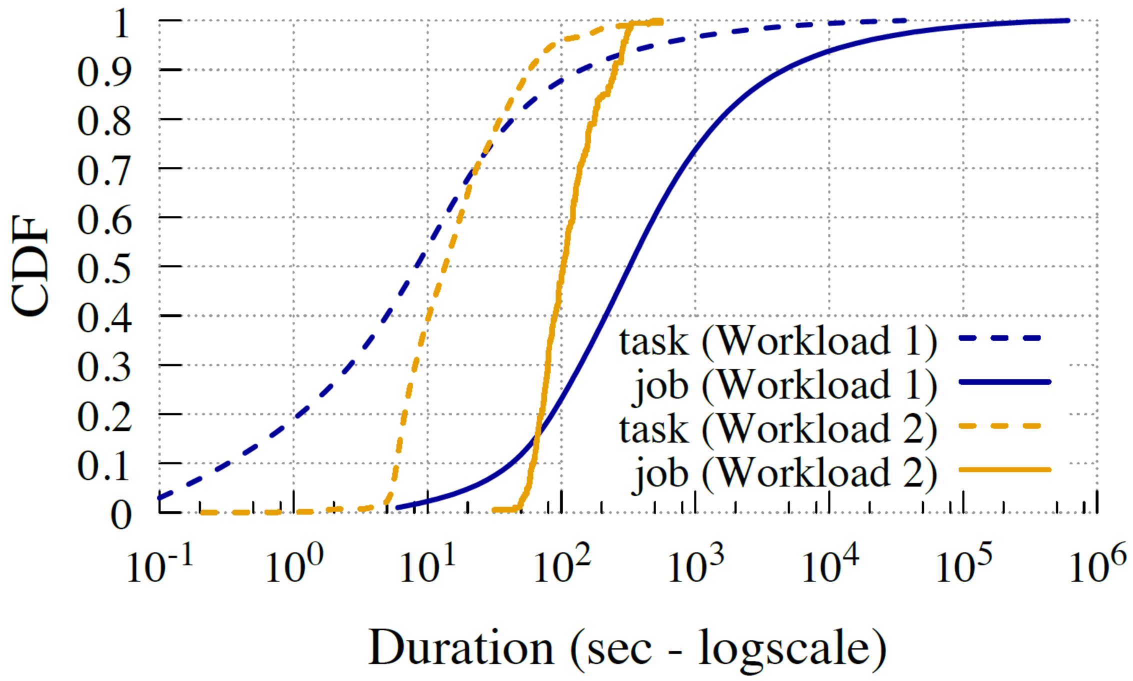
Scale to 500000 nodes

Workload heterogeneity

- Wide variety of workloads…
  - Production SLA jobs, best-effort jobs, services, interactive queries
- … and of task runtimes

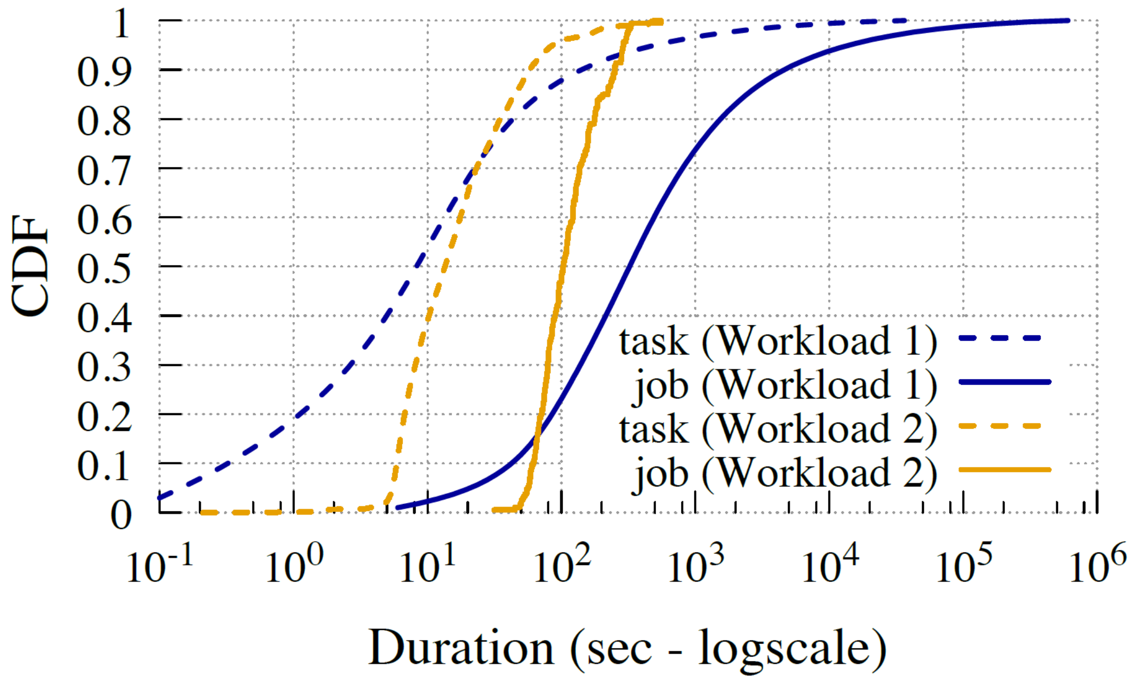
Workload heterogeneity

- Wide variety of workloads…
  - Production SLA jobs, best-effort jobs, services, interactive queries
- … and of task runtimes



Workload heterogeneity

- Wide variety of workloads…
  - Production SLA jobs, best-effort jobs, services, interactive queries
- … and of task runtimes



Workload heterogeneity in Cosmos

- Task runtime varies from sub-sec to 10,000+ sec
- 50% of tasks are shorter than 10 sec

Production jobs and predictability

- Production jobs typically have deadlines
  - "Job shows up at 3pm, deadline at 6am, requires X resources for 50 mins"
- Many SLA jobs are recurring
  - Empirically >60% of jobs in our clusters
- Predictability is crucial
  - "Why is my job running slower than yesterday?"
  - 25% of user tickets due to unpredictability
- Current work-around
  - >75% of our jobs are over-provisioned

Our solutions

4 Hadoop committers in CISL
404 patches as of last night

- Rayon/Morpheus: support SLOs via reservations
  - OSS: in Hadoop 2.6 [YARN-1051], Publications: SoCC 2014, OSDI 2016
- Mercury/Yaq: improve utilization via container types and node-side queuing
  - OSS: in Hadoop 3.0 [YARN-2877], Publications: ATC 2015, EuroSys 2016
- YARN Federation: scale-out YARN by federating multiple clusters
  - OSS: currently open-sourced [YARN-2915]
- Medea: support for long-running applications with complex placement constraints
  - Research prototype

Microsoft is transitioning its Big-Data clusters to (the above) YARN-based RM infrastructure

Our solutions

4 Hadoop committers in CISL

404 patches as of last night

- Rayon/Morpheus: support SLOs via reservations
  - OSS: in Hadoop 2.6 [YARN-1051], Publications: SoCC 2014, OSDI 2016
- Mercury/Yaq: improve utilization via container types and node-side queuing
  - OSS: in Hadoop 3.0 [YARN-2877], Publications: ATC 2015, EuroSys 2016
- YARN Federation: scale-out YARN by federating multiple clusters
  - OSS: currently open-sourced [YARN-2915]
- Medea: support for long-running applications with complex placement constraints
  - Research prototype

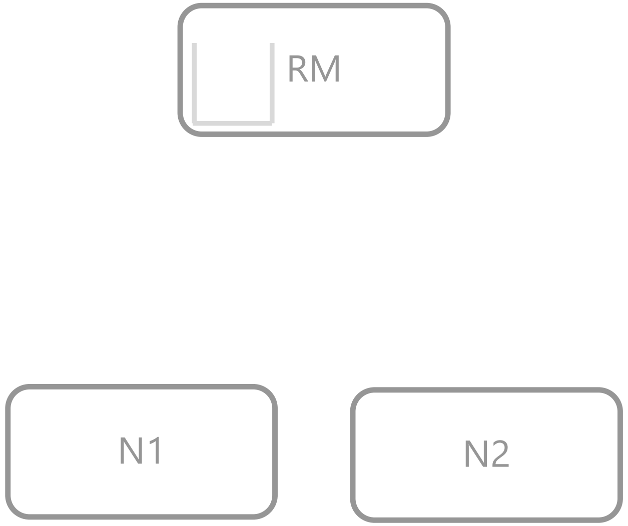
Microsoft is transitioning its Big-Data clusters to (the above) YARN-based RM infrastructure

Mercury/Yaq

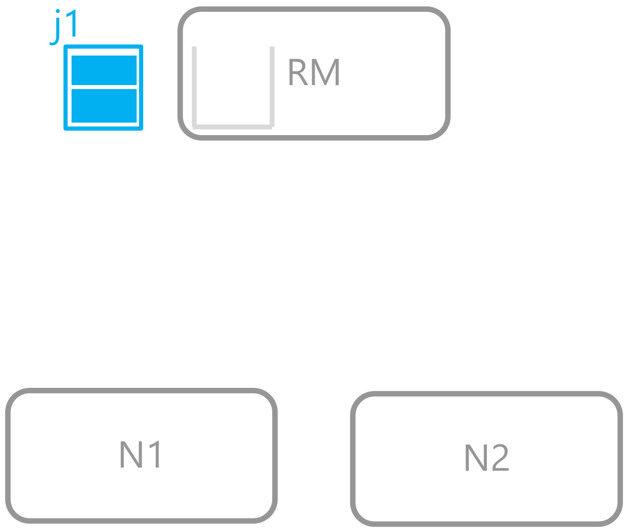
Improve resource utilization (and job completion time)

[Hadoop 3.0; ATC 2015, EuroSys 2016]

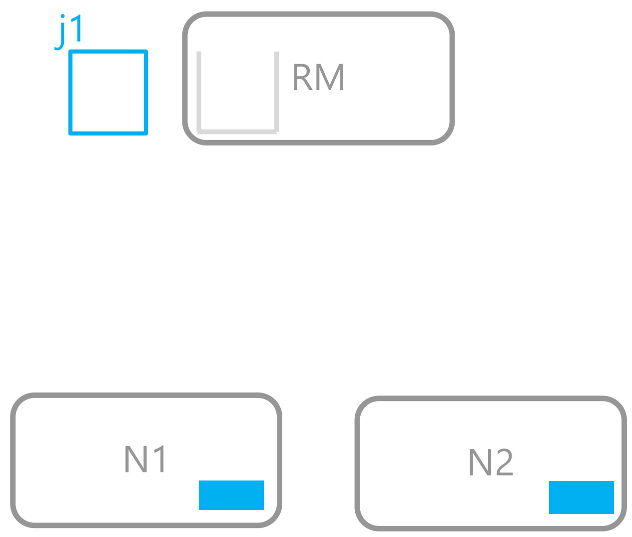
Resource utilization in YARN



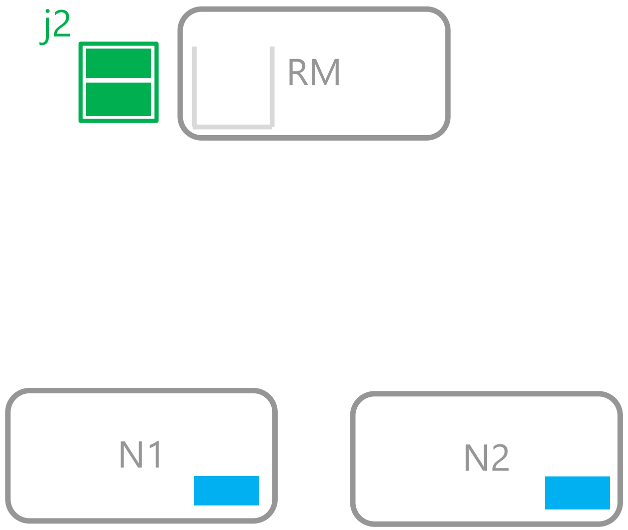
Resource utilization in YARN



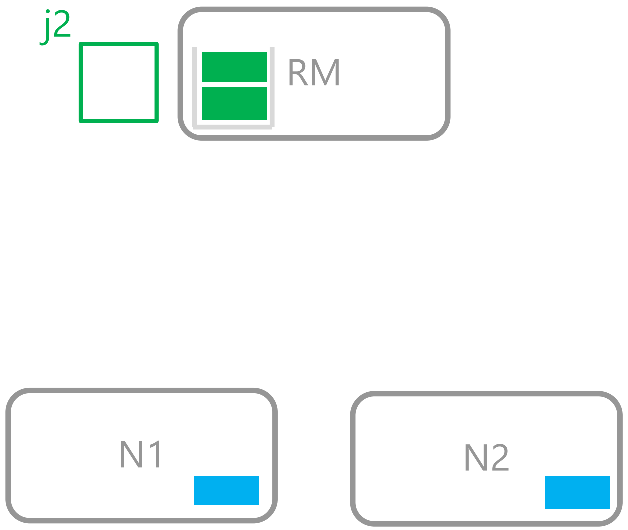
Resource utilization in YARN



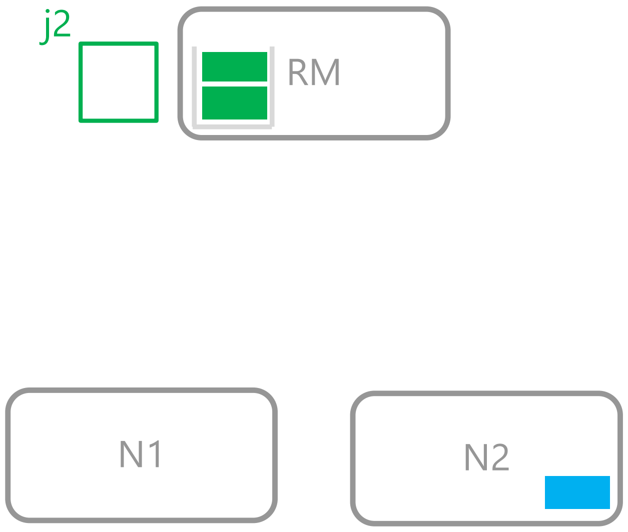
Resource utilization in YARN



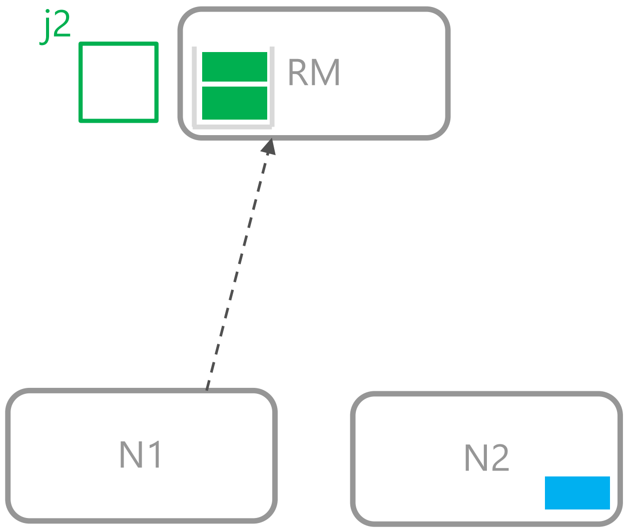
Resource utilization in YARN



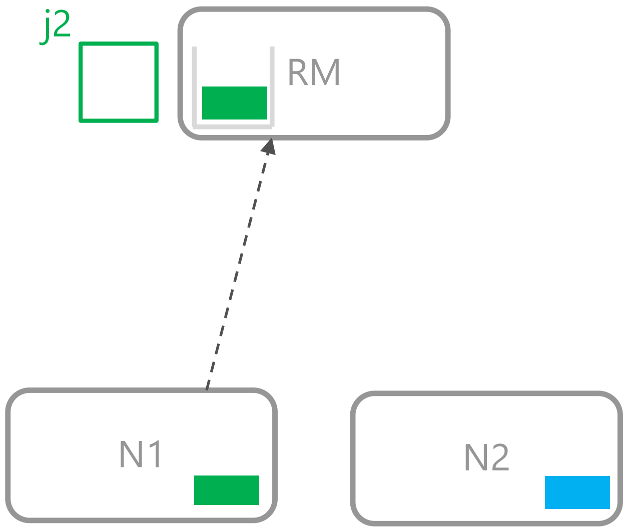
Resource utilization in YARN



Resource utilization in YARN

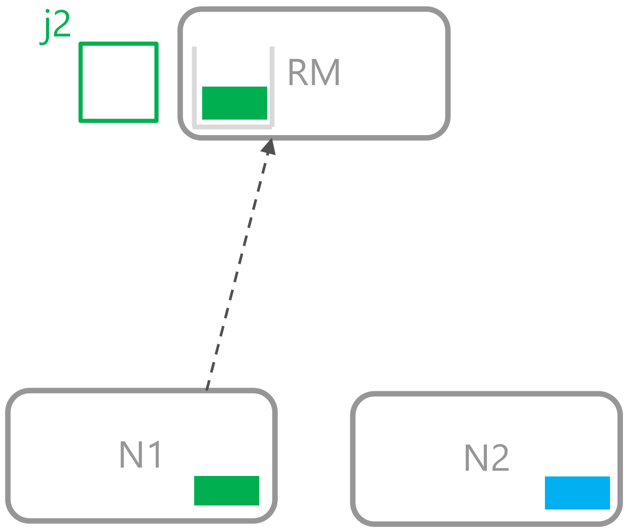


Resource utilization in YARN

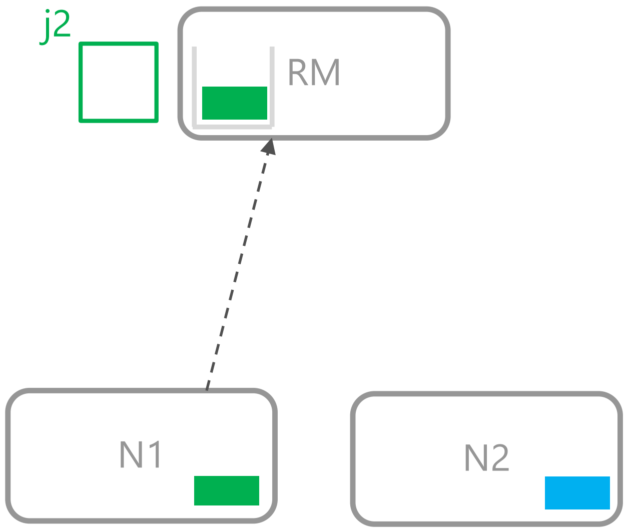


Resource utilization in YARN

- Feedback delays impact cluster utilization
  - RM in the critical path of all scheduling decisions
  - Resources can remain idle between allocations
  - Resource utilization suboptimal, especially for shorter tasks



Resource utilization in YARN

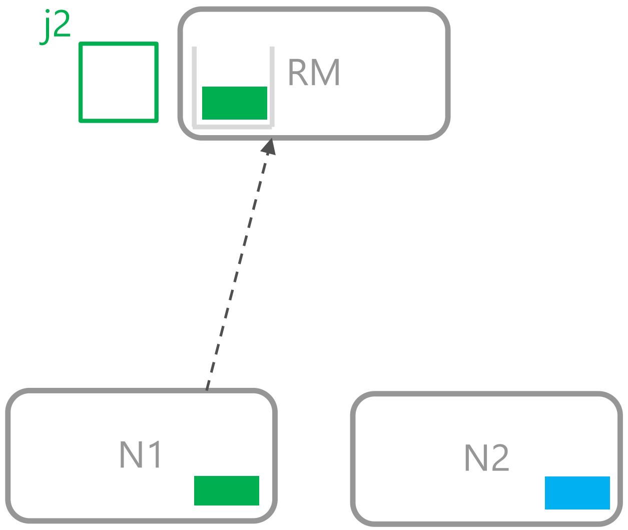


- Feedback delays impact cluster utilization
  - RM in the critical path of all scheduling decisions
  - Resources can remain idle between allocations
  - Resource utilization suboptimal, especially for shorter tasks

5 sec	10 sec	50 sec	Mixed-5-50	Cosmos-gm
60.59%	78.35%	92.38%	78.54%	83.38%

Average allocated resources for varying workloads.

Resource utilization in YARN



- Feedback delays impact cluster utilization
  - RM in the critical path of all scheduling decisions
  - Resources can remain idle between allocations
  - Resource utilization suboptimal, especially for shorter tasks

5 sec	10 sec	50 sec	Mixed-5-50	Cosmos-gm
60.59%	78.35%	92.38%	78.54%	83.38%

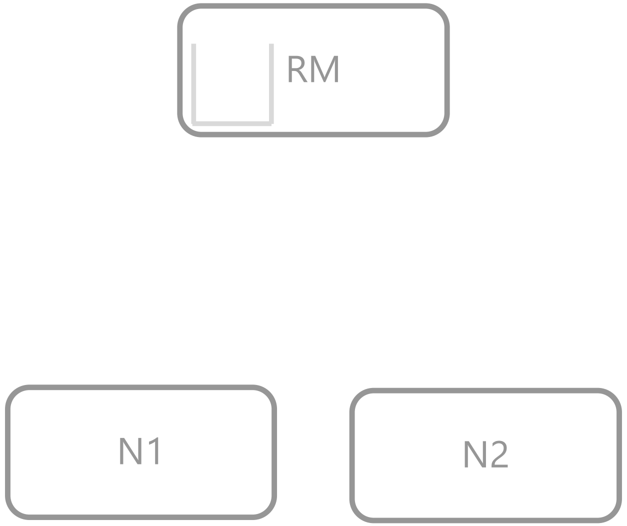
Average allocated resources for varying workloads.

- Actual resource utilization is even lower
  - E.g., a task using 1GB out of a 4GB allocated container
  - Resource overprovisioning makes matters worse

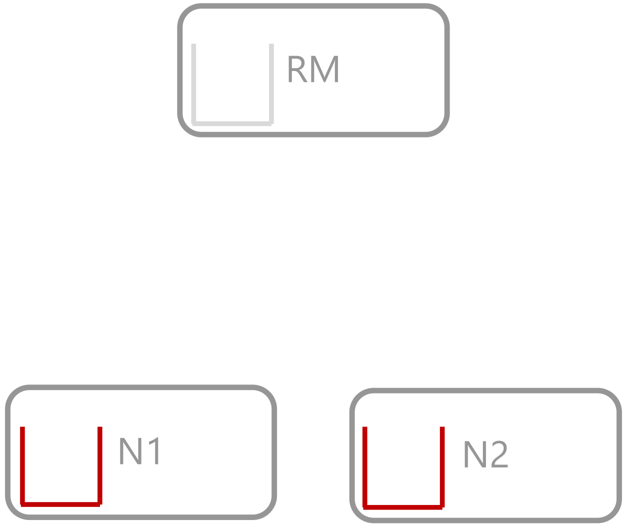
Mercury: Key ideas

- Introduce task queuing at nodes
  - Mask feedback delays
  - Improve cluster utilization
  - Improve task throughput (by up to 40%)
- Container types
  - GUARANTEED and OPPORTUNISTIC
  - Keep guarantees for important jobs
  - Use opportunistic execution to improve utilization

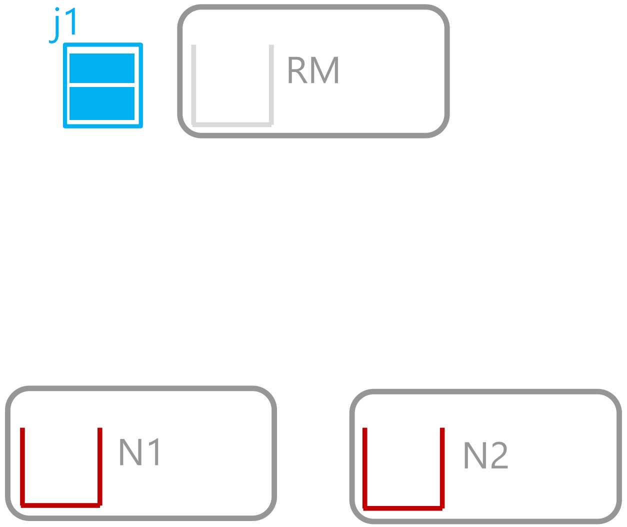
Node-side queuing



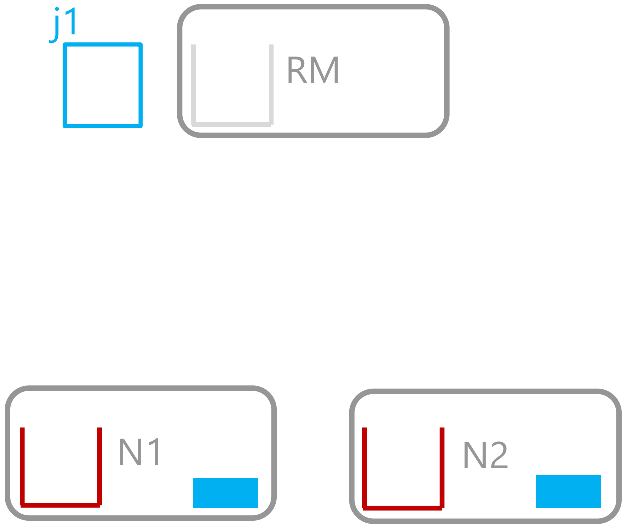
Node-side queuing



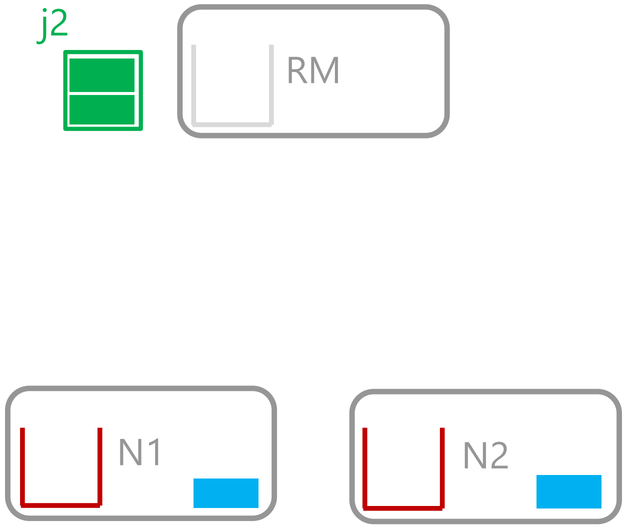
Node-side queuing



Node-side queuing

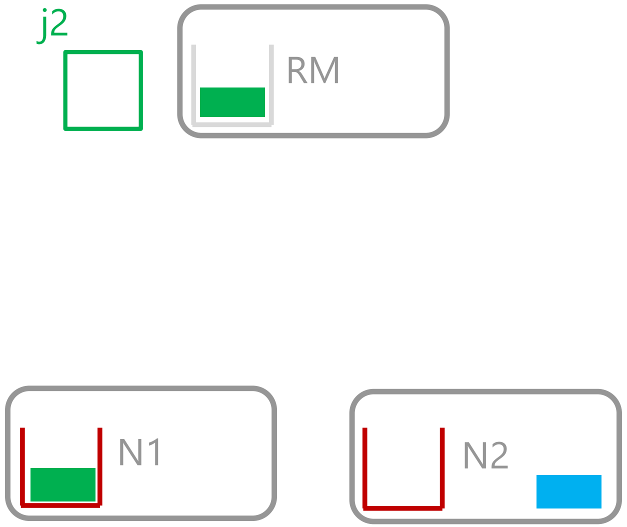


Node-side queuing

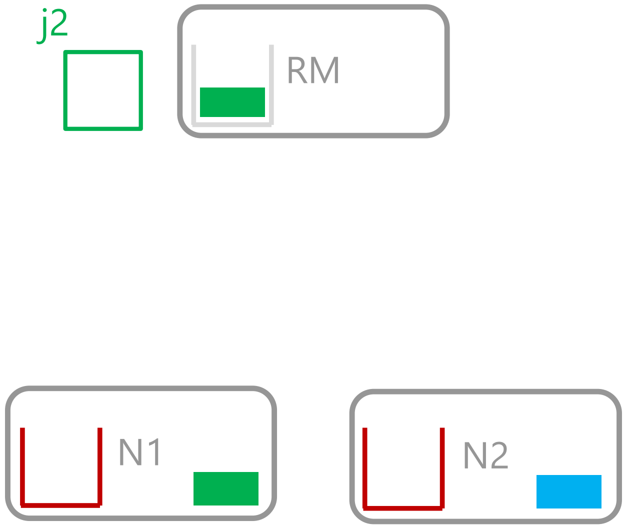


Node-side queuing

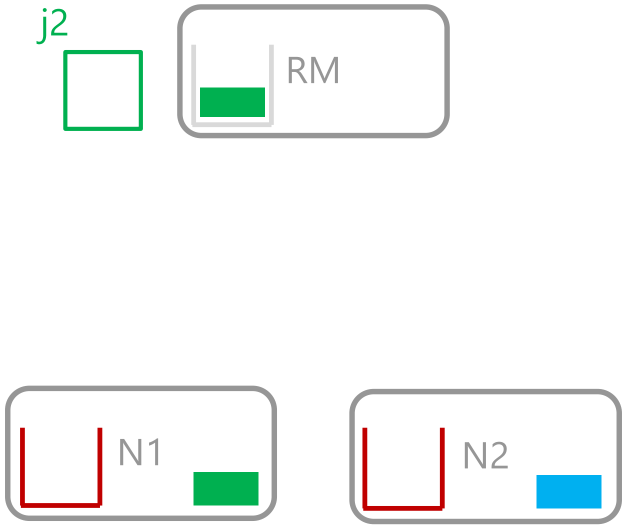
Node-side queuing



Node-side queuing

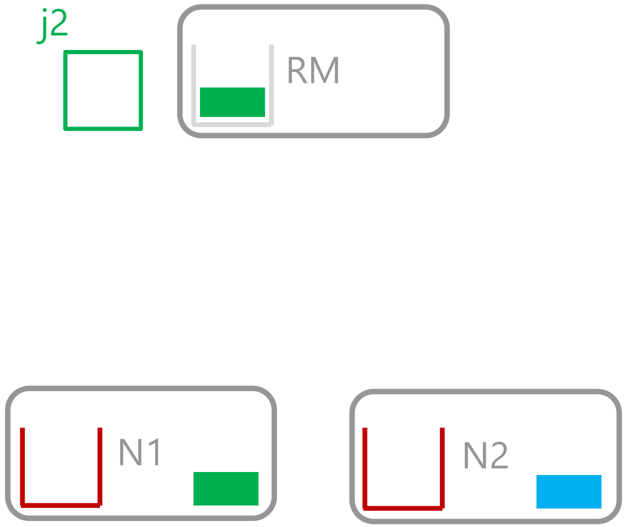


Node-side queuing



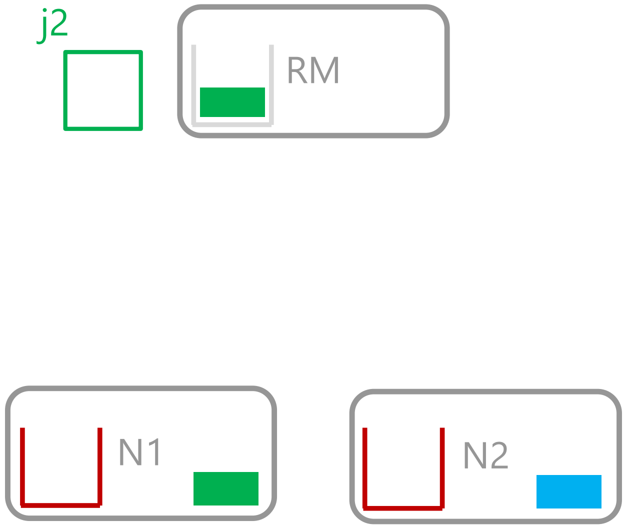
- Tasks can be queued:
  - At the resource manager (RM)
  - At the nodes

Node-side queuing



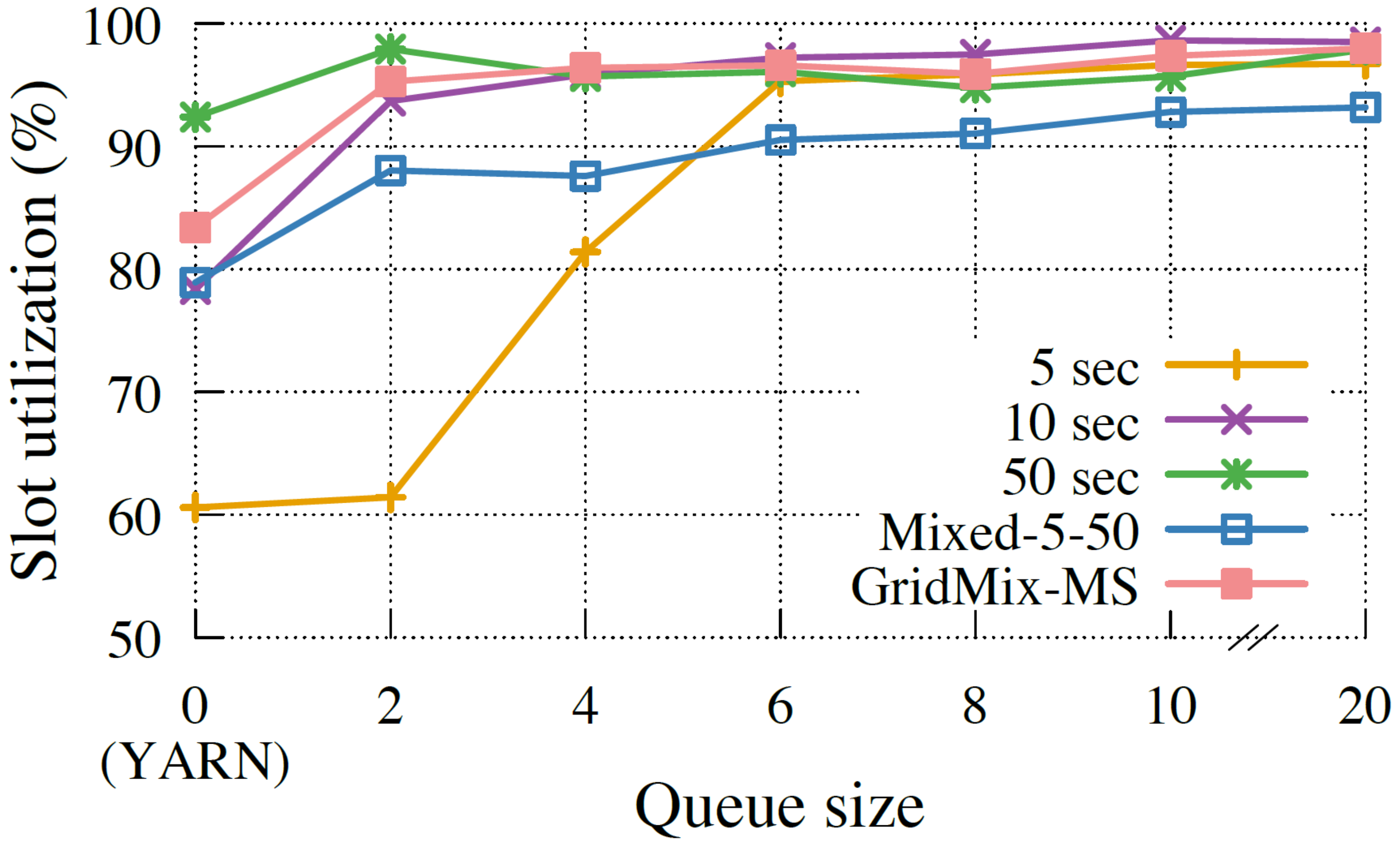
- Tasks can be queued:
  - At the resource manager (RM)
  - At the nodes
- Existing centralized schedulers do *not* queue tasks at nodes

Node-side queuing

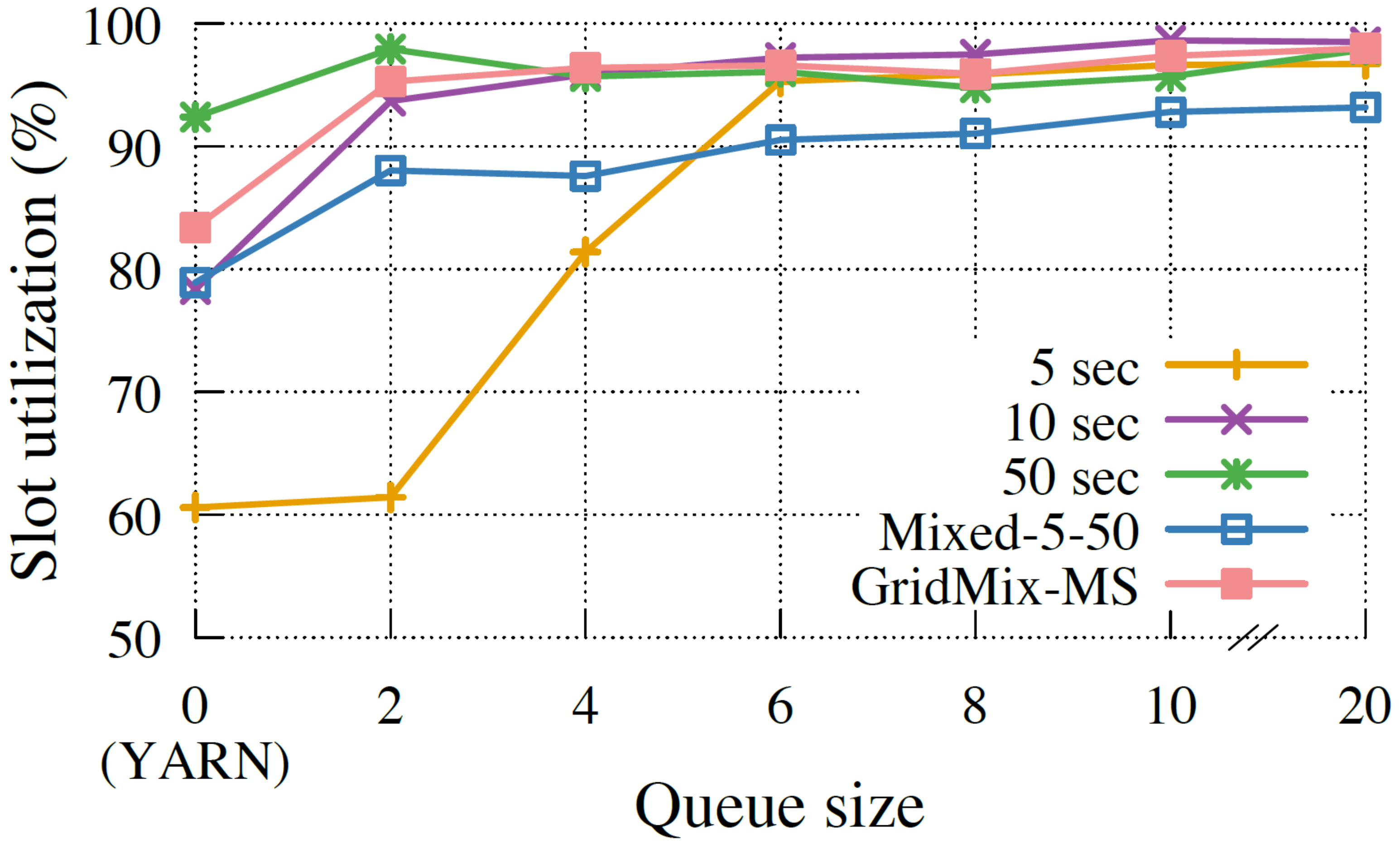


- Tasks can be queued:
  - At the resource manager (RM)
  - At the nodes
- Existing centralized schedulers do *not* queue tasks at nodes
  - Challenging to get right

Utilization gains

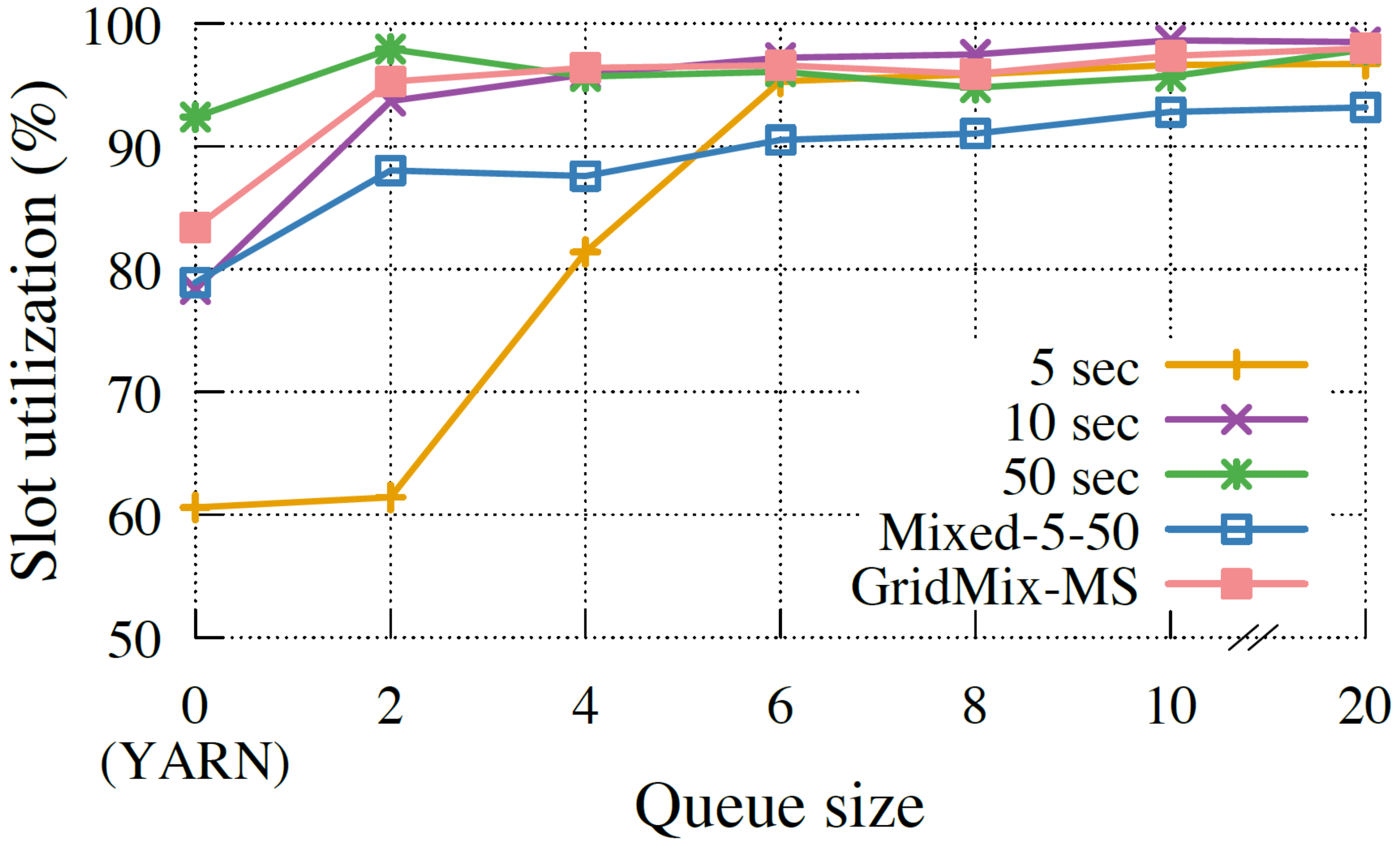


Utilization gains



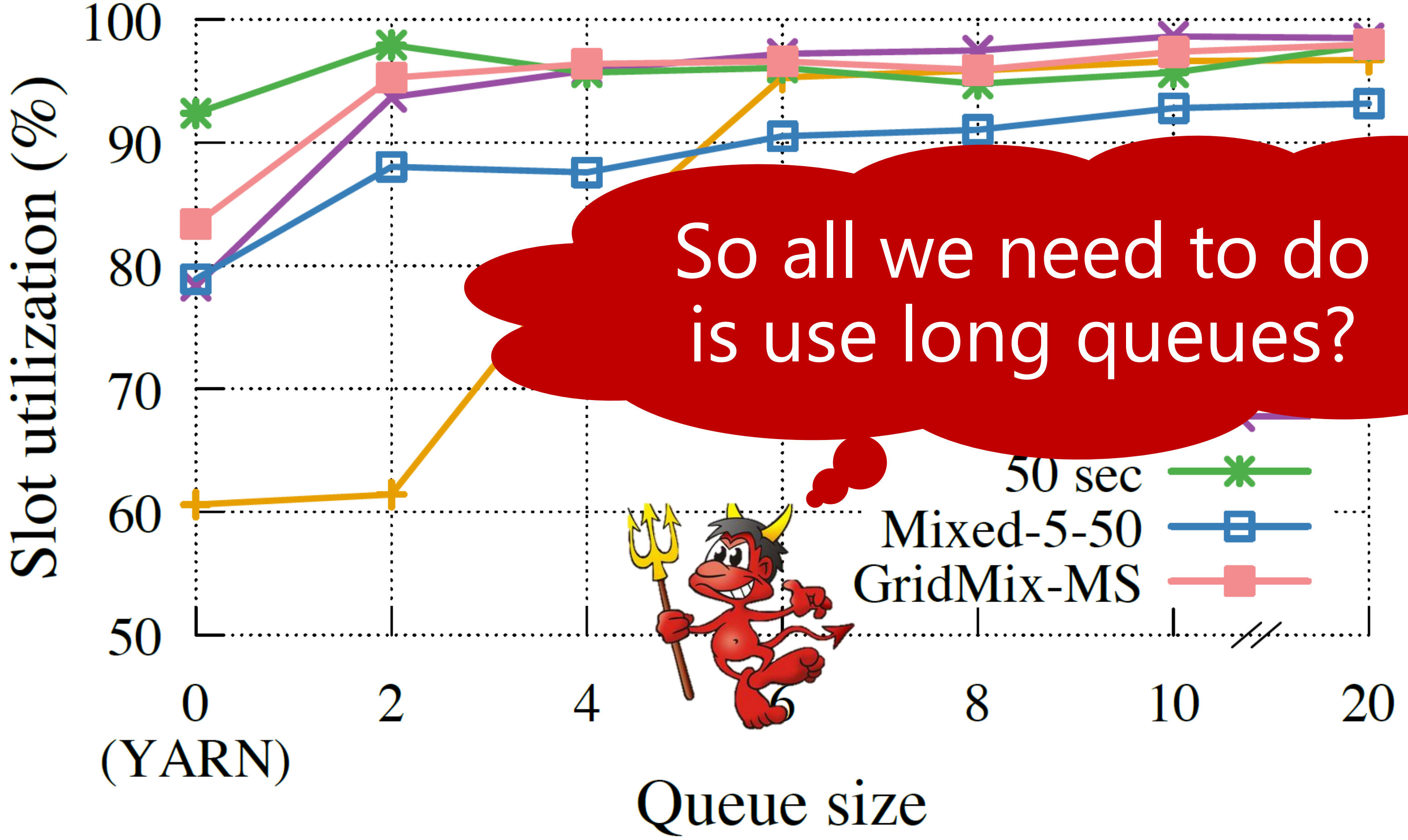
- Sufficiently long queues lead to optimal utilization

Utilization gains



- Sufficiently long queues lead to optimal utilization
- The shorter the tasks the longer the queues need to be

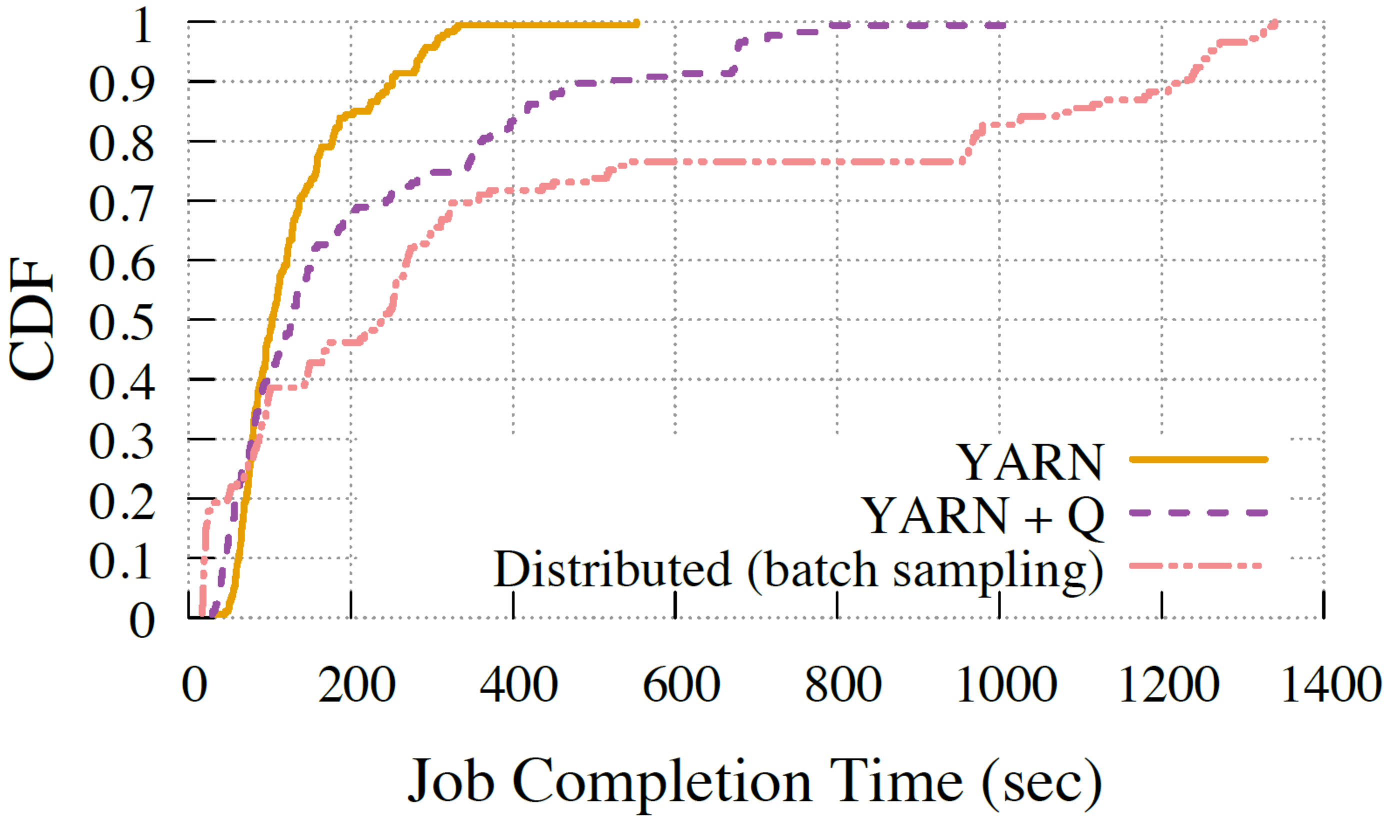
Utilization gains



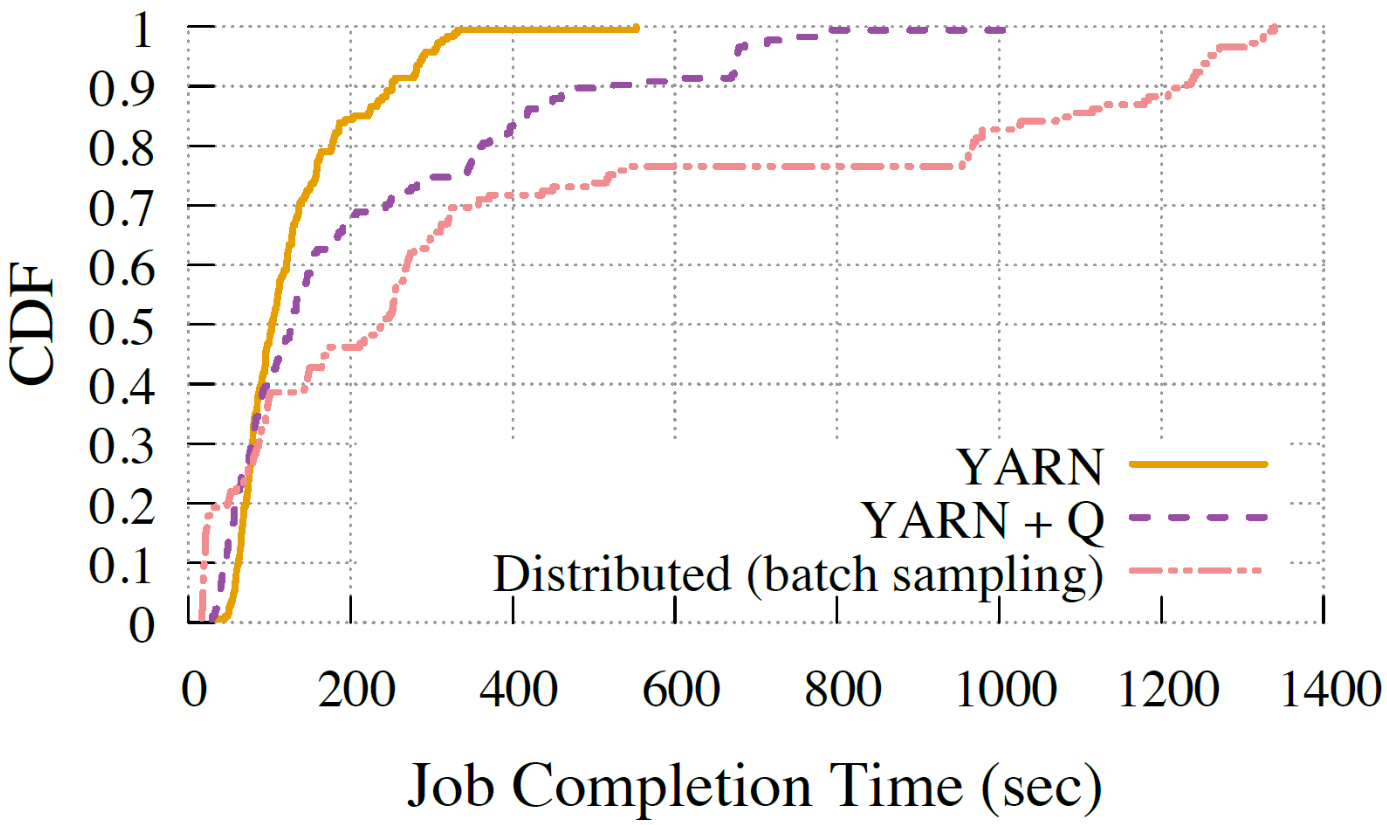
So all we need to do is use long queues?

- Sufficiently long queues lead to …imal utilization
- …e shorter the tasks the longer the queues need to be

Job completion times with node-side queuing

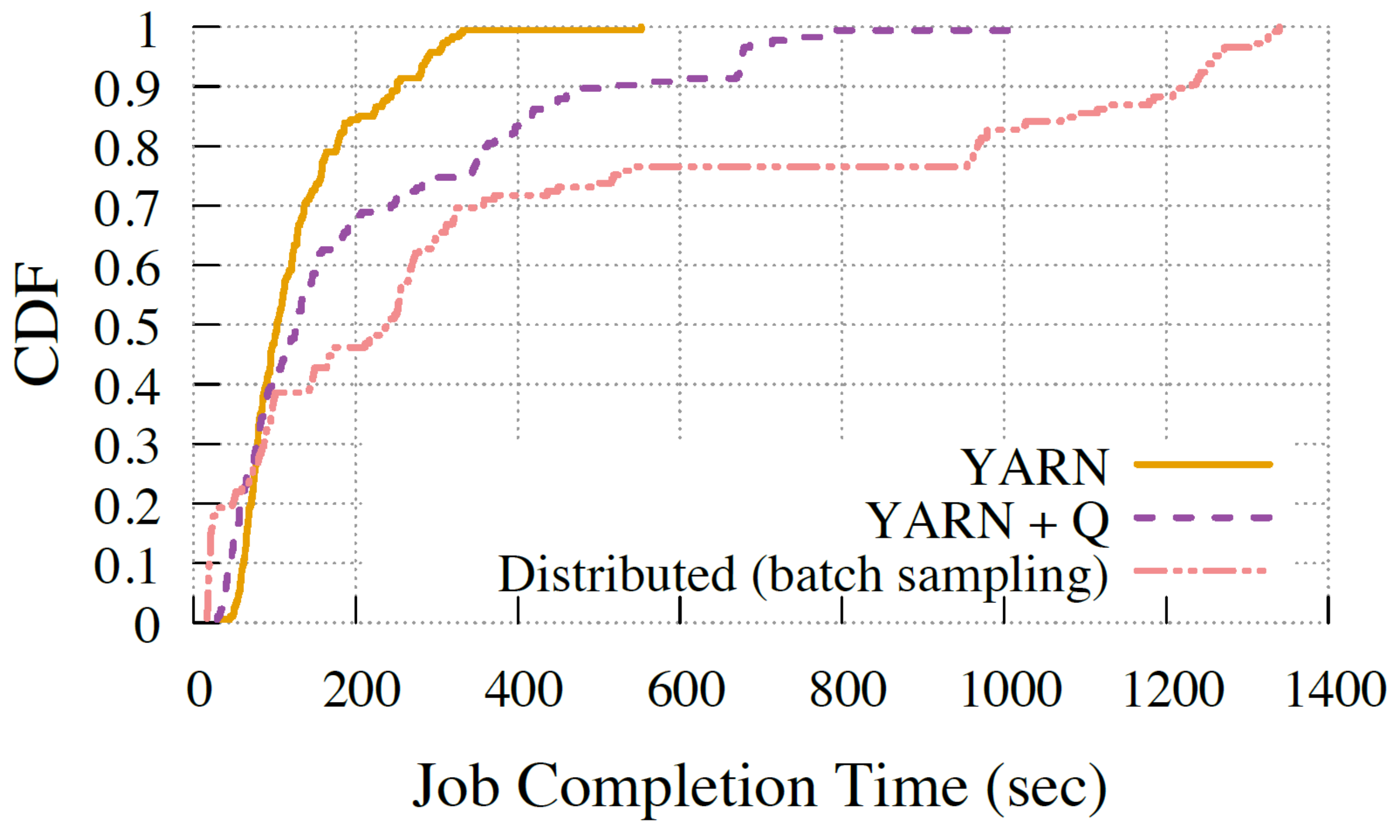


Job completion times with node-side queuing



- Naïve node-side queuing can be detrimental for job completion times
  - Despite the utilization gains

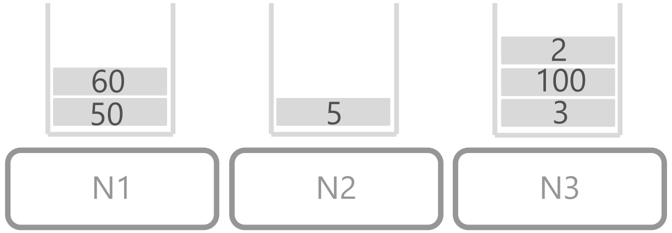
Job completion times with node-side queuing



- Naïve node-side queuing can be detrimental for job completion times
  - Despite the utilization gains

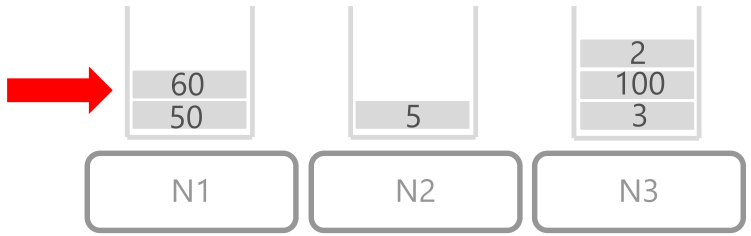
Proper queue management techniques are required

Problems with node-side queuing



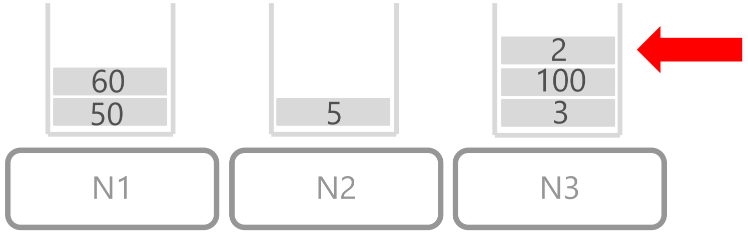
Problems with node-side queuing

- Load imbalance across nodes
  - Suboptimal task placement



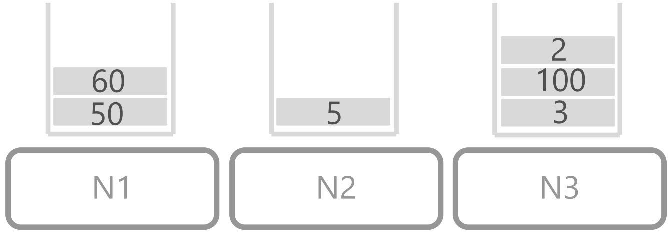
Problems with node-side queuing

- Load imbalance across nodes
  - Suboptimal task placement
- Head-of-line blocking
  - Especially for heterogeneous tasks



Problems with node-side queuing

- Load imbalance across nodes
  - Suboptimal task placement
- Head-of-line blocking
  - Especially for heterogeneous tasks
- Early binding of tasks to nodes



Yaq: Queue management techniques

Place tasks to node queues

Prioritize task execution (queue reordering)

Bound queue lengths

Yaq: Queue management techniques

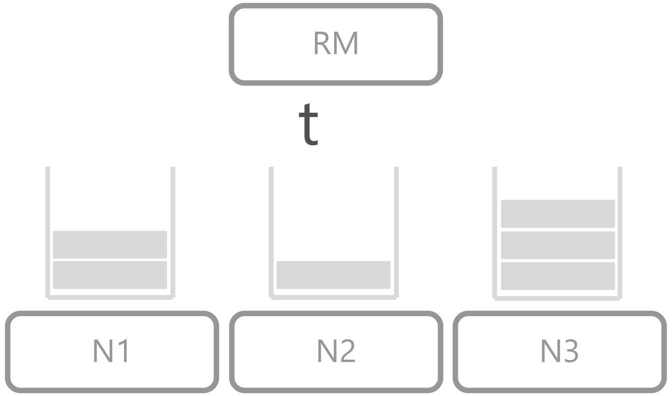
Place tasks to node queues

Prioritize task execution (queue reordering)

Bound queue lengths

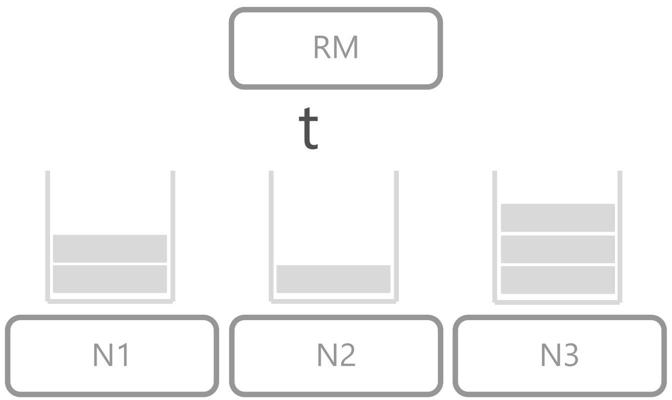
Yaq improves median job completion time by 1.7x over YARN

Placement of Tasks to Queues



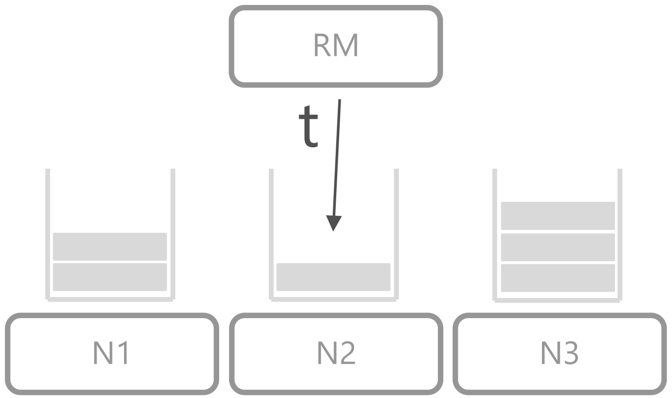
Placement of Tasks to Queues

- Placement based on queue length
  - Agnostic of task characteristics
  - Suboptimal placement for heterogeneous workloads



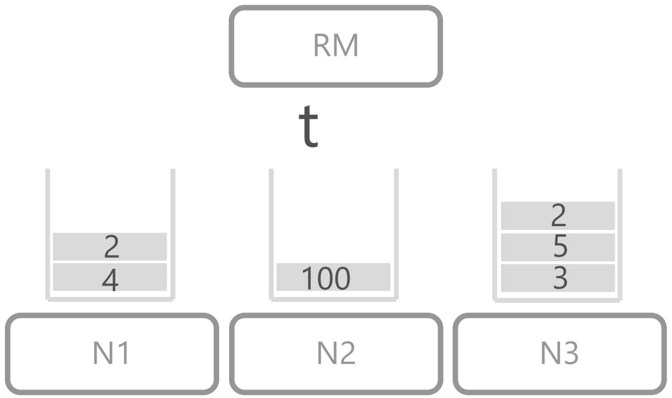
Placement of Tasks to Queues

- Placement based on queue length
  - Agnostic of task characteristics
  - Suboptimal placement for heterogeneous workloads



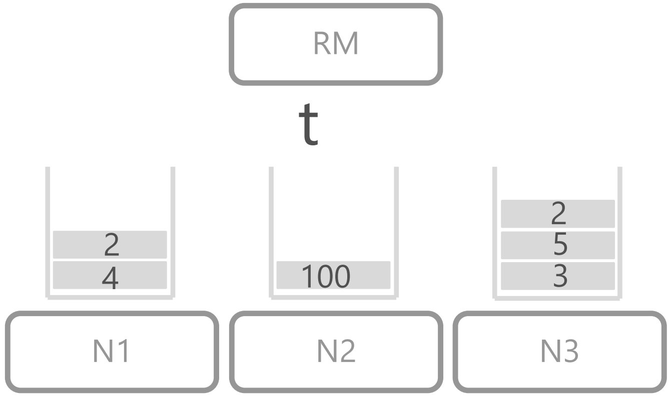
Placement of Tasks to Queues

- Placement based on queue length
  - Agnostic of task characteristics
  - Suboptimal placement for heterogeneous workloads



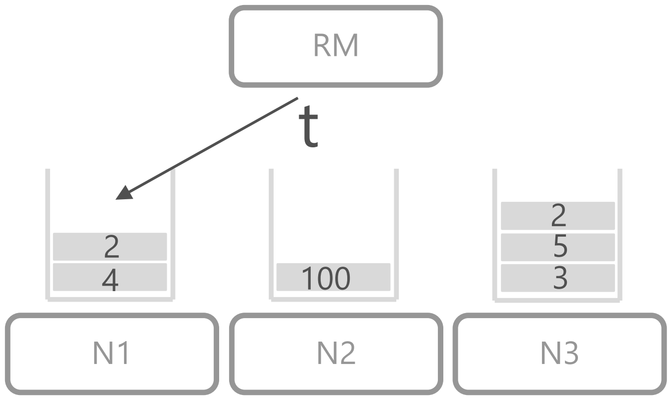
Placement of Tasks to Queues

- Placement based on queue length
  - Agnostic of task characteristics
  - Suboptimal placement for heterogeneous workloads
- Placement based on queue wait time
  - Better for heterogeneous workloads
  - Requires task duration estimates



Placement of Tasks to Queues

- Placement based on queue length
  - Agnostic of task characteristics
  - Suboptimal placement for heterogeneous workloads
- Placement based on queue wait time
  - Better for heterogeneous workloads
  - Requires task duration estimates

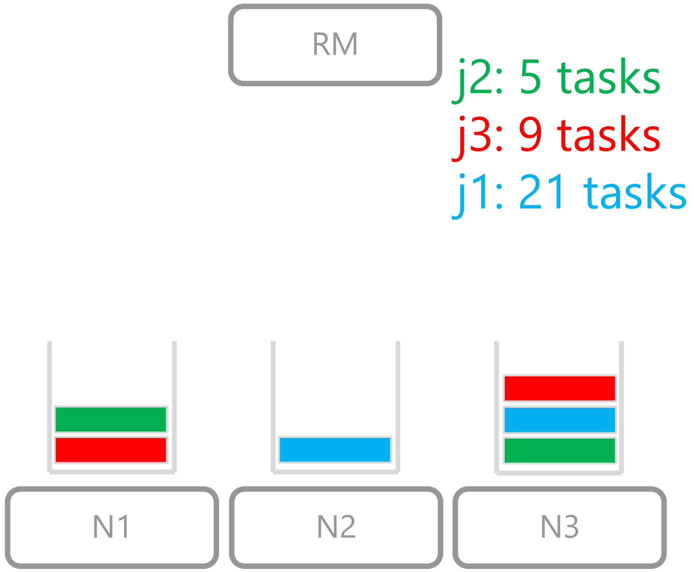


Task Prioritization

- Queue reordering strategies
  - Shortest Remaining Job First (SRJF)
  - Least Remaining Tasks First (LRTF)
  - Shortest Task First (STF)
  - Earliest Job First (EJF)

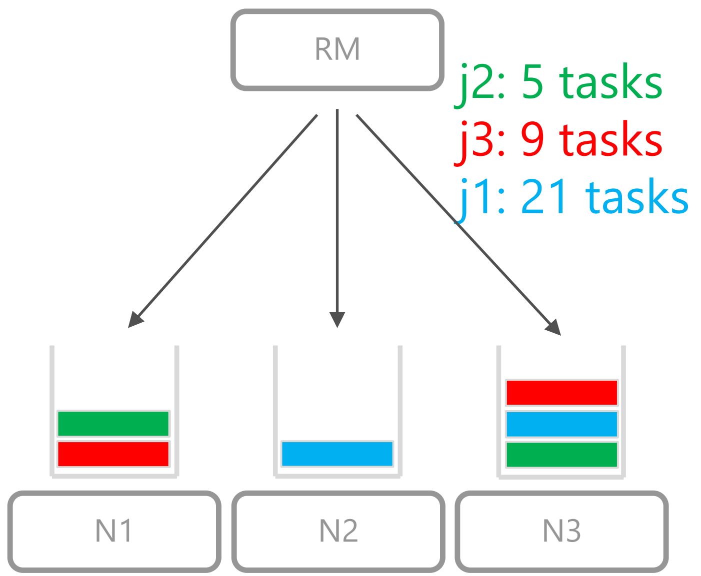
Task Prioritization

- Queue reordering strategies
  - Shortest Remaining Job First (SRJF)
  - Least Remaining Tasks First (LRTF)
  - Shortest Task First (STF)
  - Earliest Job First (EJF)



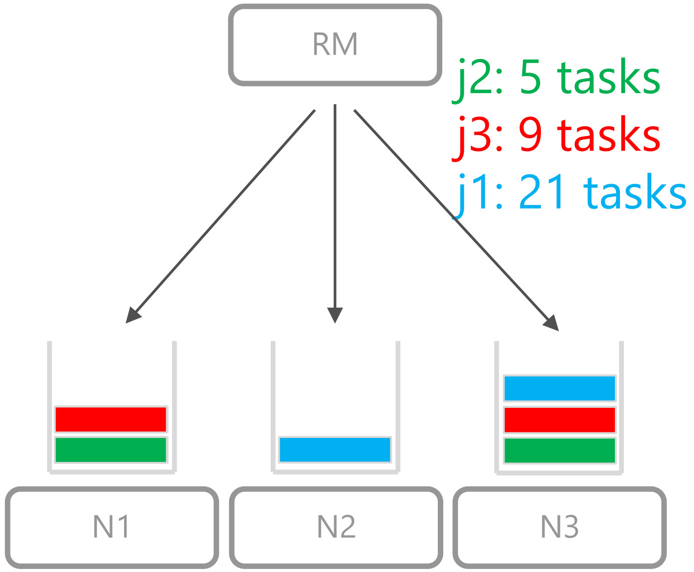
Task Prioritization

- Queue reordering strategies
  - Shortest Remaining Job First (SRJF)
  - Least Remaining Tasks First (LRTF)
  - Shortest Task First (STF)
  - Earliest Job First (EJF)



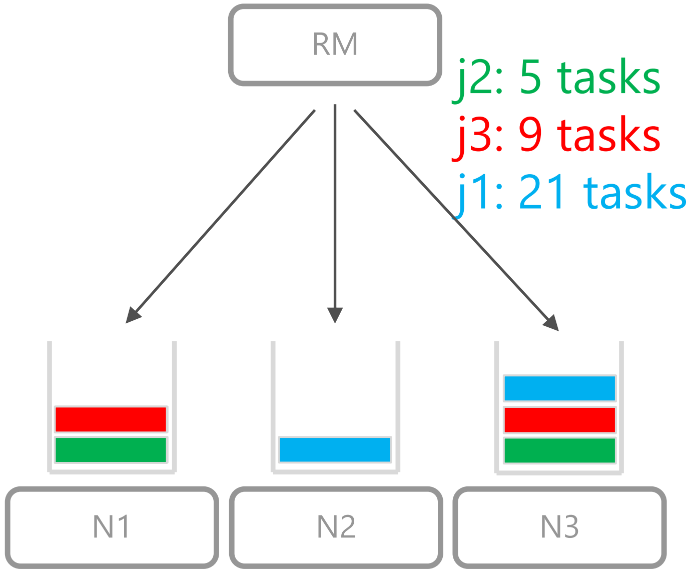
Task Prioritization

- Queue reordering strategies
  - Shortest Remaining Job First (SRJF)
  - Least Remaining Tasks First (LRTF)
  - Shortest Task First (STF)
  - Earliest Job First (EJF)



Task Prioritization

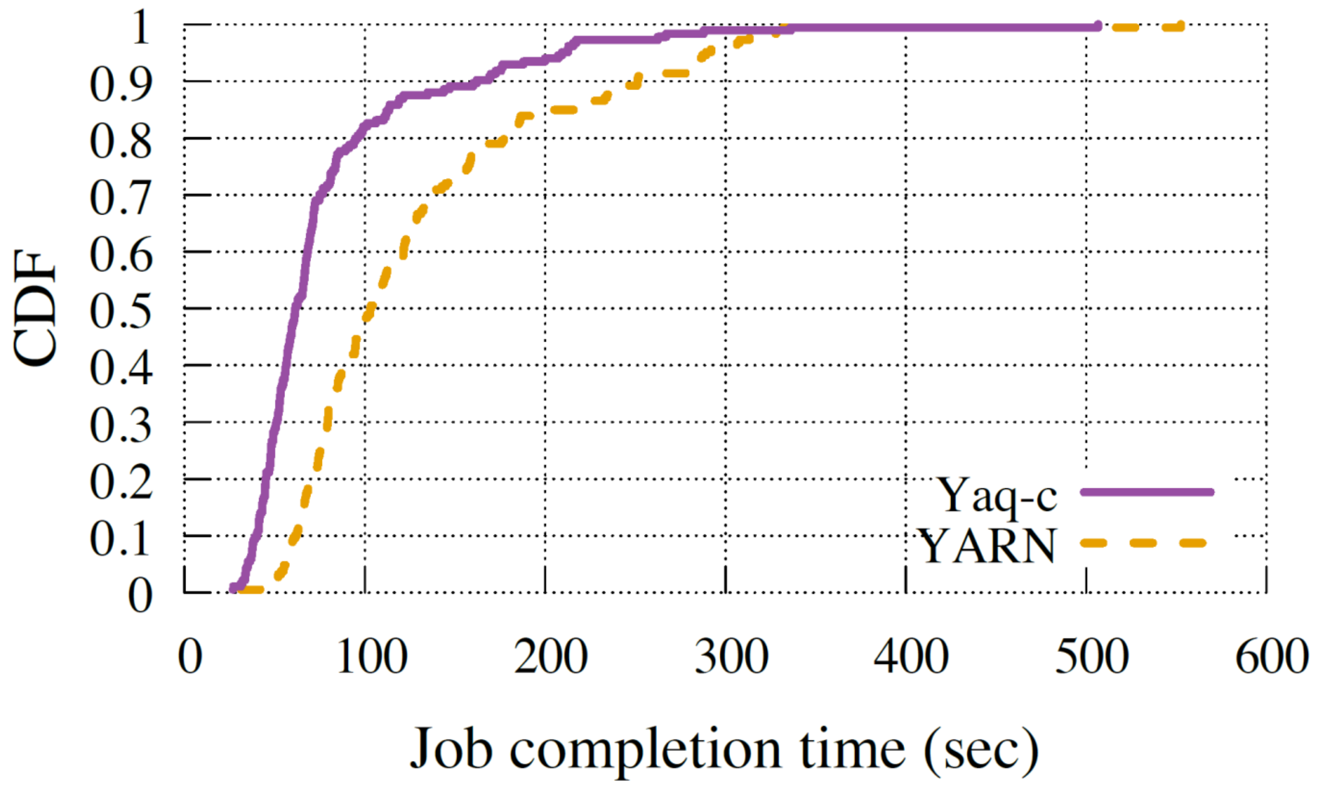
- Queue reordering strategies
  - Shortest Remaining Job First (SRJF)
  - Least Remaining Tasks First (LRTF)
  - Shortest Task First (STF)
  - Earliest Job First (EJF)
- SRJF and LRTF are job-aware
  - Dynamically reorder tasks based on job progress
- Starvation freedom
  - Give priority to tasks waiting more than *X* secs



Bounding Queue Lengths

- Determine max number of tasks at a queue
  - Trade-off between short and long queues
- Short queues
  - Resource idling
    → lower throughput
- Long queues
  - High queuing delays, early binding of tasks to queues
    → longer job completion times
- Static and dynamic queue bounding

Evaluating Yaq



	Task queuing delay (sec)		
	Mean	Stdev	Median
Yaq-c	8.5	21.4	1.1
Yaq-c (unbounded)	65.5	85.1	30.4
Yaq-c (no reorder)	53.2	78.2	25.4
YARN	-	-	-

- Setup
  - 80-node cluster
  - 185 Hive production queries
  - Queue length of 4 slots
  - Queue wait time-based placement
  - SRJF prioritization
- 1.7x improvement in median JCT over YARN
- 1.1 sec median task queuing delay
  - Both bounding and reordering are crucial

More on Mercury/Yaq

- Container types
  - Scheduling and execution
  - When to choose each type
- Support for distributed scheduling of containers
- Apply techniques on any distributed scheduler
  - 9.3x better median job completion over Sparrow-like batch sampling

- Next steps
  - Resource over-commitment
  - Support for multi-tenancy (YARN as a secondary tenant)
  - Pricing models for different container types

Mercury/Yaq: Wrap-up

- Improvement of cluster utilization
  - Queuing of tasks at NMs
  - Container types
- Need for queue management techniques
  - Queue bounding
  - Task placement to queues
  - Prioritization of tasks in queues
- Improvement in median job completion time
  - 1.7x over YARN
  - 9.3x over Sparrow-like batch sampling

Thank you!