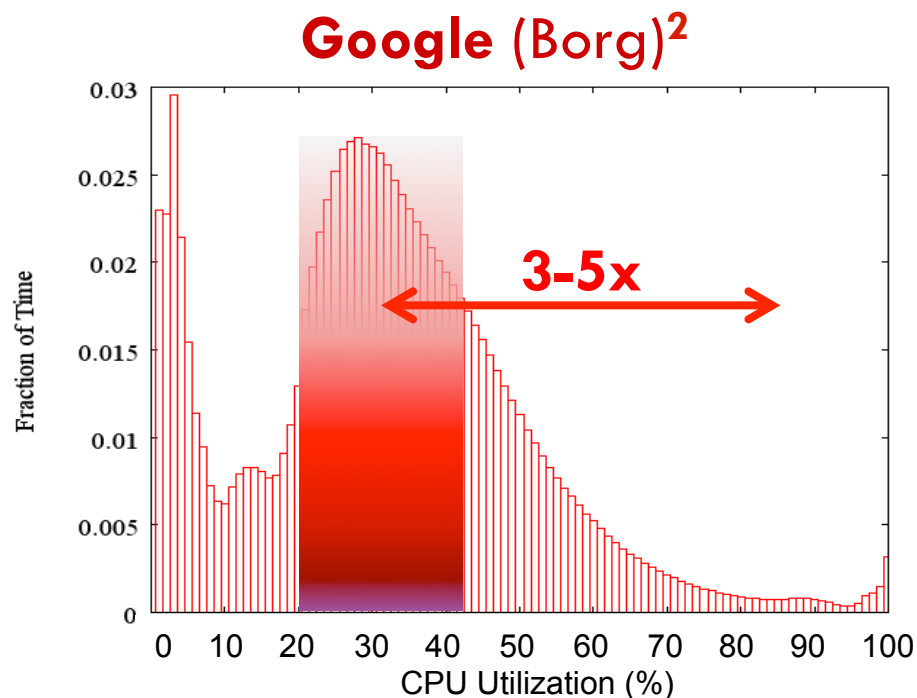
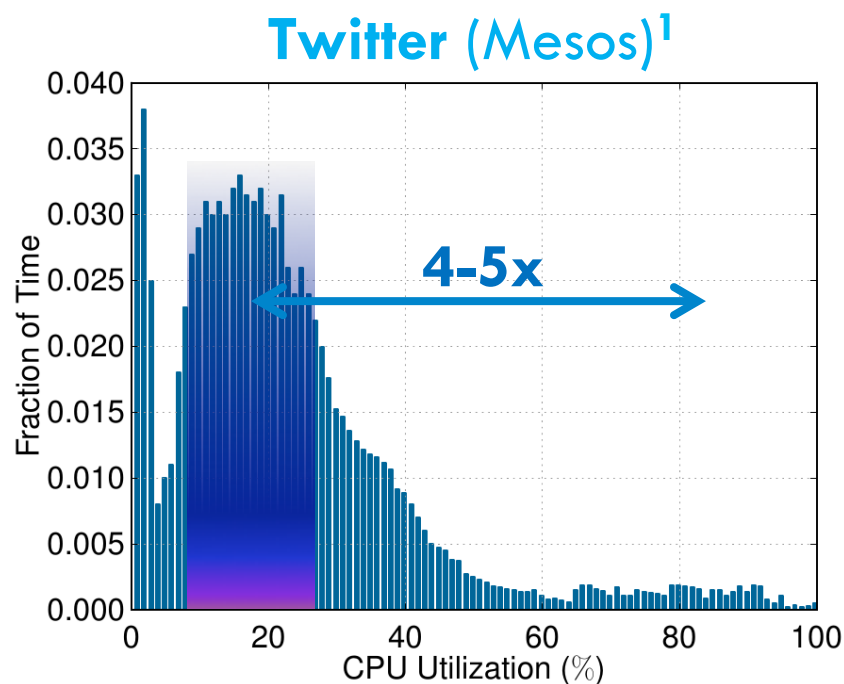


# Leveraging Approximation to Improve Resource Efficiency in the Cloud

Neeraj Kulkarni, Feng Qi, Glyfina Fernando  
and Christina Delimitrou

*Cornell University*

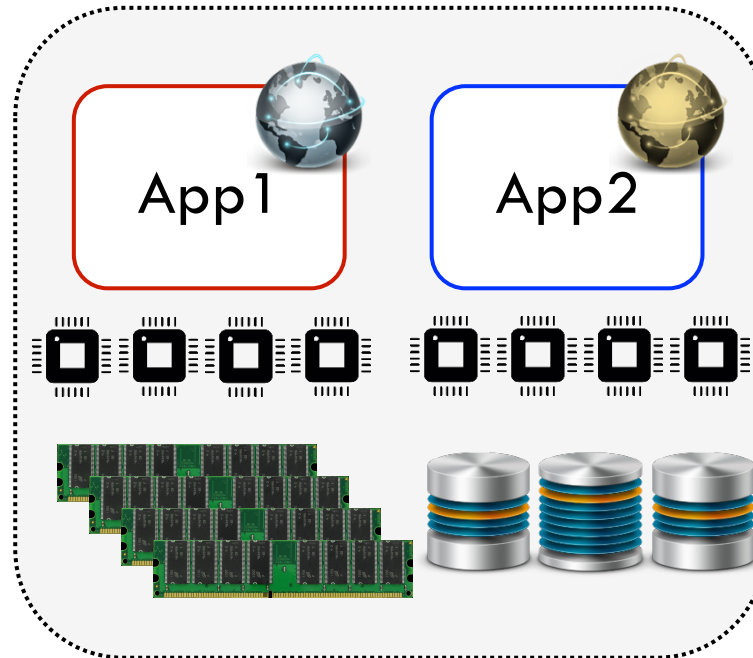
# Datacenter Underutilization



<sup>1</sup> C. Delimitrou and C. Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management, ASPLOS 2014.

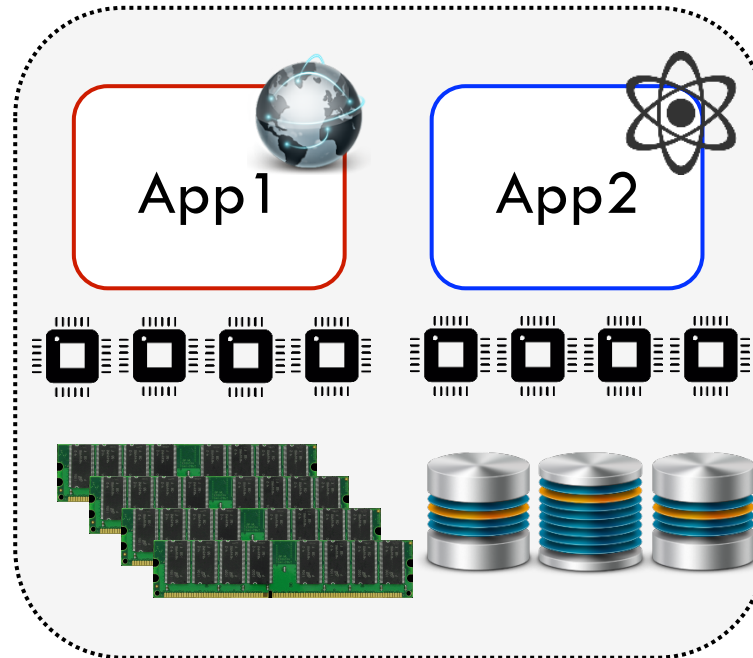
<sup>2</sup> L. A. Barroso, U. Holzle. The Datacenter as a Computer, 2013.

# A Common Approach



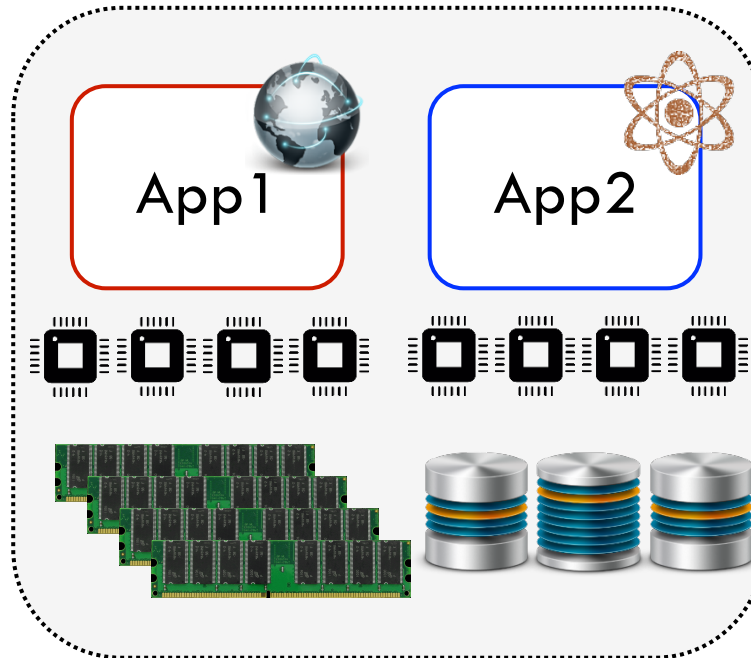
- ❑ Co-schedule multiple cloud services on same physical platform
- ❑ Often leads to resource interference, especially when sharing cores

# A Common Cure



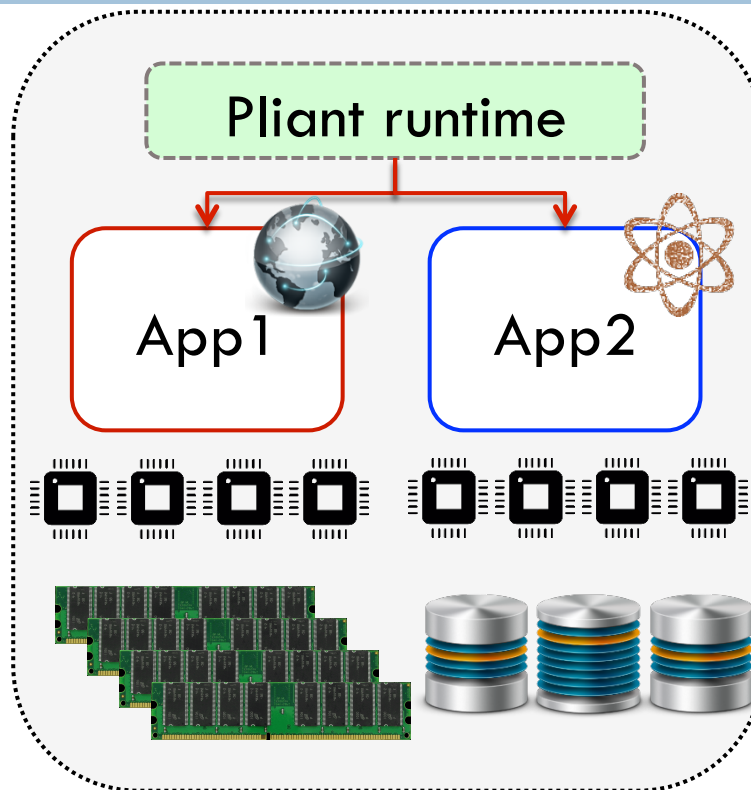
- Co-schedule one high priority and one/more best-effort apps
- Performance is non-critical for best effort jobs
- **Disadvantage: assume best-effort apps are always low priority**

# Approximate Computing Apps to the Rescue



- Approximate computing apps can absorb a loss of resources as loss of output quality instead of a loss in performance
- Advantage: performance of all co-scheduled applications is high-priority

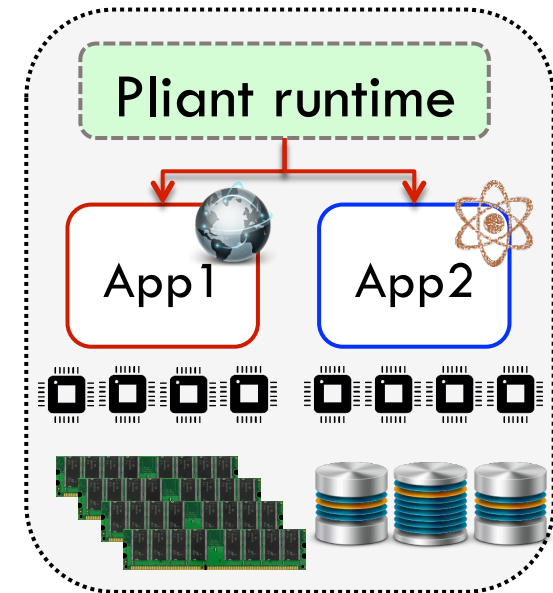
# Pliant



- Enables latency-critical & approximate apps to share resources (including cores) without penalizing their performance
- Tunes degree and type of approximation based on measured interference

# Challenges

1. **Identify opportunities for approximation**
  - ▣ ACCEPT (precision, loop perforation, sync elision), algorithmic exploration
2. **Lightweight profiling to determine when to employ approximation**
  - ▣ End-to-end latency/throughput & perf counters
3. **Determine what resource(s) to constrain?**
  - ▣ Based on measured interference
4. **Determine what type of approximation & to what extent?**
  - ▣ Based on interference and performance impact



# Pliant

Server

Client

Workload generator

Performance monitor

Pliant runtime

Interference monitor

App1

App2

DynamoRIO for switching between precise/approximate versions

- Initial implementation, overheads high but not prohibitive
- Looking into Petabricks and LLVM



# Adaptive Approximation

## □ Incremental approximation:

- ▣ Employ the minimum amount of approximation (quality loss) to restore the performance of the interactive service
- ▣ Several versions for each type of approximation, choose online

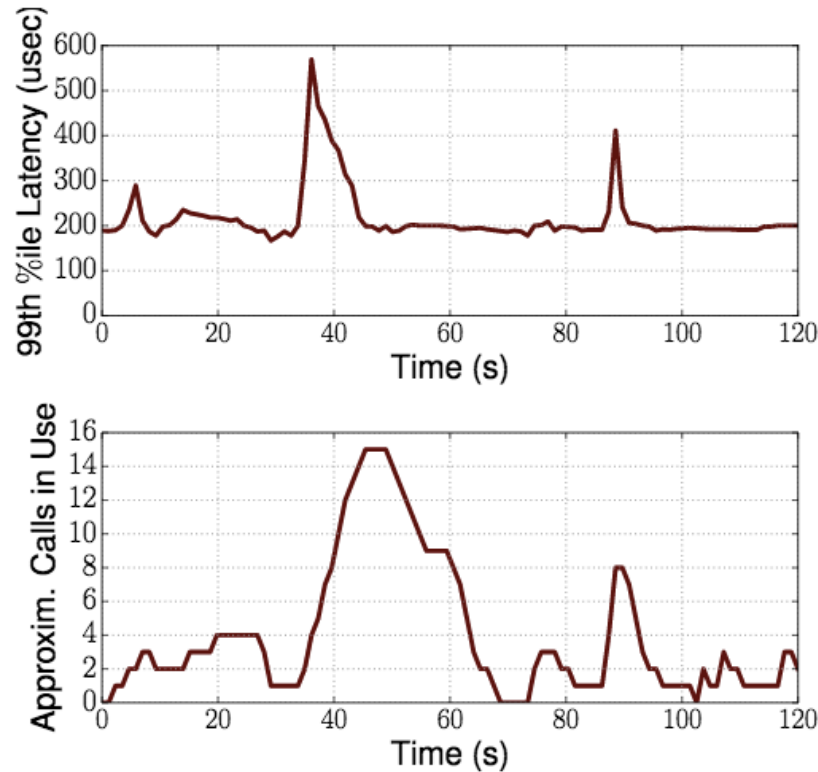
## □ Interference-aware approximation:

- ▣ Choose the type of interference that minimizes pressure in the bottlenecked resource
- ▣ Example:
  - High memory interference → prioritize algo tuning
  - High CPU interference → prioritize sync elision, loop perforation

# Methodology

- Latency-critical interactive services: memcached & nginx
  - ▣ Open-loop workload generator & performance monitor
  - ▣ Facebook traffic pattern
- Approximate computing apps: PARSEC, SPLASH, Spark MLlib
- System: 2 2-socket, 40-core servers, 128GB RAM each

# Evaluation



□ memcached sharing physical cores with PARSEC

□ Latency  → Degree of approximation 

# Conclusions

- Approximate computing: opportunity to improve cloud efficiency without loss in performance
- **Pliant**: cloud runtime to co-schedule interactive services with approximate computing apps
  - ▣ **Incremental** and **interference-aware** approximation
  - ▣ Preserves QoS for interactive service with minimal loss in quality for approximate computing application
- **Current work**:
  - ▣ DynamoRIO → Petabricks/LLVM
  - ▣ Add cloud approximate computing application
  - ▣ Explore design space exploration & optimization problem
  - ▣ Leverage hardware isolation techniques

# Questions?

- Approximate computing: opportunity to improve cloud efficiency without loss in performance
- **Pliant**: cloud runtime to co-schedule interactive services with approximate computing apps
  - ▣ **Incremental** and **interference-aware** approximation
  - ▣ Preserves QoS for interactive service with minimal loss in quality for approximate computing application
- **Current work**:
  - ▣ DynamoRIO → Petabricks/LLVM
  - ▣ Add cloud approximate computing application
  - ▣ Explore design space exploration & optimization problem
  - ▣ Leverage hardware isolation techniques