



Dependency-aware and Resource-efficient Scheduling for Heterogeneous Jobs in Clouds

Jinwei Liu* and Haiying Shen†

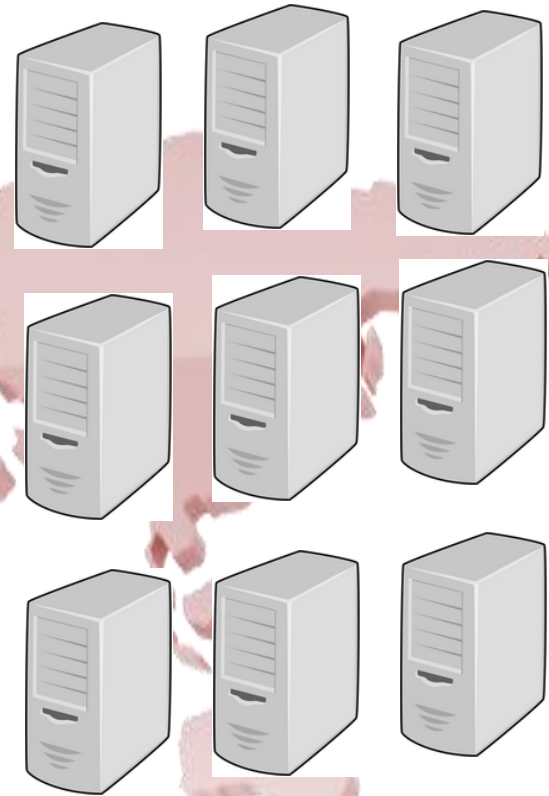
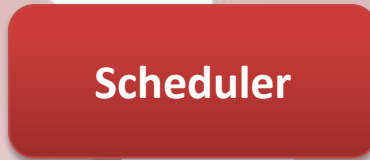
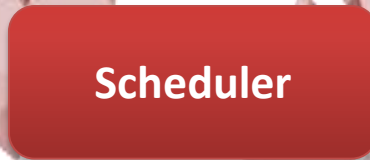
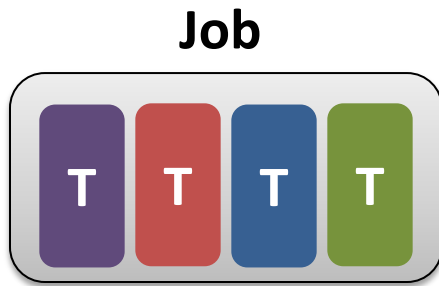
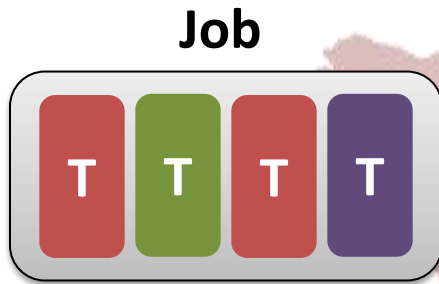
*Dept. of Electrical and Computer Engineering, Clemson University, SC, USA

†Dept. of Computer Science, University of Virginia, Charlottesville, VA, USA

Outline

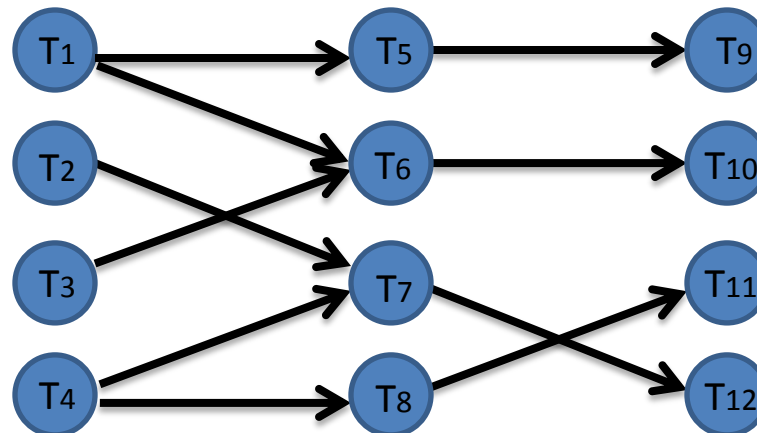
- Introduction
- Problem addressed by Dependency-aware and Resource-efficient Scheduling (DRS)
- Design of DRS
- Performance Evaluation
- Conclusions

Introduction



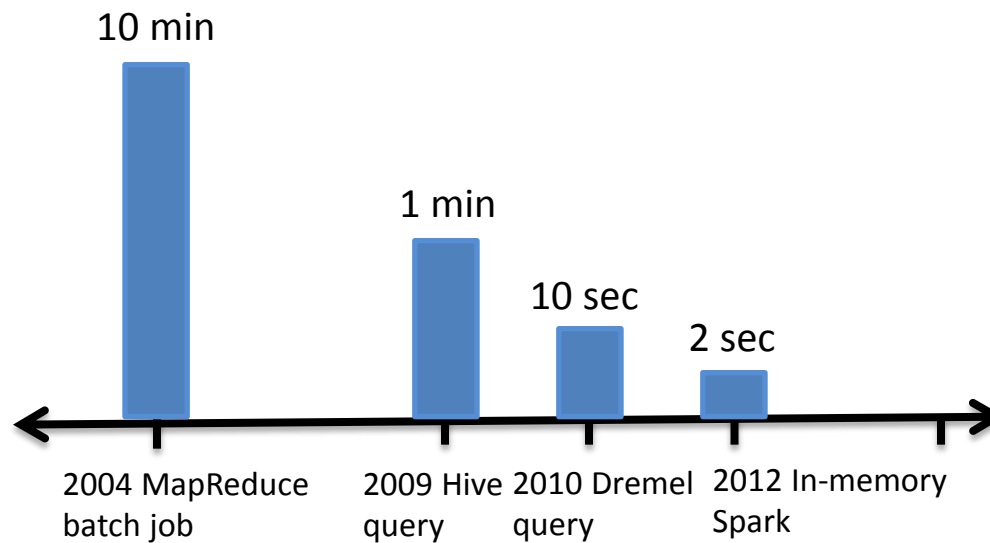
Motivation

- Large-scale data analytics frameworks are shifting towards shorter task durations and larger degrees of parallelism
- Diverse task dependency
 - A task cannot start running until its precedent tasks complete



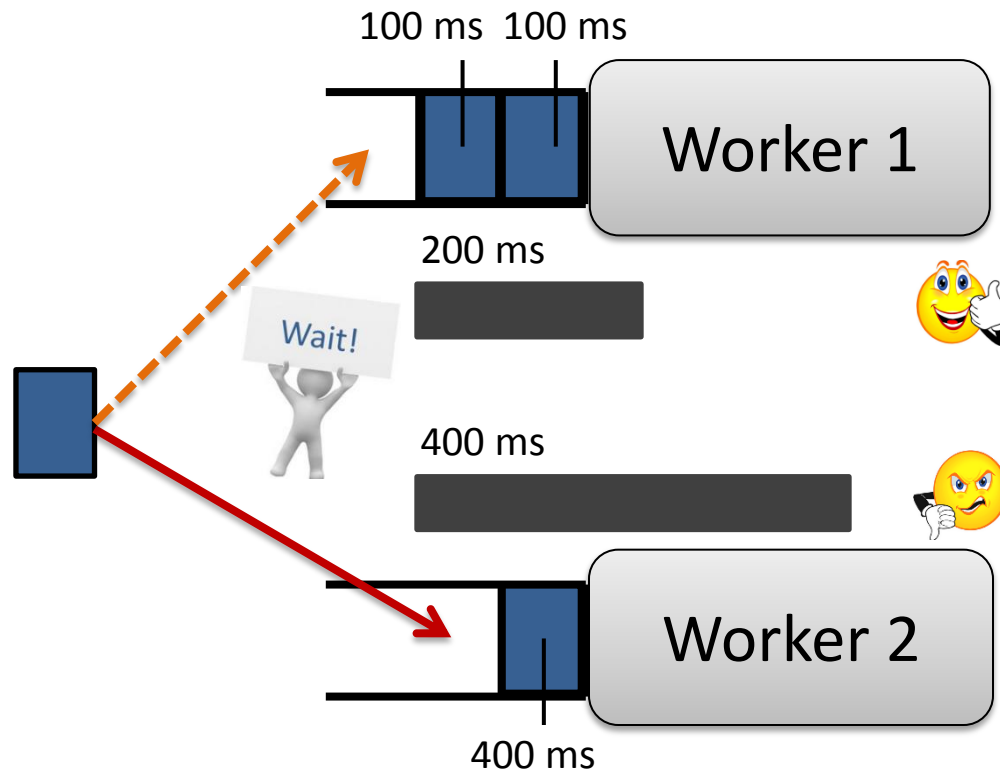
Motivation (cont.)

- High requirements on response time



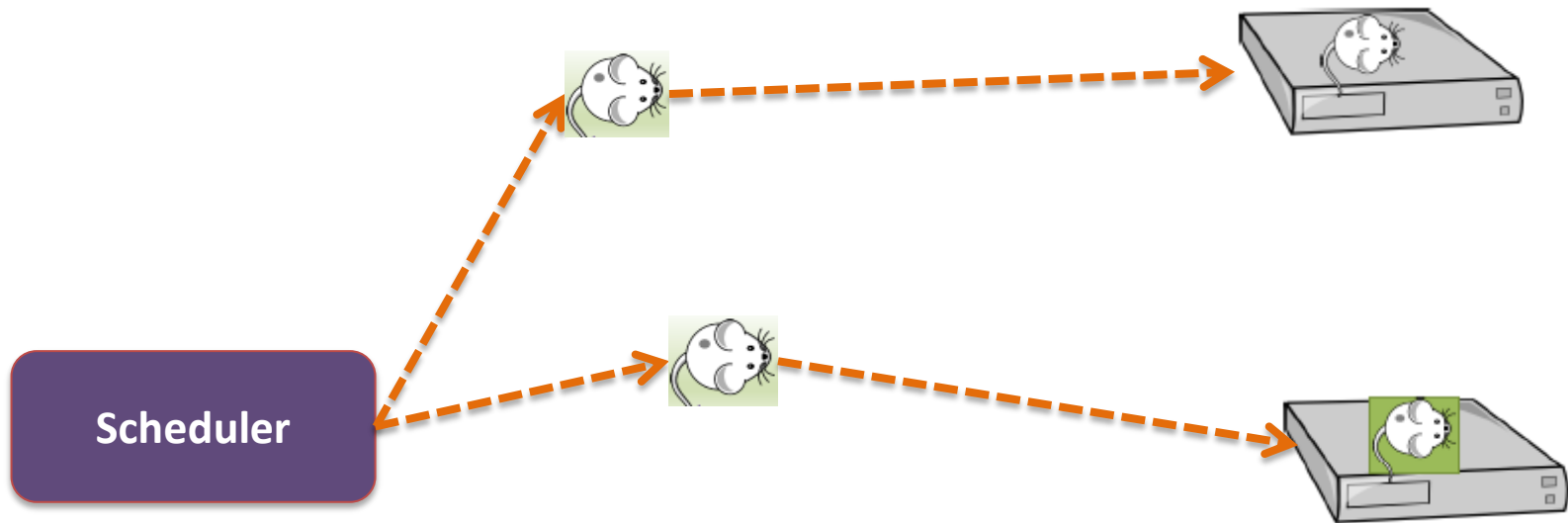
Motivation (cont.)

- Queue length: poor predictor of waiting time



Motivation (cont.)

- Resource inefficiency

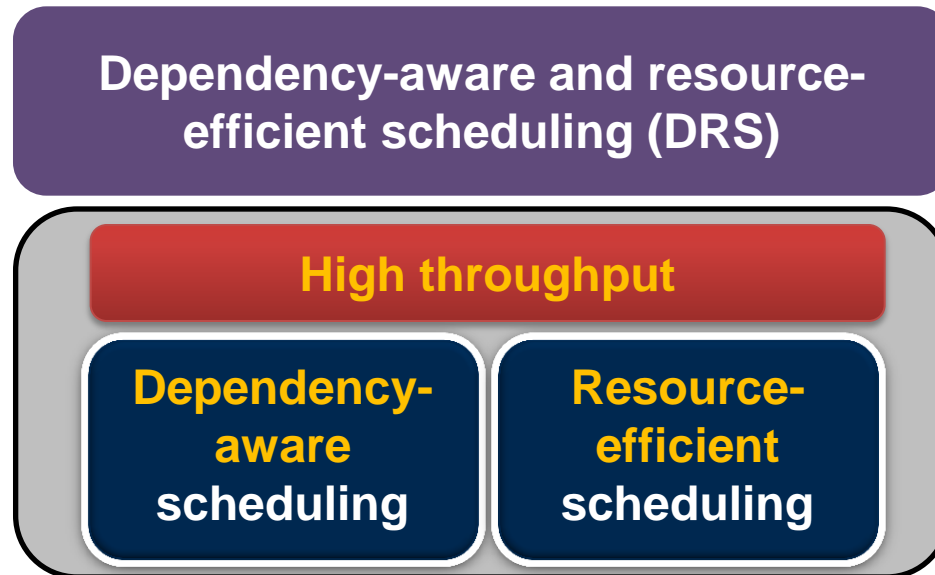


Random reservation (power of two)

- [1] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica. Sparrow: Distributed, low latency scheduling. In *Proc. of SOSP*, 2013
- [2] P. Delgado, F. Dinu, A.-M. Kermarrec, and W. Zwaenepoel. Hawk: Hybrid datacenter scheduling. In *Proc. of ATC*, 2015.


Proposed Solution

- DRS: Dependency-aware and Resource-efficient Scheduling
 - Features of DRS
 - Dependency awareness
 - Resource efficiency
 - High throughput



Framework of DRS

Outline

- Introduction
-  Problem addressed Dependency-aware and Resource-efficient Scheduling (DRS)
- Design of DRS
- Performance Evaluation
- Conclusions

DRS

- System model

- n : Total # of workers
- a : # of jobs in \mathbf{J}
- t_{ij} : The j th task of J_i
- m : # of tasks per job

- \mathbf{J} : A set of jobs
- J_i : The i th job in \mathbf{J}
- S_{ij} : Task t_{ij} 's size
- C_{COS} : Resource cost

- *Waiting time*: The time from when a task is submitted to the worker machine until when the task starts executing
- *Service time*: The time from when a task starts executing until when the task finishes executing on a worker machine
- *Job response time*: The time from when a job is submitted to the scheduler until the tail task of the job finishes executing
- *Throughput*: The total number of jobs completing their executions within the job deadlines per unit of time

DRS

- Problem statement
 - Given a set of jobs consisting of tasks, constraints of tasks (i.e., dependency constraints, resource constraints, response time constraints of jobs) and a set of heterogeneous worker machines, how to schedule these jobs so that the resource cost and the response time can be reduced as much as possible?
- Goal
 - Design an efficient scheduling method for heterogeneous jobs with task dependency constraints for reducing resource cost and response time.

DRS

- Linear programming (ILP) model

$$\min C_{cos} = \sum_{i=1}^a \sum_{j=1}^m \sum_{r=1}^l \sum_{k=1}^n (d_{ij,r} \cdot c_r \cdot t_{ij,k}^s \cdot y_{ij,k})$$

$$s.t. \sum_{i=1}^a \sum_{j=1}^m d_{ij,r} \sum_{k=1}^n t_{ij,k}^s y_{ij,k} - \sum_{i=1}^a \sum_{j=1}^m m_{kr} \sum_{k=1}^n t_{ij,k}^s y_{ij,k} \leq 0 \quad (r \in \{1, \dots, l\}) \quad (4)$$

$$(t_{ij}^b + t_{ij,k}^s) \cdot y_{ij,k} \leq (t_{uv}^b + (1 - x_{ij,uv,k}) \cdot E) \cdot y_{uv,k} \quad (\forall u \in \{1, \dots, a\}, v \in \{1, \dots, m\}) \quad (5)$$

$$t_{ij}^b + \sum_{k=1}^n t_{ij,k}^s \cdot y_{ij,k} \leq t_i^d \quad (t_{ij} \in C_i) \quad (6)$$

$$t_{ij}^b + \sum_{k=1}^n t_{ij,k}^s \cdot y_{ij,k} \leq t_{iq}^b \quad (q \in \{1, \dots, m\}) \quad (7)$$

$$x_{ij,uv,k} + x_{uv,ij,k} = 1 \quad (9)$$

$$x_{ij,uv,k} \in \{0, 1\} \quad (10)$$

$$y_{ij,k} \in \{0, 1\} \quad (10)$$

$$\sum_{k=1}^n y_{ij,k} = 1 \quad (11)$$

- ILP model
 - Constraint (4): ensures that the cumulative resource usage of R_r on a worker machine does not exceed the capacity of this resource type during the period of task execution
 - Constraint (5): ensures the execution order of tasks on a worker
 - Constraint (6): ensures jobs can complete within their specified deadlines
 - Constraint (7): ensures the dependency relation between tasks
 - Constraint (11): ensures that a task can be assigned to only one worker machine
- CPLEX linear program solver

Challenges

- Challenges of DRS design
 - How to schedule tasks with constraints (e.g., dependency) to achieve low response time and high resource utilization
 - How to accurately estimate tasks' waiting time in the queues of workers
 - How to reduce the communication overhead in scheduling

Outline

- Introduction
- Problem addressed by Dependency-aware and Resource-efficient Scheduling (DRS)
- Design of DRS
- Performance Evaluation
- Conclusions

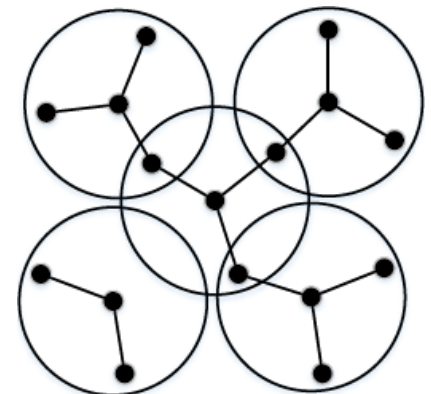


Design of DRS

- Reduce cost
 - Utilize the ILP model to reduce resource cost
- Reduce response time
 - Leverage task dependency information and schedule tasks that are independent of each other to different workers or different processors of a worker so that independent tasks can run in parallel
 - Use the reinforcement learning-based approach to estimate tasks' waiting time for scheduling
 - Utilize scheduler domains and Gossip protocol to reduce the communication overhead

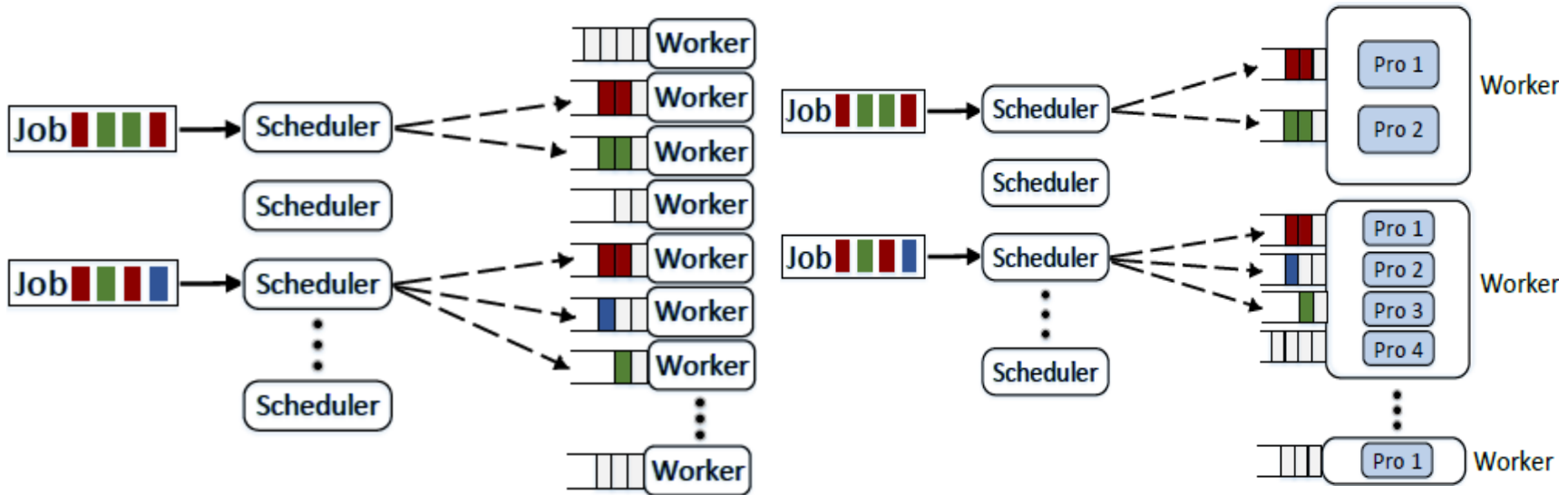
Reduce Communication Overhead

- Schedulers are scattered in a distributed system
- When users submit their jobs, the jobs are delivered to the schedulers near them
- If the scheduler is heavily loaded, the job will be delivered to the lightly loaded neighbor of the heavily loaded scheduler to achieve load balance
- Scheduler Manager Communication
 - Split the schedulers into different sets, each set is a scheduler domain
 - A scheduler manager exists in each scheduler domain
 - Scheduler manager communicate with each other instead of schedulers, reducing communication overhead



Reduce Response Time

- Running independent tasks in parallel



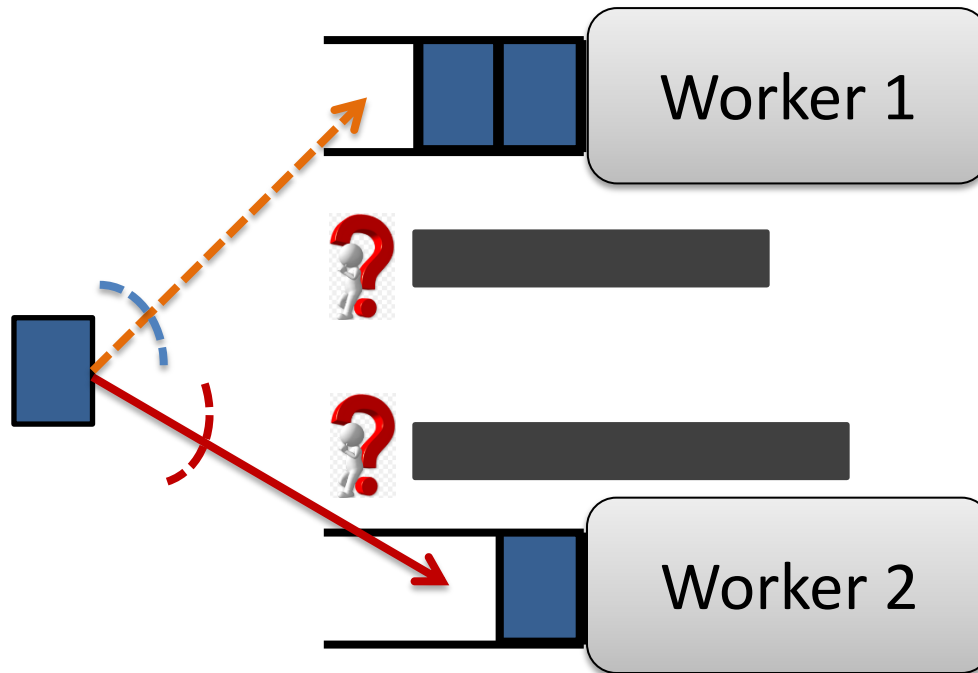
Running tasks in parallel on multiple single processor workers

Running tasks in parallel on multiprocessor workers

- Job classification: CPU intensive, memory intensive, GPU intensive, etc.

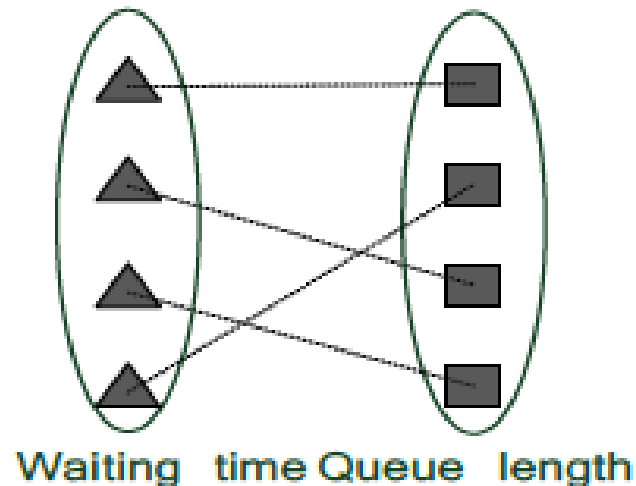
Reduce tasks' waiting time

- Predict tasks waiting time



Predicting Tasks' duration

- Reinforcement learning-based approach
 - DRS uses the mutual reinforcement learning-based approach to accurately estimate tasks' duration in worker's queues, and utilizes it for task scheduling



Outline

- Introduction
- Dependency-aware and Resource-efficient Scheduling (DRS)
- Design of DRS
- Performance Evaluation
- Conclusions



Performance Evaluation

- Methods for comparison

- CLR [1]: implements opportunistic allocation of spare resources to jobs, and it chooses the tasks consuming the most resources to be preempted.
[1] G. Ananthanarayanan, C. Douglas, R. Ramakrishnan, S. Rao, and I. Stoica. True elasticity in multi-tenant data-intensive compute clusters. In *Proc. SoCC*, 2012.
- Natjam [2]: assigns higher priority to production jobs and lower priority to research jobs in scheduling. It uses production jobs to preempt research jobs. Natjam first preempts the tasks of the job with the maximum deadline (which have the lowest priority). For tasks with the same job priority, Natjam uses two task preemption policies: Shortest Remaining Time (SRT) and Longest Remaining Time (LRT).
[2] 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.
- SNB [3]: gives preference to requests for small size files or requests with short remaining file size. Specifically, it uses the linear combination of waiting time and the remaining time for a job (i.e., estimated time for completing the remaining part of the job) to determine the priority of a job.
[3] A. Balasubramanian, A. Sussman, and N. Sadeh. Decentralized preemptive scheduling across heterogeneous multi-core grid resources. In *Proc. of JSSPP*, 2015.
- SRPT [4]: gives preference to those requests which are short, or have small remaining processing requirements, in accordance with the SRPT (Shortest Remaining Processing Time) scheduling policy.
[4] 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.

Experiment Setup

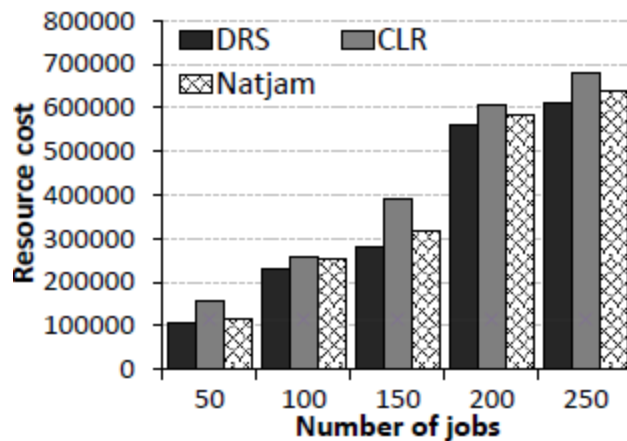
- Parameter settings

Parameter	Meaning	Setting
N	# of servers	50/30
a	# of jobs	50-1000
m	# of tasks / job	10-20
l	# of resource types	2
α	Weight for CPU size	0.5
β	Weight for memory size	0.5
γ	Weight for waiting time	0.5
θ	Weight for queue length	0.5

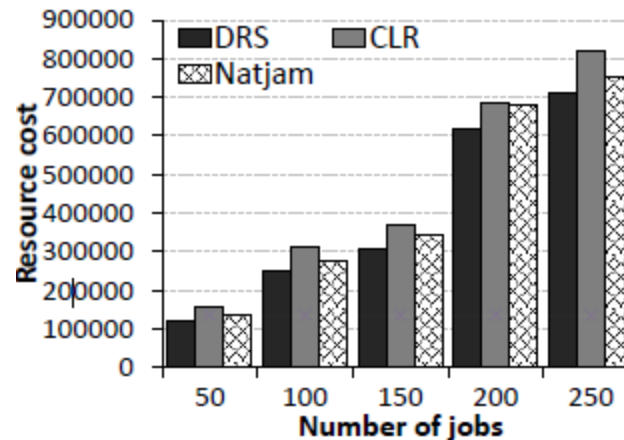
- Palmetto cluster in Clemson and Amazon EC2

Evaluation (cont.)

- Resource cost on the cluster



(a) 15 tasks / job

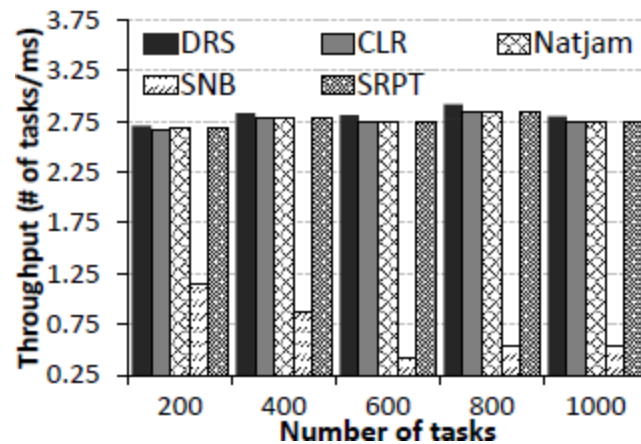


(b) 20 tasks / job

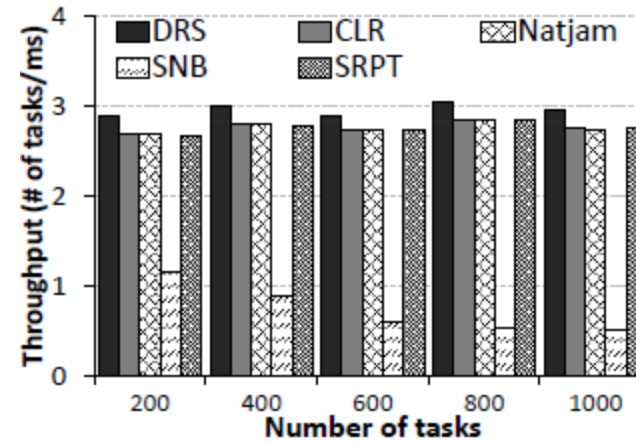
Result: Resource cost increases as the number of jobs increases; resource cost follows $DRS < Natjam < CLR$

Evaluation (cont.)

- Throughput on the cluster



(a) 5 jobs / second

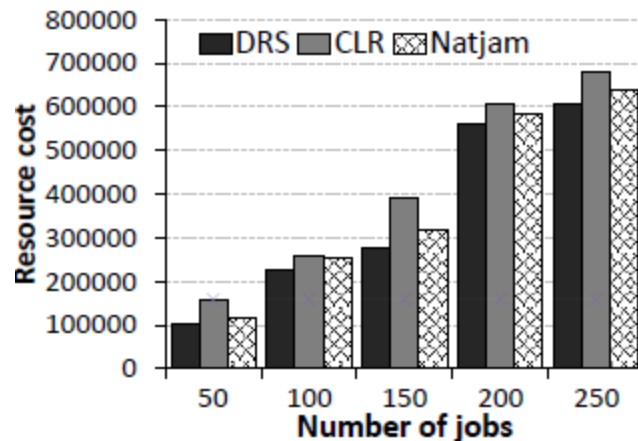


(b) 50 jobs / second

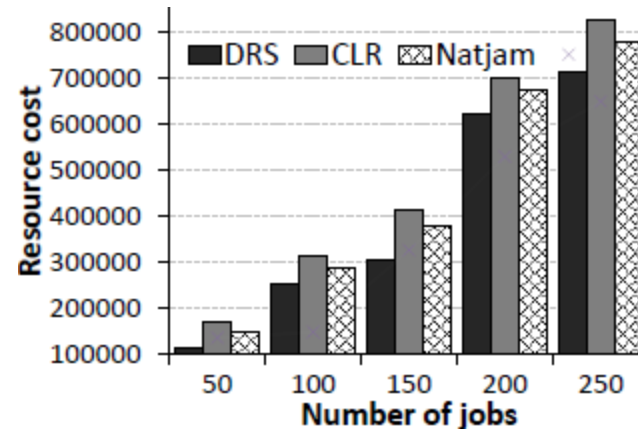
Result: Throughput follows $SNB < SRPT < CLR \approx Natjam < DRS$

Evaluation (cont.)

- Resource cost on Amazon EC2



(a) 15 tasks / job

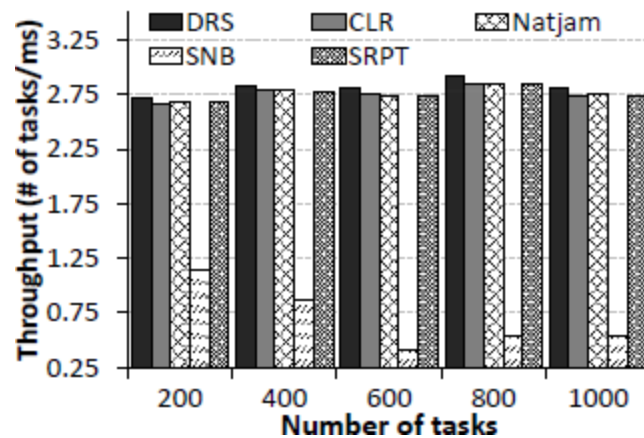


(b) 20 tasks / job

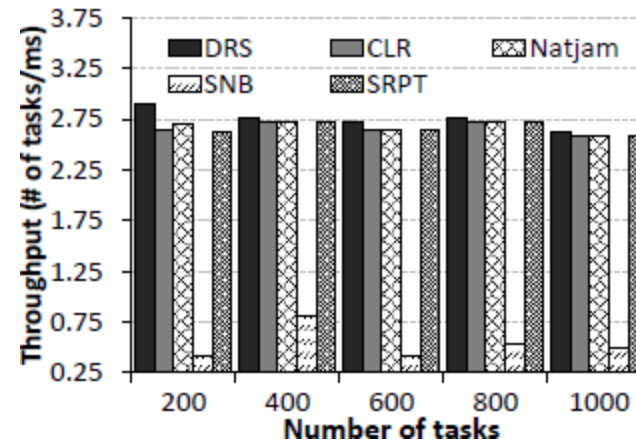
Result: Resource cost increases as the number of jobs increases; resource cost follows $DRS < Natjam < CLR$

Evaluation (cont.)

- Throughput on Amazon EC2



(a) 5 jobs / second



(b) 50 jobs / second

Result: Throughput follows SNB < SRPT < Natjam < CLR < DRS

Outline

- Introduction
- Problem addressed Dependency-aware and Resource-efficient Scheduling (DRS)
- Design of DRS
- Performance Evaluation
- Conclusions



Conclusions

- Our contributions
 - Build a linear programming model to minimize the resource cost and increase the resource utilization
 - Consider task dependency for task assignment, utilize scheduler domains and Gossip protocol to reduce the communication overhead
 - Present a reinforcement learning-based approach to estimate tasks' waiting time in the queue of workers, and then assign tasks to workers to reduce the response time
 - Conduct extensive trace-driven experiments and show the performance of DRS
- Future work
 - Consider preemption
 - Consider the tolerance of failures

Thank you!
Questions & Comments?



Dr. Haiying Shen

hs6ms@Virginia.edu

Associate Professor

Pervasive Communication Laboratory

University of Virginia