Falcon: An Efficient Dependency-Aware Scheduling for High Throughput and Resource Utilization in Clouds

The 2025 IEEE World Forum on Public Safety Technology (WF-PST)
(23 - 25, September 2025, Orlando, Florida, USA)

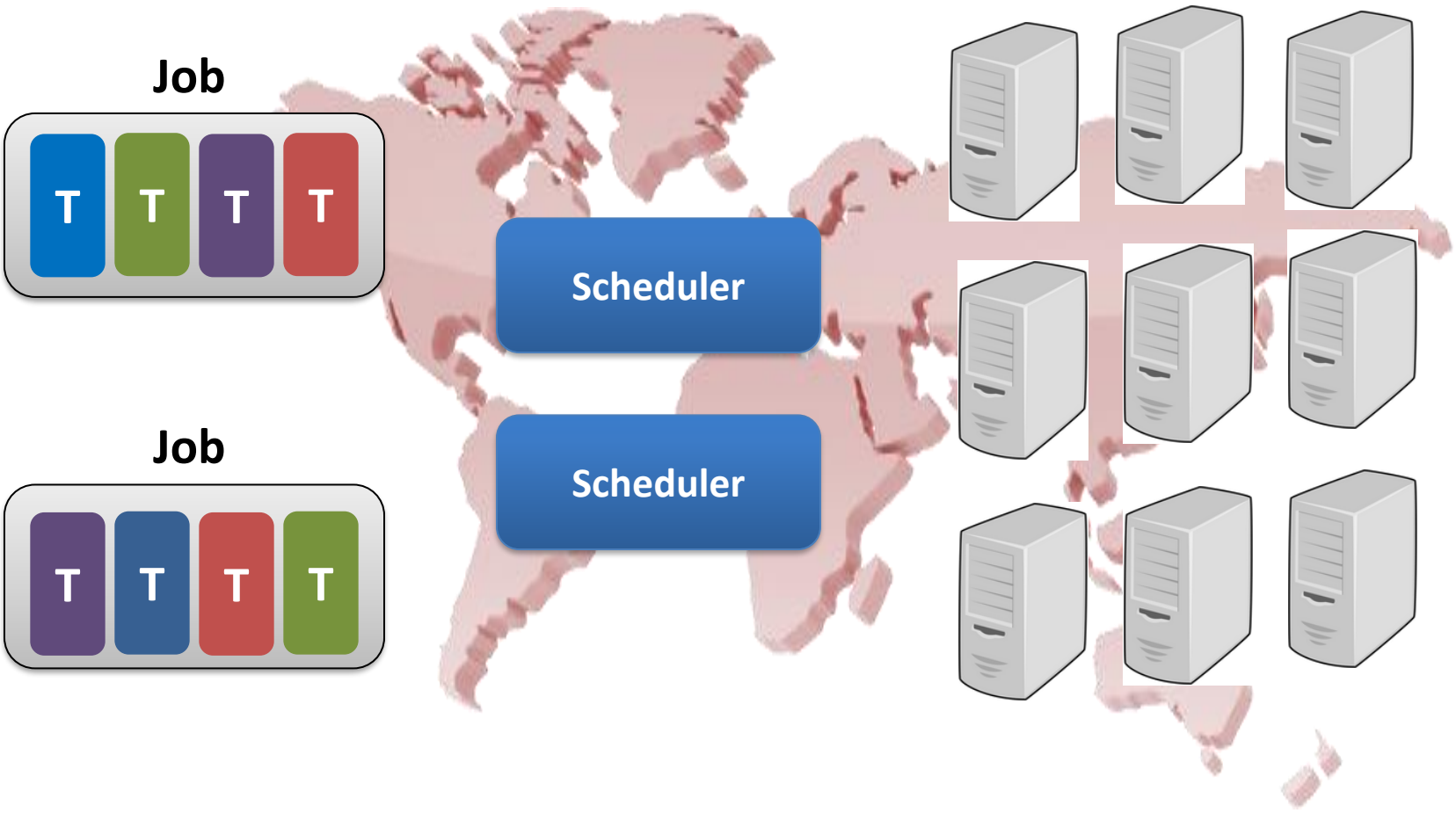
Jinwei Liu*, Rui Gong†, Richard A. Alo*, Wei Dai‡,
Richard A. Long§, Pierre Ngnepieba§

*Dept. of Computer and Information Sciences, Florida A&M University, Tallahassee, FL, USA
†Dept. of Informatics and Mathematics, Mercer University, Macon, GA, USA
‡Dept. of Computer Science, Purdue University Northwest, Hammond, IN, USA
§College of Science and Technology, Florida A&M University, Tallahassee, FL, USA

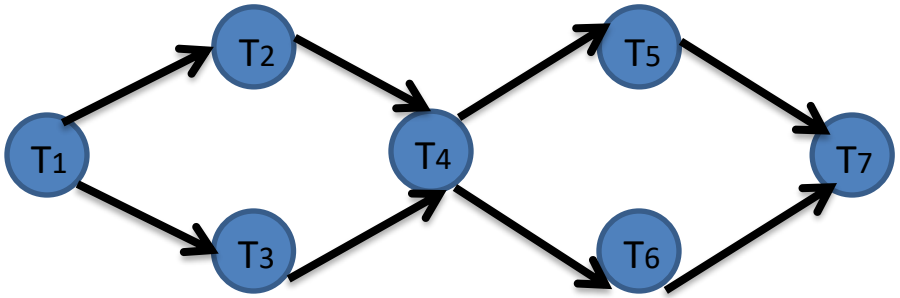
Talk Outline

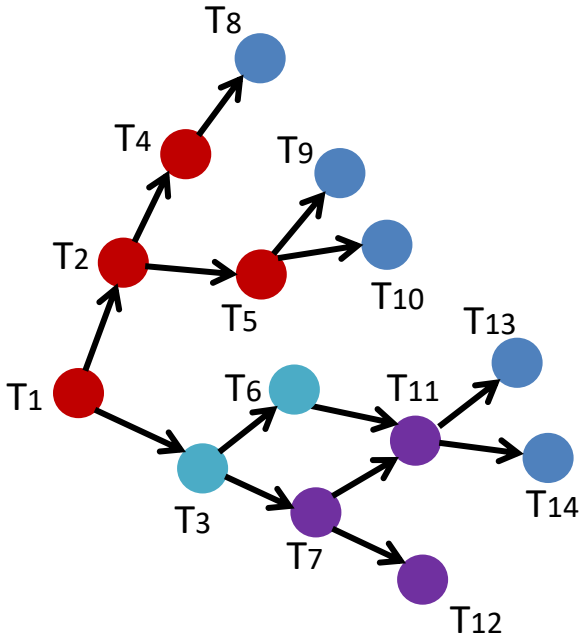
- **Introduction**
- Problem Statement
- Proposed Approach: Falcon
- Design of Falcon
- Performance Evaluation
- Conclusions and Future Work

Introduction



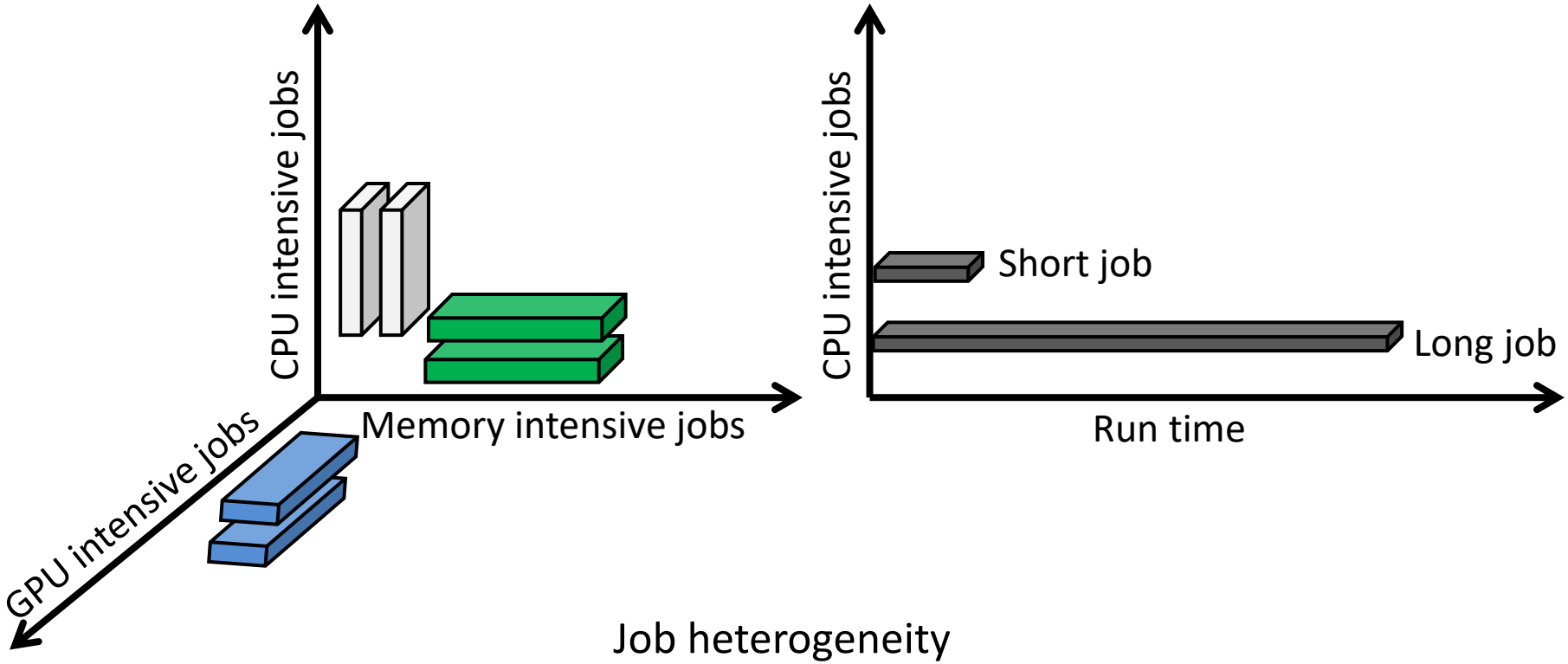
Motivation





Motivation (cont.)

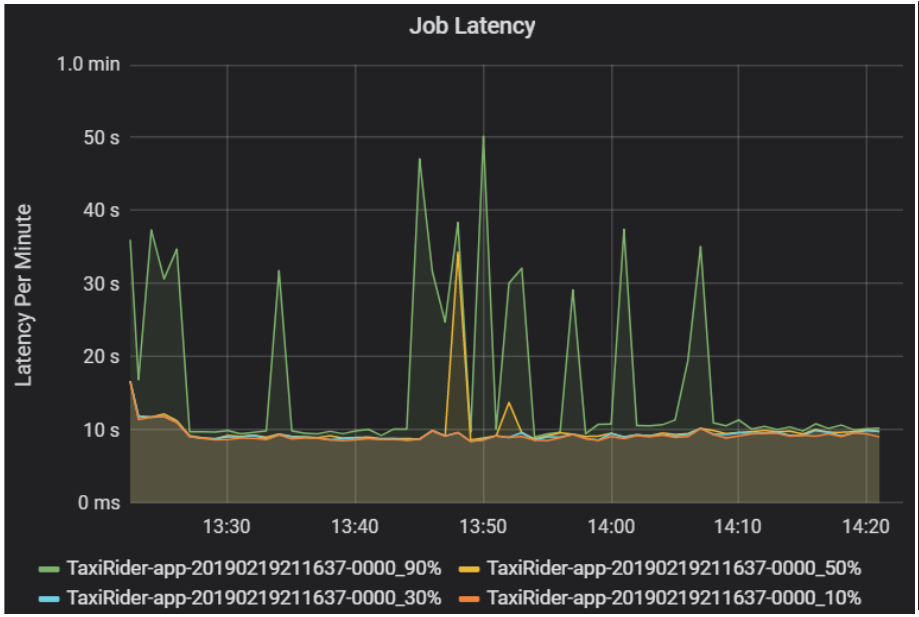
- Job heterogeneity challenges the performance improvement on latency, throughput and resource utilization [1, 2]

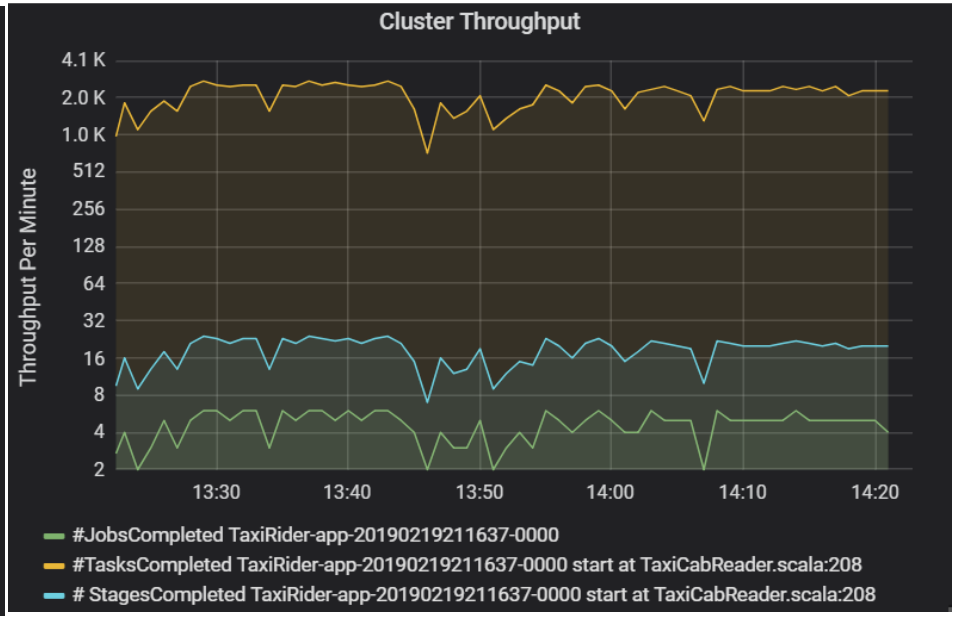


[1] Z. Wang, H. Li, Z. Li, X. Sun, J. Rao, H. Che, and H. Jiang. Pigeon: an effective distributed, hierarchical datacenter job scheduler. In Proc. of *ACM SoCC*, Santa Cruz, 2019.
[2] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding. Mlaas in the wild: Workload analysis and scheduling in large-scale heterogeneous gpu clusters. In *NSDI*, 2022.

Motivation (cont.)

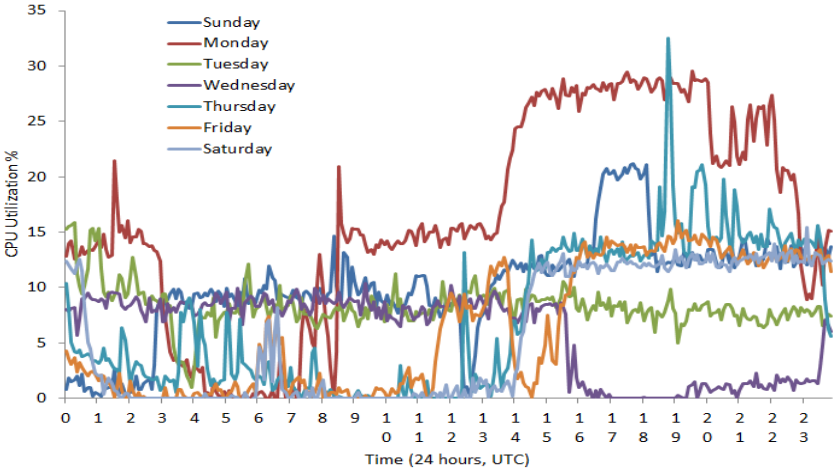
- Performance bottlenecks in Azure Databricks
  - ➢ High job/task latency
  - ➢ Low cluster throughput



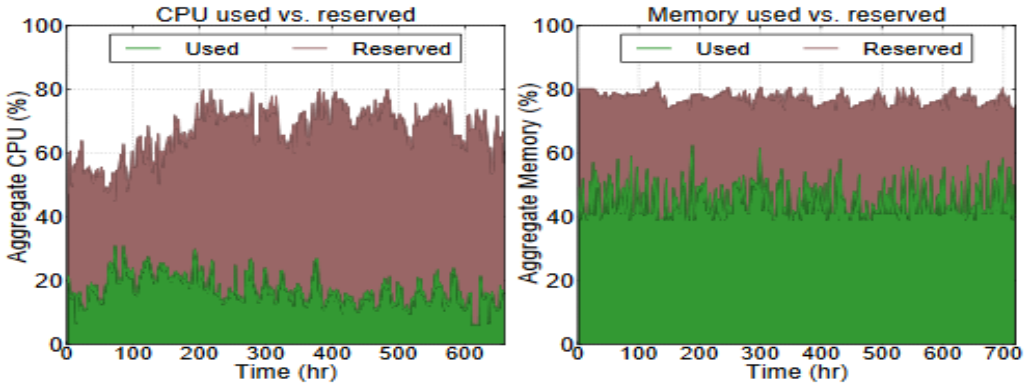


Motivation (cont.)

- Low resource utilization in real systems
  - ➢ Amazon EC2 servers have low resource utilization
  - ➢ Production cluster at Twitter has low resource utilization



Amazon EC2 servers

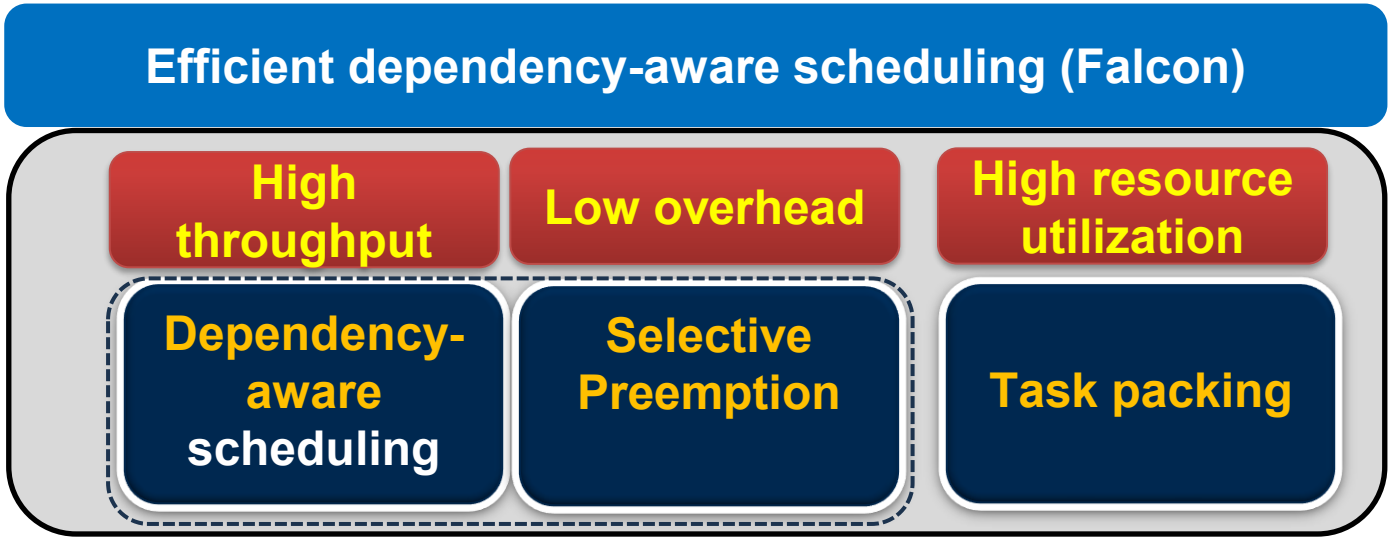


Production cluster at Twitter [3]

[3] C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *Proc. of ASPLOS*, 2014.

This Research

- **Falcon:** an efficient dependency-aware scheduling for high throughput and low overhead in clouds
  - ➢ **Features of Falcon**
    - High throughput
    - High resource utilization
    - Low overhead
    - Dependency awareness



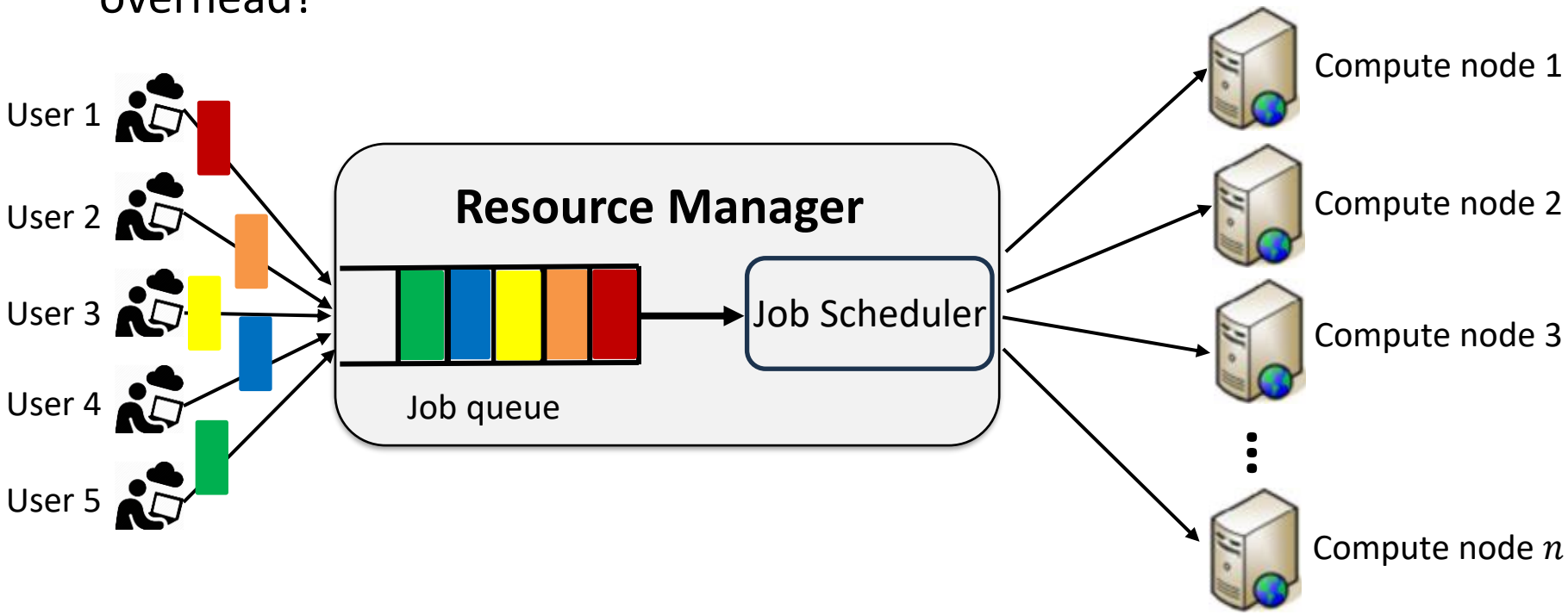
Framework of Falcon

Talk Outline

- Introduction
- **Problem Statement**
- Proposed Approach: Falcon
- Design of Falcon
- Performance Evaluation
- Conclusions and Future Work

Problem Statement

- **Problem statement:** Given a set of jobs consisting of tasks, constraints of jobs and tasks (e.g., task precedence constraints), priorities of jobs with tasks, and a number of heterogeneous worker machines, what is the makespan? Then, how to design an efficient scheduler to schedule these jobs so that the throughput can be maximized while satisfying high priority tasks and reducing time overhead?



Talk Outline

- Introduction
- Problem Statement
- **Proposed Approach: Falcon**
- Design of Falcon
- Performance Evaluation
- Conclusions and Future Work

Proposed Approach: Falcon

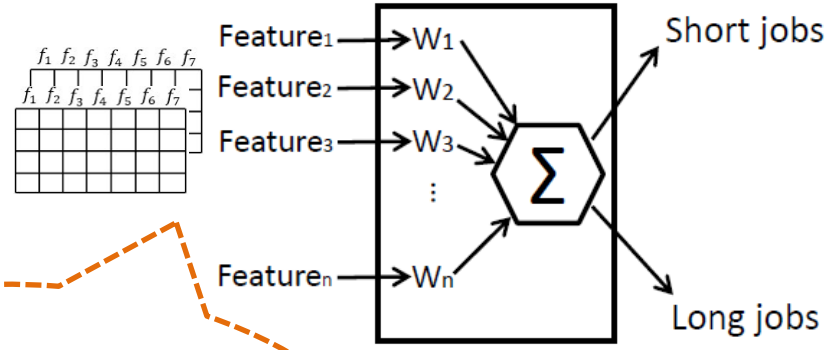
- **Key idea:** Use machine learning algorithm to classify the jobs into short jobs and long jobs based on the extracted features; split the jobs into tasks and distribute the tasks to the master nodes based on task dependency and the load of masters; utilize the dependency information of tasks to determine task priority, and pack complementary tasks; the workers calculate the priority of tasks assigned to them and run tasks based on the Selective Preemption.

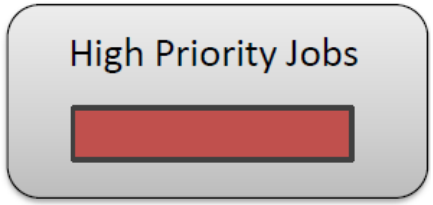
- **Rationale of task packing, job classification and dependency consideration**
  - Task packing can help reduce resource wastage caused by resource fragmentation
  - Jobs can vary significantly in size; different jobs have different resource demands; short jobs and long jobs have different priorities
  - Leveraging task dependency in scheduling can help improve throughput
  - Selectively allowing higher priority tasks to preempt lower priority tasks can satisfy high priority tasks while reducing overhead

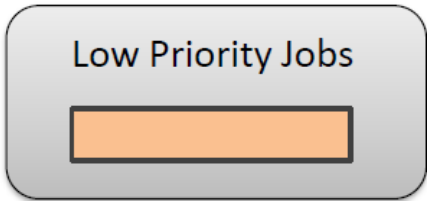
Job Classification

- **Job classification:** classify jobs into two categories based on job's execution time

- Job priority determination: short jobs have higher priorities and long jobs have lower priorities



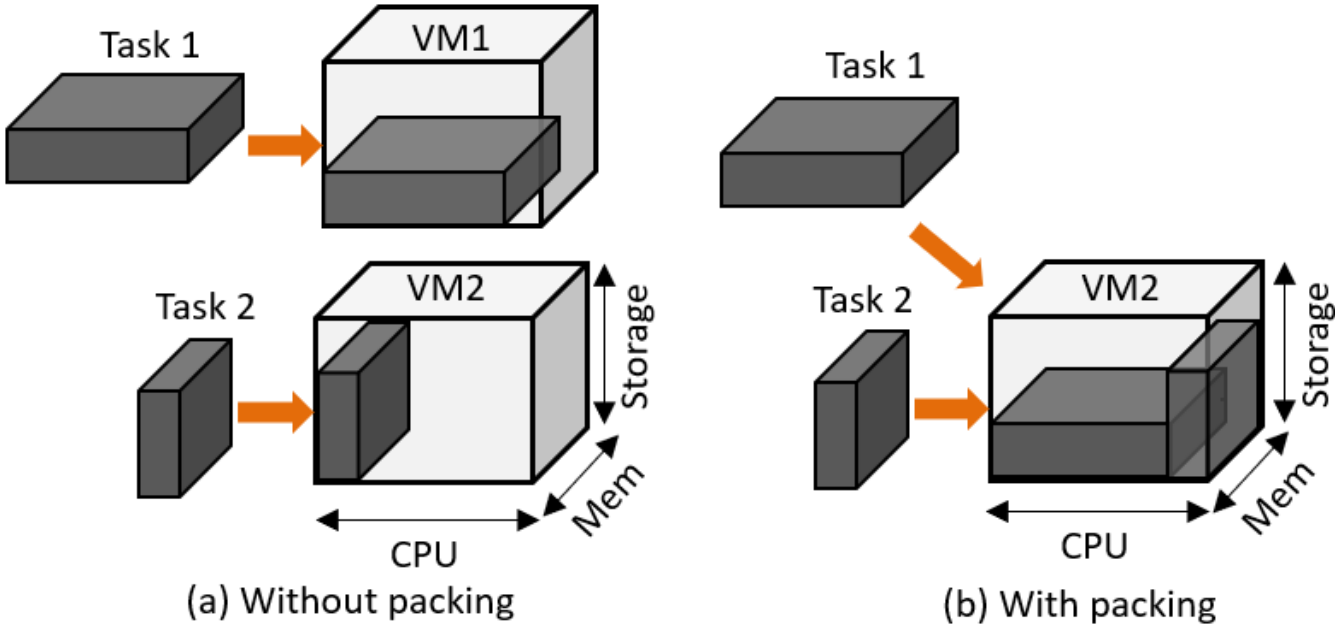




Feature	Description
Job-level features	
Required CPU	Amount of CPU resource required by the job
Required MEM	Amount of MEM resource required by the job
Required storage	Amount of storage resource required by the job
Required GPU	Amount of GPU resource required by the job
# of tasks	Number of tasks which the job contains
System-level features	
CPU utilization	VM's CPU utilization
MEM utilization	VM's MEM utilization
Storage utilization	VM's storage utilization
GPU utilization	VM's GPU utilization

Task Packing

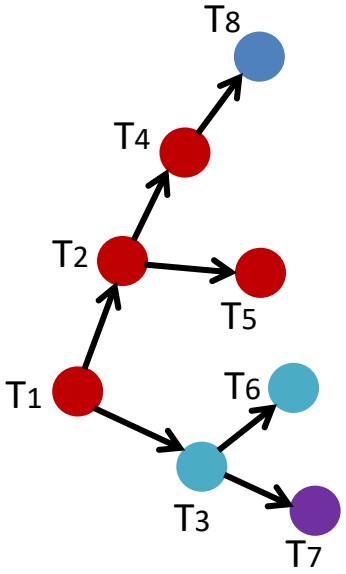
- **Task packing:** pack tasks by leveraging the complementary of tasks' requirements on different resource types and task dependency



$$DV(j,i) = \sum_{h=i}^{l}\left(\left(d_{jh} - \frac{d_{jh}+d_{ih}}{2}\right)^2 + \left(d_{ih} - \frac{d_{jh}+d_{ih}}{2}\right)^2\right) \quad (1)$$

Dependency-aware scheduling

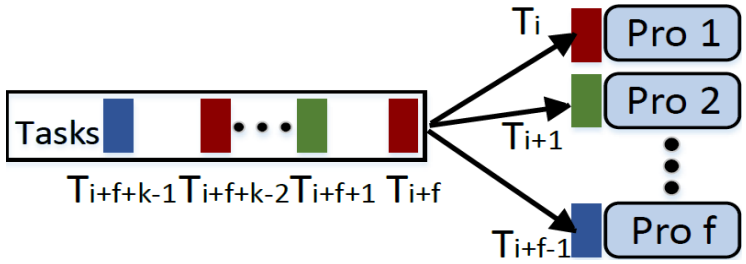
- **Dependency-aware scheduling**
  - **Dependency-aware task priority determination**



  - **Rationale**: Choosing tasks with more dependent tasks to run enables more runnable tasks; more runnable task options enable to select a better task that can more efficiently reduce the latency
  - Priorities assigned by other methods W/o considering dependency
    - $T_1 < T_3 < T_2 < T_4 < T_5 < T_6 < T_7 < T_8$
  - Priorities assigned by Falcon
    - $T_8 < T_7 < T_6 < T_5 < T_4 < T_3 < T_2 < T_1$

Selective Preemption

- **Selective Preemption (SP)**
  - Preemption for multiple tasks running on multiple processors



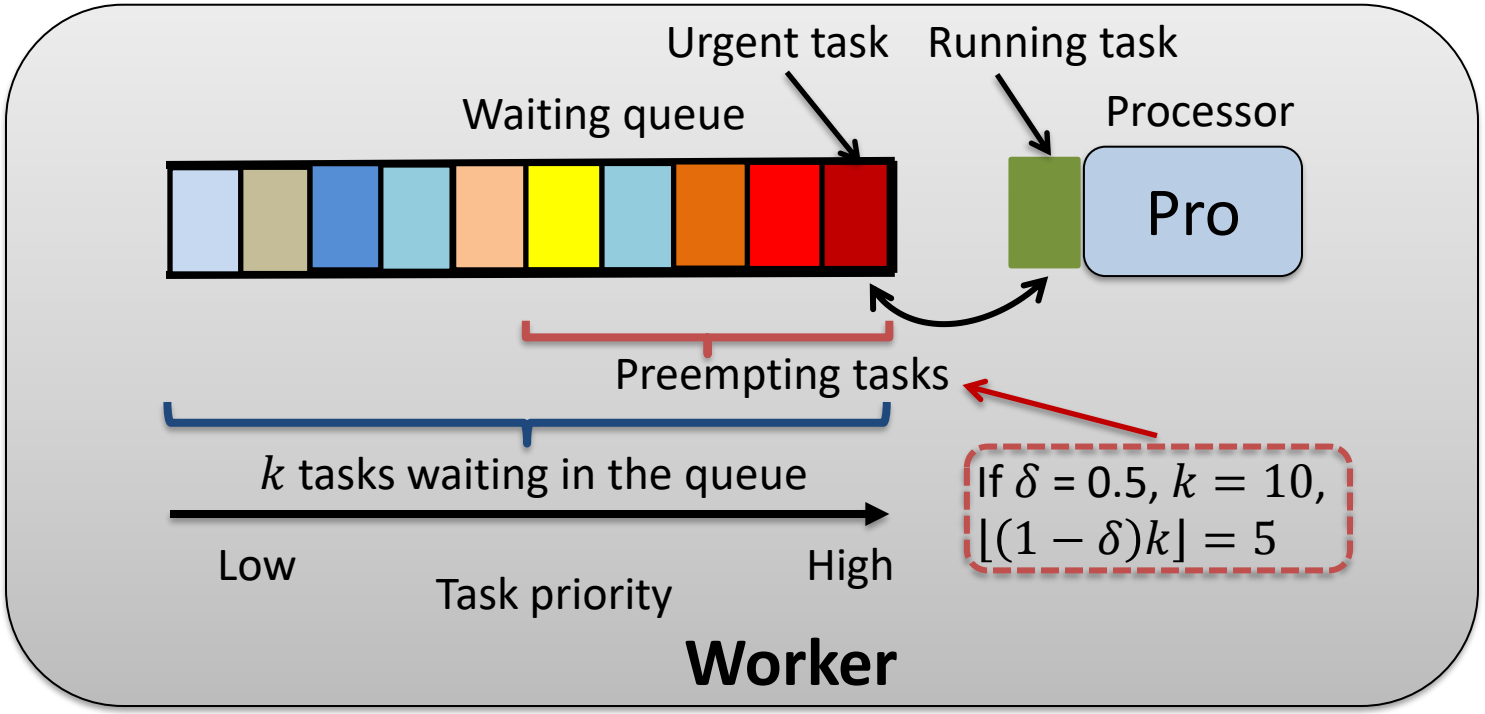
  - Task T_{i+f} can execute only if the following conditions are satisified
    - ✓ $\boldsymbol{C_1}$: the priority of the waiting task T_{i+f} is higher than that of the running task, i.e., $P_{i+f} > P_j$ $(j \in \{i, i+1, \dots, i+f-1\})$
    - ✓ $\boldsymbol{C_2}$: the waiting task T_{i+f} is indepedent of the running task T_j $(j \in \{i, i+1, \dots, i+f-1\})$
    - ✓ $\boldsymbol{C_3}$: the rank of T_{i+f}'s priority among all of the waiting tasks in the queue is top $\lfloor (1-\delta)k \rfloor$, where δ (minimum required ratio between 0 and 1) is related to practical applicaiton
  - SP uses the following formulas to check if $\boldsymbol{C_3}$ is satisfied

$$\frac{\sum_{j=i+f}^{i+f+k-1} I_{i+f,j}}{k} \geq \delta, \qquad (2)$$

$$I_{ij} = \begin{cases} 1, & P_i > P_j \\ 0, & P_j \leq P_j \end{cases} \qquad (3)$$

Selective Preemption (cont.)

- **Selective Preemption**
  - **Selective Preemption:** at most $\lfloor (1-\delta)k \rfloor$ tasks are allowed for preemption



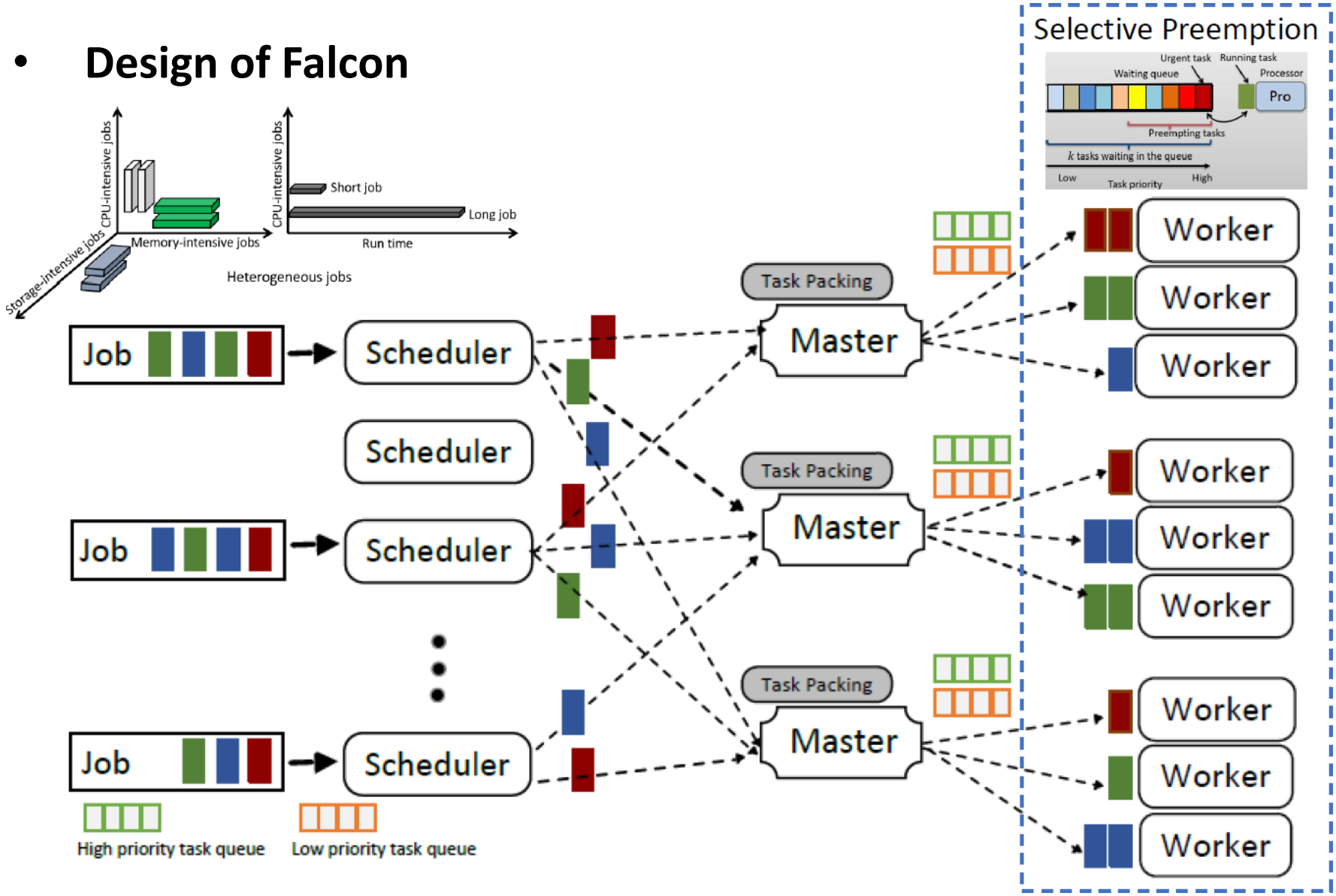
  - **Advantage: Significantly reduce overhead caused by preemption**

Talk Outline

- Introduction
- Problem Statement
- Proposed Approach: Falcon
- **Design of Falcon**
- Performance Evaluation
- Conclusions and Future Work

Design of Falcon

- **Design of Falcon**



Talk Outline

- Introduction
- Problem Statement
- Proposed Approach: Falcon
- Design of Falcon
- **Performance Evaluation**
- Conclusions and Future Work

Performance Evaluation

- **Methods for comparison**
  - **Amoeba [4]**: a system that enables lightweight elasticity in data-intensive computing (DISC) frameworks; it dynamically adapts the resource usage of a job during its execution
  - **Natjam [5]**: a system that supports arbitrary job priorities and preemption for MapReduce clusters
  - **SRPT [6]**: a size-based scheduling for improving the performance of web servers servicing static HTTP requests
  - **SNB [7]**: A decentralized preemptive scheduling algorithm (with backfilling) for multi-core grid resources

[4] G. Ananthanarayanan, C. Douglas, R. Ramakrishnan, S. Rao, and I. Stoica. True elasticity in multi-tenant data-intensive compute clusters. In *Proc. SoCC*, 2012.
[5] B. Cho, M. Rahman, T. Chajed, I. Gupta, C. Abad, N. Roberts, and P. Lin. Natjam: Design and evaluation of eviction policies for supporting priorities and deadlines in mapreduce clusters. In *Proc. SoCC*, 2013.
[6] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal. Size-based scheduling to improve web performance. *ACM Trans. on Computer Systems*, 21(2):207--233, 2003.
[7] A. Balasubramanian, A. Sussman, and N. Sadeh. Decentralized preemptive scheduling across heterogeneous multi-core grid resources. In *Proc. of JSSPP*, 2015.

Trace Data

- **Trace data**

Google Cluster Workload Traces

  - **Google Cluster Trace [8]**
    - The Google Cluster trace records resource usage on a cluster of about 11,000 machines from May 2011 for 29 days. The trace also describes job submission, scheduling decision, task priority, tasks' resource request, and tasks' resource usage. We randomly chose tasks from the jobs in the period between May 1 to May 7.

[8] Google trace. https://code.google.com/p/googleclusterdata/

Experimental Setup

- **Trace-driven experiments on a real cluster & Amazon EC2**
  - **Node deployment**
    - 50 Nodes in the real cluster [9]
    - 30 Nodes in Amazon EC2 [10]

amazon web services™ | EC2

  - **Trace from Google cluster [8]**

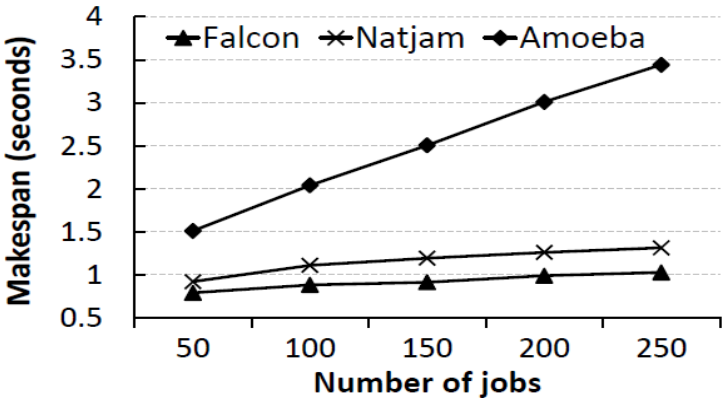
[9] Palmetto cluster. https://docs.rcd.clemson.edu/palmetto/about/.
[10] Amazon EC2. http://aws.amazon.com/ec2.

Experimental Setup (cont.)

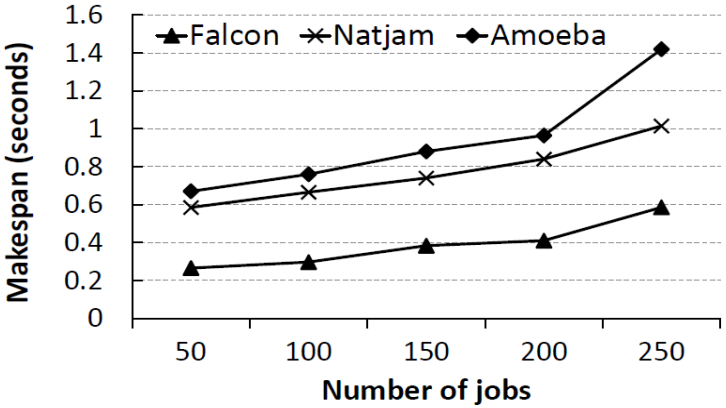
Parameter	Meaning	Setting
n	# of servers	10-50
h	# of jobs	50-1000
n_t	# of tasks of a job	10-20
δ	Minimum required ratio	0.35
ω_1	Weight for task's remaining time	0.5
ω_2	Weight for task's waiting time	0.5

Evaluation of Falcon

- **Makespan**



(a) Makespan of various methods w/ 10 workers on a real cluster

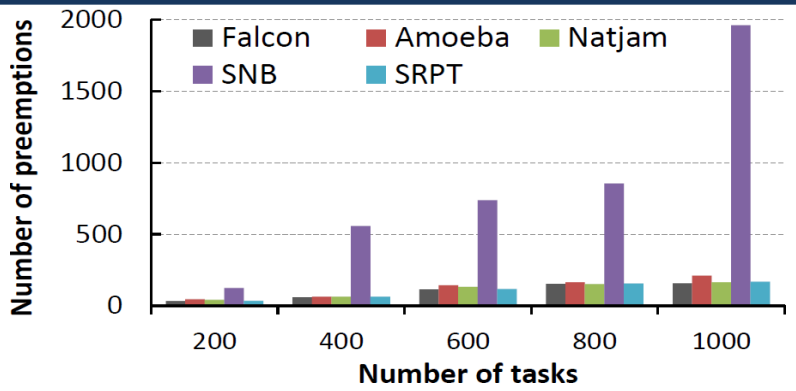


(b) Makespan of various methods w/ 50 workers on a real cluster

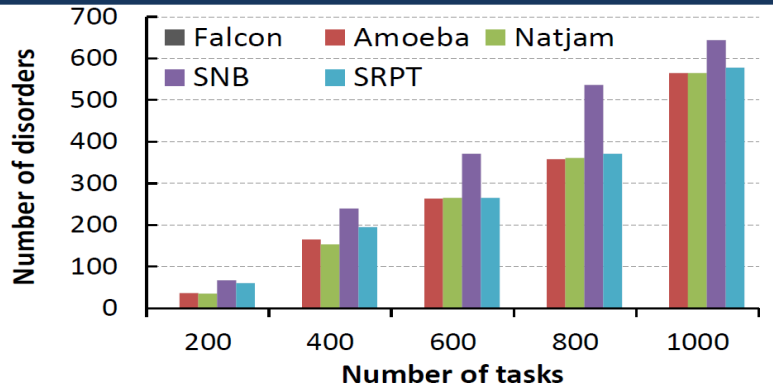
Result: The makespan increases as the number of jobs increases; the makespans of Natjam, Amoeba and Falcon follow Falcon<Natjam<Amoeba

Reason: With a fixed number of workers, the more the jobs submitted, the more time the workers require to finish executing all the jobs; Falcon assigns tasks to workers with the consideration of task dependency; Falcon uses the Selective Preemption to reduce the time overhead caused by preemption

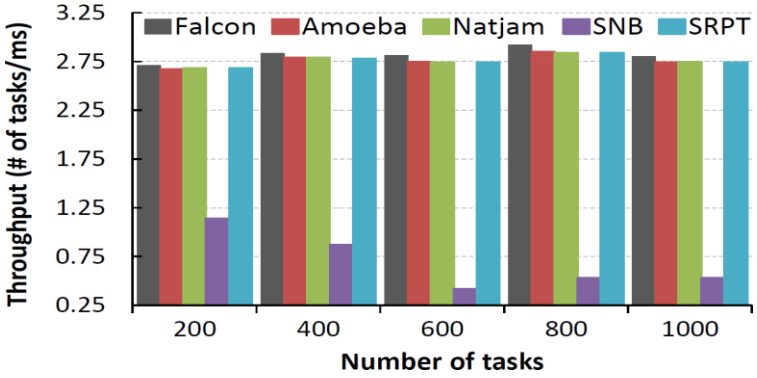
Evaluation of Falcon (cont.)



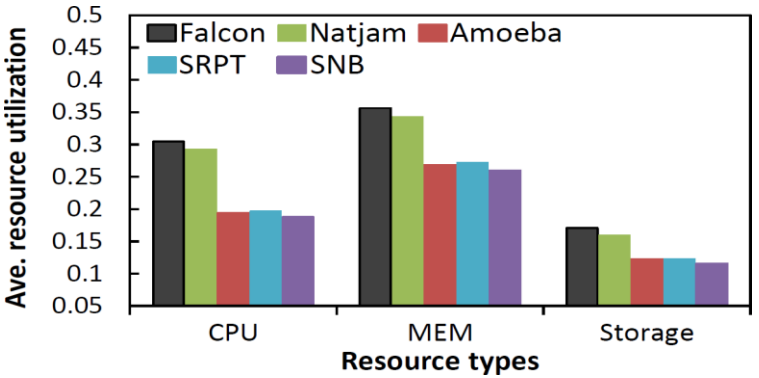
(a) Overhead vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on a real cluster



(b) Number of disorders vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on a real cluster



(c) Throughput vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on a real cluster



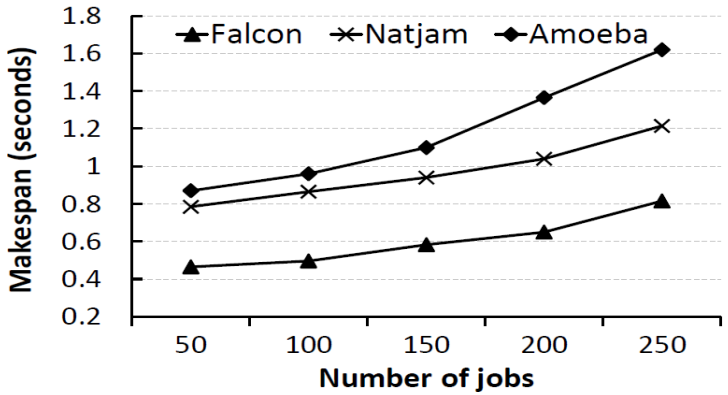
(d) Resource utilization vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on a real cluster

Result: SNB has the highest overhead, and the overhead of Falcon in general is the lowest; the number of disorders follows Falcon < Natjam < Amoeba < SRPT < SNB; the throughput follows SNB < SRPT < Amoeba ≈ Natjam < Falcon; resource utilization in general follows SNB < Amoeba ≈ SRPT < Natjam < Falcon

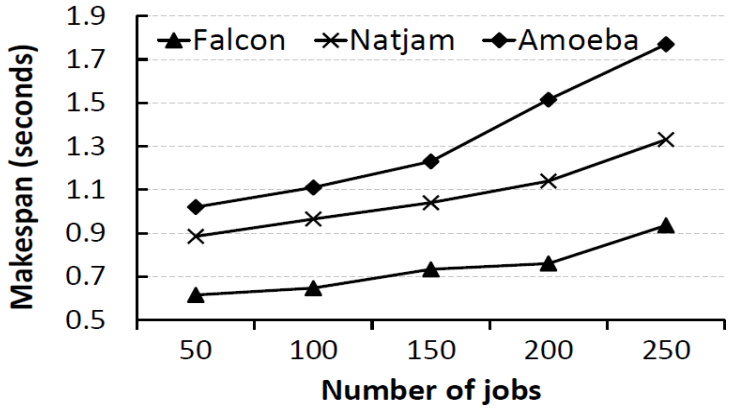
Reason: Falcon uses the SP method; Falcon considers the dependencies in scheduling; Falcon uses a task packing strategy for improving the resource utilization

Evaluation of Falcon (cont.)

- **Makespan**



(a) Makespan of various methods w/ 20 workers on Amazon EC2

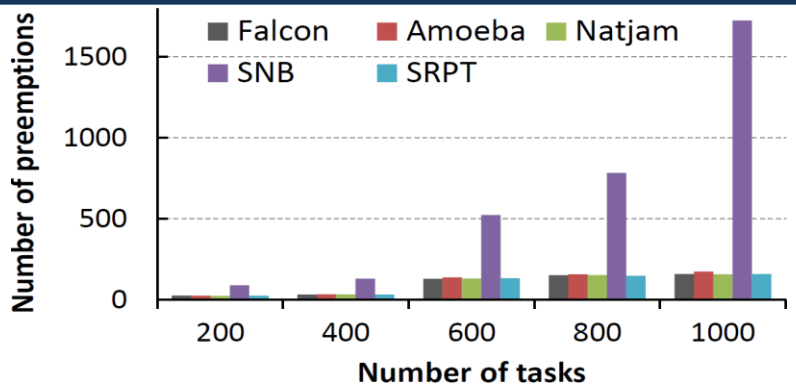


(b) Makespan of various methods w/ 30 workers on Amazon EC2

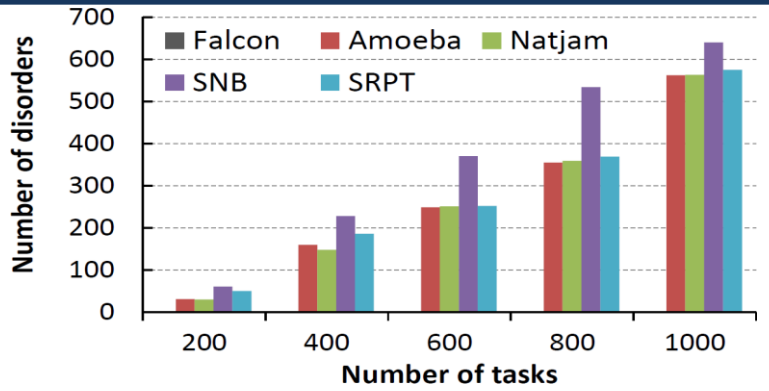
Result: The makespan increases as the number of jobs increases; the makespans of Natjam, Amoeba and Falcon follow Falcon < Natjam < Amoeba

Reason: With a fixed number of workers, the more the jobs submitted, the more time the workers require to finish executing all the jobs; Falcon assigns tasks to workers with the consideration of task dependency; Falcon uses the Selective Preemption to reduce the time overhead caused by preemption

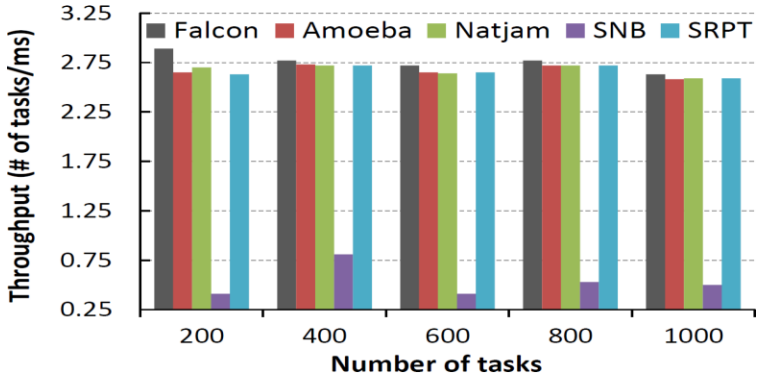
Evaluation of Falcon (cont.)



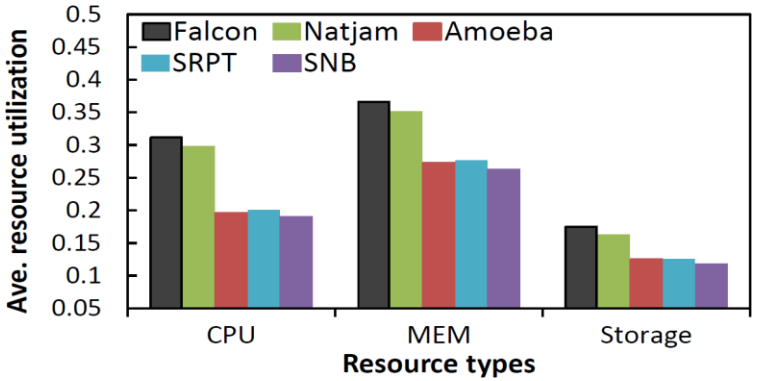
(a) Overhead vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on Amazon EC2



(b) Number of disorders vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on Amazon EC2



(c) Throughput vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on Amazon EC2

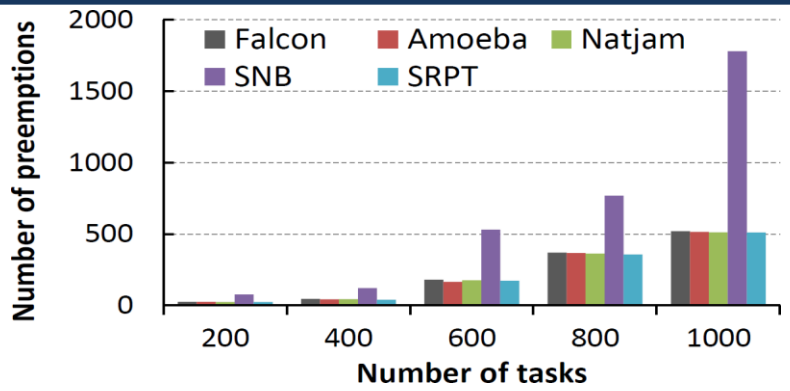


(d) Resource utilization vs. number of tasks w/ job submission rate 5 jobs/sec and 30 workers on Amazon EC2

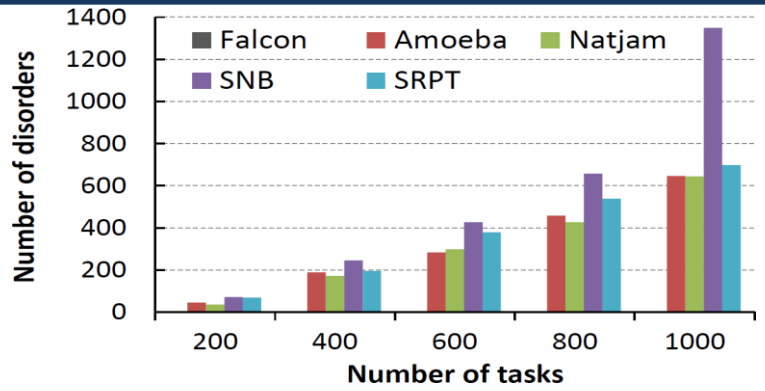
Result: SNB has the highest overhead, and the overhead of Falcon in general is the lowest; the number of disorders follows Falcon < Natjam < Amoeba < SRPT < SNB; the throughput follows SNB < SRPT < Amoeba ≈ Natjam < Falcon; resource utilization in general follows SNB < Amoeba ≈ SRPT < Natjam < Falcon

Reason: Falcon uses the SP method; Falcon considers the dependencies in scheduling; Falcon uses a task packing strategy for improving the resource utilization

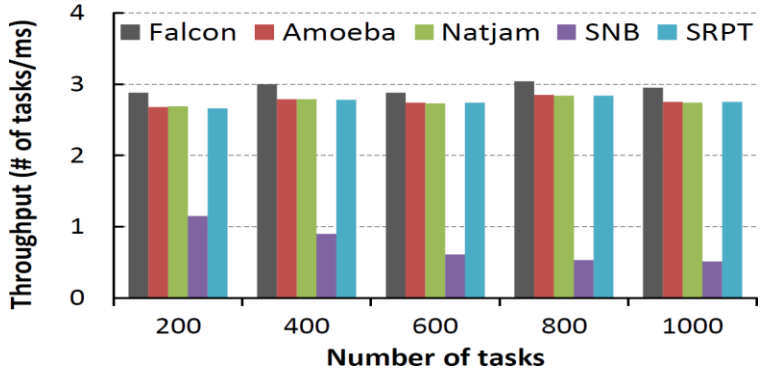
Evaluation of Falcon (cont.)



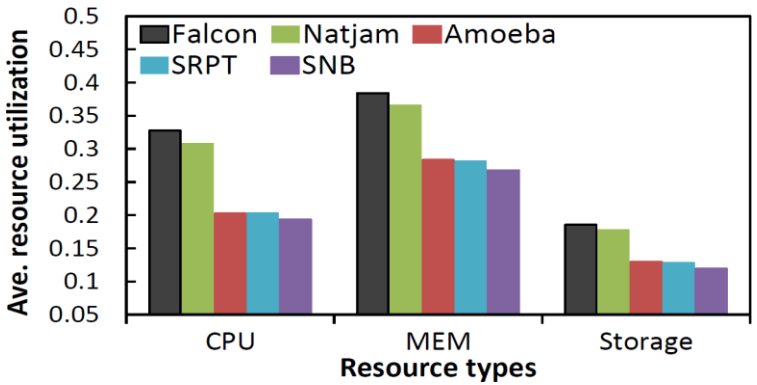
(a) Overhead vs. number of tasks w/ job submission rate 50 jobs/sec and 30 workers on Amazon EC2



(b) Number of disorders vs. number of tasks w/ job submission rate 50 jobs/sec and 30 workers on Amazon EC2



(c) Throughput vs. number of tasks w/ job submission rate 50 jobs/sec and 30 workers on Amazon EC2



(d) Resource utilization vs. number of tasks w/ job submission rate 50 jobs/sec and 30 workers on Amazon EC2

Result: SNB has the highest overhead, and the overhead of Falcon in general is the lowest; the number of disorders follows Falcon < Natjam < Amoeba < SRPT < SNB; the throughput follows SNB < SRPT < Amoeba ≈ Natjam < Falcon; resource utilization in general follows SNB < Amoeba ≈ SRPT < Natjam < Falcon

Talk Outline

- Introduction
- Problem Statement
- Proposed Approach: Falcon
- Design of Falcon
- Performance Evaluation
- **Conclusions and Future Work**

Conclusions and Future Work

- **Our contributions**
  - Propose an efficient dependency-aware scheduling, which can increase the throughput and resource utilization while reducing the overhead in clouds
  - Leverage task dependency to determine task priority and adequately consider task dependency in scheduling to improve throughput
  - Consider the heterogeneity of jobs/tasks, and propose a task packing strategy to reduce the resource fragmentation and improve the resource utilization
  - Propose Selective Preemption (SP), which can satisfy high-priority tasks and reduce the time overhead caused by preemption

- **Future work**
  - Use different cloud/cluster workloads (including machine learning workloads or GPU-based workloads) to fully verify the performance of our method
  - Consider data locality, fairness, tail latency, and resource harvesting
  - Consider fault tolerance and energy efficiency for designing a robust and efficient scheduling system

Thank you!

Questions & Comments?

Jinwei Liu, Assistant Professor

jinwei.liu@famu.edu

Department of Computer and Information Sciences

Florida A&M University