QUASAR: RESOURCE-EFFICIENT AND QOS-AWARE CLUSTER MANAGEMENT

Christina Delimitrou and Christos Kozyrakis

Stanford University

http://mast.stanford.edu

ASPLOS – March 3rd 2014

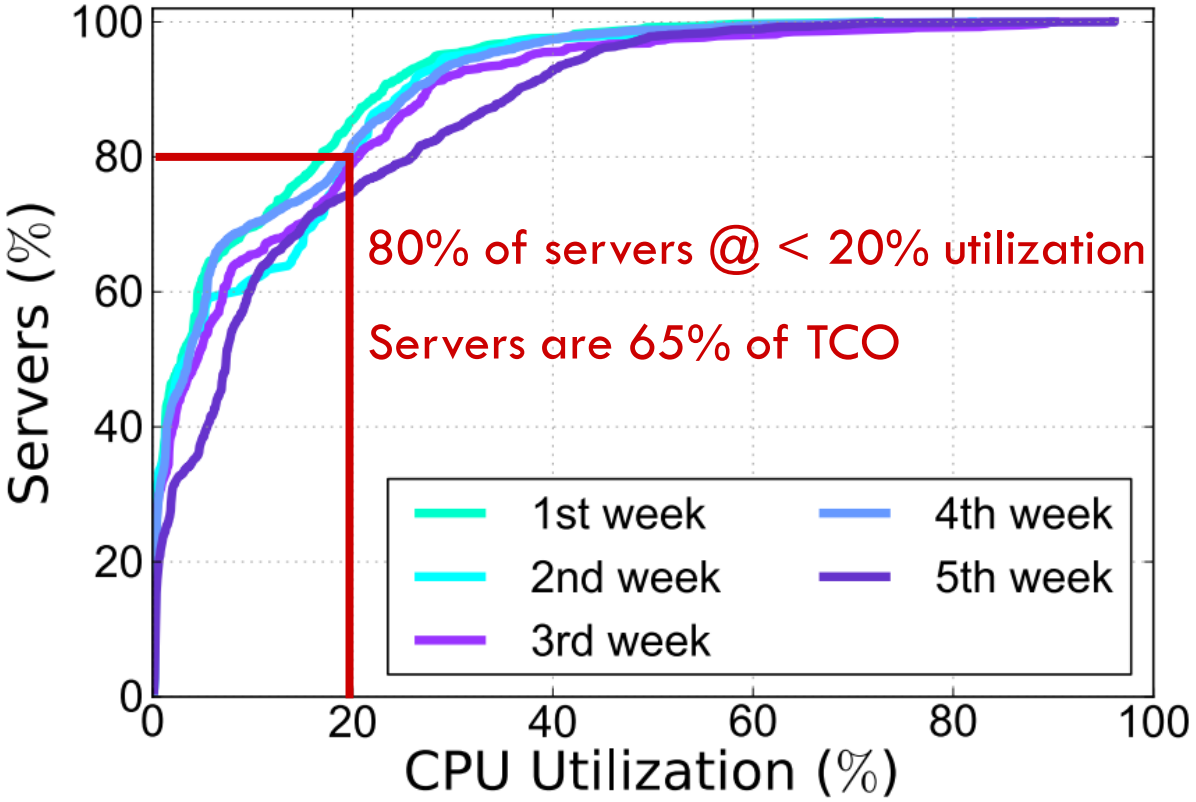
Executive Summary

- Problem: low datacenter utilization
  - Overprovisioned reservations by users
- Problem: high jitter on application performance
  - Interference, HW heterogeneity

- Quasar: resource-efficient cluster management
  - User provides ~~resource reservations~~ performance goals
  - Online analysis of resource needs using info from past apps
  - Automatic selection of number & type of resources
  - High utilization and low performance jitter

Datacenter Underutilization

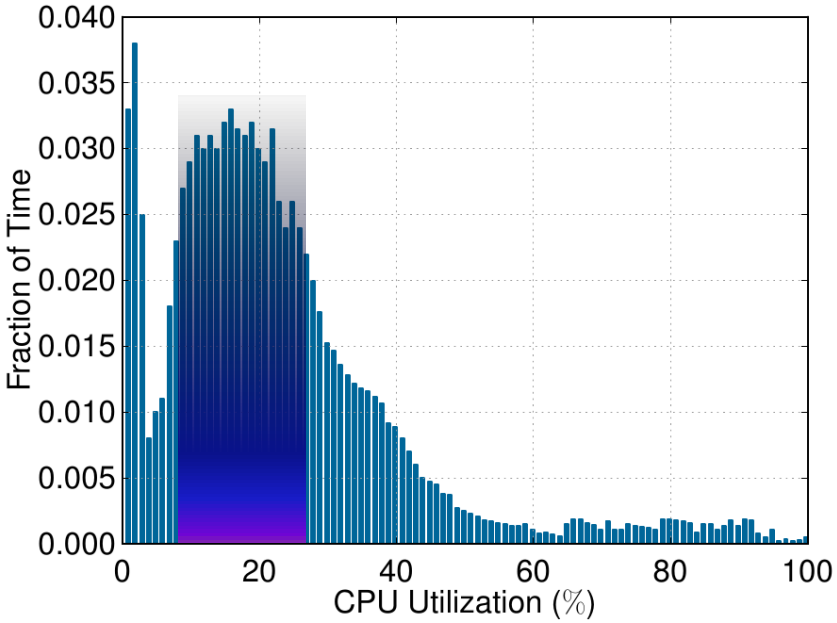
- A few thousand server cluster at Twitter managed by Mesos
- Running mostly latency-critical, user-facing apps



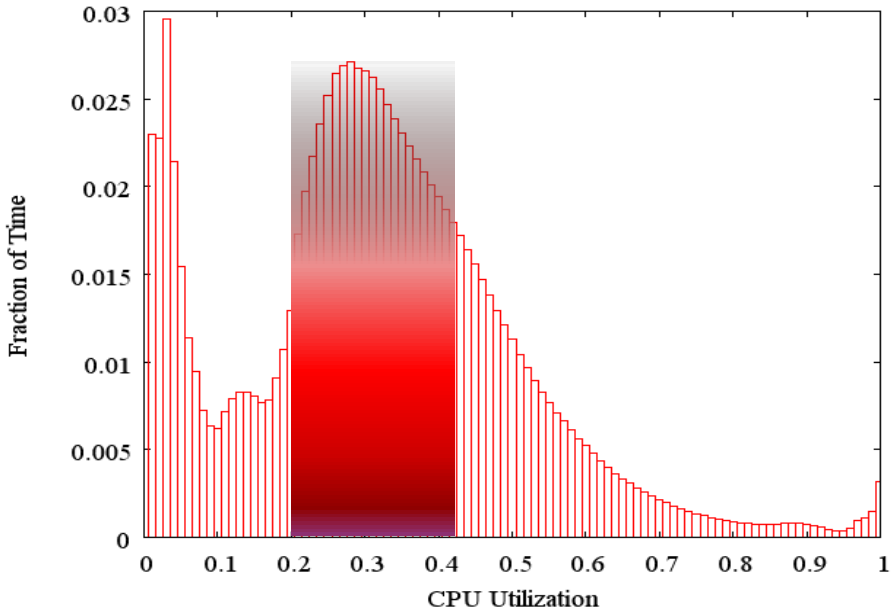
Datacenter Underutilization

- Goal: raise utilization without introducing performance jitter

Twitter

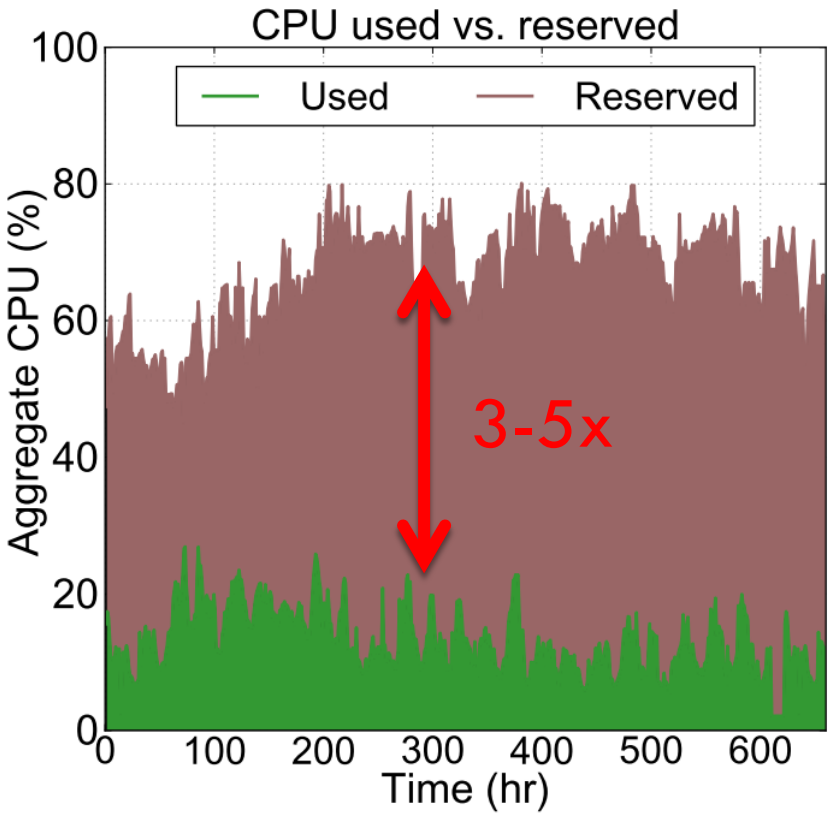


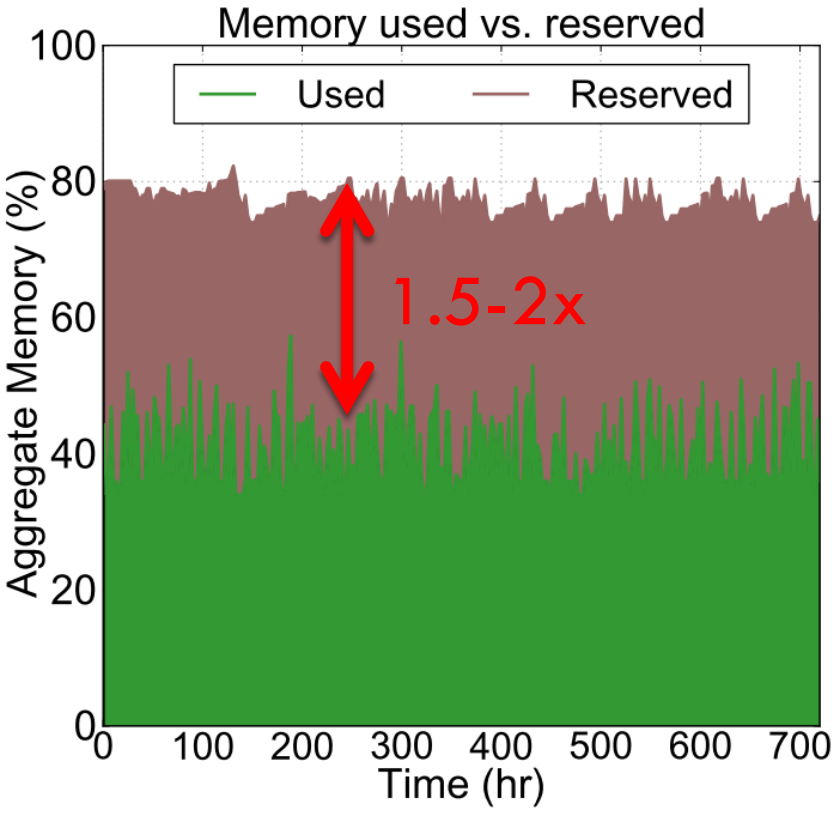
Google[1]



[1] L. A. Barroso, U. Holzle. The Datacenter as a Computer, 2009.

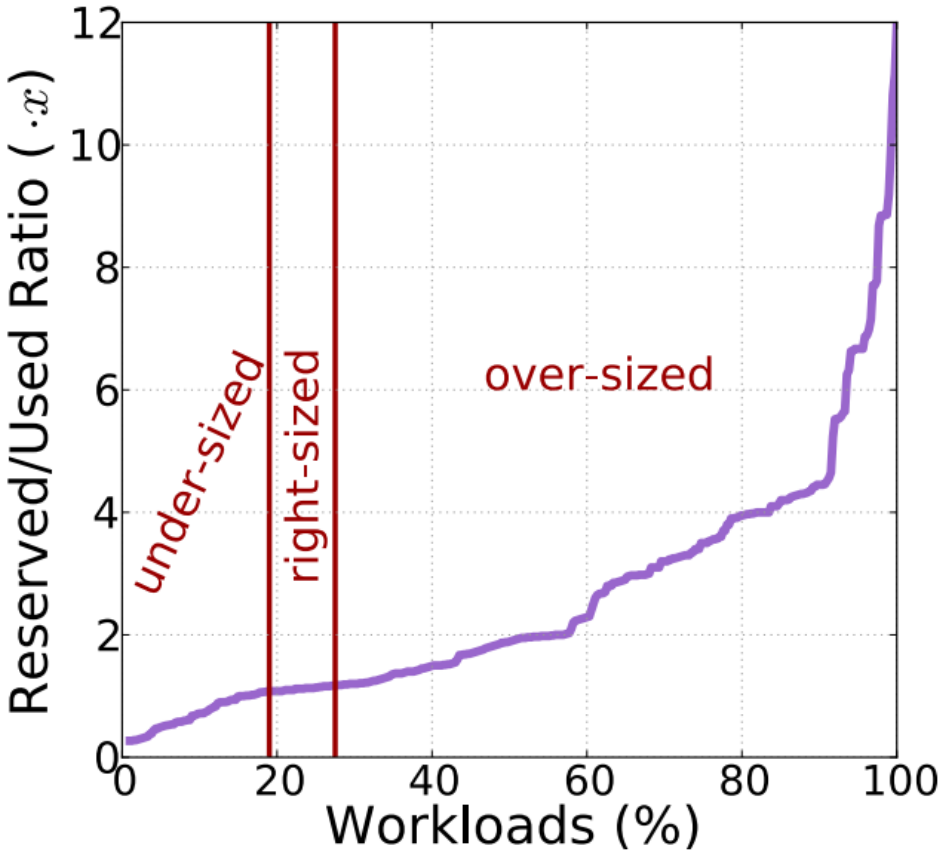
Reserved vs. Used Resources





- Twitter: up to 5x CPU & up to 2x memory overprovisioning

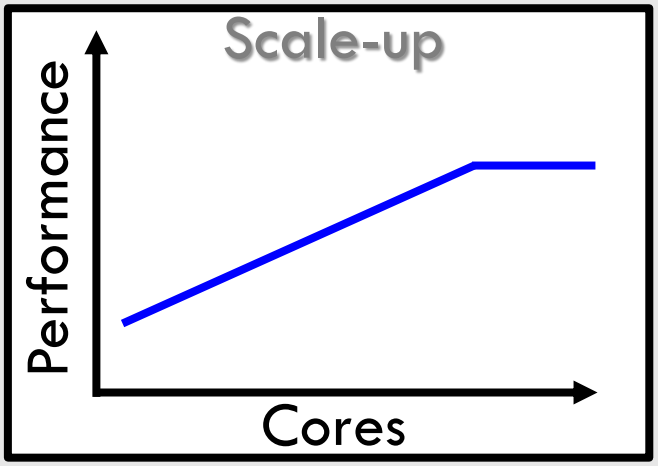
Reserved vs. Used Resources



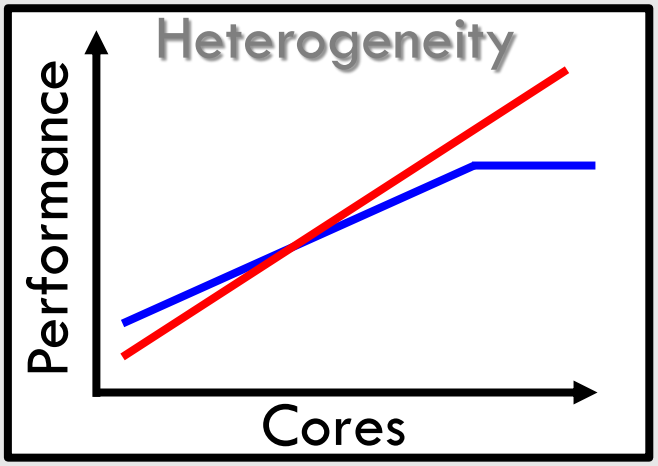
- 20% of job under-sized, **~70% of jobs over-sized**

Rightsizing Applications is Hard

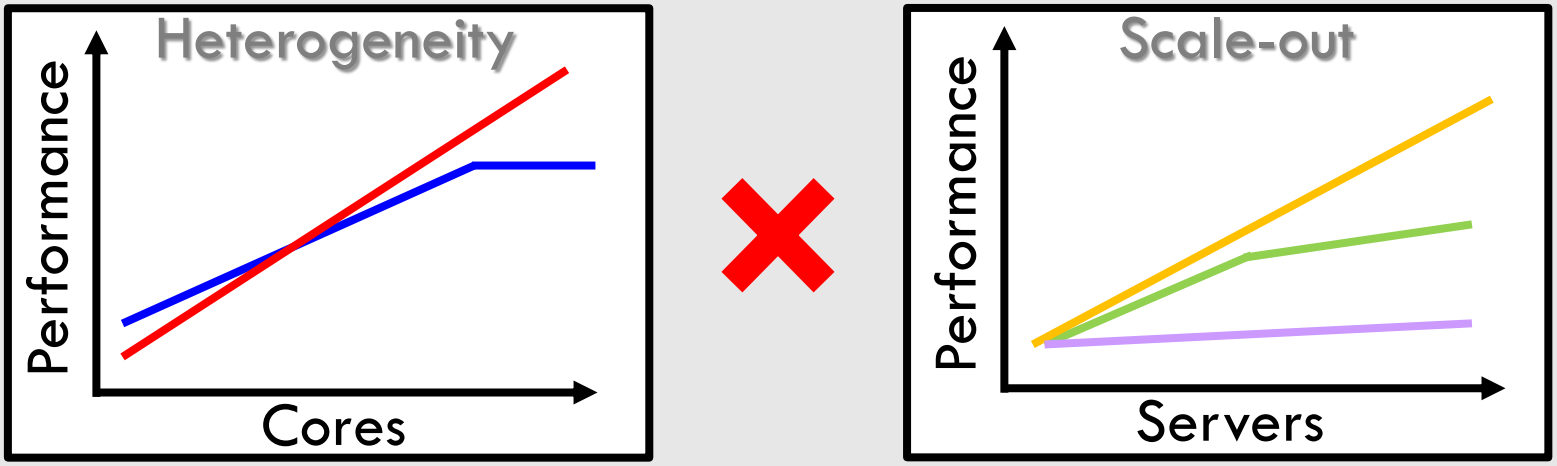
Performance Depends on



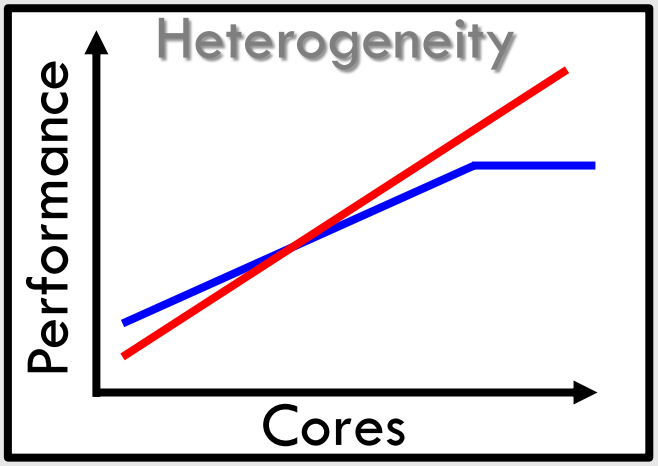
Performance Depends on

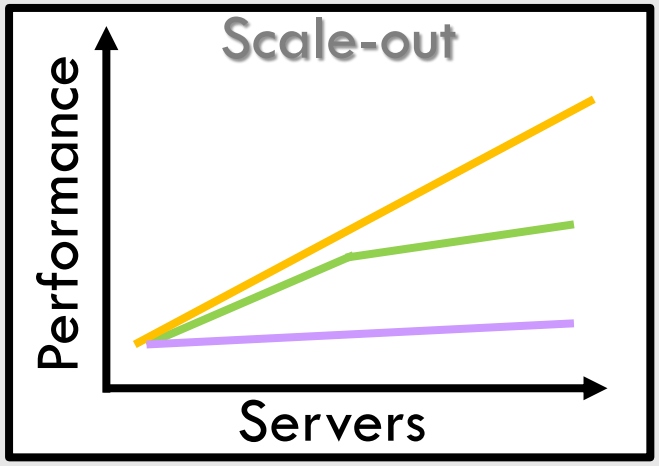


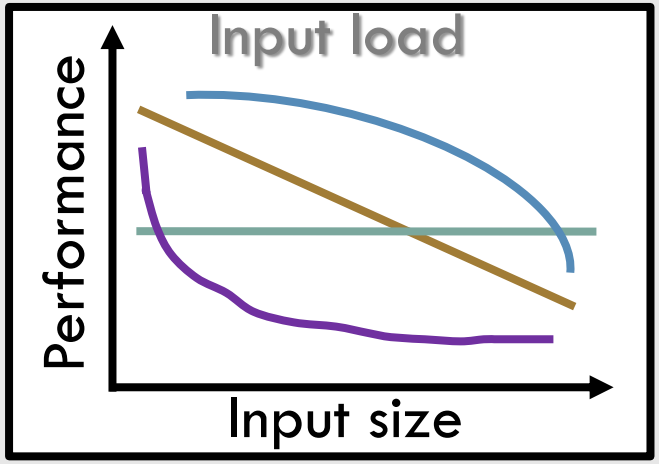
Performance Depends on



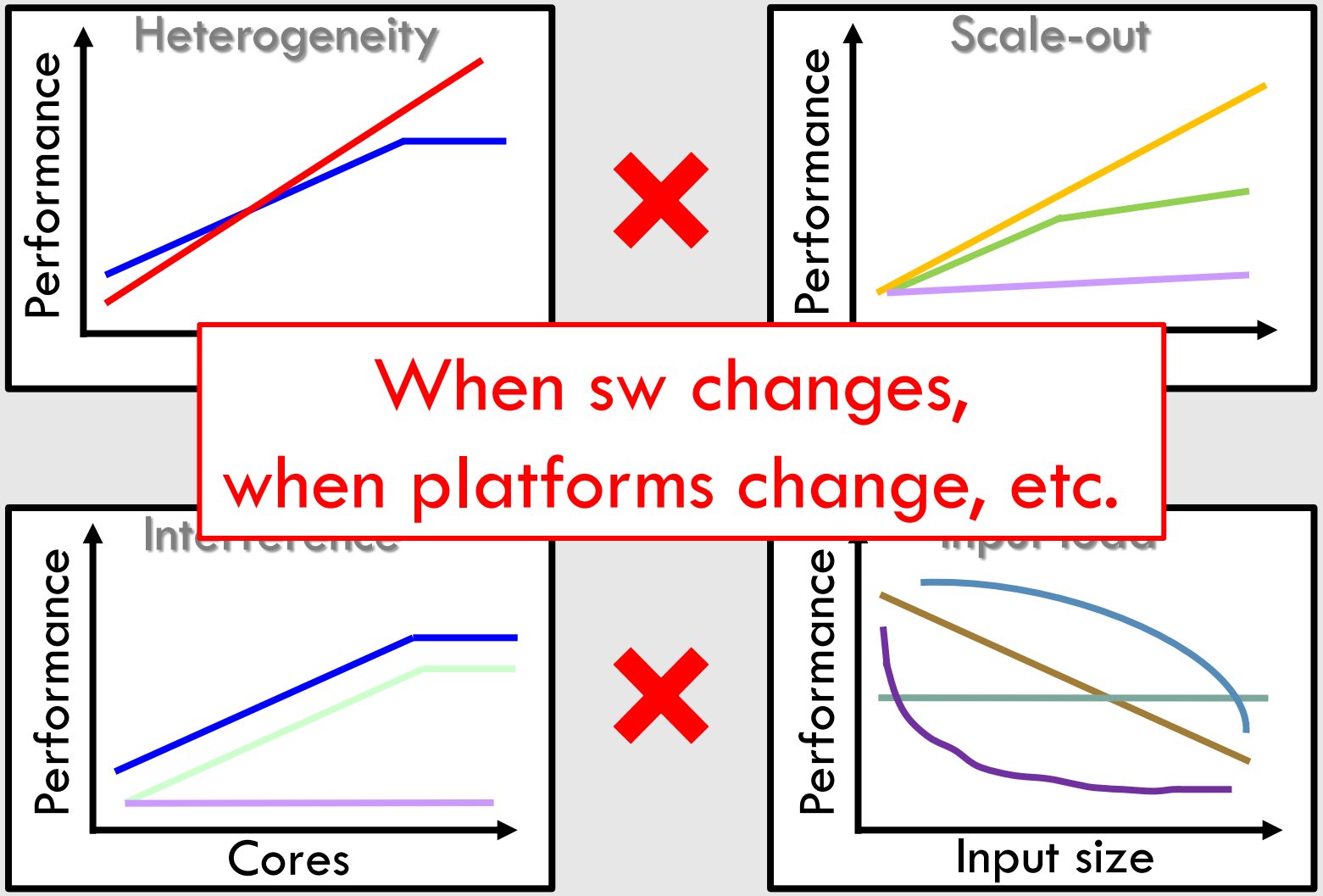
Performance Depends on



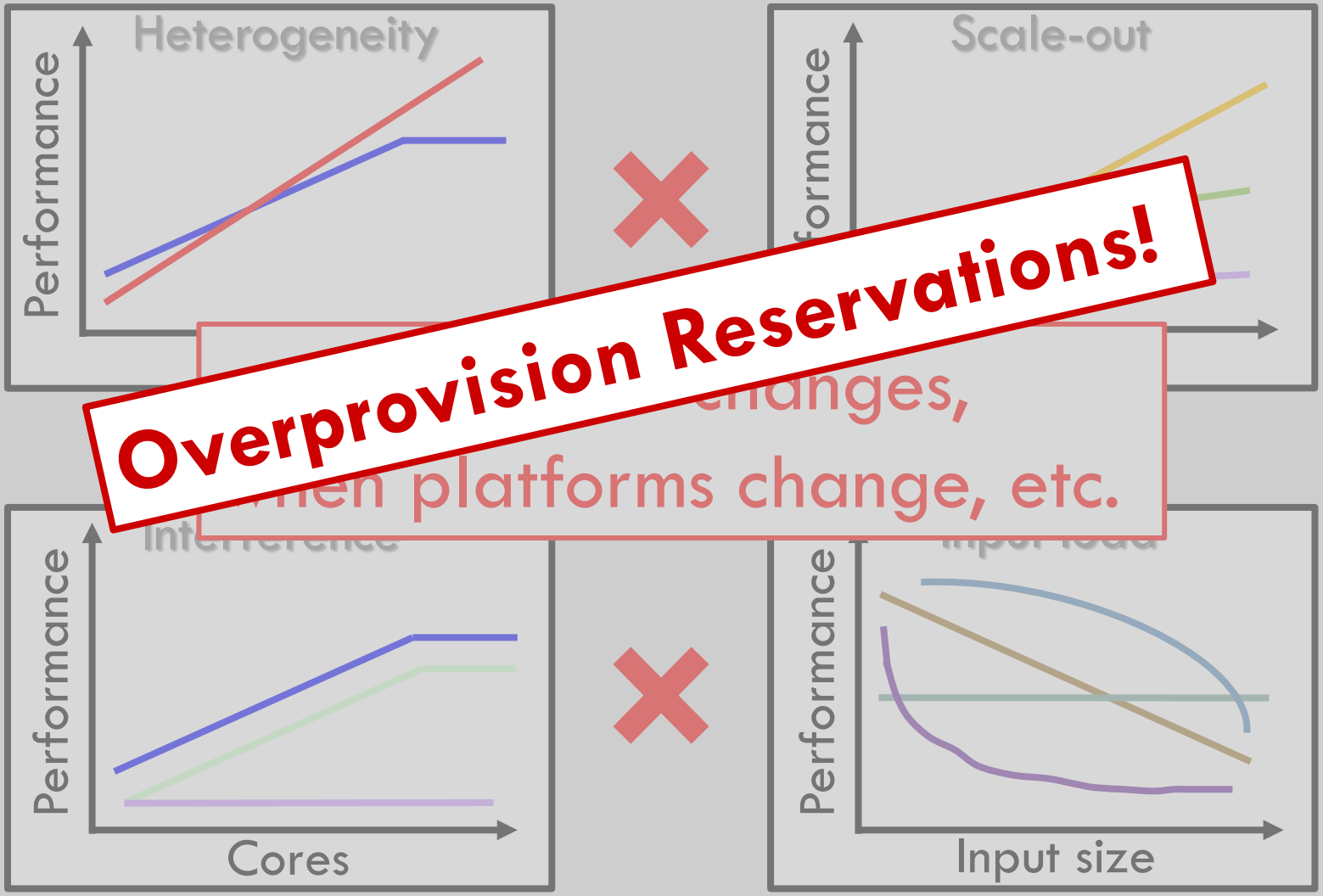




Performance Depends on



Performance Depends on

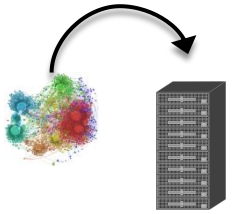


Rethinking Cluster Management

- User provides ~~resource reservations~~ performance goals
- Joint allocation and assignment of resources
  - Right amount depends on quality of available resources
  - Monitor and adjust dynamically as needed
- **But wait…**
  - **The manager must know the resource/performance tradeoffs**

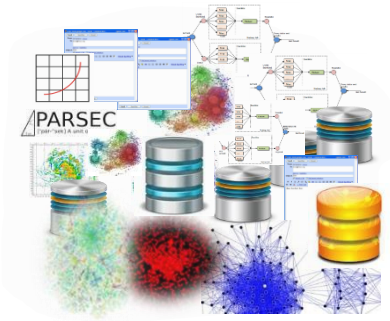
Understanding Resource/Performance Tradeoffs

- Combine:
  - Small signal from short run of new app
  - Large signal from previously-run apps
- Generate:
  - Detailed insights for resource management
  - Performance vs scale-up/out, heterogeneity, …
- Looks like a classification problem



Small app signal

+

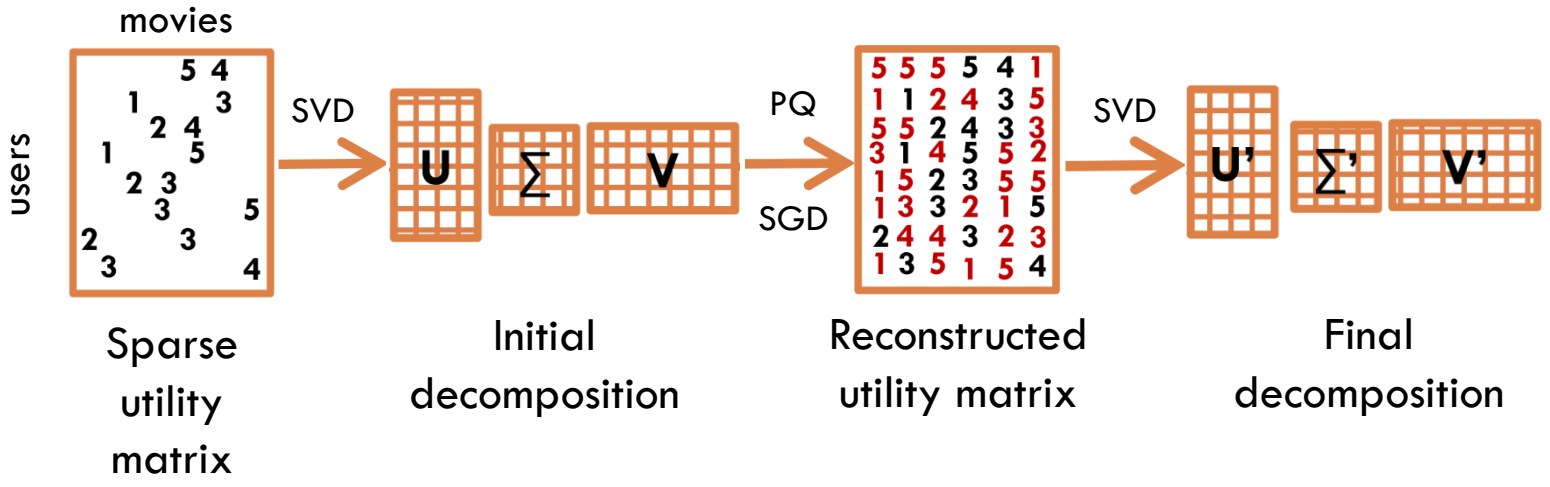


Big cluster data

Resource/performance tradeoffs

Something familiar…

- Collaborative filtering – similar to Netflix Challenge system
  - Predict preferences of new users given preferences of other users
  - Singular Value Decomposition (SVD) + PQ reconstruction (SGD)
  - High accuracy, low complexity, relaxed density constraints



Application Analysis with Classification

	Rows	Columns	Recommendation
Netflix	Users	Movies	Movie ratings
Heterogeneity			
Interference			
Scale-up			
Scale-out			

- 4 parallel classifications
  - Lower overheads & similar accuracy to exhaustive classification

Heterogeneity Classification

	Rows	Columns	Recommendation
Netflix	Users	Movies	Movie ratings
Heterogeneity	Apps	Platforms	Server type
Interference			
Scale-up			
Scale-out			

- Profiling on two randomly selected server types
- Predict performance on each server type

Interference Classification

	Rows	Columns	Recommendation
Netflix	Users	Movies	Movie ratings
Heterogeneity	Apps	Platforms	Server type
Interference	Apps	Sources of interference	Interference sensitivity
Scale-up			
Scale-out			

- Predict sensitivity to interference
  - Interference intensity that leads to >5% performance loss
- Profiling by injecting increasing interference

Scale-Up Classification

	Rows	Columns	Recommendation
Netflix	Users	Movies	Movie ratings
Heterogeneity	Apps	Platforms	Server type
Interference	Apps	Sources of interference	Interference sensitivity
Scale-up	Apps	Resource vectors	Resources/node
Scale-out			

- Predict speedup from scale-up
- Profiling with two allocations (cores & memory)

Scale-Out Classification

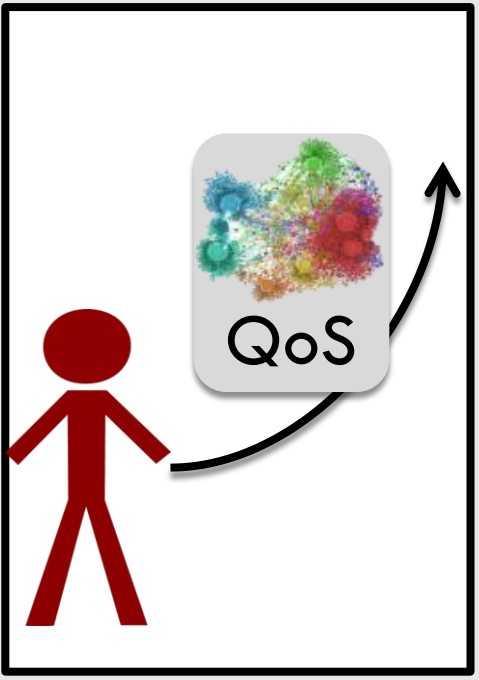
	Rows	Columns	Recommendation
Netflix	Users	Movies	Movie ratings
Heterogeneity	Apps	Platforms	Server type
Interference	Apps	Sources of interference	Interference sensitivity
Scale-up	Apps	Resource vectors	Resources/node
Scale-out	Apps	Nodes	Number of nodes

- Predict speedup from scale-out
- Profiling with two allocations (1 & N>1 nodes)

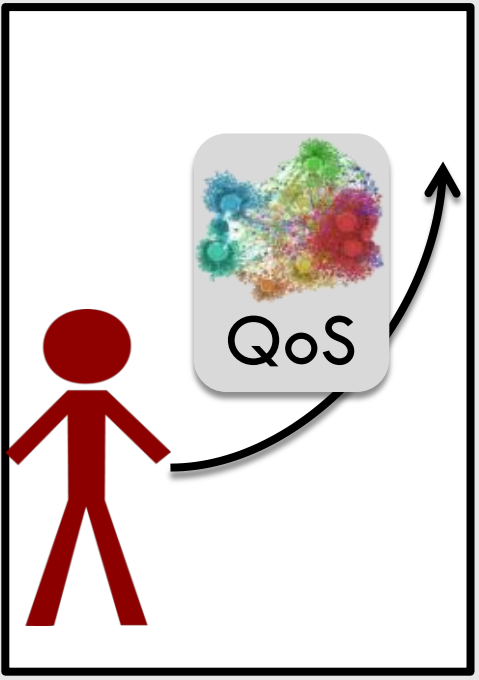
Classification Validation

	Heterogeneity		Interference		Scale-up		Scale-out	
	avg	max	avg	max	avg	max	avg	max
Single-node	4%	8%	5%	10%	4%	9%	-	-
Batch distributed	4%	5%	2%	6%	5%	11%	5%	17%
Latency-critical	5%	6%	7%	10%	6%	11%	6%	12%

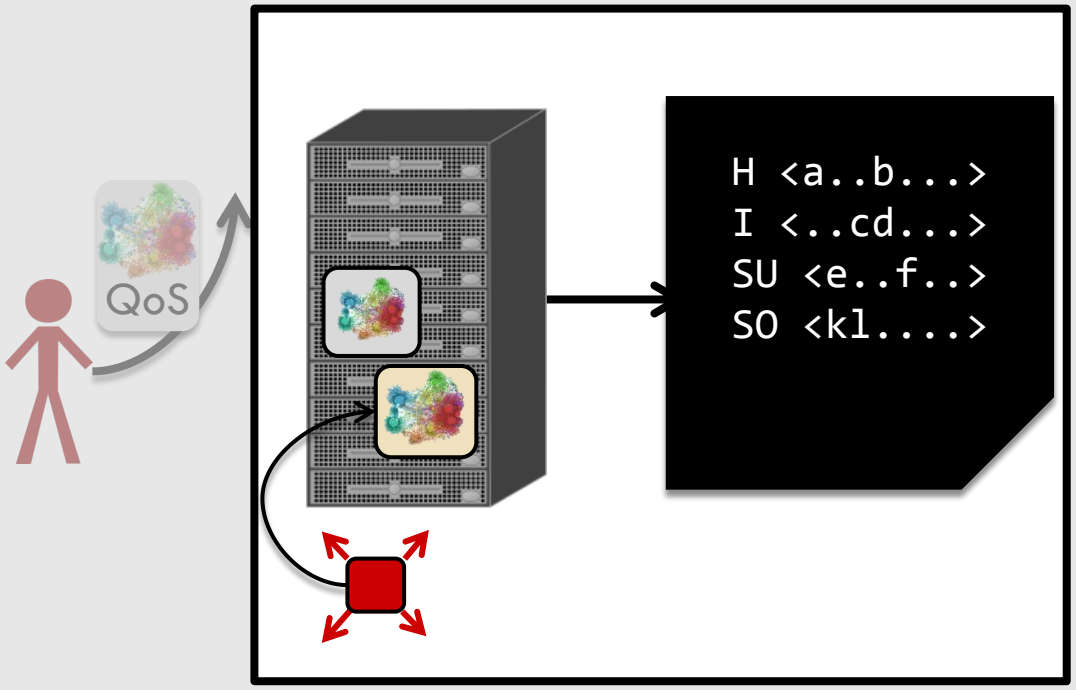
Quasar Overview



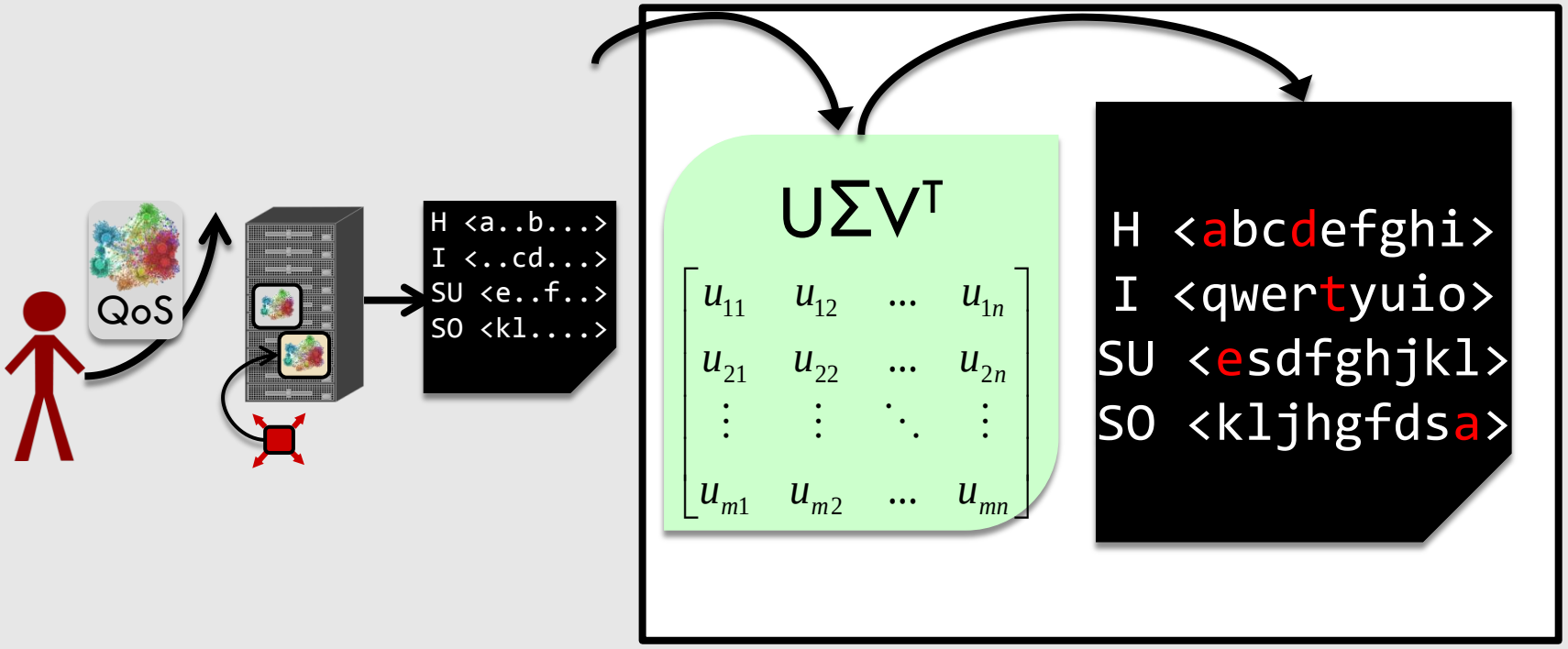
Quasar Overview



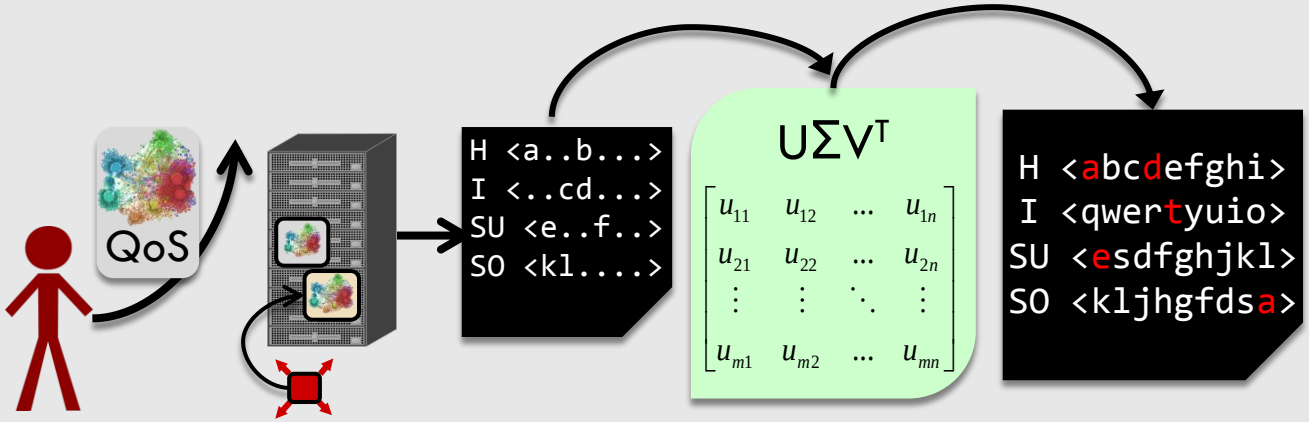
Quasar Overview



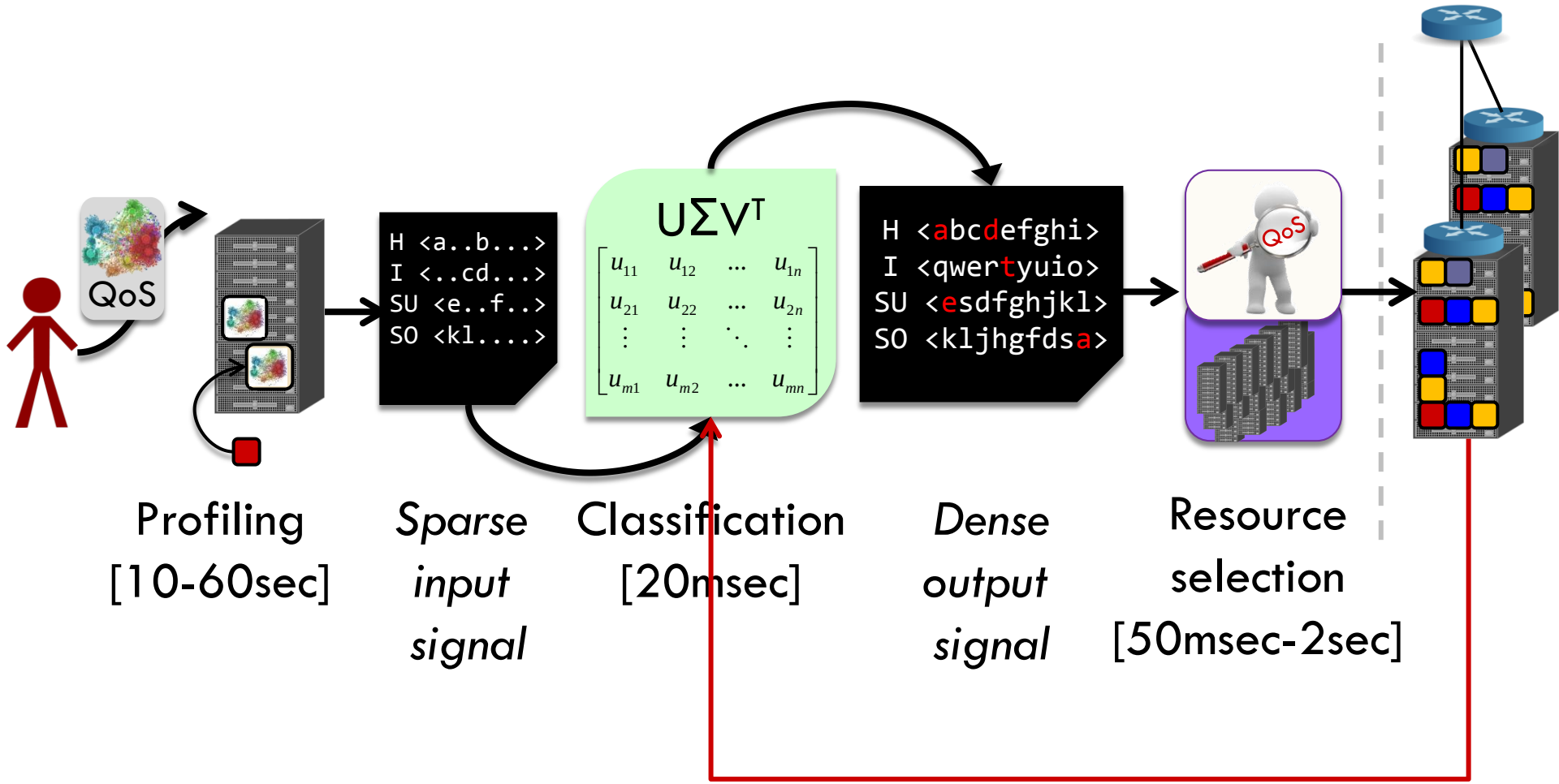
Quasar Overview



Quasar Overview



Quasar Overview



Greedy Resource Selection

- Goals
  - Allocate least needed resources to meet QoS target
  - Pack together non-interfering applications

- Overview
  - Start with most appropriate server types
  - Look for servers with interference below critical intensity
    - Depends on which applications are running on these servers
  - First scale-up, next scale-out

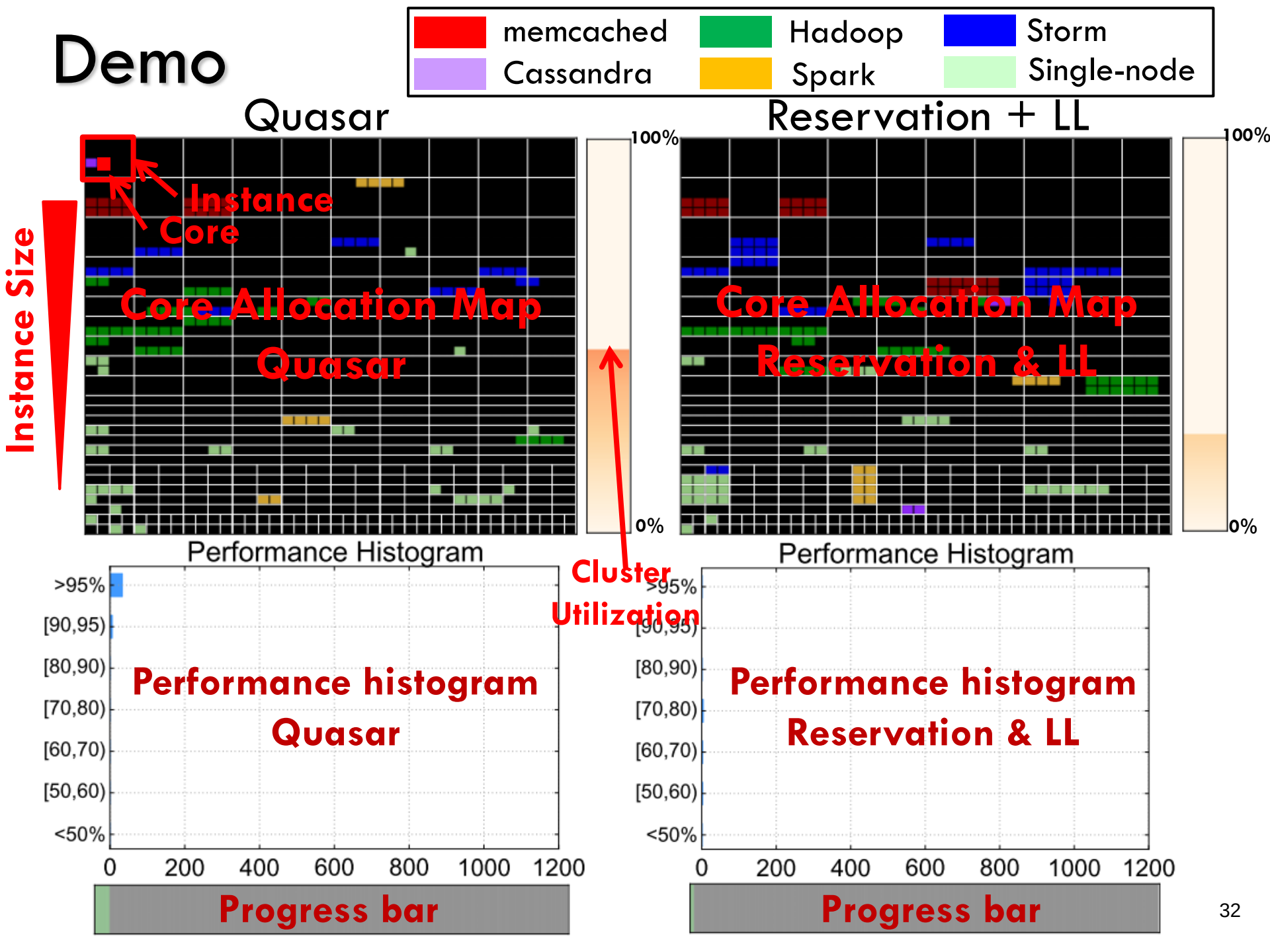
Quasar Implementation

- 6,000 loc of C++ and Python
- Runs on Linux and OS X
- Supports frameworks in C/C++, Java and Python
  - ~100-600 loc for framework-specific code
- Side-effect free profiling using Linux containers with chroot

Evaluation: Cloud Scenario

- Cluster
  - 200 EC2 servers, 14 different server types
- Workloads: 1,200 apps with 1sec inter-arrival rate
  - Analytics: Hadoop, Spark, Storm
  - Latency-critical: Memcached, HotCrp, Cassandra
  - Single-threaded: SPEC CPU2006
  - Multi-threaded: PARSEC, SPLASH-2, BioParallel, Specjbb
  - Multiprogrammed: 4-app mixes of SPEC CPU2006
- Objectives: high cluster utilization and good app QoS

Demo


memcached
Hadoop
Storm
Cassandra
Spark
Single-node
Quasar
Reservation + LL
Instance
Core
Instance Size
Core Allocation Map
Quasar
Core Allocation Map
Reservation & LL
100%
0%
Cluster
Utilization
Performance Histogram
>95%
[90,95)
[80,90)
[70,80)
[60,70)
[50,60)
<50%
0
200
400
600
800
1000
1200
Performance histogram
Quasar
Performance histogram
Reservation & LL
Progress bar
Progress bar

Demo

Cloud Scenario Summary

Quasar achieves:

- 88% of applications get >95% performance
- ~10% overprovisioning as opposed to up to 5x
- Up to 70% cluster utilization at steady-state
- 23% shorter scenario completion time

Conclusions

- Quasar: high utilization, high app performance
  - From reservation to performance-centric cluster management
  - Uses info from previous apps for accurate & online app analysis
  - Joint resource allocation and resource assignment

- See paper for:
  - Utilization analysis of Twitter cluster
  - Detailed validation & sensitivity analysis of classification
  - Further evaluation scenarios and features
    - E.g., setting framework parameters for Hadoop

Questions??

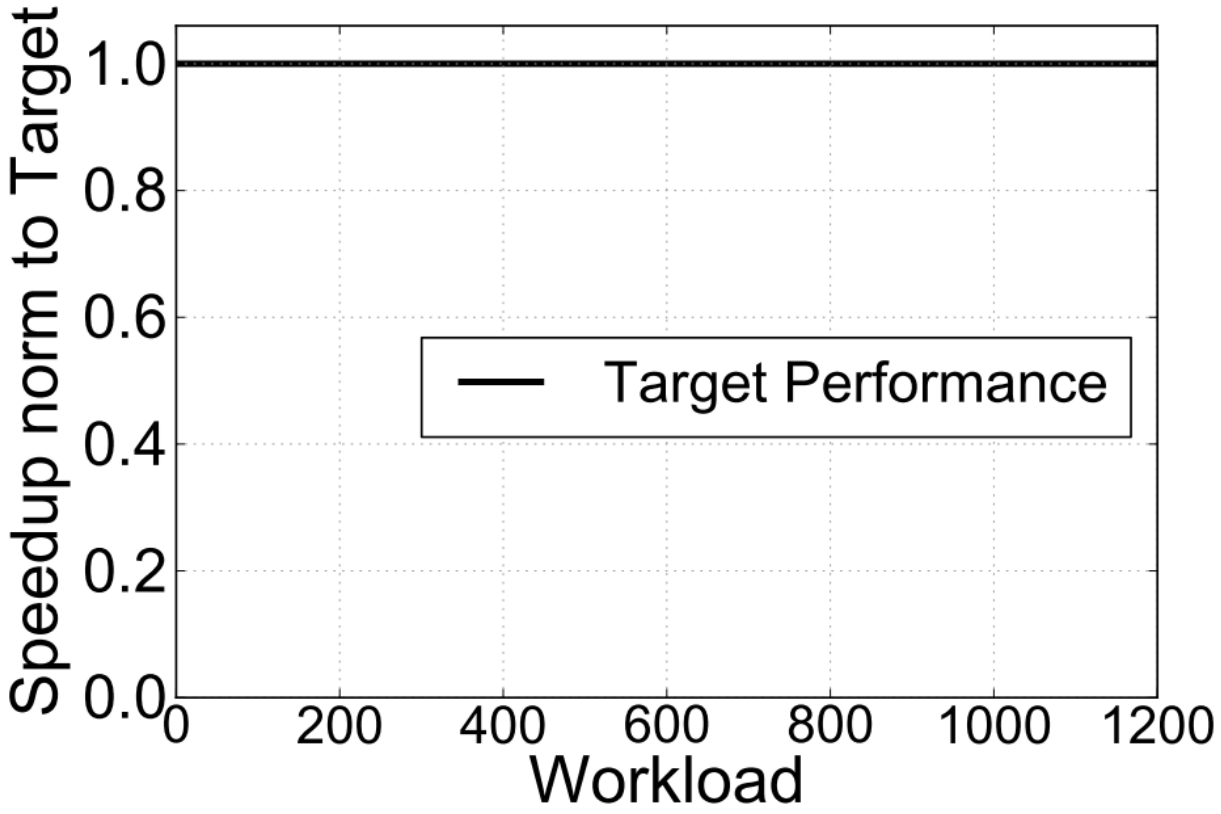
- Quasar: high utilization, high app performance
  - From reservation to performance-centric cluster management
  - Uses info from previous apps for accurate & online app analysis
  - Joint resource allocation and resource assignment

- See paper for:
  - Utilization analysis of Twitter cluster
  - Detailed validation & sensitivity analysis of classification
  - Further evaluation scenarios and features
    - E.g., setting framework parameters for Hadoop

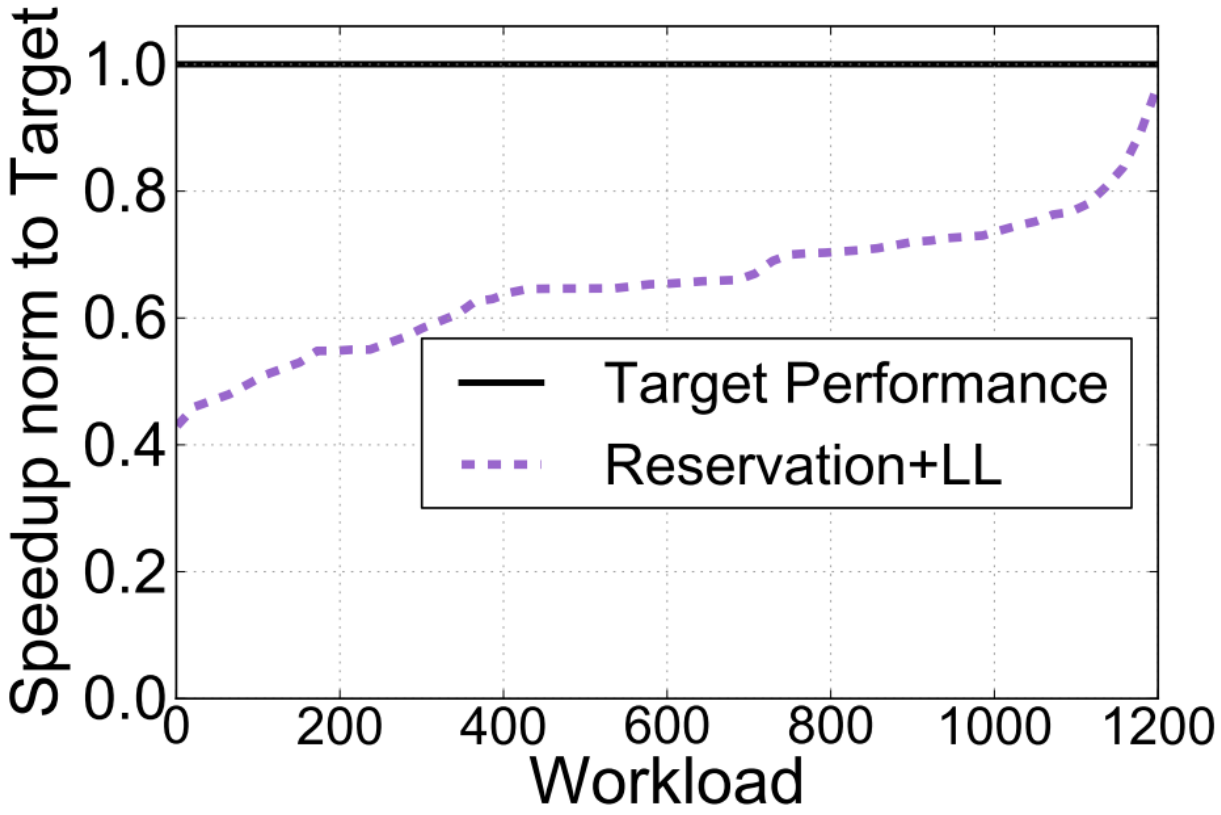
Questions??

Thank you

Cloud Provider: Performance

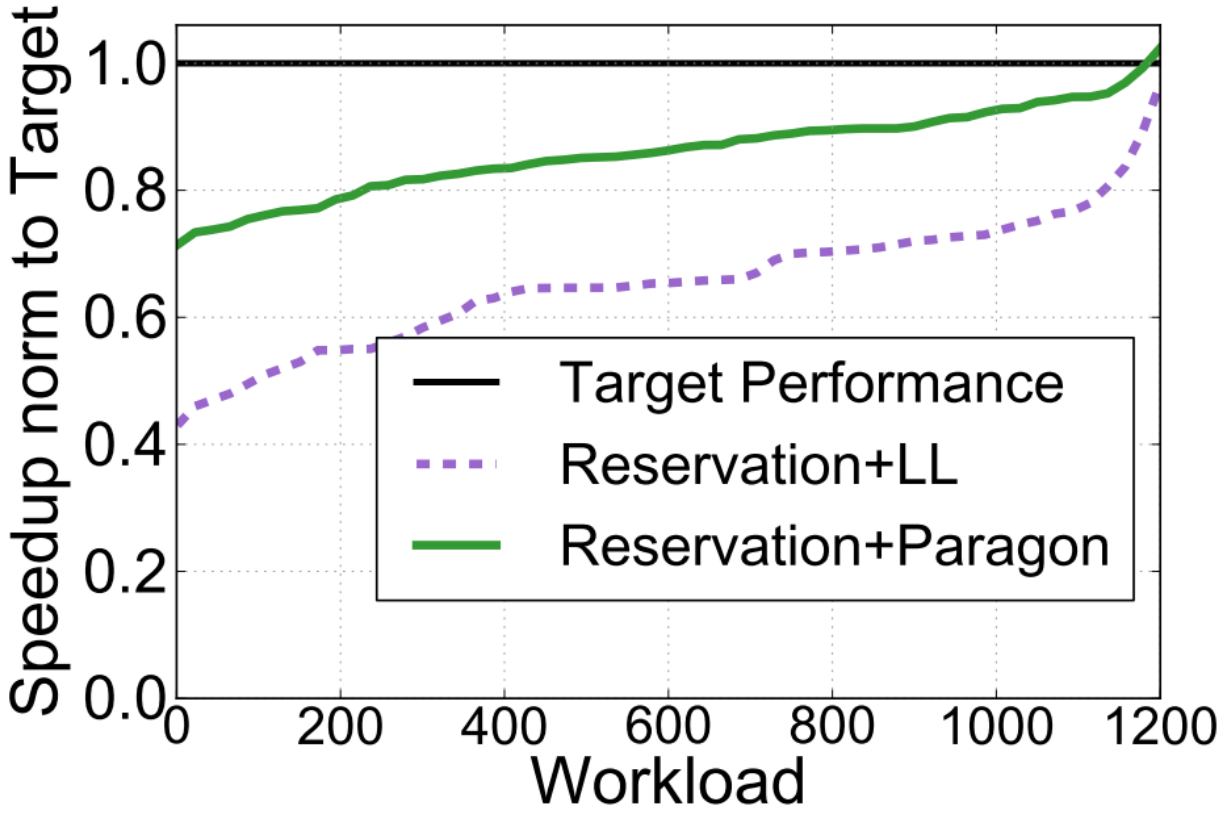


Cloud Provider: Performance



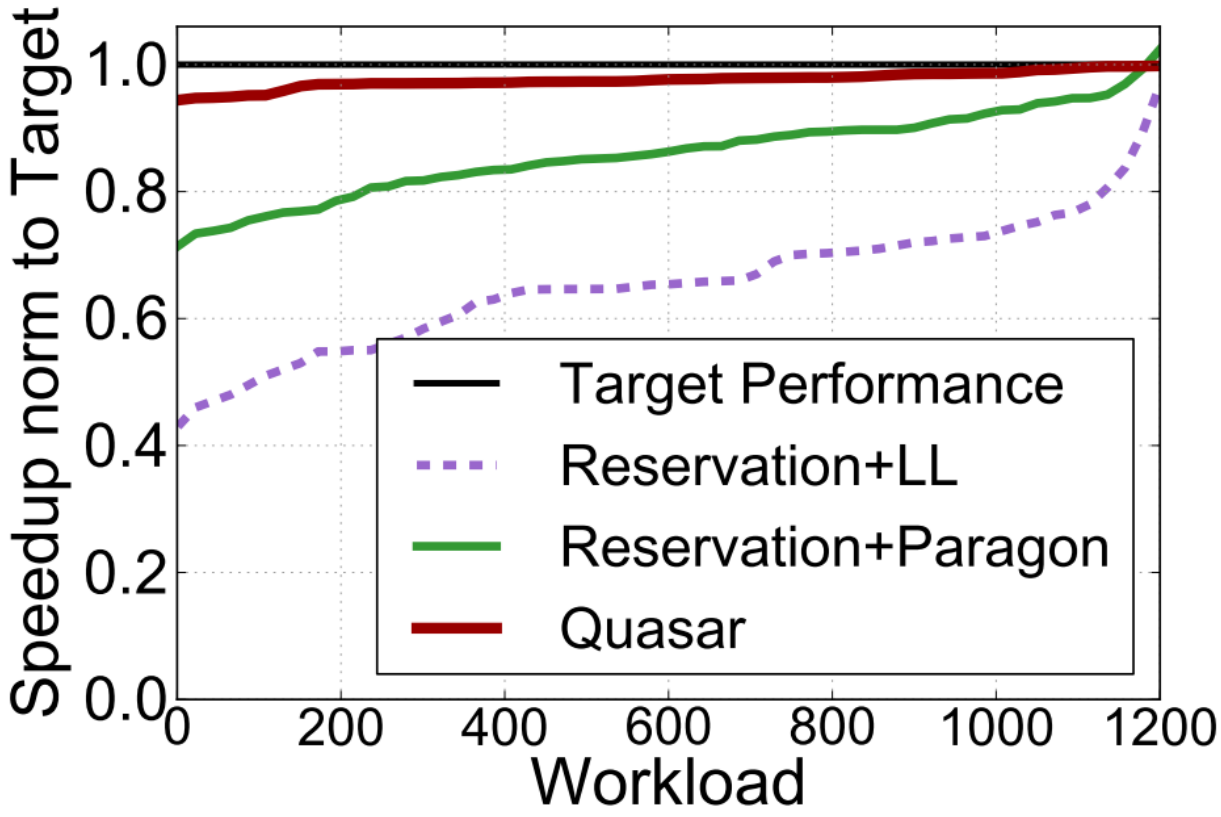
- Most applications violate their QoS constraints

Cloud Provider: Performance



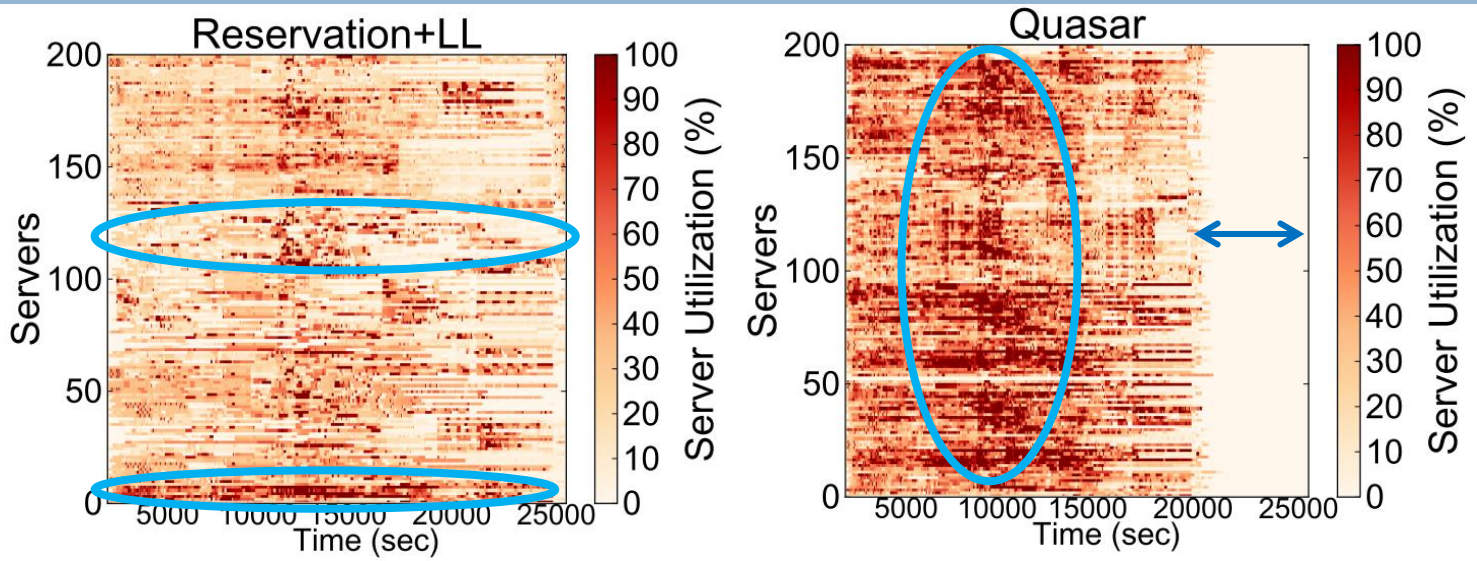
- 83% of performance target when only assignment is heterogeneity & interference aware

Cloud Provider: Performance



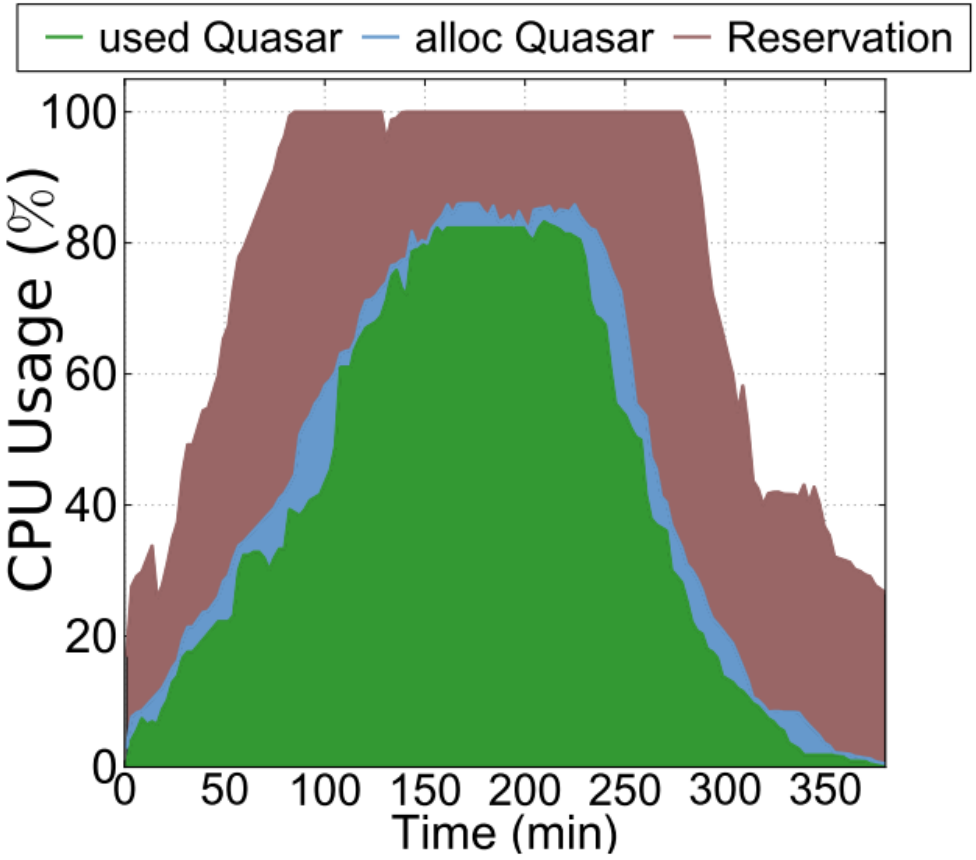
- 98% of performance target on average

Cluster Utilization



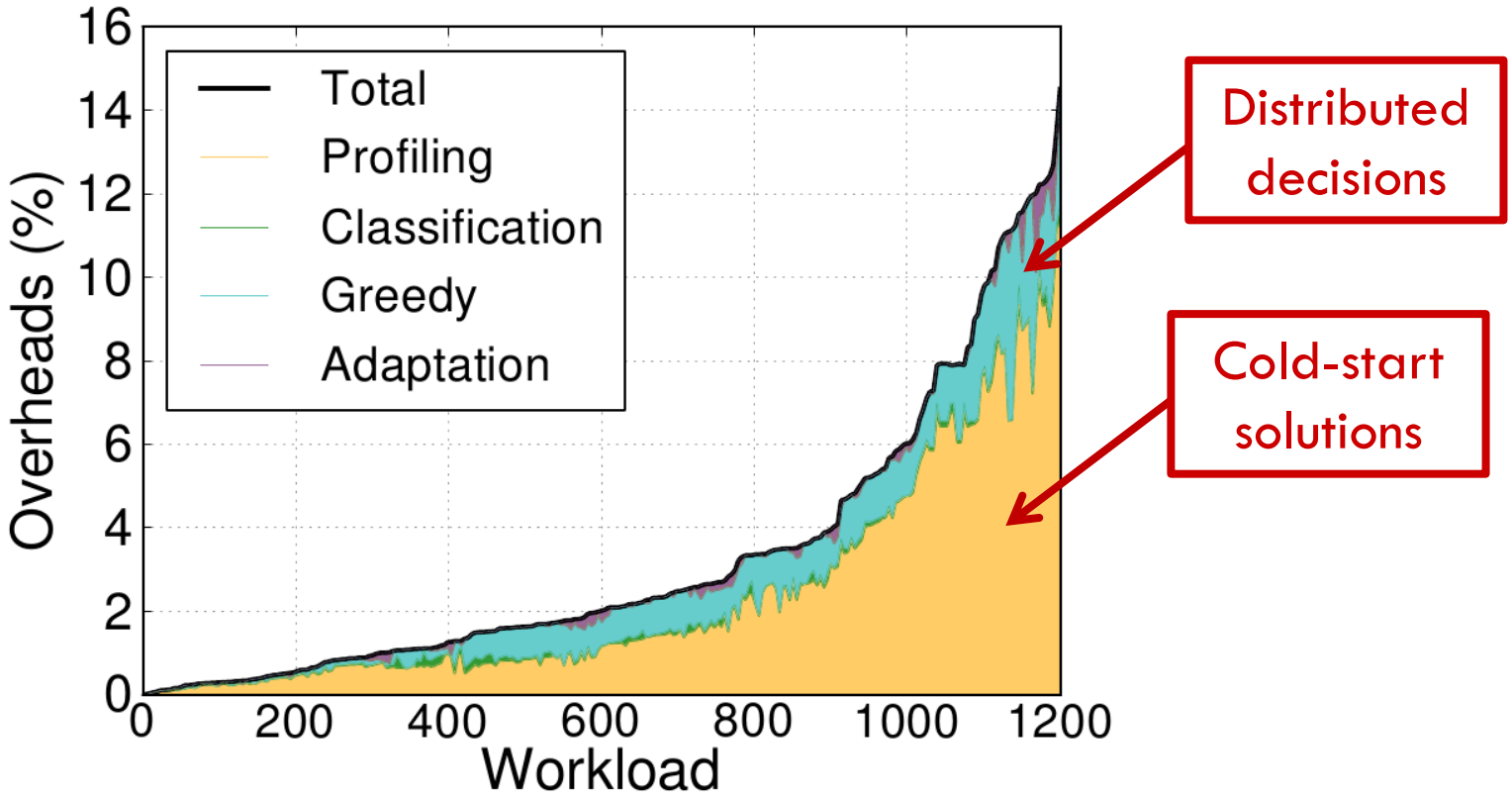
- Baseline (Reservation+LL):
  - Imbalance in server utilization
  - Per-app QoS violations + higher execution time
- Quasar increases server utilization by 47%
  - High performance for user
  - Better utilization for DC operator → resource efficiency

Reducing Overprovisioning



- ~10% overprovisioning, compared to 40%-5x for Reservation+LL

Scheduling Overheads



- 4.1% of execution time on average, up to 15% for short-lived workloads – mostly from profiling