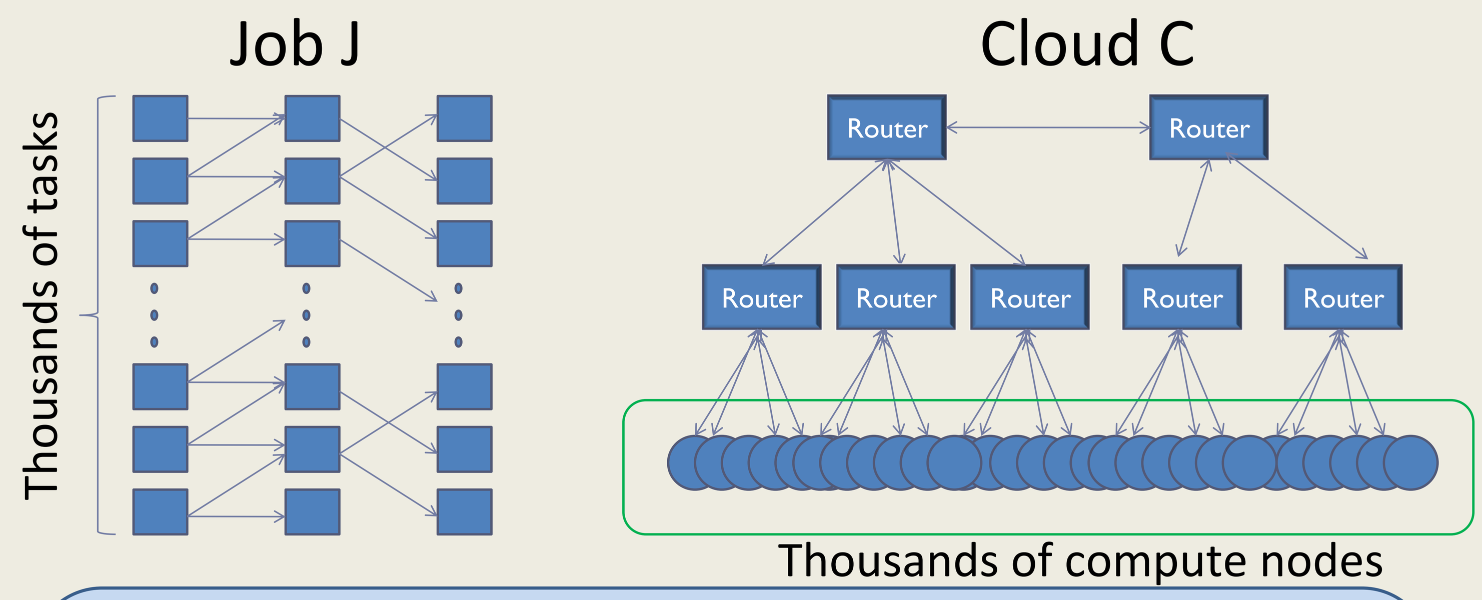
Scheduling Large Jobs by Abstraction Refinement

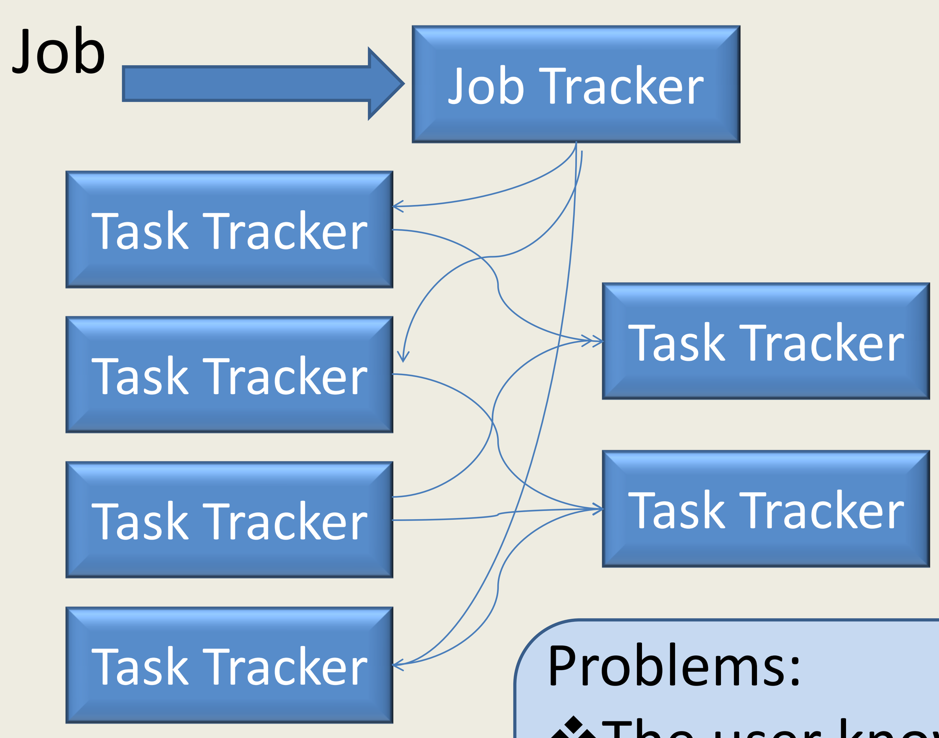
The Cloud Scheduling Problem



Scheduling a job J on a cloud C:

- ❖Every task has to be scheduled on a compute node
- ❖A good solution shall efficiently use the cloud resources and finish the job in reasonable time

Existing Dynamic Scheduling based Solutions (e.g. Hadoop)



A compute node is assigned as the JobTracker, and all other compute nodes as TaskTrackers. The client program sends the job to the JobTracker. Several TaskTrackers are registered at the JobTracker. The JobTracker sends the tasks to the TaskTrackers for processing and checks progress

Problems:

- ❖The user knows the amount to pay only after job completion
- ❖There is a large communication overhead involved in dynamic scheduling.

Our Proposal: Static Scheduling in Clouds

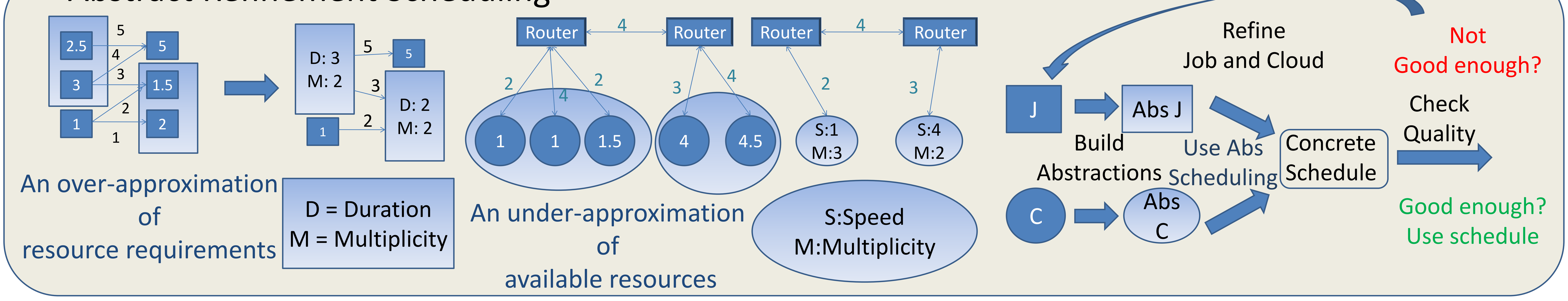
Core idea:

- ❖We assume that we know an estimate of the maximum time required for every task.
- ❖Static scheduling based on task duration estimates
- ❖Dynamic scheduling techniques to utilize unused intervals

Challenge:

- ❖ Computing optimal schedule is NP hard
- ❖ Most Heuristics: N • T, where N is the number of compute nodes in the cloud and T is the number of tasks in the job
- ❖ N and T are very large for the cloud scheduling problem

Abstract Refinement Scheduling



THE ABS SCHEDULER FISCH

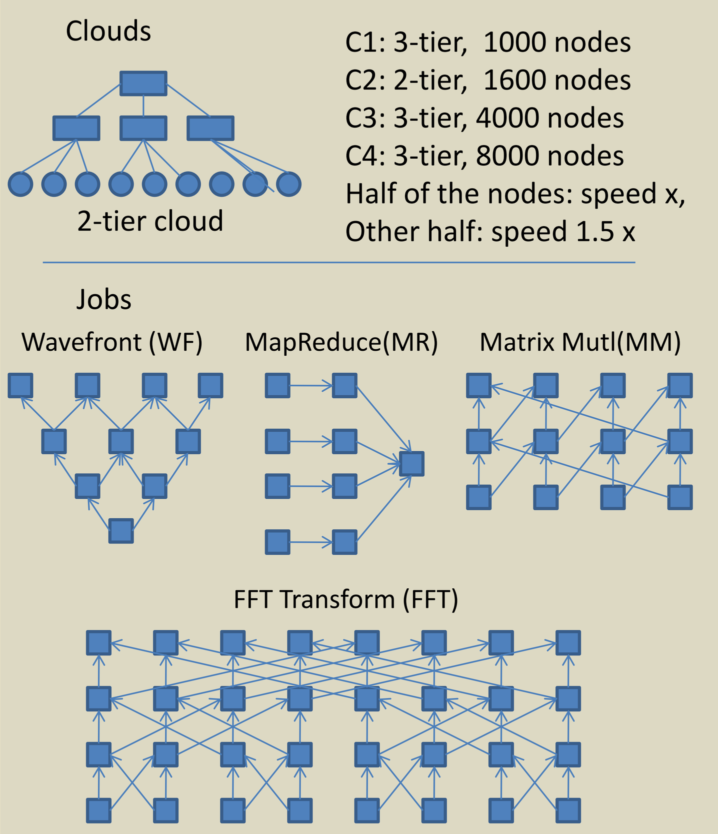
- ❖Create a fixed abstraction of the data center
- ❖Maintain the free intervals in every abstract node as an inverted index (this allows efficient retrieval of the first *n* free intervals in an abstract node)
- ❖ Choose abstract tasks in topological order
- ❖ Schedule all tasks in one abstract task in the abstract node that allows to finish the tasks at the earliest (greedy schedule).
- ❖ If schedule does not meet requirements, refine job abstraction

THE ABS SCHEDULER BLIND

- ❖ Based on the idea of buddy lists
- ❖ Starts with a coarse data center abstraction as a single abstract node
- ❖ Refines the data center abstraction on scheduling every abstract task
- ❖ Avoids too refined data center abstraction using optimzations like view coarsening

Simulation

Experimental setup



We create jobs with 1000 tasks with random computation and data transfer requirements

Results

Scheduling Latency per task (in seconds) and Cloud Utilization on C2

Job	Latency	Util	Latency	Util
MR	0.34	86%	0.32	93%
MM	1.34	55%	1.95	77%
FFT	1.89	68%	1.40	78%
WF	1.57	49%	0.71	62%

Cloud utilization on different clouds after scheduling the sequence of 1000 jobs

Scheduler	C1	C2	C3	C4
FISCH	90%	71%	73%	76%
BLIND	85%	80%	77%	79%

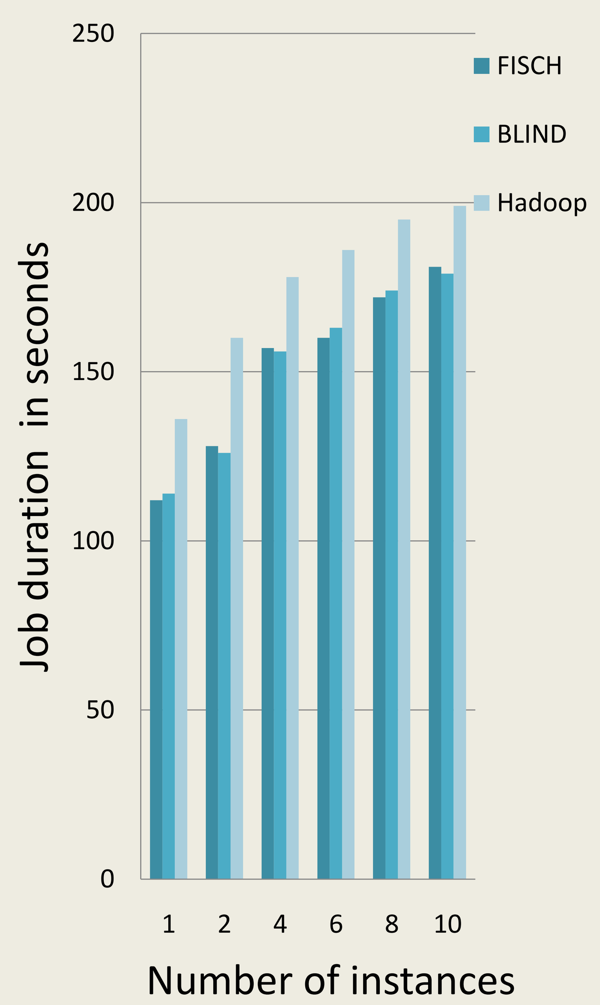
Comparing AR Schedulers to Hadoop

Experimental setup

Job Description:
A MapReduce Job
Size of each job: 4 MB
Mapper: An image transformation, requires 8.1 seconds on average, set the estimate to 40 seconds
Reducer: Identity operation

Cloud:
We rent Amazon EC2 m1.xlarge
(15 GB RAM, 4 virtual cores, 64-bit)
Number of compute nodes: 50N
(N is the number of instances)

Hadoop: Version 0.19.0



Tom Henzinger Vasu Singh Thomas Wies Damien Zufferey
IST Austria