SURF

Dutch Research Center #3322

I am an Ad!

Exploring Portfolio Scheduling for Long-term Execution of Scientific Workloads in IaaS Clouds

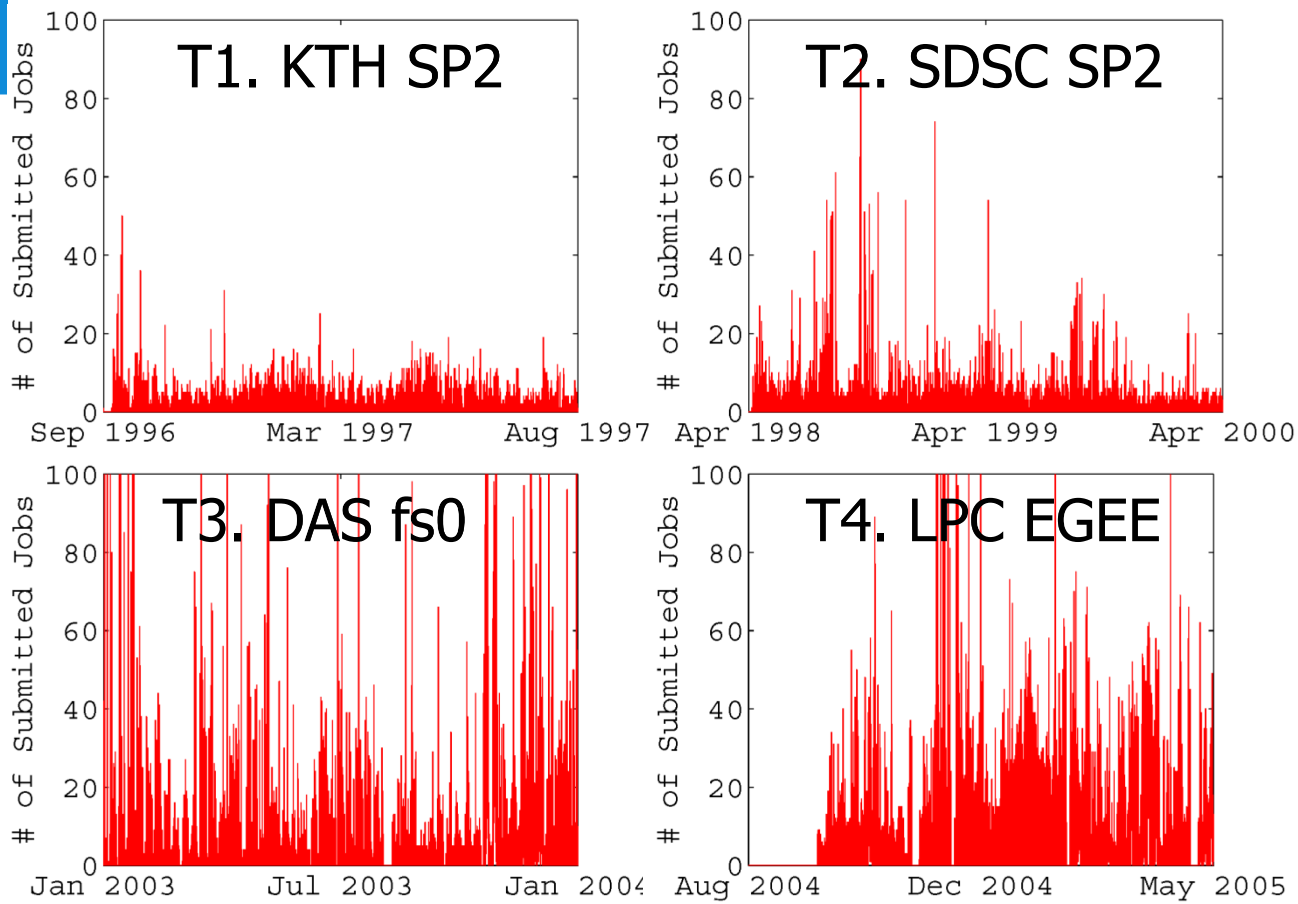
19 November 2013

Kefeng Deng, Junqiang Song, and Kaijun Ren
National University of Defense Technology
Changsha, China

Alexandru Iosup
Delft University of Technology
Delft, the Netherlands

What do These Workloads Have in Common?

Hint: ok, they're scientific and bursty, but also …



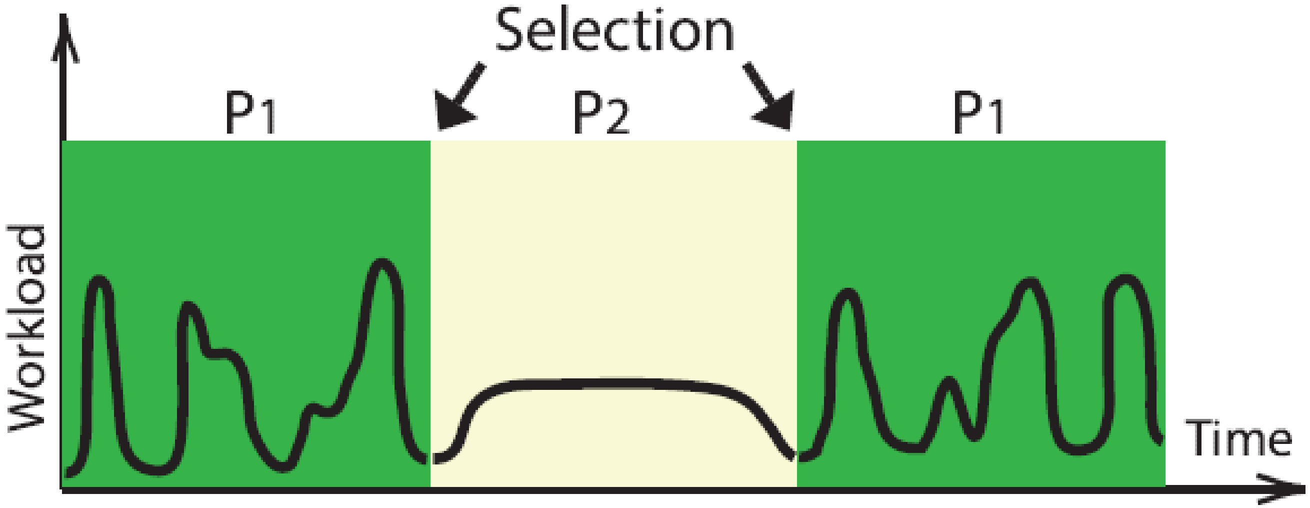
- Sources:
    - The Parallel Workloads Archive
    - The Grid Workloads Archive

Why Portfolio Scheduling?

- **Old scheduling aspects**
  - Workloads evolve over time and exhibit periods of distinct characteristics
  - No one-size-fits-all policy: hundreds exist, each good for specific conditions
- **Data centers increasingly popular (also not new)**
  - Constant deployment since mid-1990s
  - Users moving their computation to IaaS-cloud data centers
  - Consolidation efforts in mid- and large-scale companies
- **New scheduling aspects**
  - New workloads
  - New data center architectures
  - New cost models

- **Developing a scheduling policy is risky and ephemeral**
- **Selecting a scheduling policy for your data center is difficult**
- **Combining the strengths of multiple scheduling policies is …**

What is Portfolio Scheduling? In a Nutshell, for Data Centers



- Create a set of scheduling policies
  - Resource provisioning and allocation policies, in this work
- Online selection of the active policy, at important moments
  - Periodic selection, in this work
- Same principle for other changes: pricing model, system, …

Agenda

1. Why portfolio scheduling?
2. What is portfolio scheduling? In a nutshell…
3. **Our periodic portfolio scheduler for the data center**
    1. Generic process
    2. A portfolio scheduler architecture, in practice
    3. **Time-constrained simulation**
4. Experimental results
   How useful is our portfolio scheduler? How does it work in practice?
5. Our ongoing work on portfolio scheduling
6. How novel is our portfolio scheduler? A comparison with related work
7. Conclusion

Please read

Deng, Verboon, Ren, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP'13.

Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

What is Portfolio Scheduling? The Generic Process

Which policies to include?

Creation

Which policy to activate?
Explain to sysadmin

Selection

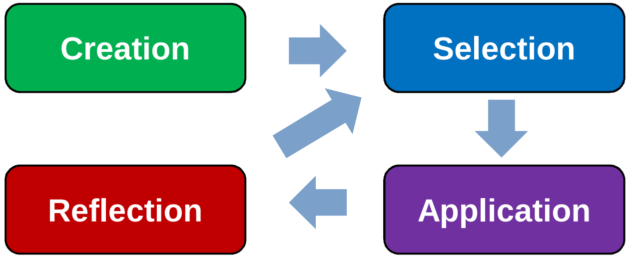
Application

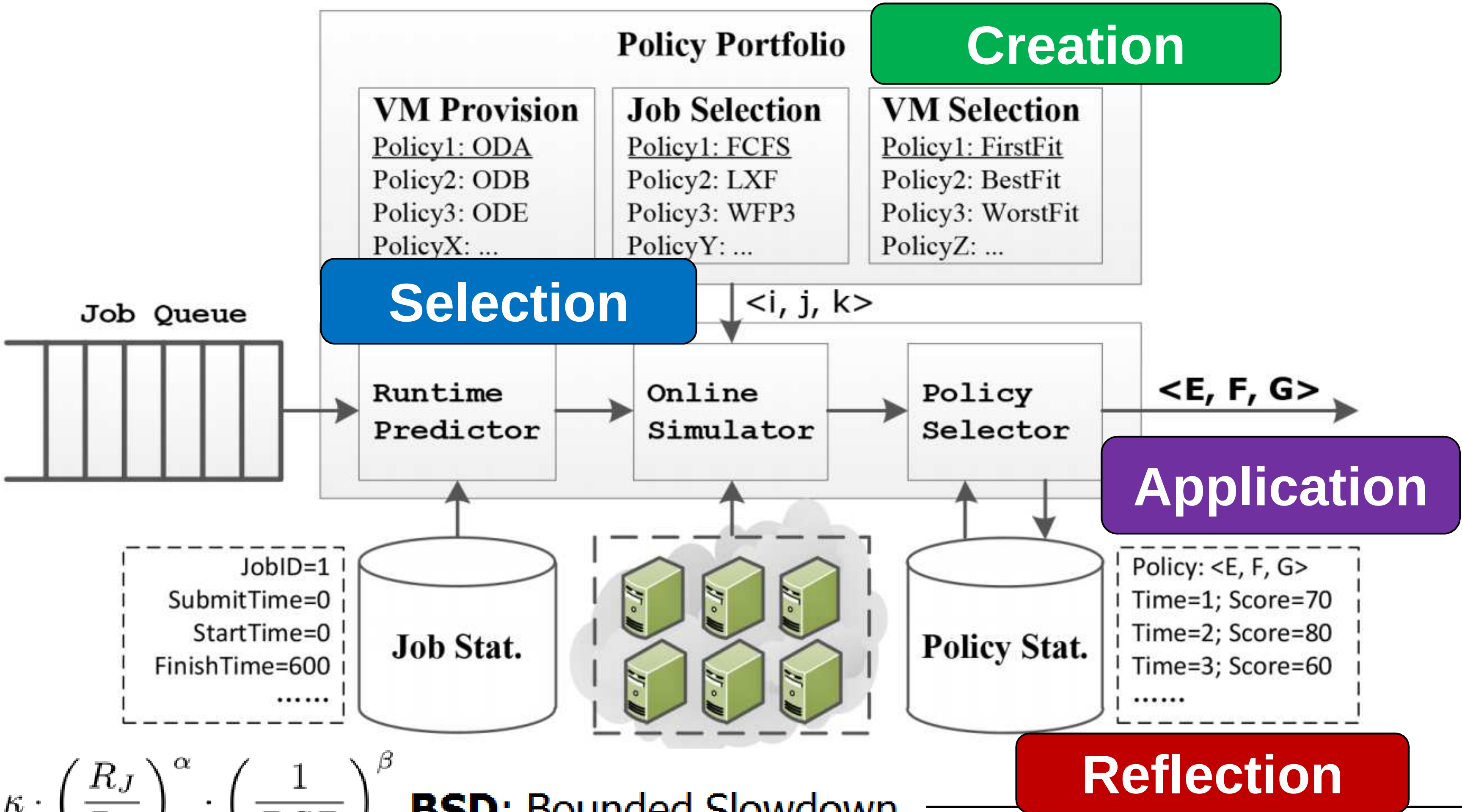
Which resources? What to log?
Validate selection immediately

Reflection

Which changes to the portfolio?

The Portfolio Scheduler





$$U = \kappa \cdot \left(\frac{R_J}{R_V}\right)^{\alpha} \cdot \left(\frac{1}{BSD}\right)^{\beta}$$ **BSD**: Bounded Slowdown

R_J: Total Runtime of Jobs; **R_V**: Total Runtime of VMs

The Creation of a Policy Portfolio (1)

Can add any policy here. The portfolio scheduler ideally combines strengths by always selecting well.

- **60** Policies = **5** Provisioning × **4** Job Selection × **3** VM Selection

- **5** VM provisioning policies:
  1) **ODA** (**O**n-**D**emand **A**ll): baseline policy, leases whenever there are available VMs
  2) **ODB** (**O**n-**D**emand **B**alance): tries to keep the number of required VMs and the number of rented VMs balanced ~ DawningCloud
  3) **ODE** (**O**n-**D**emand **E**xecTime): leases VMs for every queued job ~ our prev. work
  4) **ODM** (**O**n-**D**emand **M**aximum): leases the maximum number of VMs requested by jobs currently in the queue, so at least one demanding job can start
  5) **ODX** (**O**n-**D**emand **X**Factor): rents the required number of VMs for every job once its expected bounded slowdown exceeds a threshold of 2 ~ Quincy

Job: runtime r_i, wait time q_i, bounded slowdown $\frac{q_i+\max(r_i,10)}{\max(r_i,10)}$

Deng, Verboon, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP’13.

The Creation of a Policy Portfolio (2)

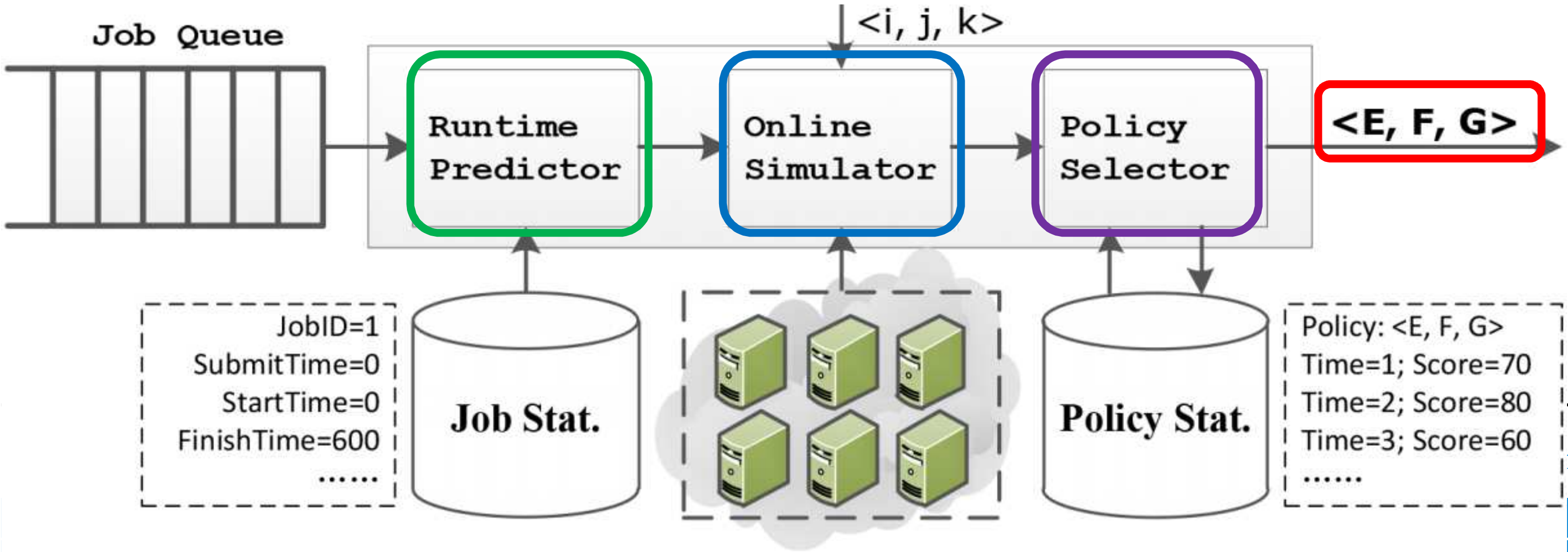
- **4** job selection policies (based on job runtime r_i, job wait time q_i, and parallelism n_i, higher priority is better):
  1) First-Come-First-Serve ($\mathbf{FCFS}$): prioritized by wait time $\mathbf{q_i}$, baseline
  2) Largest-Slowdown-First ($\mathbf{LXF}$): prioritized by slowdown $(\mathbf{q_i} + \mathbf{t_i})/\mathbf{t_i}$
  3) $\mathbf{WFP3}$: prioritized by function $(\mathbf{q_i}/\mathbf{t_i})^3 \cdot \mathbf{n_i}$, to trade-off preference for large jobs with emphasis on job slowdown
  4) $\mathbf{UNICEF}$: prioritized by function $\mathbf{q_i}/(\mathbf{log_2}(\mathbf{n_i}) \cdot \mathbf{t_i})$, to prefer small-scale jobs with short runtime

- **3** VM selection policies (cost model ~ Amazon EC2):
  1) First Fit ($\mathbf{FF}$): selects idle VMs without distinction
  2) Best Fit ($\mathbf{BF}$): selects idle VMs with minimum remaining time
  3) Worst Fit ($\mathbf{WF}$): selects idle VMs with maximum remaining time

Other Ingredients of a Portfolio Scheduler

Q: Can you see a problem with running the simulator for each policy?

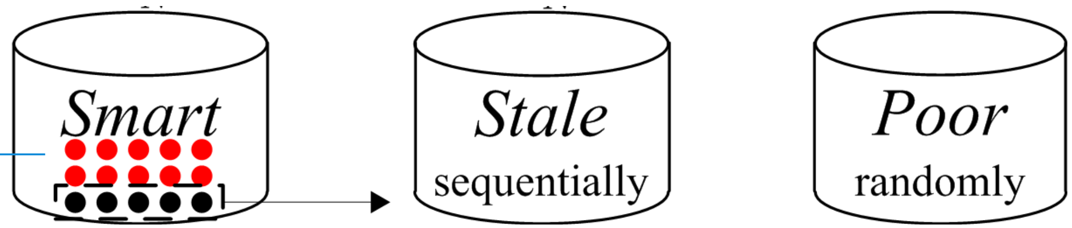
- Online Simulator and Policy Selector
  - For each policy, **simulate** scheduling **all** the **queued jobs**, then output an utility value (score)
  - **Select** the **policy** with the highest score for **real-world operation**



Time-Constrained Simulation (1)

- Given
  - N: Total number of policies and Δ: Constrained simulation time
- Approach (for uniform execution time of per-policy simulation)
  - Three classes of policies, Smart, Poor, Stale (not recently explored)
  - Explore speculatively policies from the three classes

To be simulated



Cannot simulate now

Simulated &
Sorted by score

$K = \|Q\| = 10$

$\lambda = 0.6$

$\|Smart\| = \lambda K$

Time-Constrained Simulation (2)

Q: Why keep running policies from Poor (historically a bad performer)?

A policy in Poor (historically performing badly) can still deliver excellent performance in the future!

Agenda

1. Why portfolio scheduling?
2. What is portfolio scheduling? In a nutshell…
3. Our periodic portfolio scheduler for the data center
    1. Generic process
    2. A portfolio scheduler architecture, in practice
    3. Time-constrained simulation
4. **Experimental results**
   How useful is our portfolio scheduler? How does it work in practice?
5. Our ongoing work on portfolio scheduling
6. How novel is our portfolio scheduler? A comparison with related work
7. Conclusion

Please read

Deng, Verboon, Ren, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP'13.

Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

Performance Evaluation

Experimental Setup (1): the system

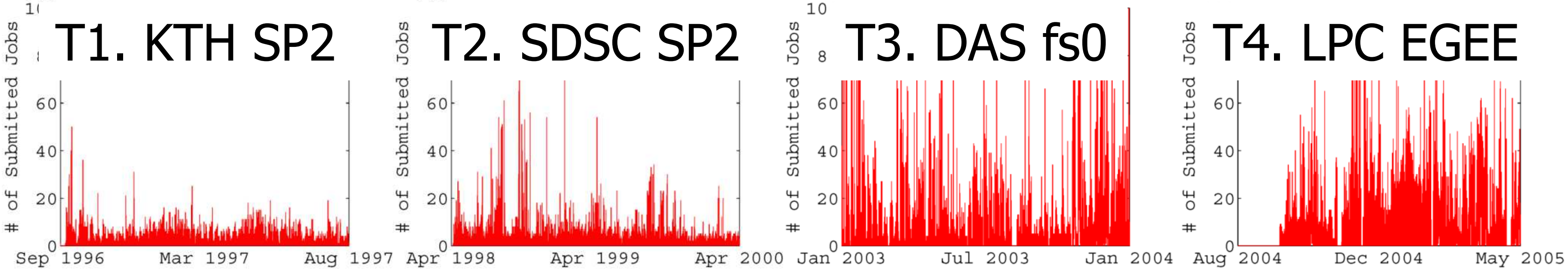
- Simulation Software: **DGSim**
- Simulation Environment:
  - A virtual cluster comprised of **homogeneous** VM instances
  - The maximum number of concurrent VMs that can be rented is **256**
  - **120** seconds delay for VM instance acquisition and booting
- Performance metrics:
  - Average bounded job slowdown
  - Charged cost: runtime of rented VMs (rounded up to the next hour)
  - Utility score: $U = \kappa \cdot \left(\frac{R_J}{R_V}\right)^{\alpha} \cdot \left(\frac{1}{BSD}\right)^{\beta}$ (*Default, κ=100, α=1, β=1*)

Iosup et al., Performance Analysis of Cloud Computing Services for Many Tasks Scientific Computing, (IEEE TPDS 2011).

Performance Evaluation

Experimental Setup (2): workload traces

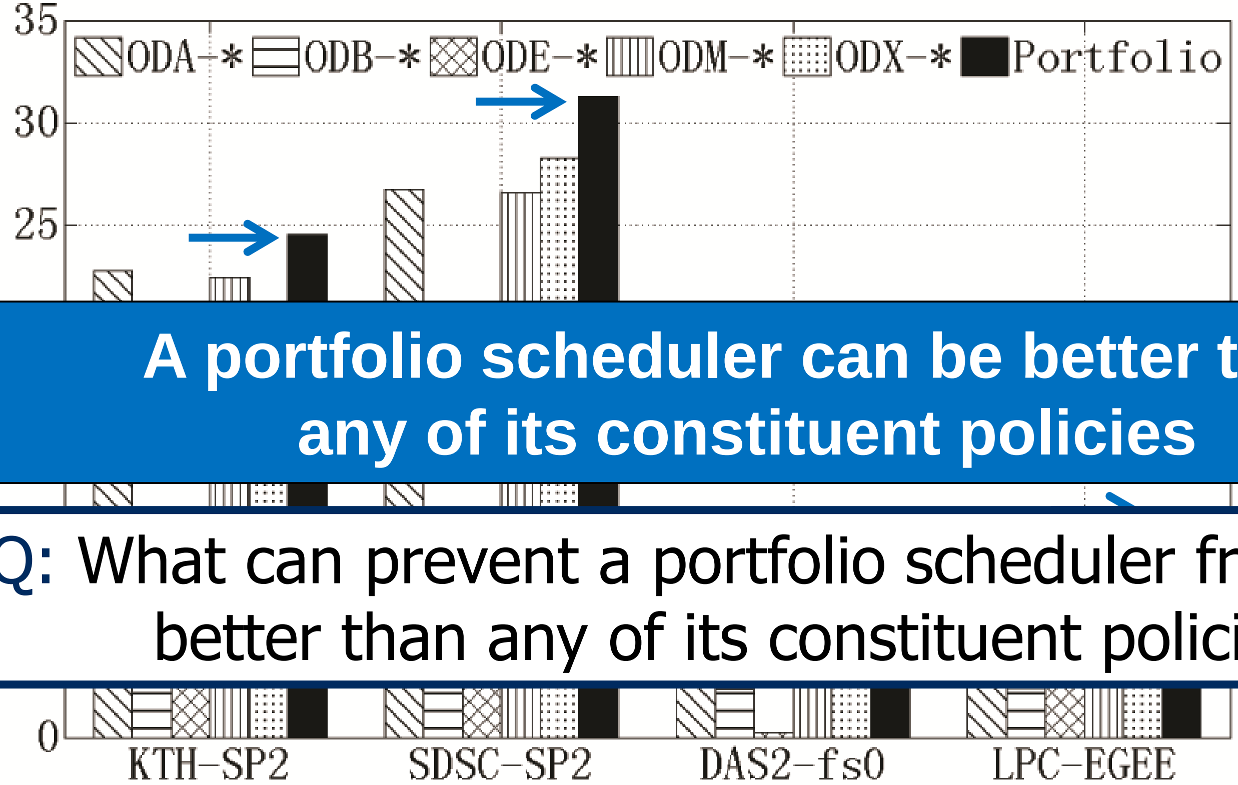
- Four traces from the Parallel Workloads Archive (PWA)
- Use from these traces jobs requesting up to **64** processors



Trace#. Name	Time [mo.]	Jobs –	Jobs ≤ 64	Jobs %	CPUs	Load [%]
T1. KTH SP2	11	28,480	28,158	98.9	100	70.4
T2. SDSC SP2	24	53,911	53,548	99.3	128	83.5
T3. DAS2 fs0	12	215,638	206,925	96.0	144	14.9
T4. LPC-EGEE	9	214,322	214,322	100	140	20.8

Performance Evaluation

1) Effect of Portfolio Scheduling (1)



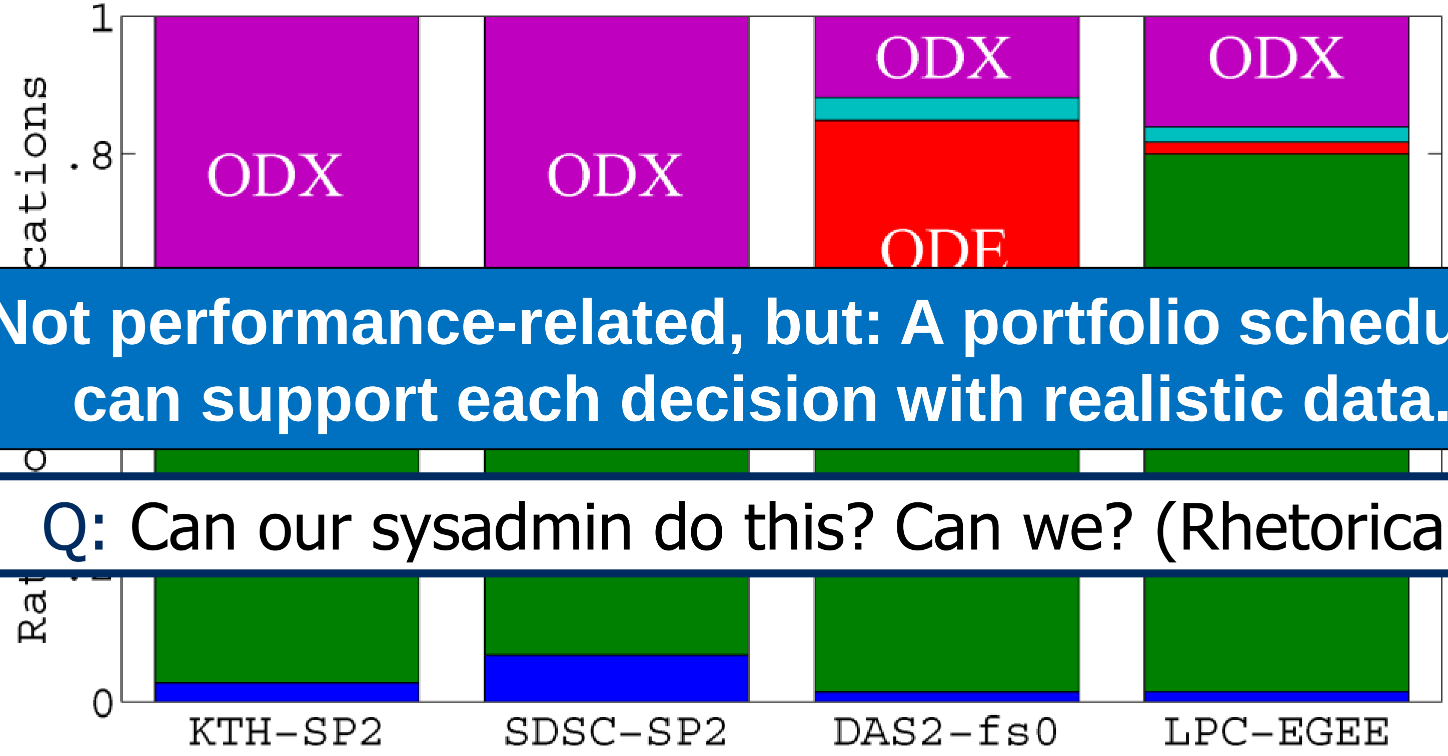
A portfolio scheduler can be better than any of its constituent policies

Q: What can prevent a portfolio scheduler from being better than any of its constituent policies?

- Portfolio scheduling is **8%**, **11%**, **45%**, and **30%** better than the best constituent policy

Performance Evaluation

1) Effect of Portfolio Scheduling (2)

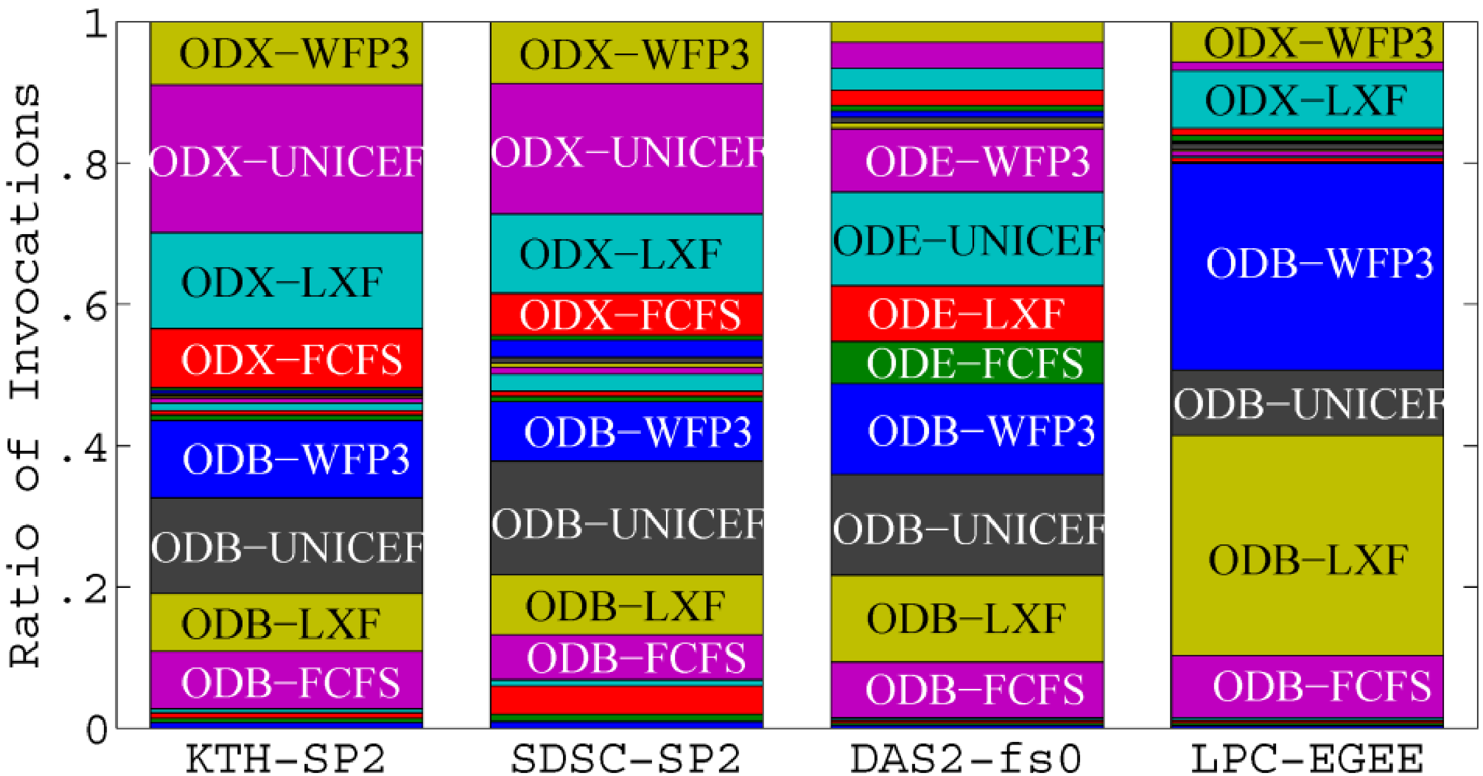


Not performance-related, but: A portfolio scheduler can support each decision with realistic data.

Q: Can our sysadmin do this? Can we? (Rhetorical)

- For KTH-SP2 and SDSC-SP2, **ODB** and **ODX** are dominant ~ many long jobs
- For DAS-fs0 and LPC-EGEE, **ODB** and **ODE** are dominant ~ short jobs, load

Q: What prevents a portfolio scheduler from being better than any of its constituent policies?

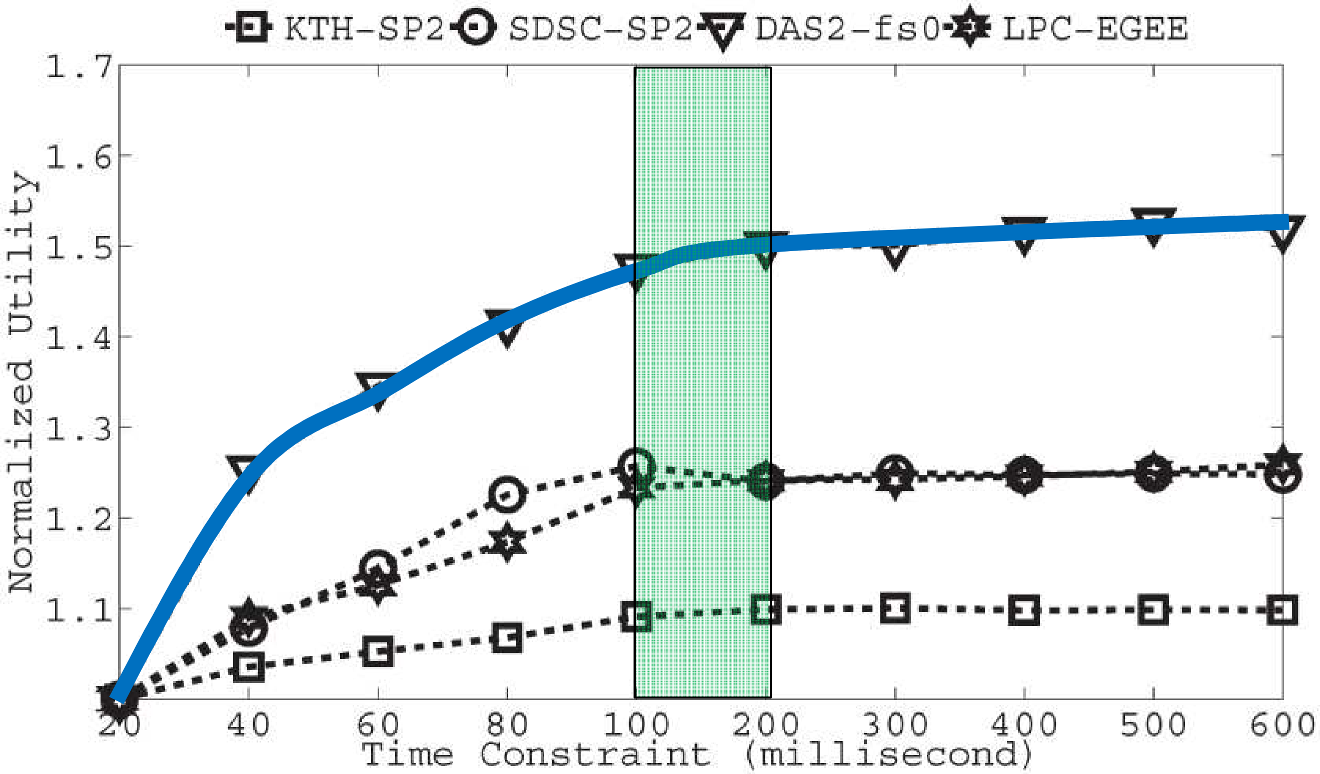


• ... he

Q: How well do you think a single (provisioning, job selection, VM selection) policy would perform? Will it be dominant? (Rhetorical)

Performance Evaluation

5) Impact of Simulation Time Constraint



- Expectedly, having more time to simulate leads in general to better results
- Here, sufficient to simulate 10—20 policies (nearly all dominant policies selected)
- Job slowdown shows different sensitivity (please read article)
- The charged cost exhibits a similar trend (please read article)

Agenda

1. Why portfolio scheduling?
2. What is portfolio scheduling? In a nutshell…
3. Our periodic portfolio scheduler for the data center
    1. Generic process
    2. A portfolio scheduler architecture, in practice
    3. Time-constrained simulation
4. Experimental results
5. **Our ongoing work on portfolio scheduling**
    Portfolio scheduling for different workloads and constituent policies
6. How novel is our portfolio scheduler? A comparison with related work
7. Conclusion

Deng, Verboon, Ren, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP'13.

Please read

Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

Portfolio Scheduling for Online Gaming and Scientific Workloads

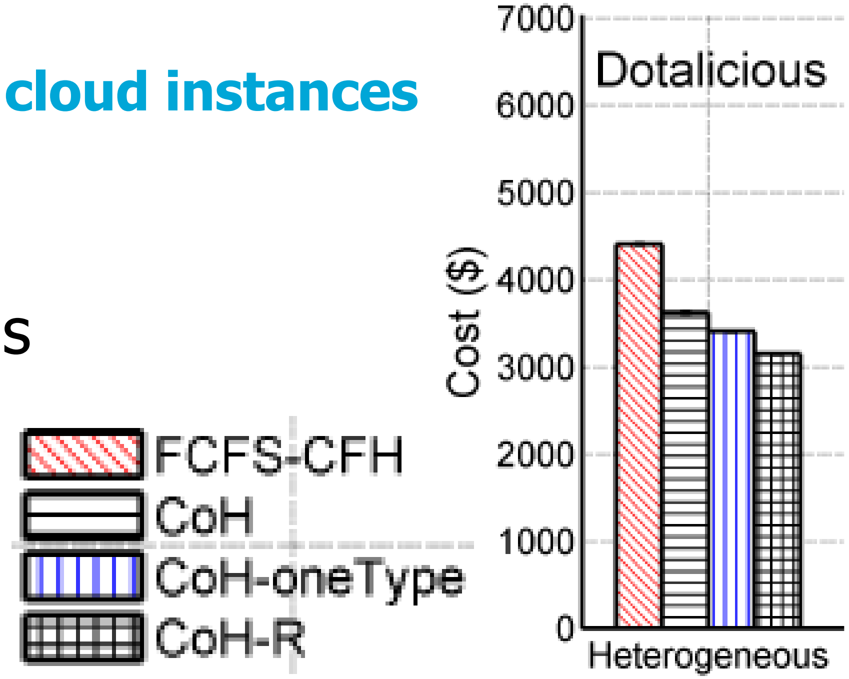
CoH = Cloud-based, online, Hybrid scheduling

- Intuition: keep rental cost low by finding good mix of machine configurations and billing options
- Main idea: **portfolio scheduler** = run *both* solver of an Integer Programming Problem and various heuristics, then pick best schedule at deadline
- Additional feature: Can use **reserved cloud instances**

- Promising early results, for

Gaming (and scientific) workloads

Trace	#jobs	average runtime [s]
Grid5000	200,450	2728
LCG	188,041	8971
DotaLicious	109,251	2231



Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

Agenda

1. Why portfolio scheduling?
2. What is portfolio scheduling? In a nutshell…
3. Our periodic portfolio scheduler for the data center
    1. Generic process
    2. A portfolio scheduler architecture, in practice
    3. Time-constrained simulation
4. Experimental results
5. Our ongoing work on portfolio scheduling
6. **How novel is our portfolio scheduler? A comparison with related work**
7. Conclusion

Please read

Deng, Verboon, Ren, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP'13.

Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

Related Work

- Computational portfolio design
  - Huberman'97, Streeter et al.'07 '12, Bougeret'09, Goldman'12, Gagliolo et al.'06 '11, Feitelson et al. JSSPP'13 **(Intel's clusters)**
  - We focus on dynamic, scientific workloads
  - We use an utility function that combines slowdown and utilization

- Modern portfolio theory in finance
  - Markowitz'52, Magill and Constantinides'76, Black and Scholes'76
  - Dynamic problem set vs fixed problem set
  - Different workloads and utility functions
  - Selection and Application very different

- General scheduling
  - Rice'76: algorithm selection problem

- Hyper-scheduling, meta-scheduling
  - The learning rule may defeat the purpose, due to historical bias to dominant policy
  - Different processes (esp. Selection, Reflection)

Agenda

1. Why portfolio scheduling?
2. What is portfolio scheduling? In a nutshell…
3. Our periodic portfolio scheduler for the data center
   1. Generic process
   2. A portfolio scheduler architecture, in practice
   3. Time-constrained simulation
4. Experimental results
5. Our ongoing work on portfolio scheduling
6. How novel is our portfolio scheduler? A comparison with related work
7. **Conclusion**

Please read

Deng, Verboon, Ren, Iosup. A Periodic Portfolio Scheduler for Scientific Computing in the Data Center. JSSPP'13.

Shen, Deng, Iosup, and Epema. Scheduling Jobs in the Cloud Using On-demand and Reserved Instances, EuroPar'13.

~~Conclusion~~ Take-Home Message

- http://www.st.ewi.tudelft.nl/~iosup/ - A.Iosup@tudelft.nl
- http://www.pds.ewi.tudelft.nl/ - DengKefeng@nudt.edu.cn

Portfolio Scheduling = set of scheduling policies, online selection

- **Creation, Selection, Application, Reflection**
- **Time constraints, here in Selection step**

Alexandru Iosup

- **Periodic portfolio scheduler for data centers**
  - Explored Creation, Selection, simple Reflection
  - Portfolio scheduler in general better than its constituent policies
  - Good results for real traces (also for synthetic)
  - Easy to setup, easy to trust

SURF #3322

- **JSSPP'13, EuroPar'13, SC'13, (future) new workload types and constituent policies + there is still much to explore about process**
- **Reality Check (future): we will apply it in our DAS multi-cluster. How about your system?**

Information

- PDS group home page and publications database: www.pds.ewi.tudelft.nl
- **KOALA** web site: www.st.ewi.tudelft.nl/koala
- **Grid Workloads Archive** (GWA): gwa.ewi.tudelft.nl
- **Failure Trace Archive** (FTA): fta.inria.fr

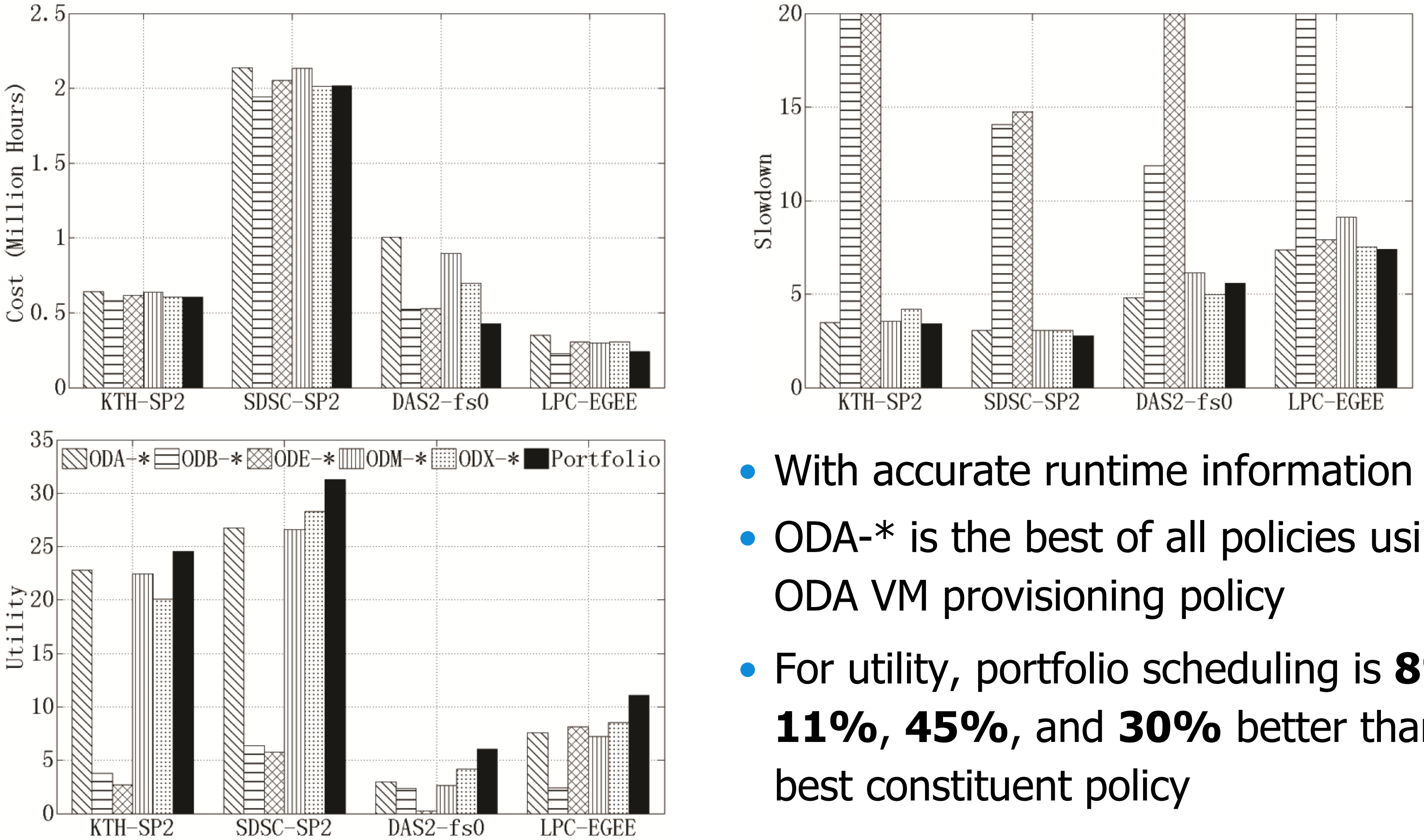
Time-Constrained Simulation (3)

Q: Why keep running policies from Poor (historically a bad performer)?

A policy in Poor (historically performing badly) can still deliver excellent performance in the future!

Performance Evaluation

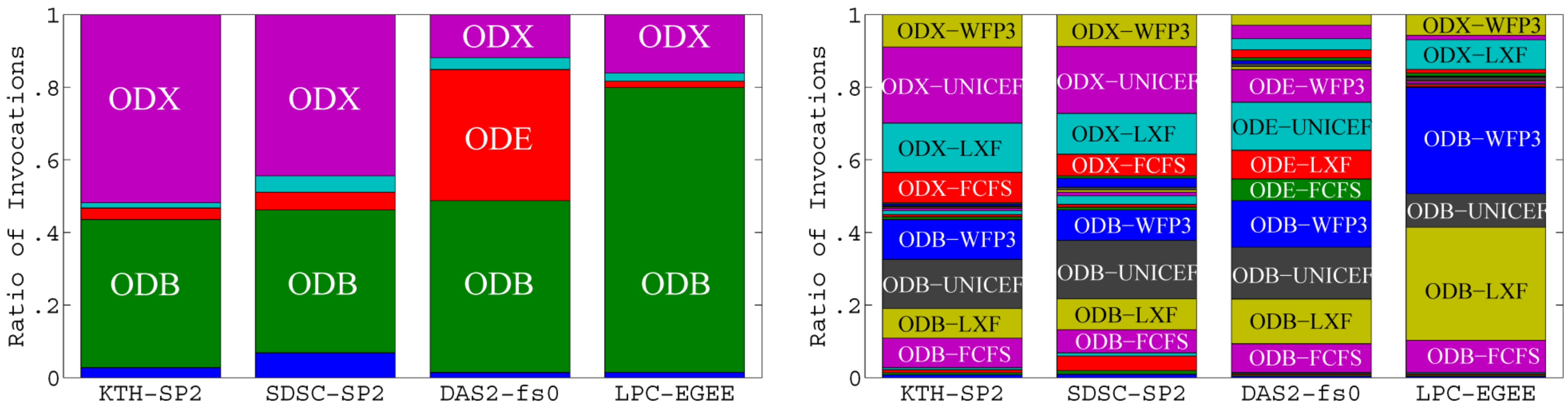
1) Effect of Portfolio Scheduling (1)



- With accurate runtime information
- ODA-* is the best of all policies using ODA VM provisioning policy
- For utility, portfolio scheduling is **8%**, **11%**, **45%**, and **30%** better than the best constituent policy

Performance Evaluation

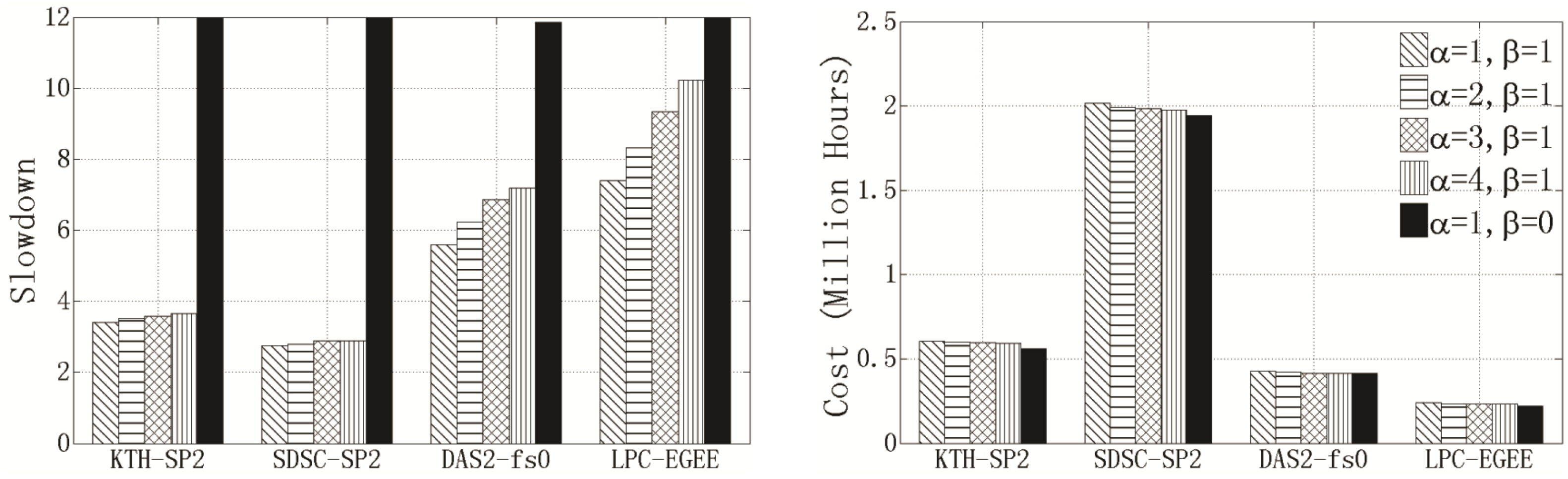
1) Effect of Portfolio Scheduling (2)



- For KTH-SP2 and SDSC-SP2, **ODB** and **ODX** are the dominant policies (a result of many long jobs)
- For DAS-fs0 and LPC-EGEE, **ODB** and **ODE** are the dominant policies (as the majority of the jobs are very short)
- Job selection policies such as **UNICEF** and **LXF** that favor short jobs have the best performance

Performance Evaluation

2) Effect of Utility Function (α—Cost-Efficiency)

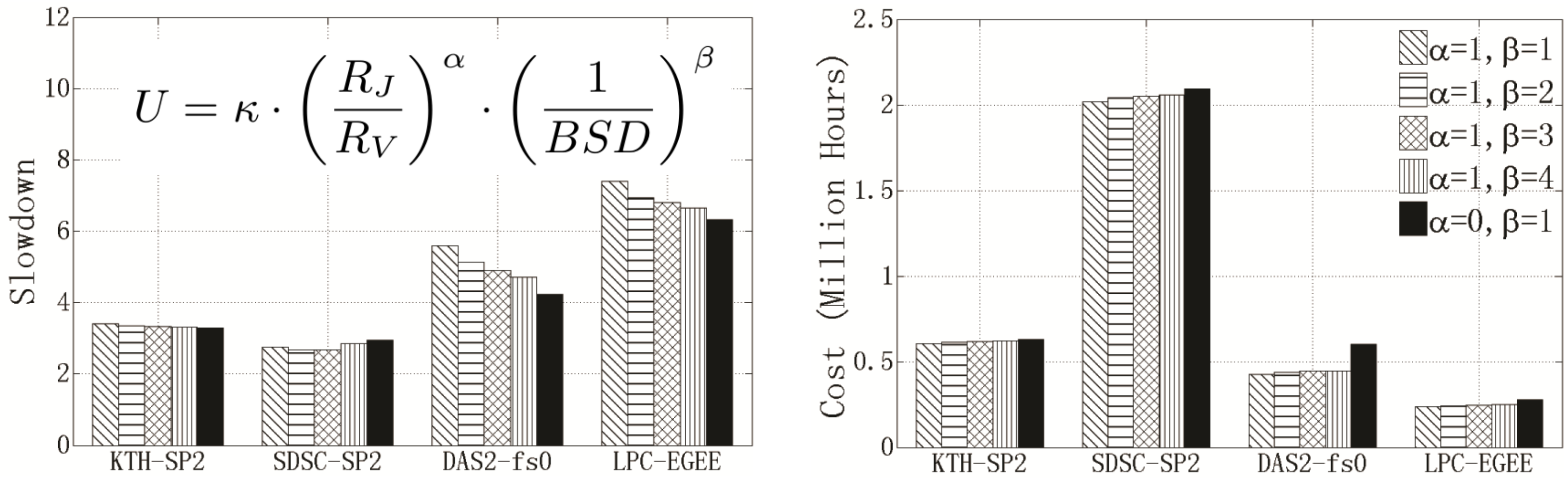


- Keep the task-urgency factor β = 1 and change the cost-efficiency factor α from 1 to 4 (the extreme setting β = 0—ignoring the job slowdown)

$$U = \kappa \cdot \left(\frac{R_J}{R_V}\right)^{\alpha} \cdot \left(\frac{1}{BSD}\right)^{\beta}$$

Performance Evaluation

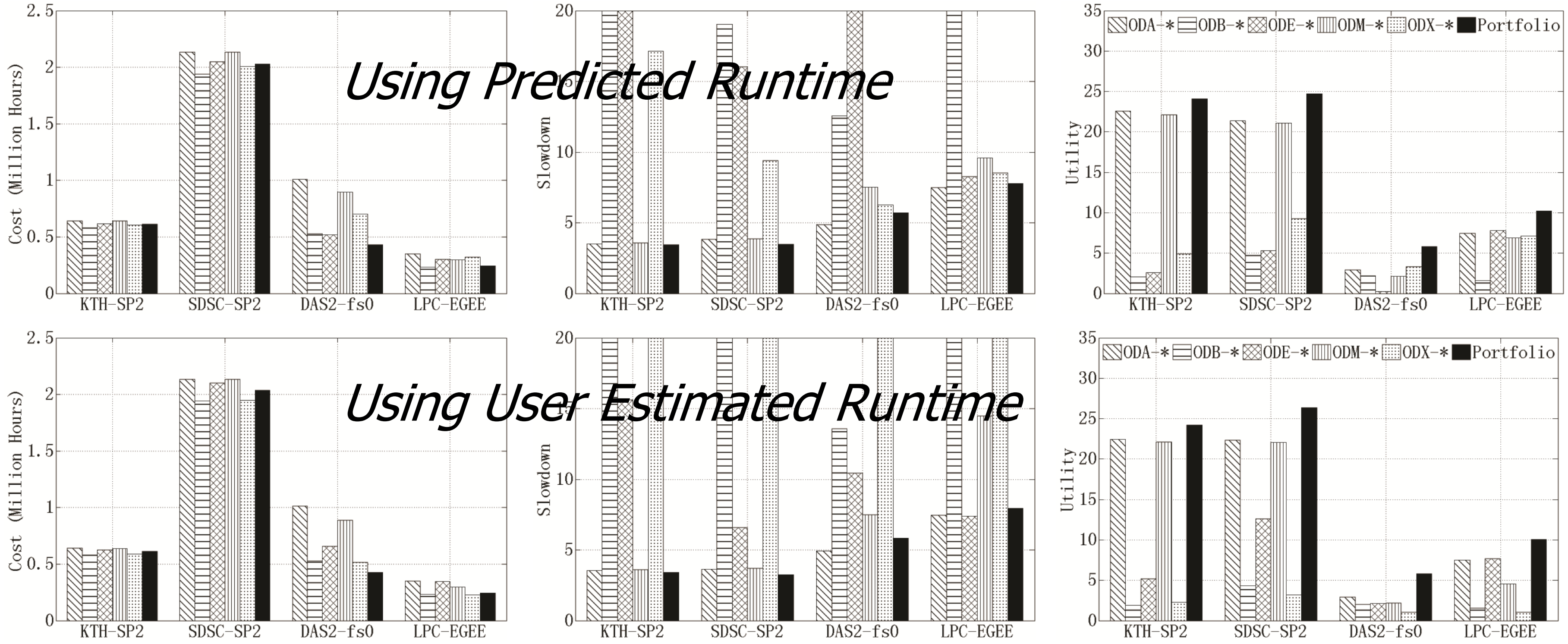
2) Effect of Utility Function (β—Task-Urgency)



- Vary the task-urgency factor, in the same way as cost-efficiency factor
- **Suggestion**: instead of putting effort to find sophisticated algorithms to reduce the cost, it is more worthwhile to find methods to improve the performance metrics that users are interested in, such as job slowdown and wait time.

Performance Evaluation

3) Impact of Runtime Prediction Inaccuracy

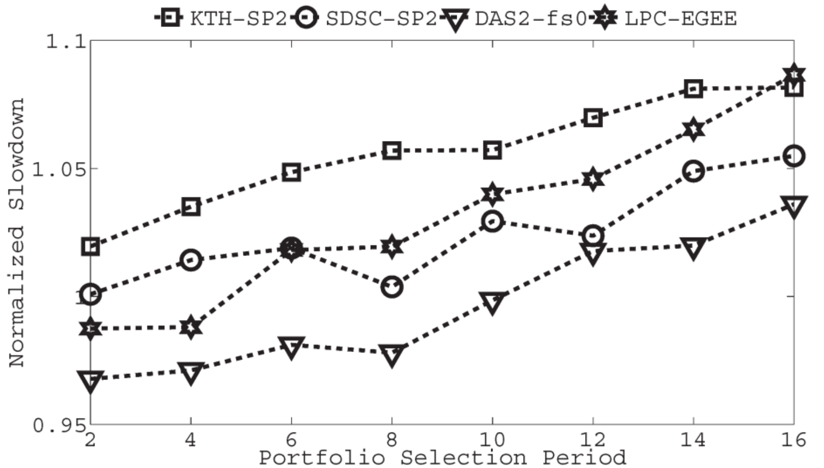


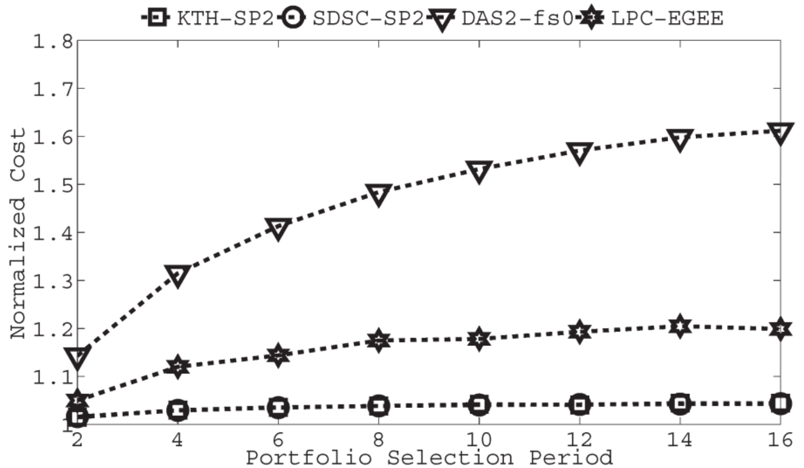
- Portfolio scheduling is not sensitive to the inaccurate runtime estimation
- Policies using job runtime are adversely affected by inaccuracy

Performance Evaluation

4) Impact of Portfolio Selection Period (1)

- **Portfolio selection period**: the interval between two consecutive selection processes (multiple times of the scheduling periods, which is 20 seconds)



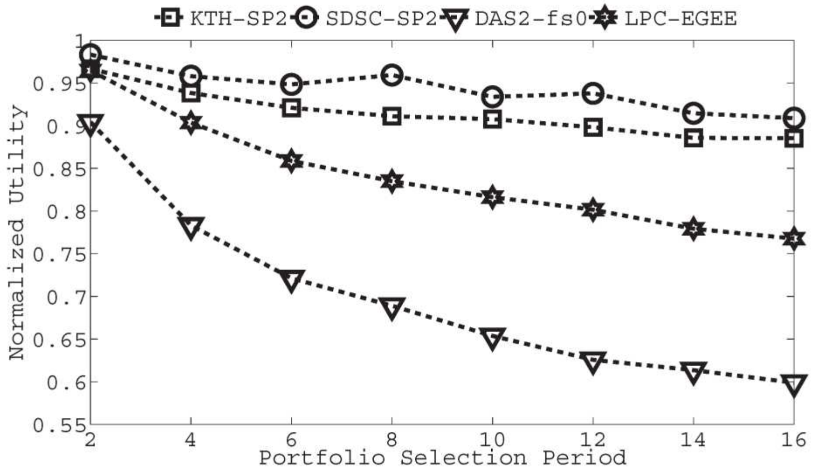


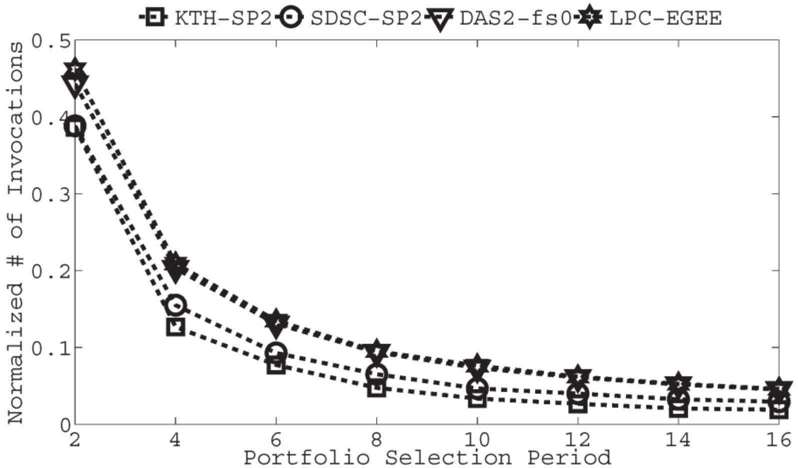
- The selection period has an insignificant impact on job slowdown (**<10%**)
- The impact on charged cost differs (**50%** for DAS2-fs0; **15%** for LPC-EGEE)

Performance Evaluation

4) Impact of Portfolio Selection Period (2)

- **Portfolio selection period**: the interval between two consecutive selection processes (multiple times of the scheduling periods, which is <u>20 seconds</u>)

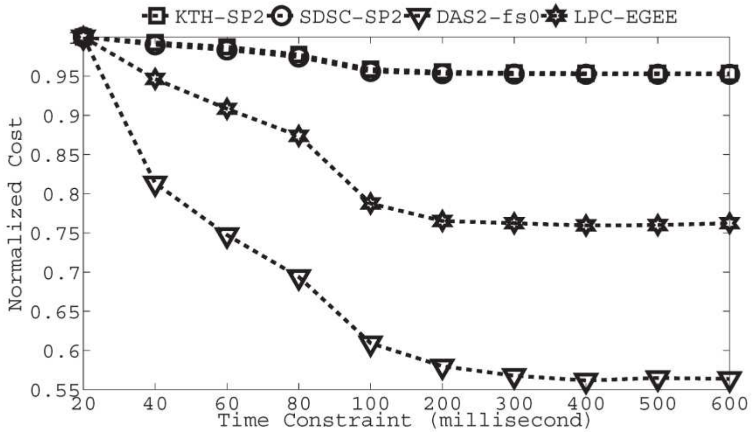


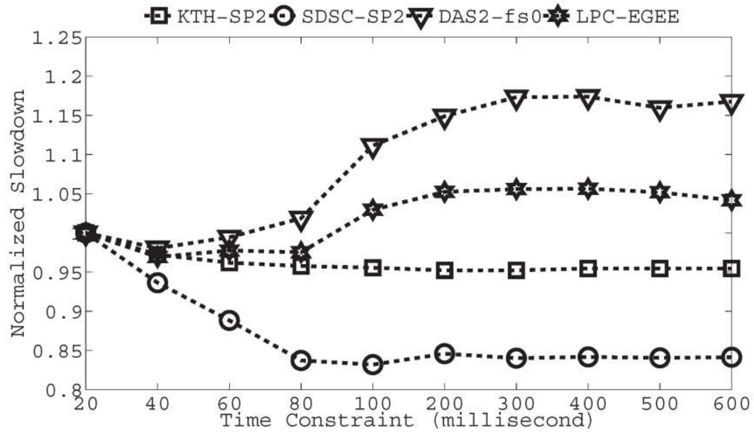


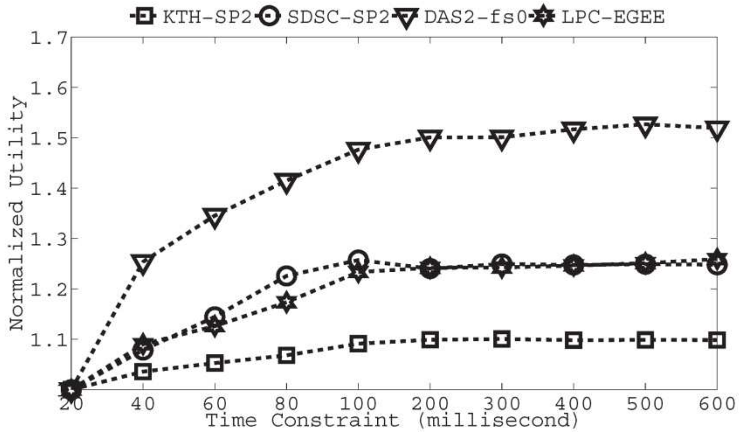
- The utility has an opposite trend in comparison with the charged cost
- The # of invocations decreases near-exponentially with the selection period
- **Suggestion**: **8** for KTH-SP2 and SDSC-SP2; **2** for LPC-EGEE; **1** for DAS2-fs0

Performance Evaluation

5) Impact of Simulation Time Constraint







- Add a **10 milliseconds** overhead for each scheduling policy
- Job slowdown shows different sensitivity
- The charged cost exhibits a similar trend
- According to the utility, simulating **1/3** of policies (60) is sufficient, as it covers almost all the dominant policies