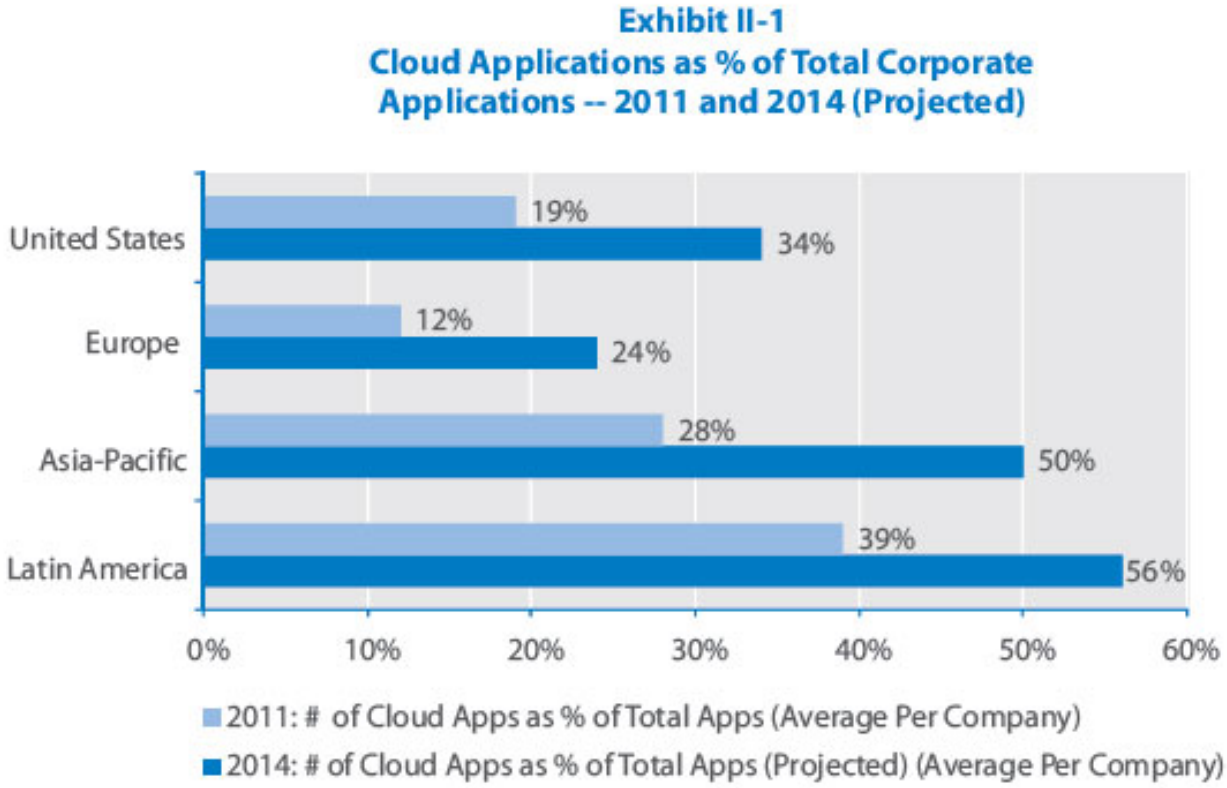
Scheduling Jobs in the Cloud Using On-demand and Reserved Instances

Siqi Shen[1], Kefeng Deng[2], Alexandru Iosup[1], and Dick Epema[1]

1 Parallel and Distributed Systems Group
Delft University of Technology, The Netherlands
2 National University of Defense Technology, China

Context

Deploying applications in leased cloud infrastructure is increasingly popular



Source: TCS cloud study
http://sites.tcs.com/cloudstudy/the-state-of-adoption-of-cloud-applications#.Ug-T8NJHIiA

Challenges of IaaS cloud adoption

- Multiple virtual machine (VM) configurations
  - General purpose: m1.small, m1.large
  - Compute-optimized: c1.medium, c1.large
  - Memory-optimized: m2.xlarge, m2.2xlarge, m2.4xlarge
  - IO-optimized: hi1.4xlarge
- VMs are not priced linearly with the performance
- Multiple billing option: on-demand, reserved, spot
  - On-demand: pay hourly usage cost
  - Reserved instance: reserved for a long term with an upfront cost but cheaper hourly cost

Research question

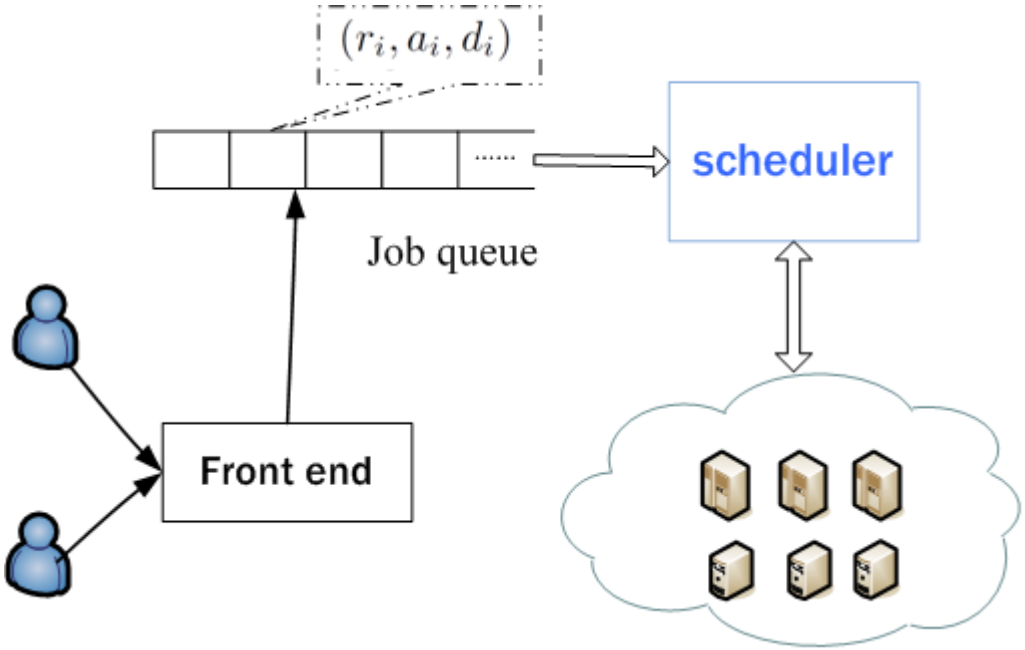
- Finding scheduling policy that can schedule diverse workloads with low waiting time yet cheaply is the focus of this work

Agenda

- System model
- Scheduling jobs using on-demand instances
- Using on-demand and reserved instances
- Experimental results
- Conclusion & future works

System model

Job 1 (2, 1, 10)
Job 2 (8, 4, 20)



- independent jobs; not migration, preemption
- r_i is the resource requirement of job i, a_i is the arrival time, and d_i is its departure time
- operation and billing model, Amazon EC2.

Assumption

- Once a job arrives at the system, the runtime and the resource requirements of the job are known.
- We rely on a predictor to give us the job requirement and run time. Currently we assume that the prediction is perfect.

Tasks of Scheduler

- Scheduler runs periodically

1. Predicting workloads
2. Proactively provisioning VM in advance from cloud
3. Allocating jobs to VMs
4. Releasing VMs
5. Start VMs if the wait time of jobs are high

CoH: a Cloud-base, online, Hybrid scheduler

- to find a combination of VMs, and a mapping between jobs and VMs
- CoH is designed as a portfolio scheduler
  - Create a set of scheduling policies
  - Online selection of the active policy, periodic selection

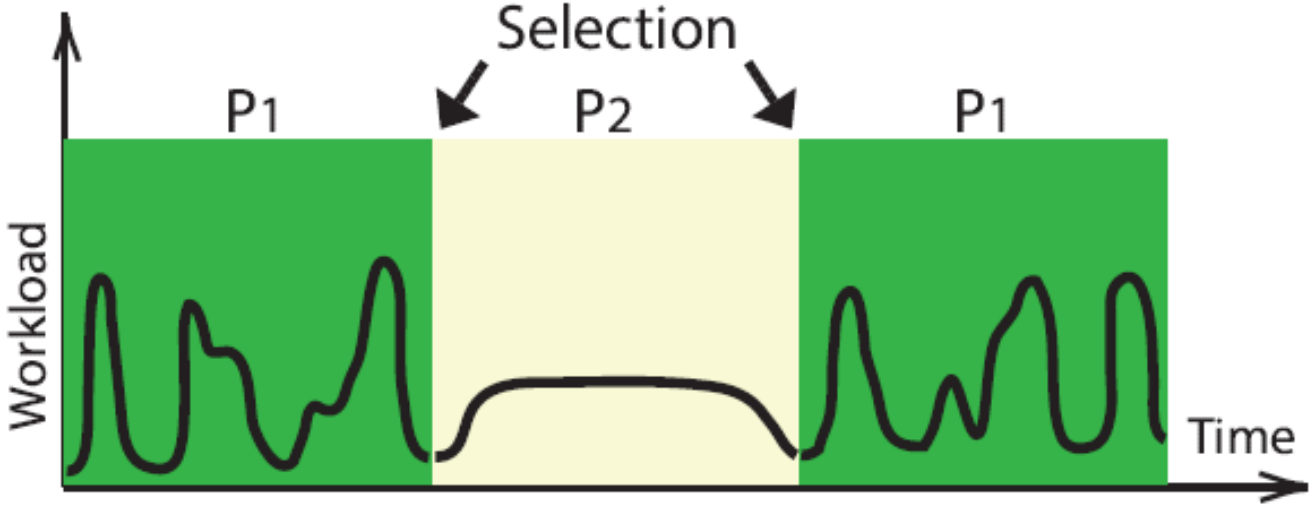
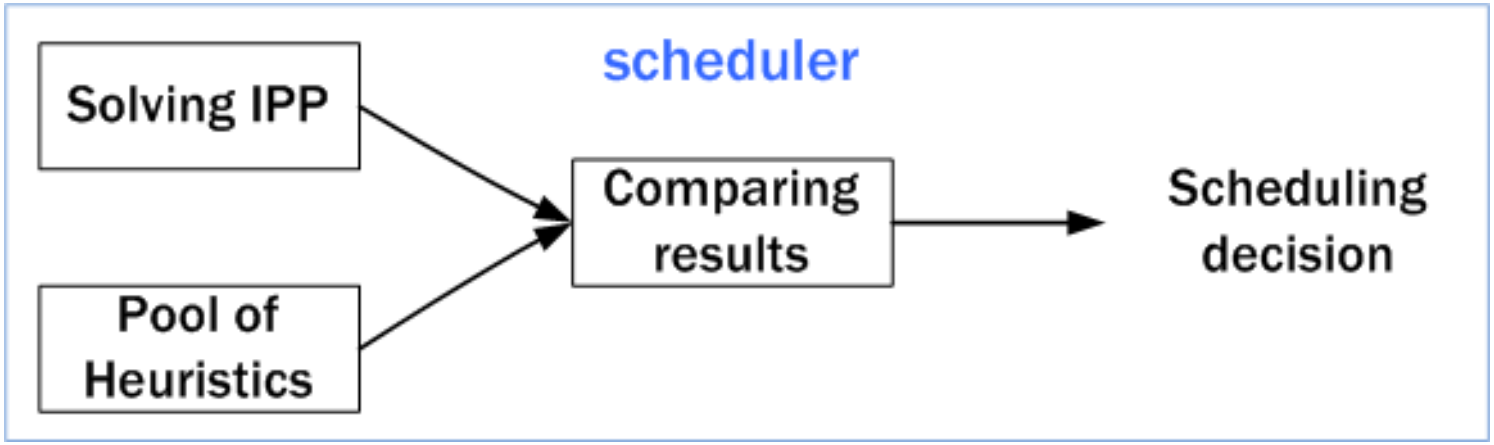


Figure source: Deng, Verboon, Iosup
http://www.pds.ewi.tudelft.nl/~iosup/Presentations/2013/2013-05-24_portfolio-sched13jsspp.pptx

CoH: a Cloud-base, online, Hybrid scheduler

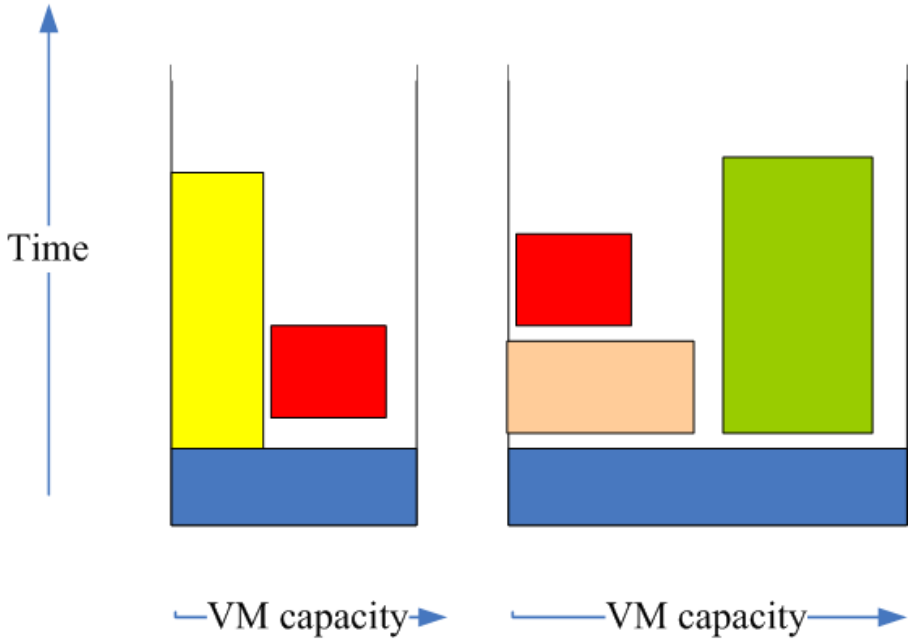
- Formulating the scheduling problems as an Integer programming problem and make the decision through math software and heuristics



- Solving IPP may take too many times, we limit the time to obtain a solution.

Integer programming problem

- To pack as much jobs as possible with minimal cost.
- Can be viewed as a variation of bin-packing problem with time factor and varying sizes of bin.



The IPP

Minimize

$$\sum_{k=1}^{K}\sum_{j=1}^{M}(y_{j_k} \times \lceil \max_{i\in\mathbb{M}}(d_i \times x_{ij_k}) - ct \rceil \times c_k) + R \quad (1)$$

subject to

$$\sum_{i=1}^{M} z_{ij} \times r_i \leq w_j \qquad \forall j \in \mathbb{N} \quad (2)$$

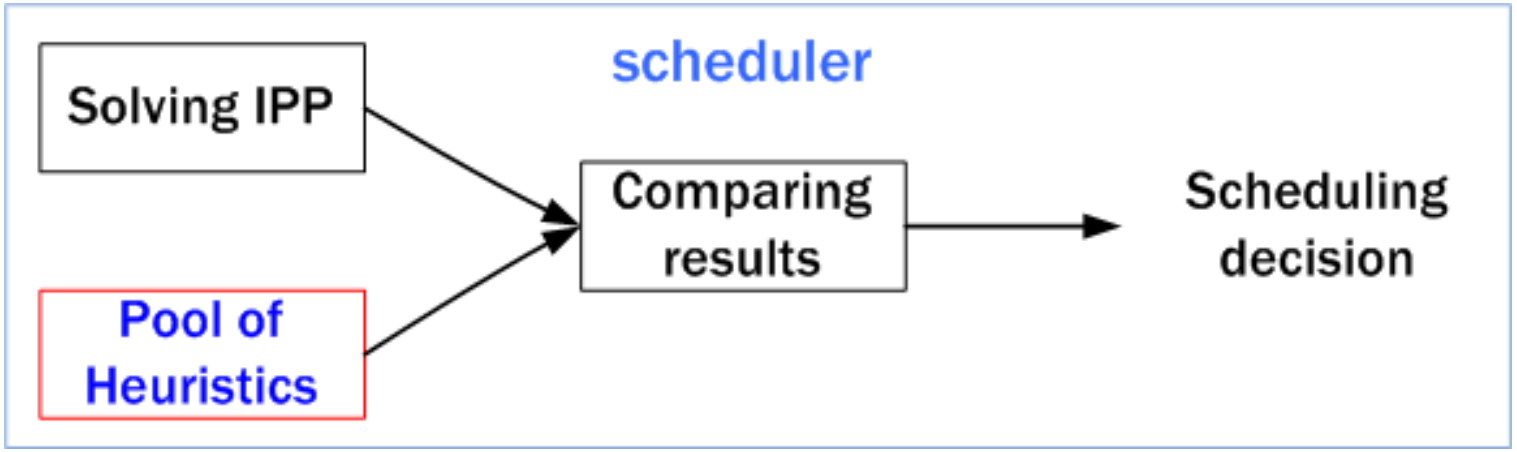
$$\sum_{k=1}^{K}\sum_{i=1}^{M} x_{ij_k} \times r_i \leq fc_k \times y_{j_k} \qquad \forall j \in \mathbb{M} \quad (3)$$

$$\sum_{j=1}^{N} z_{ij} + \sum_{k=1}^{K}\sum_{j=1}^{M} x_{ij_k} = 1 \qquad \forall i \in \mathbb{M} \quad (4)$$

$$x_{ij_k} \leq y_{j_k} \qquad \forall i,j \in \mathbb{M}, \forall k \in \mathbb{K} \quad (5)$$

x_{ij_k}	whether job i is assigned to jth VM of type k, $x_{ik_j} \in \{0,1\}$
y_{j_k}	whether jth VM of type k is to-be-provisioned, $y_{k_j} \in \{0,1\}$
z_{ij}	whether job i is assigned to jth running VM, $z_{ij} \in \{0,1\}$
c_k	hourly cost of VM of type k
c_j	hourly cost of running VM j

Heuristics



- Heuristic = Provisioning policy + Allocation policy
- Provision policy: When and which type of VM to boot
- Allocation policy: mapping jobs to VMs
- Provision policy = Job selection criteria + VM selection criteria

Algorithm 1 Provisioning heuristic.

Input:

The set of jobs, $\{job_j\}$;
The set of running VMs, $\{VM_i\}$;

Output:

Number of to-be-provision VM n_k of each type k;

```
1: while there are un-allocated future jobs do
2:    Selecting job_j according to job selection criteria;
3:    Sorting VM_i according to VM selection criteria;
4:    for each VM_i do
5:       if the VM_i can allocate job_j then
6:          allocate job_j to VM_i;
7:          break;
8:       end if
9:    end for
10:   if job_j is still unallocated then
11:      Provision a new most cost-efficient VM;
12:      Adding the new VM into VM sets VM_i;
13:      n_k++;
14:   end if
15: end while
16: return {n_k};
```

Job and VM selection criterias

Name	Description
FCFS	First-come-first-server
RR	round-robin
LJF	Largest job first
SJF	Smallest job first
LTJF	Longest run-Time job first
STJF	Shortest run-Time job first

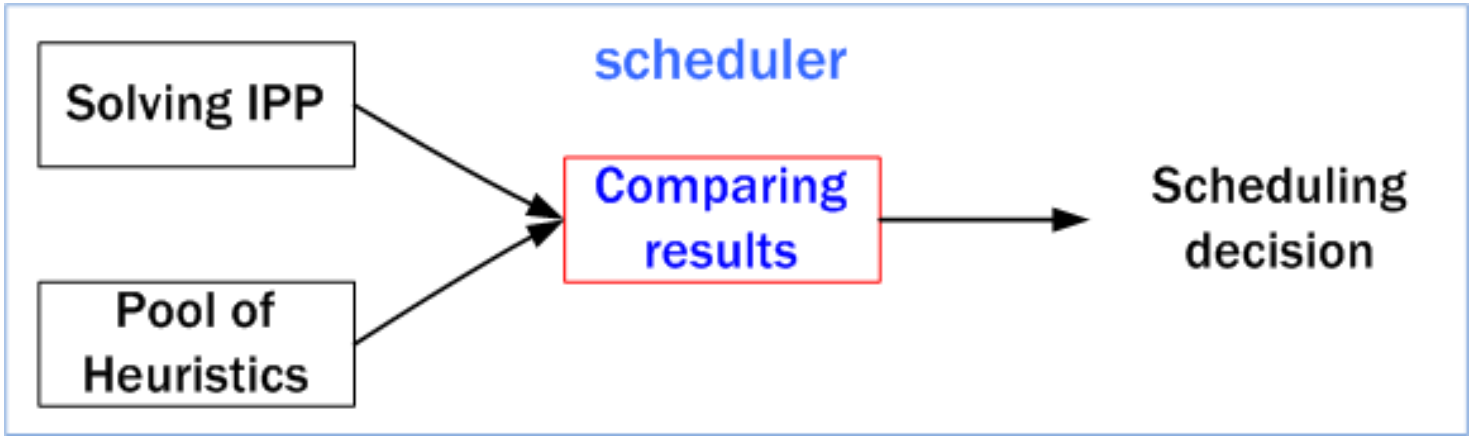
Table 2. Job selection criteria

Name	Description
Rnd	Random
LM	Largest capacity VM first
SM	Smallest capacity VM first
LA	Latest arrival time
EA	Earliest arrival time
LD	Latest departure time
ED	Earliest departure time
LAA	Latest average arrival time
EAA	Earliest average arrivial Time
CFH	Close to Full Hour
FFH	Far from full hour

Table 3. VM selection criteria

Job selection criteria does not have high impact of the final cost but VM selection criteria does.

Comparing results



- Picking the one has the best object value

CoH – R: CoH using reserved instances

- Reserved instances: need to pay an upfront cost UF_i and pay a lower on-demand cost per hour C_i
- EC2: lightly, medium, heavily utilized reserverd.
  - Uf_i from low to high, while C_i from high to low
  - m1.large on-demand 1 year cost $2102.4
  - m1.large heavily utilized 1 year cost $1166.56
- Policy
  - Always try to allocated jobs on reserved instances with lower C_i
  - When planning to start a new virtual machines, try to start an VM with lower C_i

Determining numbers and types of reserved instances

- D_t, number of VM required at time t
- N_k, number of lightly, medium, and heavily utilized instances
- C_k, hourly cost of lightly, medium, and heavily utilized instances
- $B(D_t)$the cost need to pay for demand D_t

$$\begin{cases} 0, & D \leq N_3 \\ C_2 \times (D - N_3), & N_3 \leq D \leq (N_3 + N_2) \\ C_1 \times (D - \sum_{i=2}^{3} N_i) + C_2 \times N_2, & \sum_{i=2}^{3} N_i \leq D \leq \sum_{i=1}^{3} N_i \\ C_o \times (D - \sum_{i=1}^{3} N_i) + \sum_{i=1}^{2} (C_i \times N_i), & \sum_{i=1}^{3} N_i \leq D \end{cases}$$

$$Minimize \quad \sum_{t=1}^{T} B(D_t) + \sum_{k \in \mathbb{K}} (UF_k \times N_k)$$

Experiment setup

- Via simulation using our own simulator
- Scheduler runs every 10 seconds
- We solve the IPP using CPLEX and limit the time for a solution to be 2 seconds
- Pricing model: Amazon EC2
- Each VM need 2 minutes to boot up
- VM capacity (CPU only)
  - Type A: 100
  - Type B: 410

Workloads

- Using 3 real-world traces

Trace	#jobs	average runtime [s]	duration	source
Grid5000	200,450	2728	May 2004 - May 2004	Grid workload archive [13]
LCG	188,041	8971	Nov 2005 - Dec 2005	Parallel workload archive [5]
DotaLicious	109,251	2231	Apr 2010 - Apr 2011	Game trace archive [8,9]

Table 4. Overview of traces.

- 3 workload methods for job resource consumptions
  - Heterogeneous : 10 to 100, random step of 10
  - Constant-10
  - Constant-100

Experiment results

CoH can lead to 20% to 40% lower cost than using heuristics

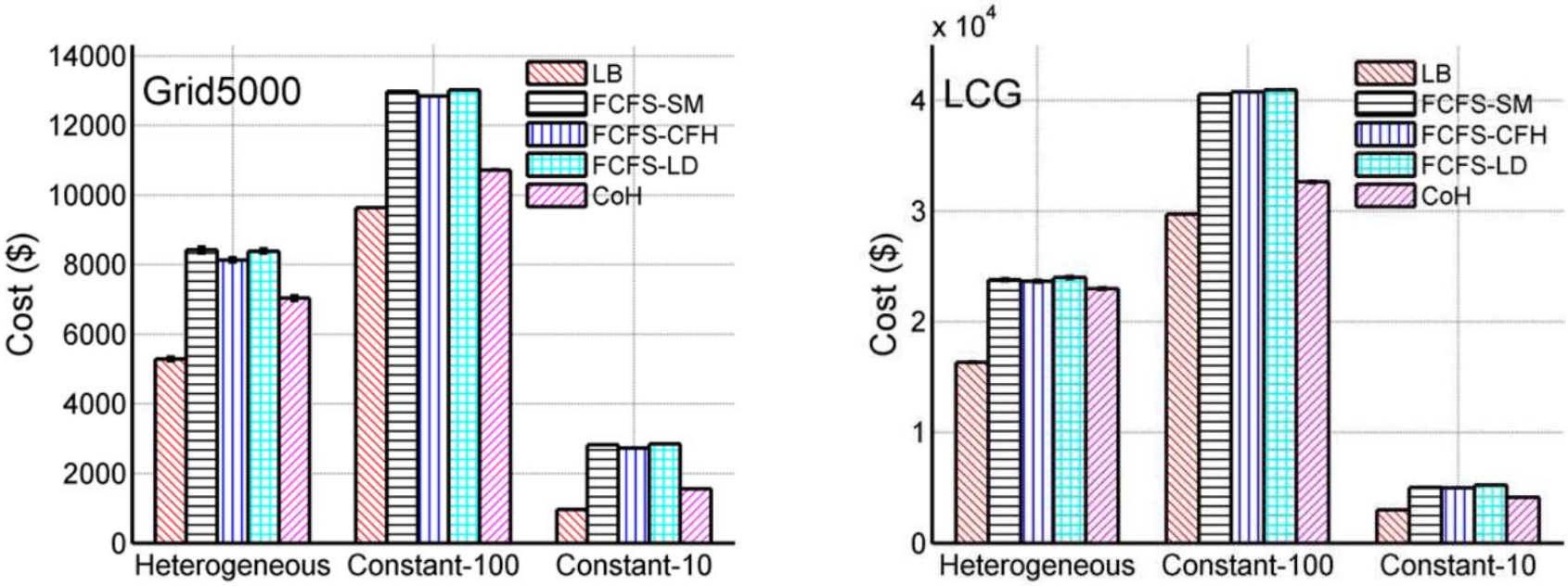
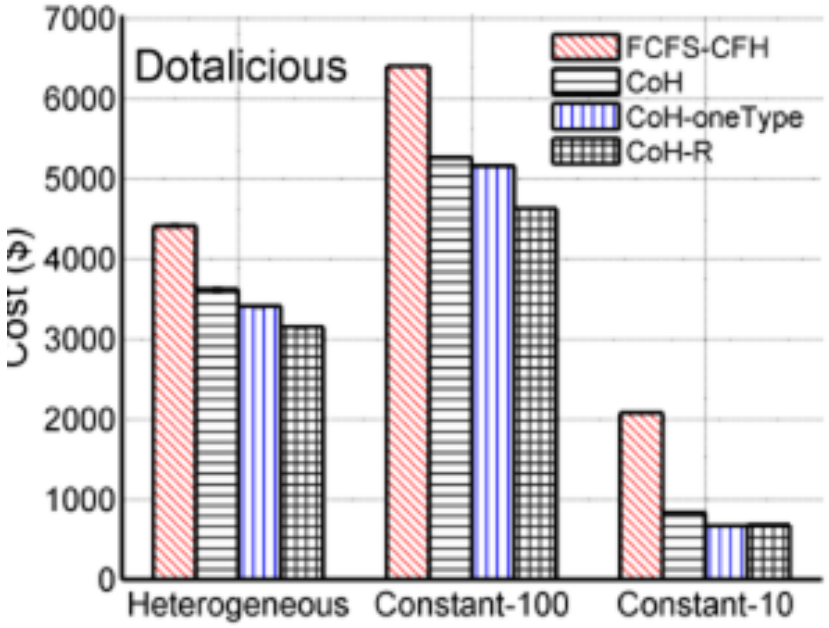
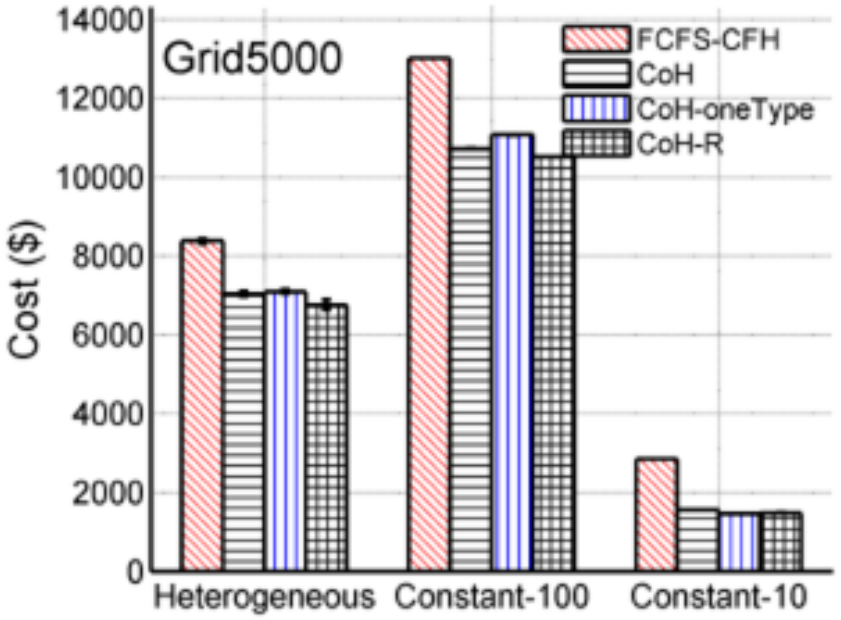


Fig. 1. Cost of various scheduling methods: Grid5000 (Left) and LCG (Right).

Experiment results

CoH-R can lead to 30% to 60% lower cost than using heuristics



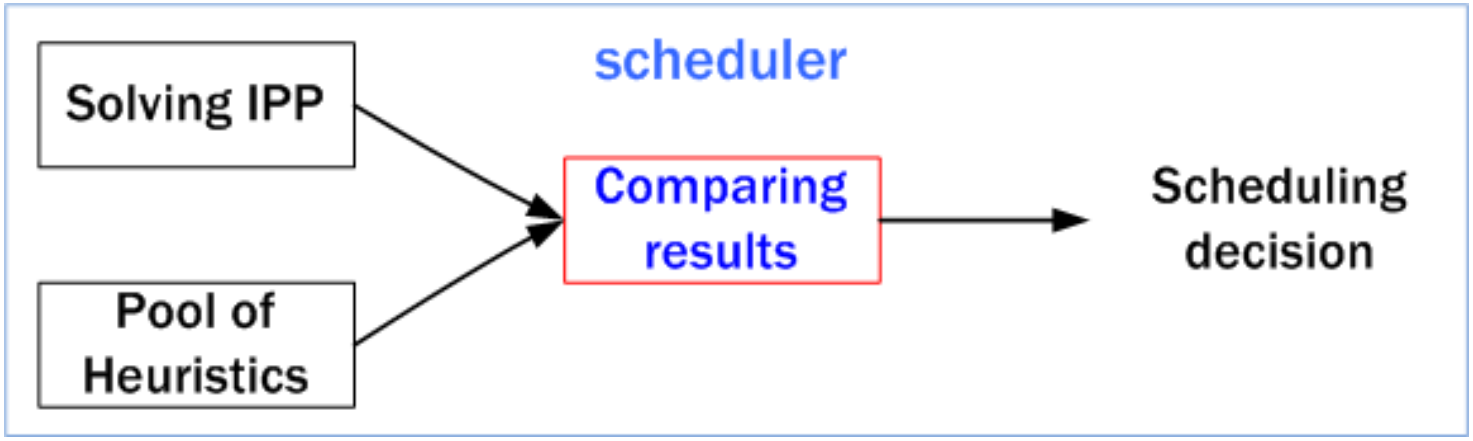


Effect of using reserved instances: Dotalicious (Left) and Grid5000 (Right).

Conclusion

- CoH, a Cloud-base, online, Hybrid scheduling policy which make uses of multiple machine configurations to plan enough capacity for users with less cost.
- CoH operates as a portfolio scheduler, make the scheduling decision by comparing the results of IPP and heuristics and pick the best results.
- CoH-R can use reserved instances to further reduce cost

Future works



- About the IPP (even step-by-step optimal does not mean global optimal)
  - Taking history records of policy into account?
- How to pick the right one, which metric to pick
- Evaluating more dimension of resource requirement
- Evaluating inaccuracy estimation of runtime

Thanks for your attention

- Any questions, comments?

- S.Shen@tudelft.nl
- http://www.pds.ewi.tudelft.nl/~siqi/

NWO Netherlands Organisation for Scientific Research

China Scholarship Council www.csc.edu.cn

973