@Large Research
Massivizing Computer Systems

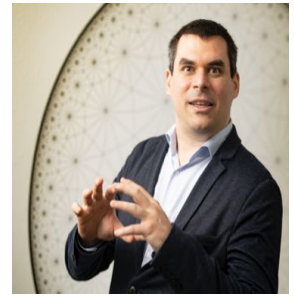
http://atlarge.science

A Reference Architecture for Datacenter Scheduler Programming: Design and Experiments

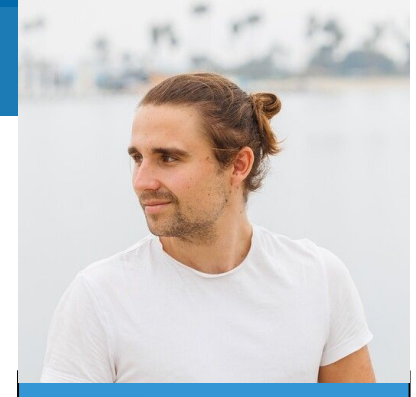
Work based on MSc Thesis

bit.ly/ref-arc-sched-pro

Prof.dr.ir.
Sacheendra
TALLURI



Prof.dr.ir.
Alexandru
IOSUP



@aratzl

MSc student. Aratz M.
LASA

VU VRIJE UNIVERSITEIT AMSTERDAM

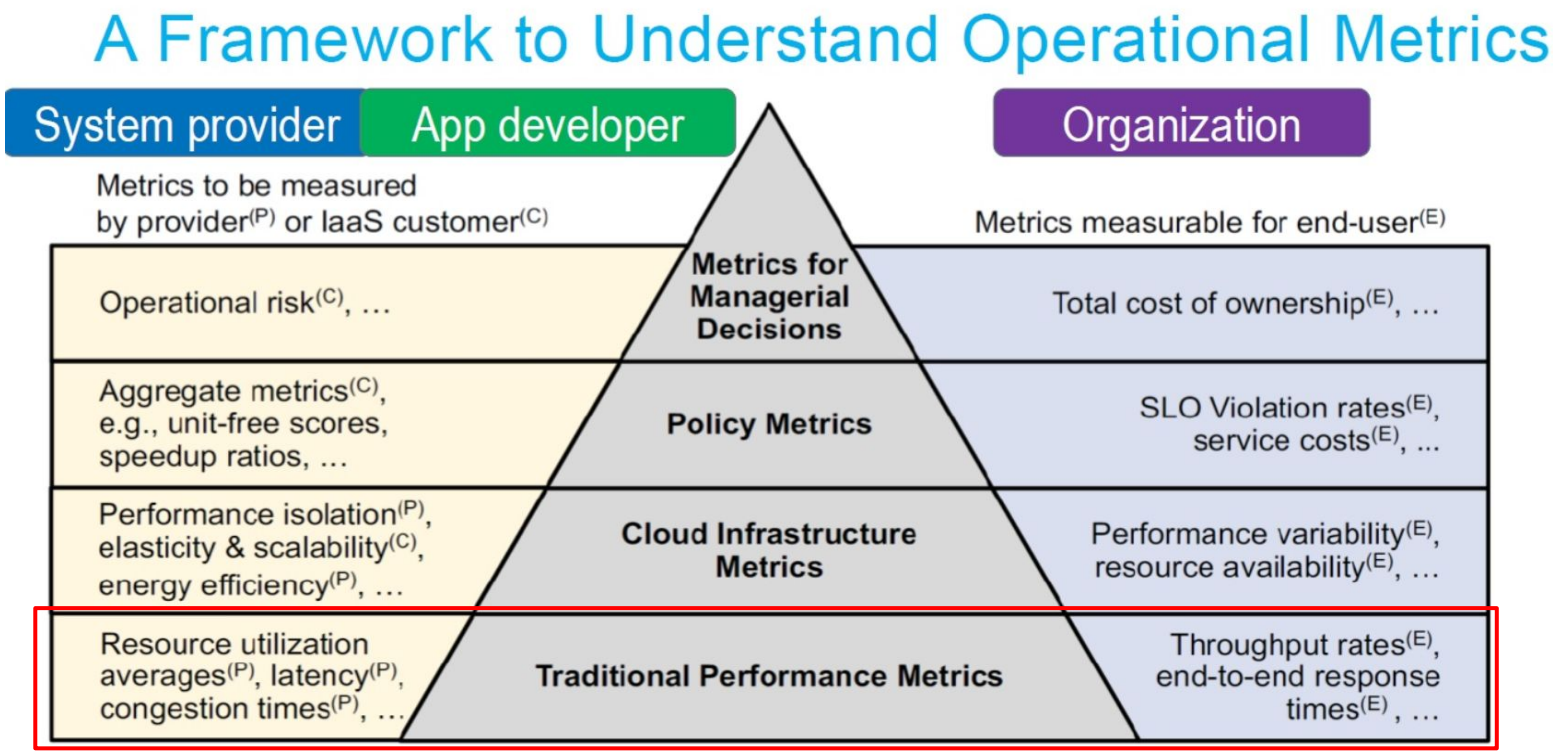
Contributions from the AtLarge team. Many thanks!
Many thanks to our collaborators, authors of all images included here.

Why schedulers are important for society?

USD **263.34 billion** in 2022

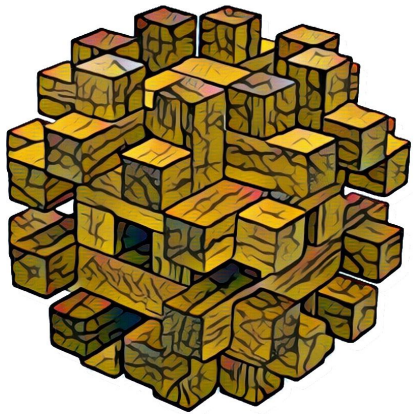
expected to reach USD 602.76 billion by 2030

What is performance?



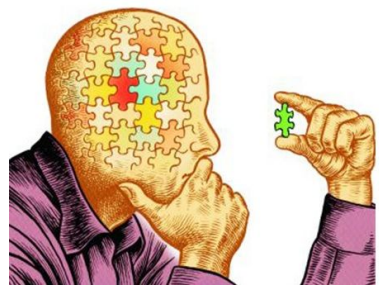
N. Herbst, A. Bauer, S. Kounev, G. Oikonomou, E. Van Eyk, G. Kousiouris, A. Evangelinou, R. Krebs, T. Brecht, C. L. Abad, A. Iosup: Quantifying Cloud Performance and Dependability: Taxonomy, Metric Design, and Emerging Challenges. TOMPECS 3(4): 19:1-19:36 (2018)

Why are schedulers important for performance?



Distributed systems increasing complexity

SLOs complexity

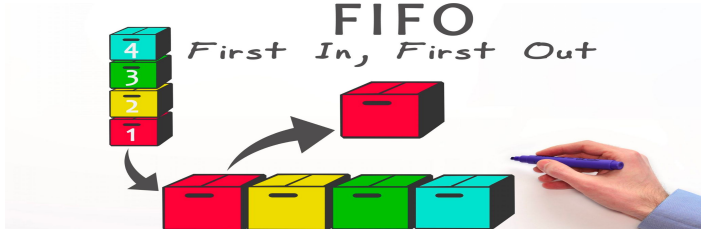


Manage complexity through global / local self-awareness

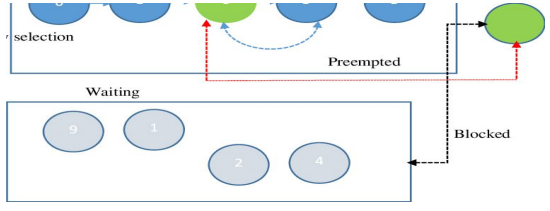
Schedulers

Scheduler's components that impact performance

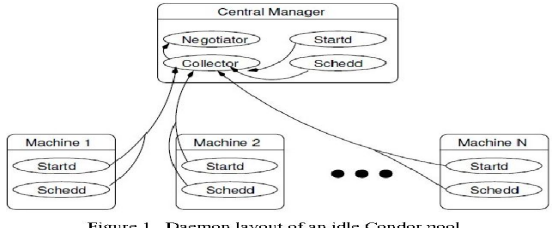
Infrastructure



Policy

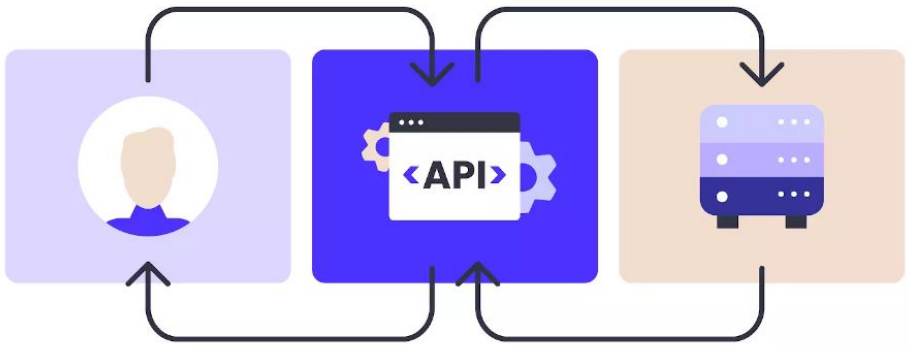


Mechanism

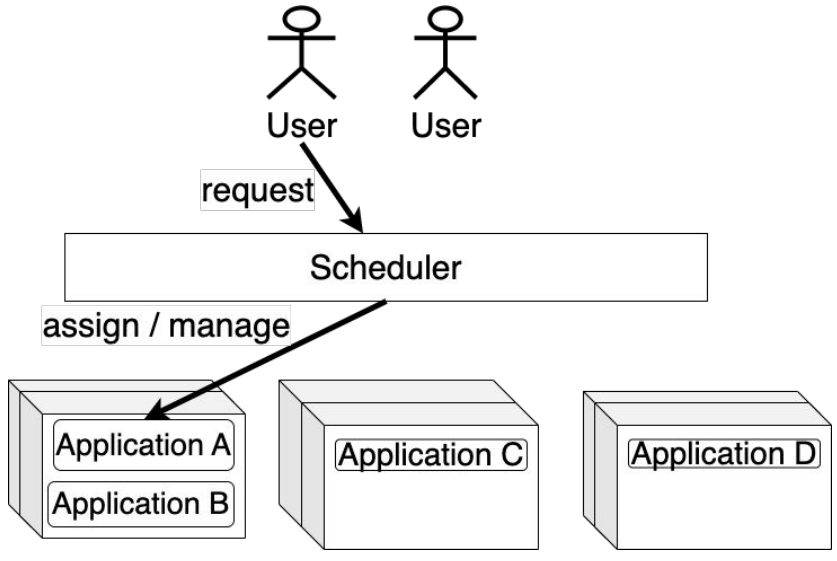


Architecture

Something else? ⟶ Programming abstraction / API



Interface between the user and the data center



Intermediary between data center resources and users

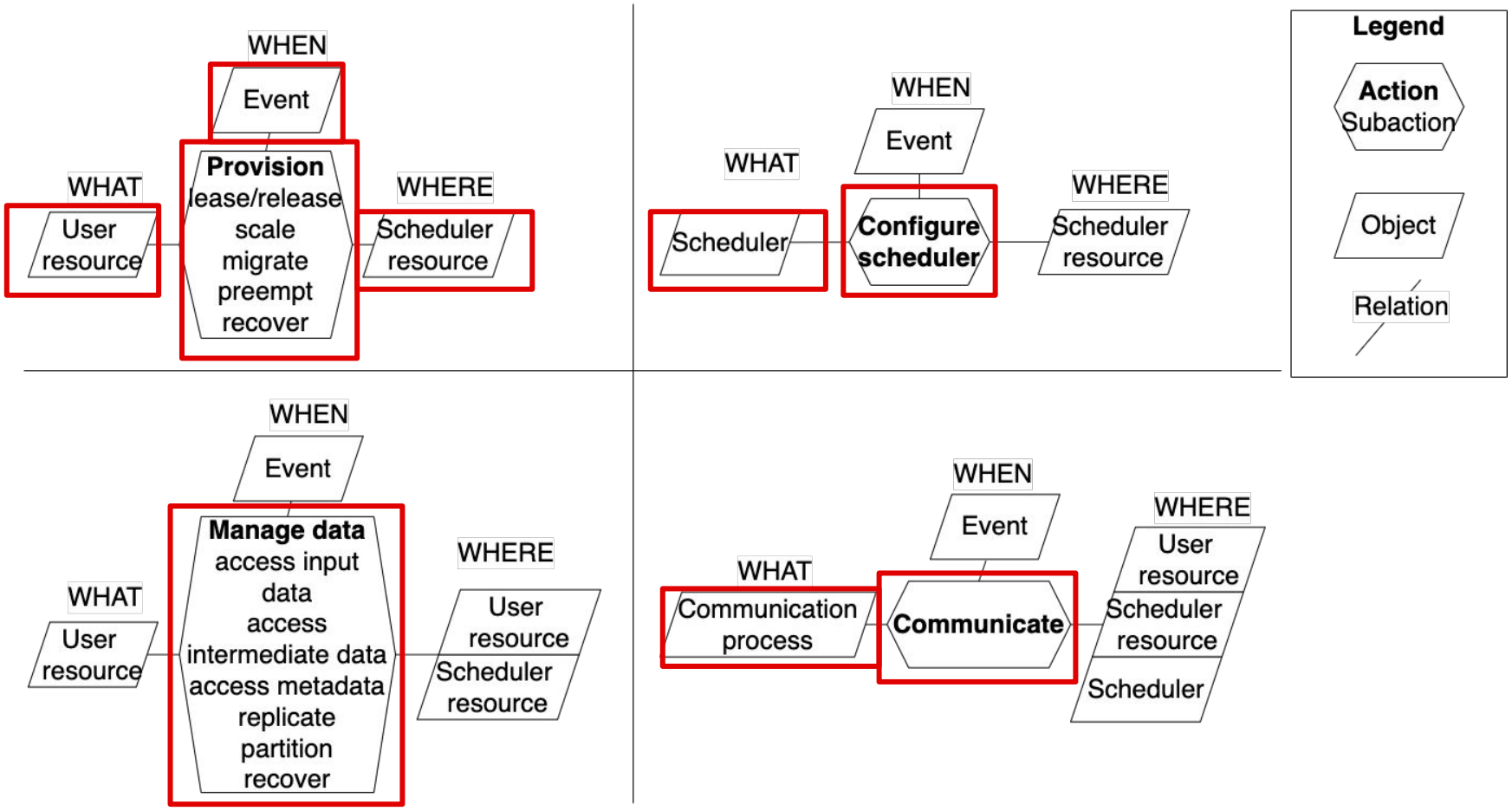
Research questions

1. How can we model scheduler programming abstractions?

2. What programming abstractions of scheduling are missing in mainstream industrial schedulers?

3. What is the performance cost of not implementing the missing programming abstractions for schedulers?

1. Reference architecture - Visual diagram



1. Reference architecture - Syntactic structure

<action> ***<object>*** **IN** ***<object>*** **WHEN** ***<object:Event>***

1. Provision:Lease UserResource<type: app, id:21, runtime-estimate: 5 days>
   IN SchedulerResource<type: vm, cpu:2.4Ghz, memory:16Gb>
   WHEN Event<day: 31, month: 12, year: 2022, hour: 00, minute: 00>

2. Provision:Scale UserResource<type: app, id: 21>
   IN SchedulerResource<type: vm, cpu:2.4Ghz, memory:16Gb>
   WHEN Event<cpu.utilization: > 80%>

3. Communicate CommunicationProcess<type: message>
   IN SchedulerResource<type: vm, id: 21>
   WHEN Event<state: failed>

2. Reference architecture - Map industrial schedulers

Action	sub-action	Schedulers				
		Kubernetes	SLURM	Spark	Condor	Airflow
Provision	lease / release	✓	✓	✓	✓	✓
	scale	✓		~		
	migrate					
	preempt	~	✓		✓	
	recover	~	~	✓	~	~
Configure scheduler		✓	~	✓	✓	✓
Manage data	access input data	✓	~	✓	✓	✓
	access intermediate data			~		
	access metadata					
	replicate			✓		
	partition			✓		
	recover	~		✓	✓	
Communicate		~	✓	~	~	~

Legend: ✓/ ~ /() = full/partial/no match.

Action	Object	Schedulers				
		Kubernetes	SLURM	Spark	Condor	Airflow
Provision	user resource	✓	~	~	~	~
	event	~	~	~	~	~
	scheduler resource	✓	✓	~	✓	✓
Configure scheduler	scheduler	~	~	✓	✓	✓
	event	~	~			
	scheduler resource	~				
Manage data	user resource	~	~	✓	~	✓
	event					
	scheduler resource		~		~	
Communicate	communication process	~	~	~	~	✓
	event	~	~	~		~
	user resource	~	~	✓	~	✓
	scheduler resource		~	✓		
	scheduler		~			

Legend: ✓/~ /() = $full/partial/no-match$.

3. Experiment - Performance cost analysis - How?

1. **Select** scheduler & missing abstraction
2. **Implement** the missing abstraction
3. **Design** a scenario where the abstraction is used 
4. **Evaluate** performance through experimentation

3. Performance cost analysis - Setup

- Traces
  - **Bitbrains** - VMs of Dutch ICT
  - **Azure** - VMs of data center
  - **Google** - single-core tasks of data center
- Experiments using **OpenDC**
  - **Open-source** data center discrete event **simulator** developed by **AtLarge**

Workload	VMs	Duration [days]	VM duration [days]		CPU cores		CPU capacity [GHz]		Memory [GBs]	
			Mean	σ	Mean	σ	Mean	σ	Mean	σ
Bitbrains	1250	30	28	5	3.27	4.04	2.7	0.16	11.75	32.6
Azure	1829	30	2	6	2.48	2.28	2.5	0.0	5.8	10.16
Google	1000000	2.5	0.0375	0.083	1.0	0.0	1.68	2.08	0.17	0.2

Table 4: Characteristics of the traces of the experiments

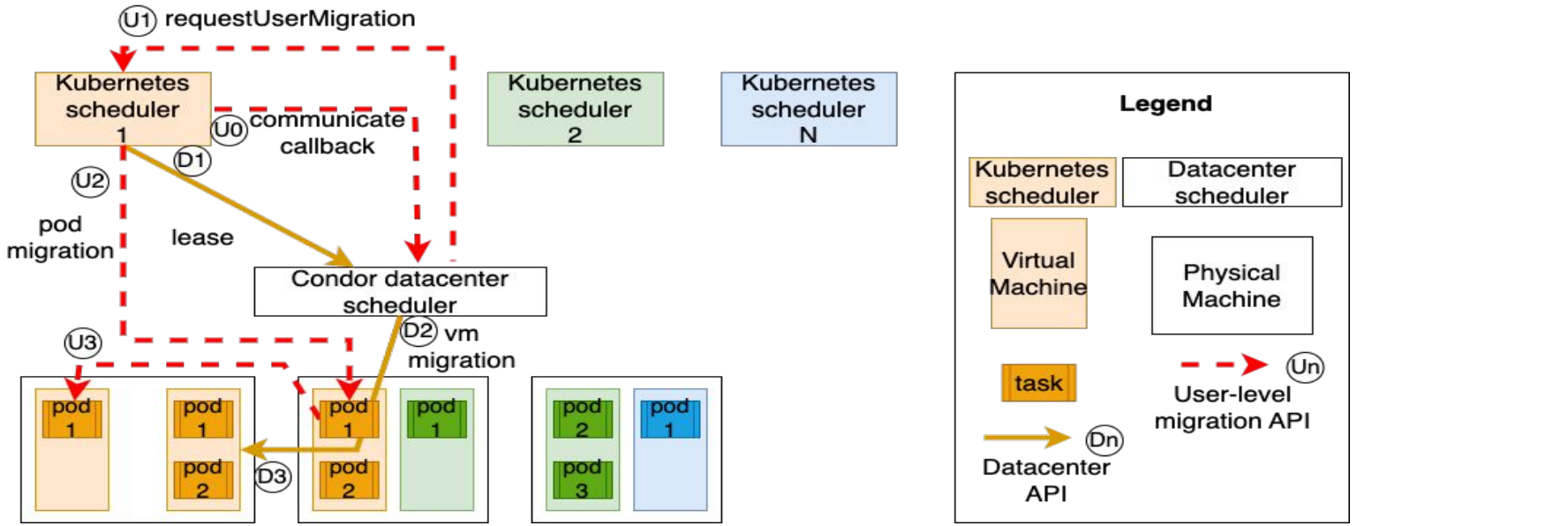
3. Experiment - System model

Scheduler: Condor
Missing API: Communicate callback
Experiment: Reducing total times using user-level migrations

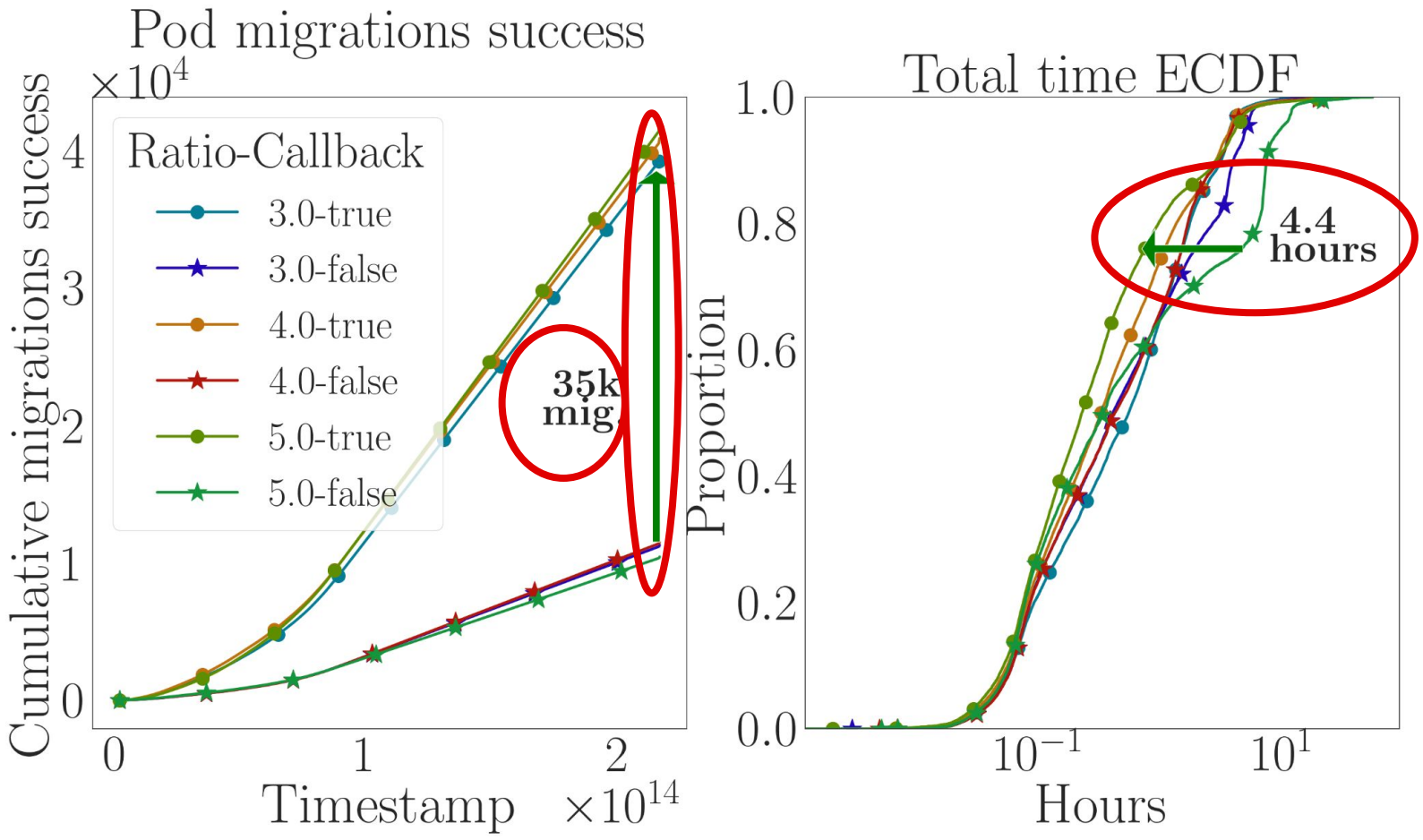
submit(callback, interferenceEvent)

+

requestUserMigration(vm, cpuCapacity)

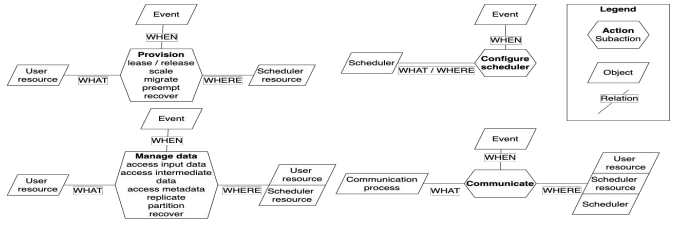


3. Experiment - Result highlights (Google)



Key takeaways

1



Model APIs → Reference Architecture

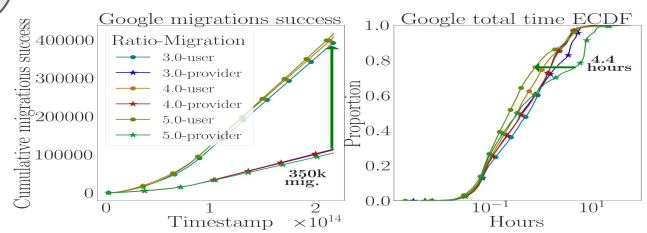
2

Action	Subaction	Schedulers				
		Kubernetes	SLURM	Spark	Condor	Airflow
Provision	lease / release	✓	✓	✓	✓	✓
	scale	✓		~		
	migrate					
	preempt	~	✓		✓	
	recover	✓	~	✓	~	~
Configure scheduler		✓	~	✓	✓	✓
Manage data	access input data	✓	~	✓	✓	✓
	access intermediate data			~		
	access metadata					
	replicate					
	partition			✓		
	recover	~		✓	✓	
Communicate		~	✓	~	~	~

Legend: ✓ / ~ / (_) = full/partial/no match.

Industrial APIs missing abstractions

3



Experimentation → existing APIs sacrifice performance

MASSIVIZING COMPUTER SYSTEMS

FURTHER READING

https://atlarge-research.com/publications.html

1. Manterola Lasa, Aratz, Sacheendra Talluri, and Alexandru Iosup. "A Reference Architecture for Datacenter Scheduler Programming Abstractions: Design and Experiments (Work In Progress Paper)." Companion of the 2023 ACM/SPEC International Conference on Performance Engineering. 2023.
2. Iosup, Alexandru, et al. "Massivizing computer systems: a vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems." 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018.
3. Iosup, Alexandru, et al. "Future Computer Systems and Networking Research in the Netherlands: A Manifesto." arXiv preprint arXiv:2206.03259 (2022).
4. Andreadis, Georgios, Laurens Versluis, Fabian Mastenbroek, and Alexandru Iosup. "A reference architecture for datacenter scheduling: design, validation, and experiments." SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2018