CSL

PARTIES:
QoS-Aware Resource Partitioning for Multiple Interactive Services

Shuang Chen, Christina Delimitrou, José F. Martínez

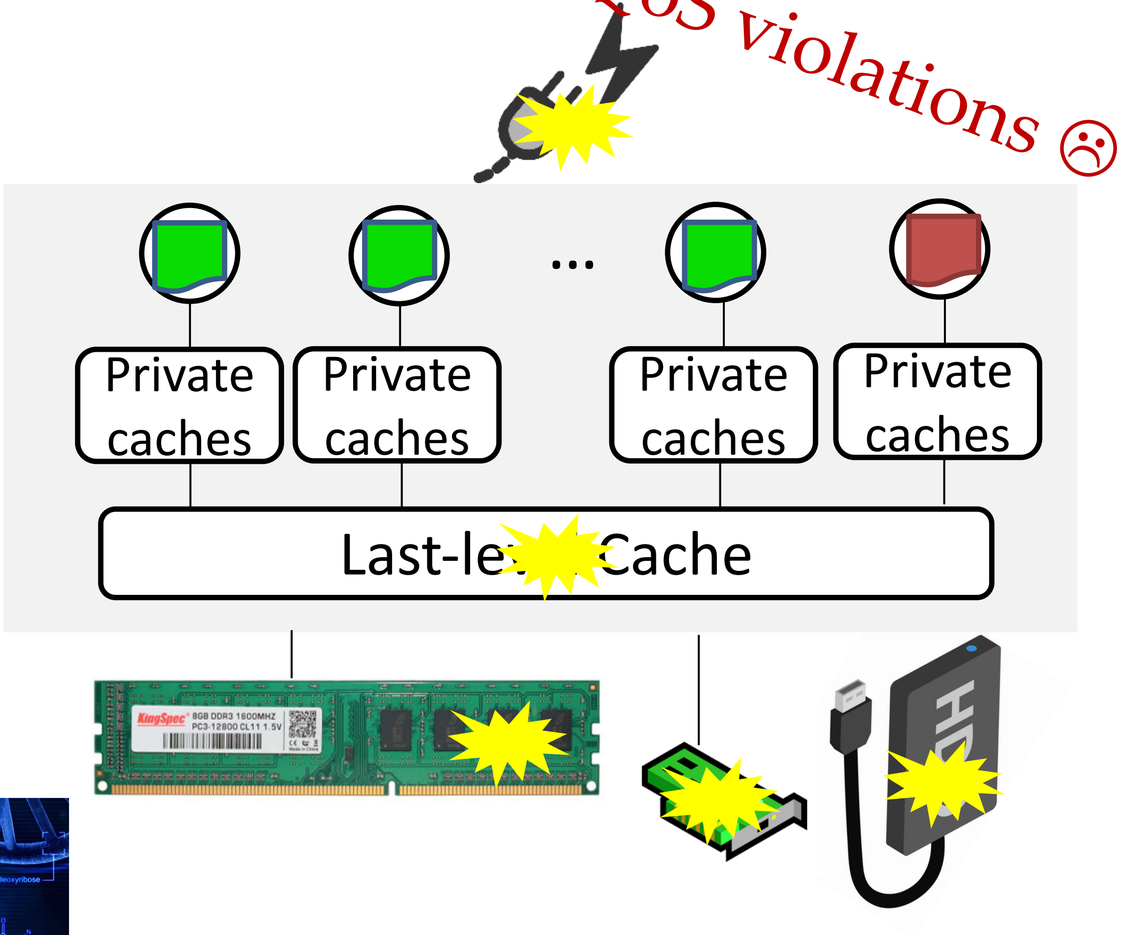
Cornell University

Colocation of Applications

Best-effort

hadoop

GraphLab



QoS violations ☹

Latency-critical

Google Maps

Bing

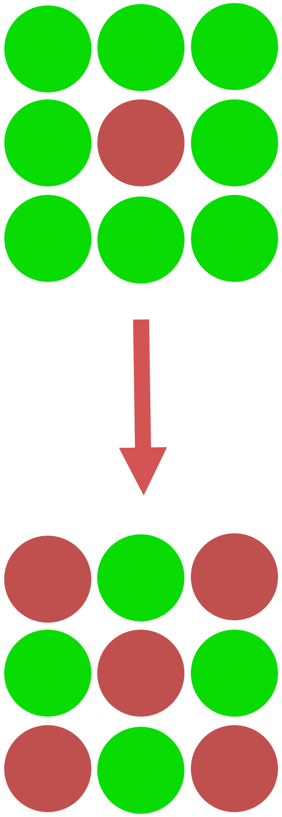
Google Translate

Prior Work

- **Interference during colocation**
- **Scheduling [Nathuji'10, Mars'13, Delimitrou'14]**
  - Avoid co-scheduling of apps that may interfere
    - May require offline knowledge
    - Limit colocation options
- **Resource partitioning [Sanchez'11, Lo'15]**
  - Partition shared resources
    - At most 1 LC app + multiple best-effort jobs

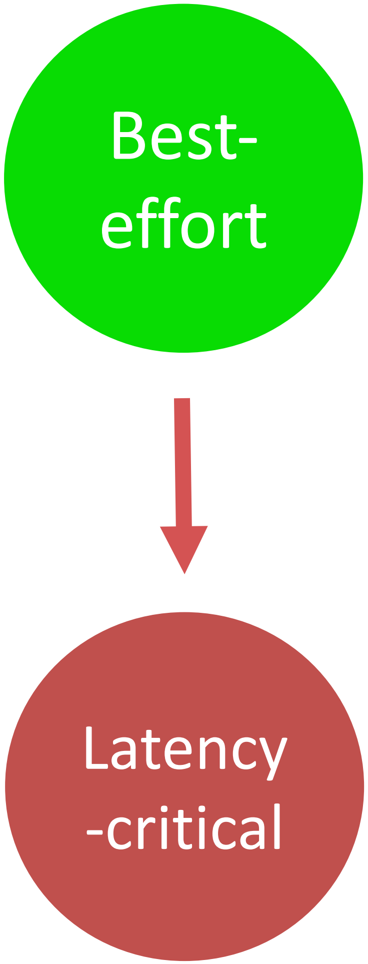
Trends in Datacenters

1 LC + many BE



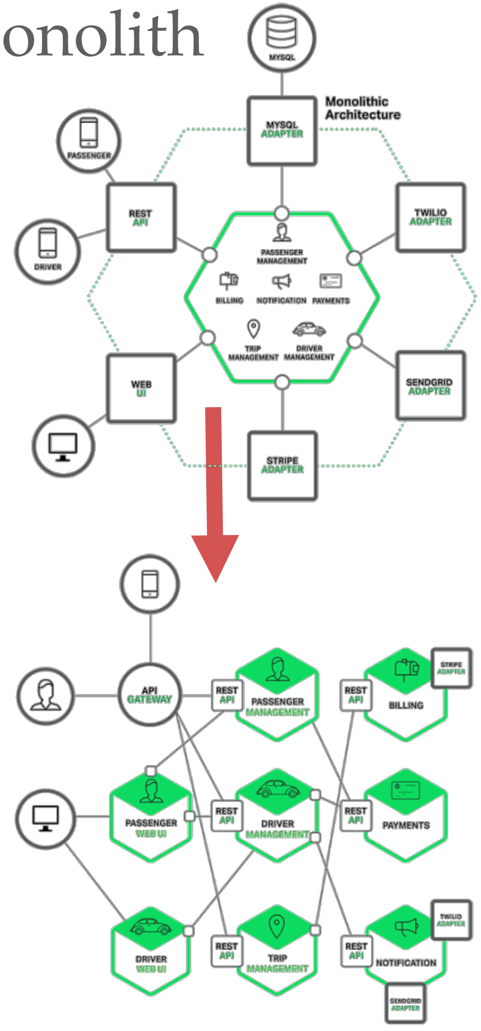
many LC + many BE

All have QoS targets



More LC jobs

Monolith



Microservices

MAIN CONTRIBUTIONS

- **Workload characterization**
  - The impact of resource sharing
  - The effectiveness of resource isolation
  - Relationship between different resources

- **PARTIES:** First QoS-aware resource manager for colocation of many LC services
  - Dynamic partitioning of 9 shared resources
  - No a priori application knowledge
  - 61% higher throughput under QoS constraints
  - Adapts to varying load patterns

INTERACTIVE LC APPLICATIONS

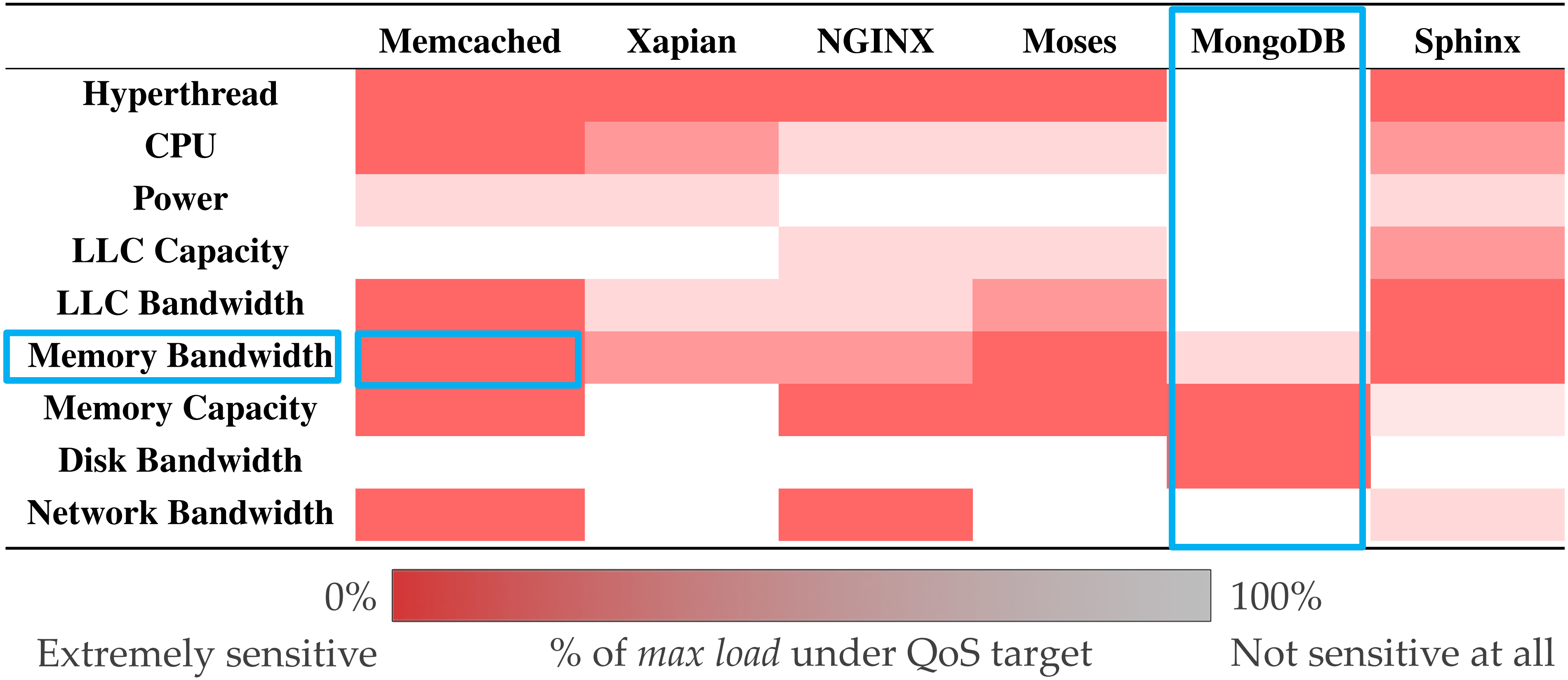
Application	Memcached	Xapian	NGINX	Moses	MongoDB	Sphinx
Domain	Key-value store	Web search	Web server	Real-time translation	Persistent database	Speech recognition
Target QoS	600us	5ms	10ms	15ms	300ms	2.5s
Max Load	1,280,000	8,000	560,000	2,800	240	14
User / Sys / IO CPU%	13 / 78 / 0	42 / 23 / 0	20 / 50 / 0	50 / 14 / 0	0.3 / 0.2 / 57	85 / 0.6 / 0
LLC MPKI	0.55	0.03	0.06	10.48	0.01	6.28
Memory Capacity	9.3 GB	0.02 GB	1.9 GB	2.5 GB	18 GB	1.4 GB
Memory Bandwidth	0.6 GB/s	0.01 GB/s	0.6 GB/s	26 GB/s	0.03 GB/s	3.1 GB/s
Disk Bandwidth	0 MB/s	0 MB/s	0 MB/s	0 MB/s	5 MB/s	0 MB/s
Network Bandwidth	3.0 Gbps	0.07 Gbps	6.2 Gbps	0.001 Gbps	0.01 Gbps	0.001 Gbps

INTERACTIVE LC APPLICATIONS

Application	Memcached	Xapian	NGINX	Moses	MongoDB	Sphinx
Domain	Key-value store	Web search	Web server	Real-time translation	Persistent database	Speech recognition
Target QoS	600us	5ms	10ms	15ms	300ms	2.5s
Max Load	1,280,000	8,000	560,000	2,800	240	14
User / Sys / IO CPU%	13 / 78 / 0	42 / 23 / 0	20 / 50 / 0	50 / 14 / 0	0.3 / 0.2 / 57	85 / 0.6 / 0
LLC MPKI	0.55	0.03	0.06	10.48	0.01	6.28
Memory Capacity	9.3 GB	0.02 GB	1.9 GB	2.5 GB	18 GB	1.4 GB
Memory Bandwidth	0.6 GB/s	0.01 GB/s	0.6 GB/s	26 GB/s	0.03 GB/s	3.1 GB/s
Disk Bandwidth	0 MB/s	0 MB/s	0 MB/s	0 MB/s	5 MB/s	0 MB/s
Network Bandwidth	3.0 Gbps	0.07 Gbps	6.2 Gbps	0.001 Gbps	0.01 Gbps	0.001 Gbps

Max load: max RPS under QoS target when running alone

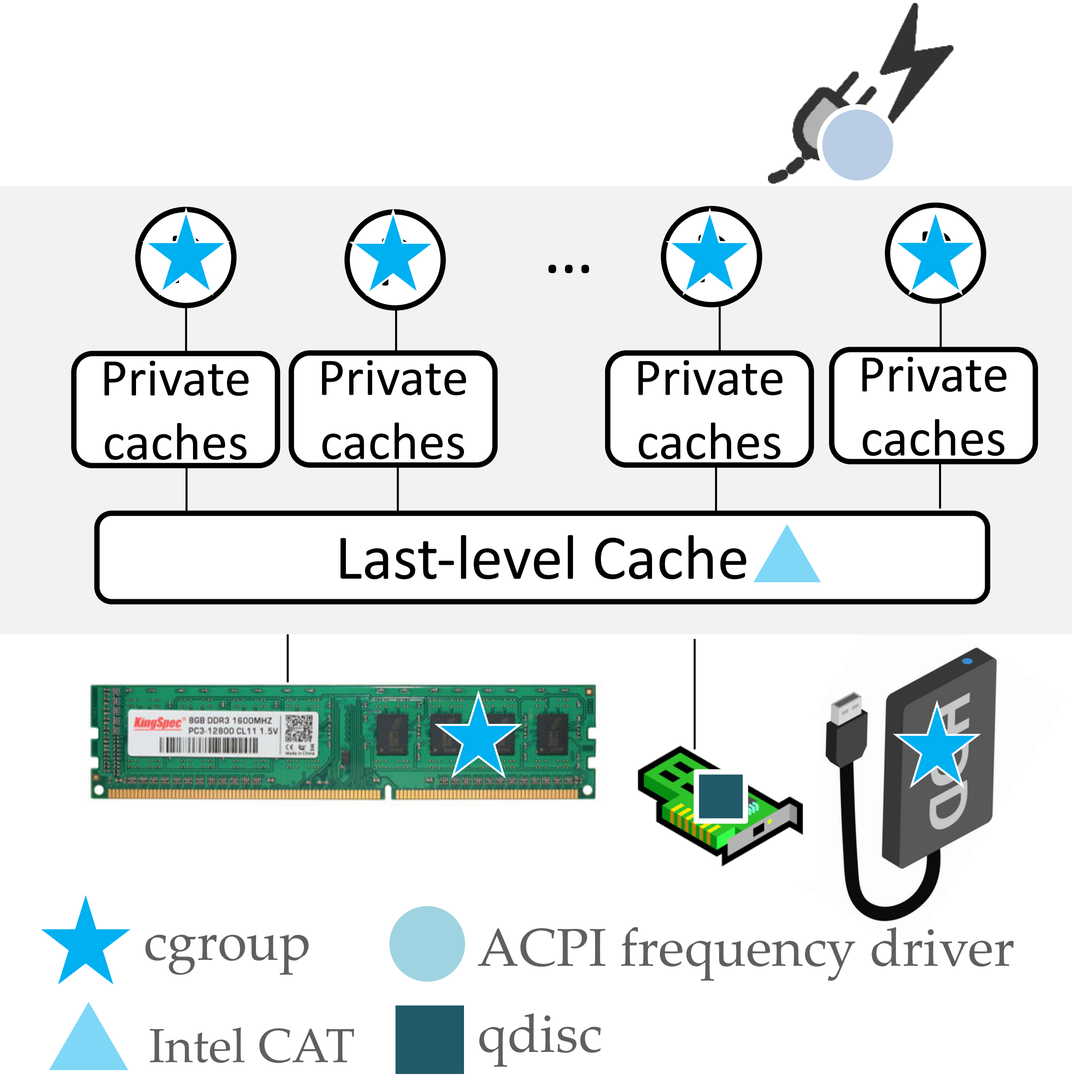
Interference Study



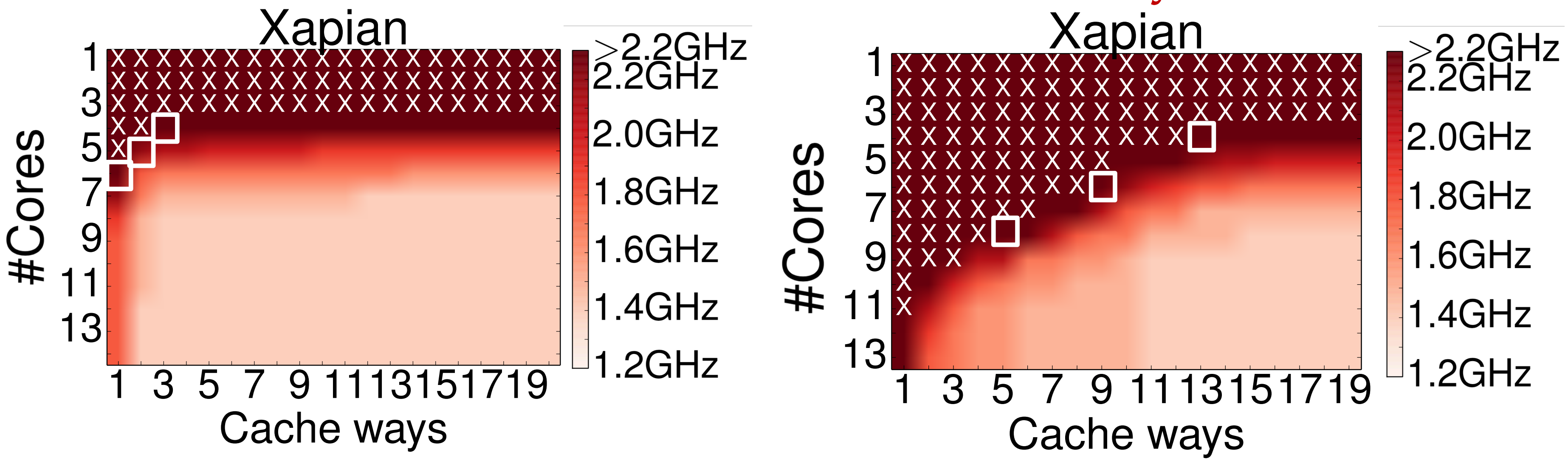
- Applications are sensitive to resources with high usage
- Applications with strict QoS targets are more sensitive

Isolation Mechanisms

- Core mapping
  - » Hyperthreads
  - » Core counts
- Memory capacity
- Disk bandwidth
- Core frequency
  - » Power
- LLC capacity
  - » Cache capacity
  - » Cache bandwidth
  - » Memory bandwidth
- Network bandwidth



Resource Fungibility



Stand-alone

With memory interference

- **Resources are *fungible***
  - More flexibility in resource allocation
  - Simplifies resource manager

PARTIES: Design Principles

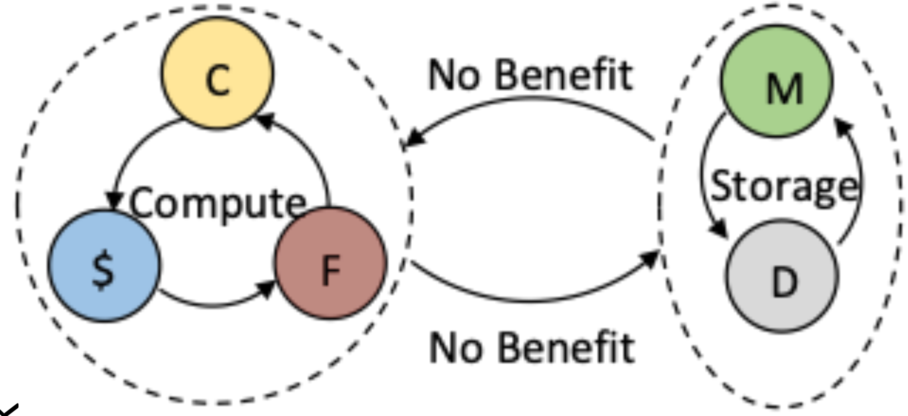
- **PARTIES**
  - PARTitioning for multiple InteractivE Services

- **Design principles**
  - LC apps are equally important
  - Allocation should be dynamic and fine-grained
  - No a priori application knowledge or offline profiling
  - Recover quickly from incorrect decisions
  - Migration is used as a last resort

PARTIES

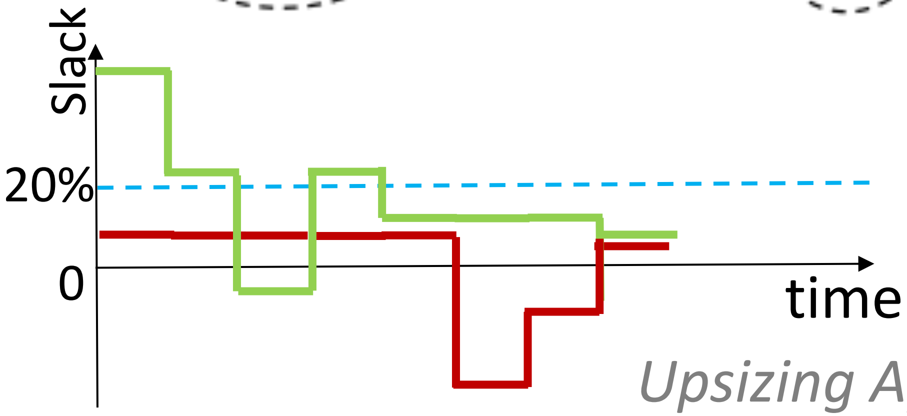
- 5 knobs organized into 2 wheels
- Start from a random resource
- Follow the wheels to visit all resources

Unallocated pool

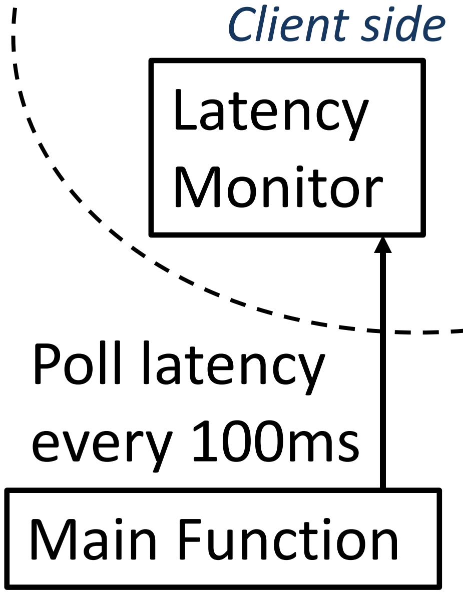


App 1 C

App 2 $



Upsizing App 1...



QoS violations?

Upsize!

Excess resources?

Downsize!

Server side

METHODOLOGY

- **Platform: Intel E5-2699 v4**
  - Single socket with 22 cores (8 IRQ cores)
- **Virtualization**
  - LXC 2.0.7
- **Load generators**
  - Open loop
  - Request inter-arrival distribution: exponential
  - Request popularity: Zipfian
- **Testing strategy**
  - Constant load: 30s warmup, 1m measurement (x5)
  - Varying load simulates diurnal load patterns

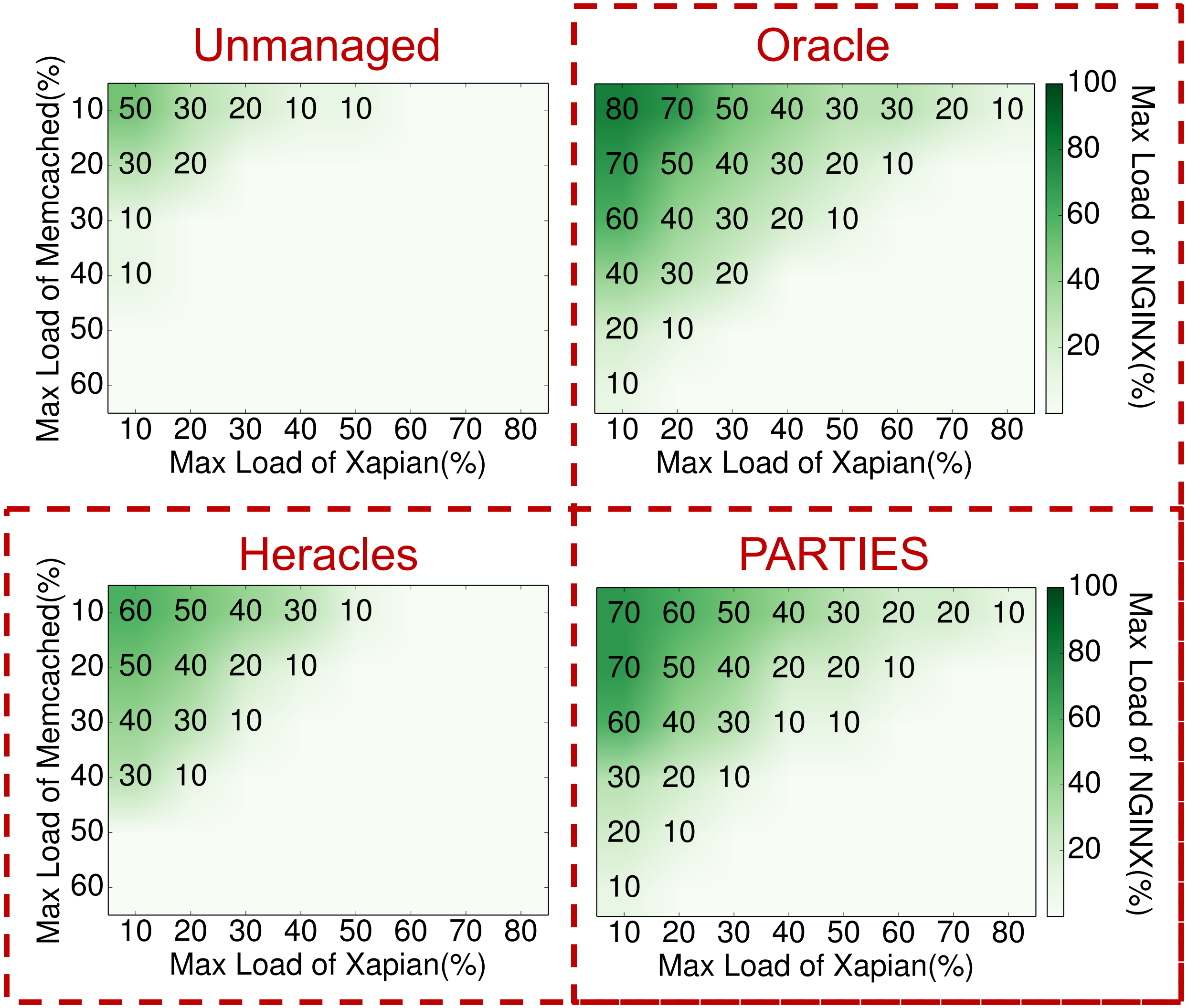
CONSTANT LOADS: MEMCACHED, XAPIAN & NGINX

Oracle

- Offline profiling
- Always finds the global optimum

Heracles

- No partitioning between BE jobs
- Suspend BE upon QoS violation
- No interaction between resources



More Evaluation

Constant loads

- All 2- and 3-app mixes under PARTIES
- Comparison with Heracles for 2- to 6-app mixes

Diurnal load pattern

- Colocation of Memcached, Xapian and Moses

PARTIES overhead

- Convergence time for 2- to 6-app mixes

Conclusions

- **Need to manage multiple LC apps**
- **Insights**
  - Resource partitioning
  - Resource fungibility
- **PARTIES**
  - Partition 9 shared resources
  - No offline knowledge required
  - 61% higher throughput under QoS targets
  - Adapts to varying load patterns

Cornell University
Computer Systems Laboratory

PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services

http://tiny.cc/parties

Shuang Chen, Christina Delimitrou, José F. Martínez

Cornell University