TS-Bat: Leveraging Temporal-Spatial Batching for Data Center Energy Optimization

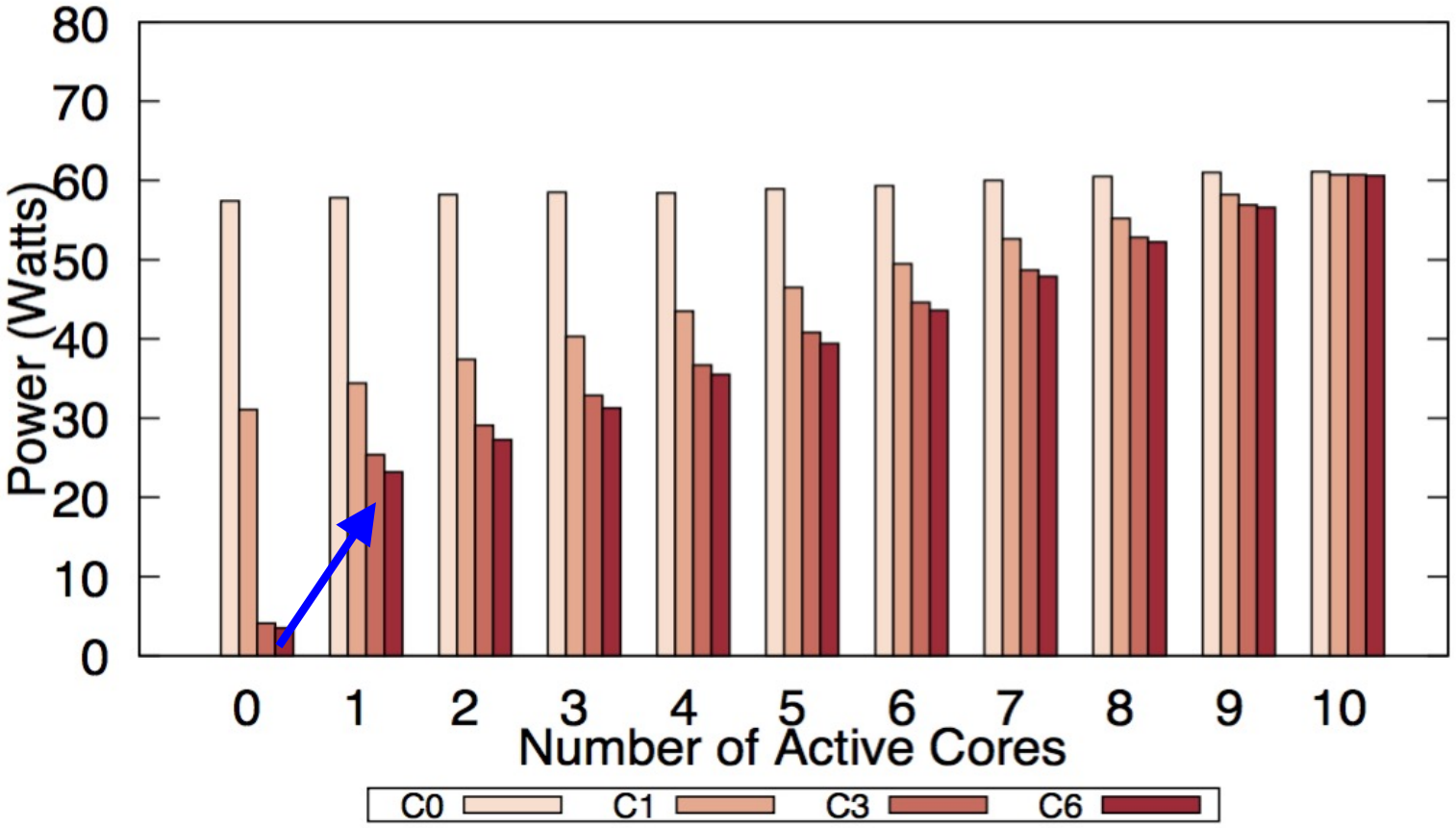
GLOBECOM 2017, Singapore

Motivation

- **Server Farms are generally over-provisioned**
  - ↗CPUs and other hardware are configured to satisfy peak load
  - ↗On average, typical server farm utilization is < 30%
- **Current server farms *lack* efficient power management**
  - ↗ CPUs are kept active more than necessary
  - ↗ Only shallow power saving states (e.g., Core C1 state) are used
- **A more effective energy optimization requires**
  - ↗A better characterization of multi-core power consumption
  - ↗Energy-aware scheduling and control of processor low power states
  - ↗Workload adaptivity (QoS constraints, job arrival rate & service time)

Understanding Multi-core Processor Power

- **Power Profile of the 10-core Intel Xeon E5 Processor**



- Cores not active are in the Corresponding C state
- Power measurements through the *RAPL* interface

Observations from Processor Power Profile

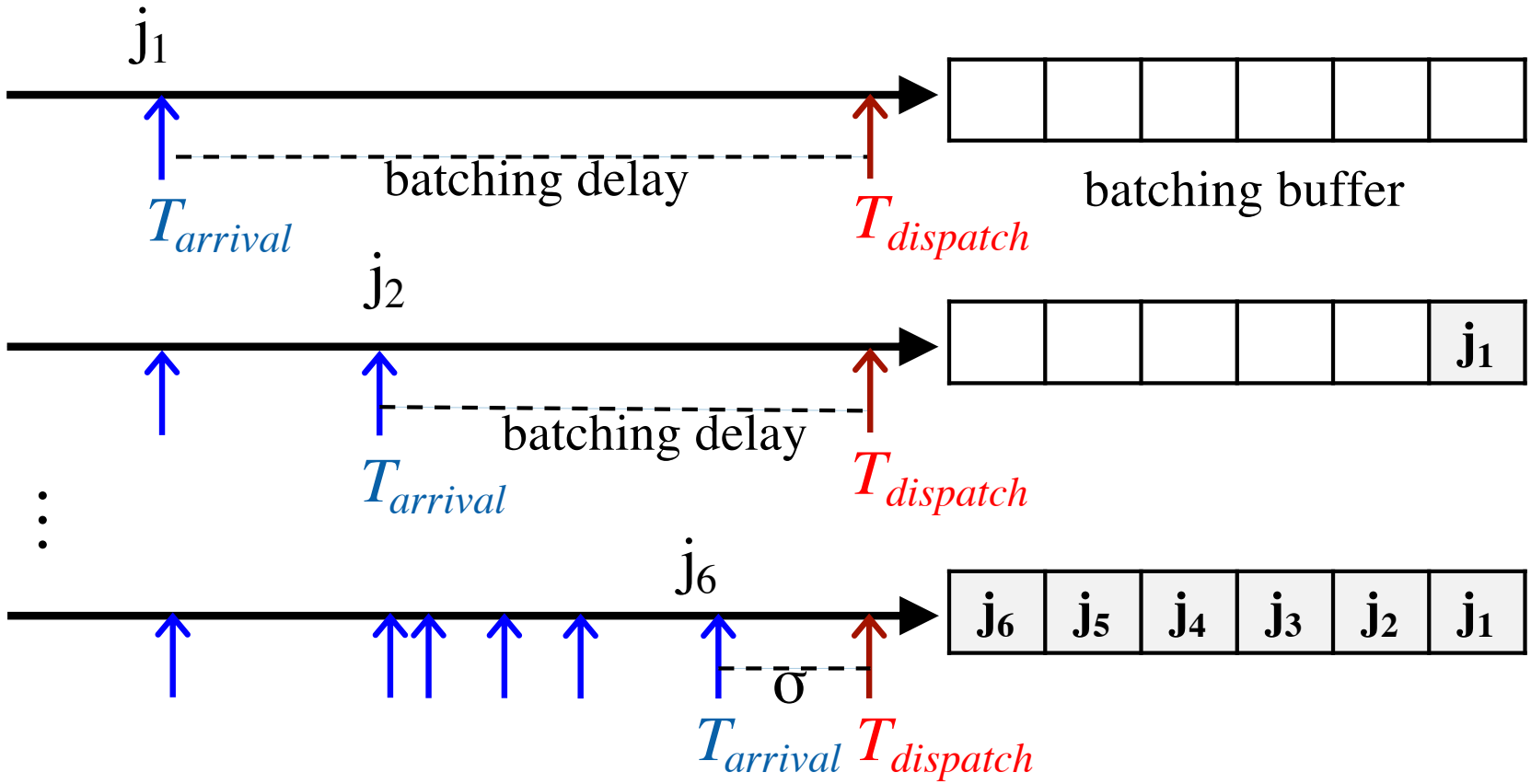
- **Processors consume considerable base power**
- **Higher utilization brings better power efficiency**

To increase power efficiency and save energy

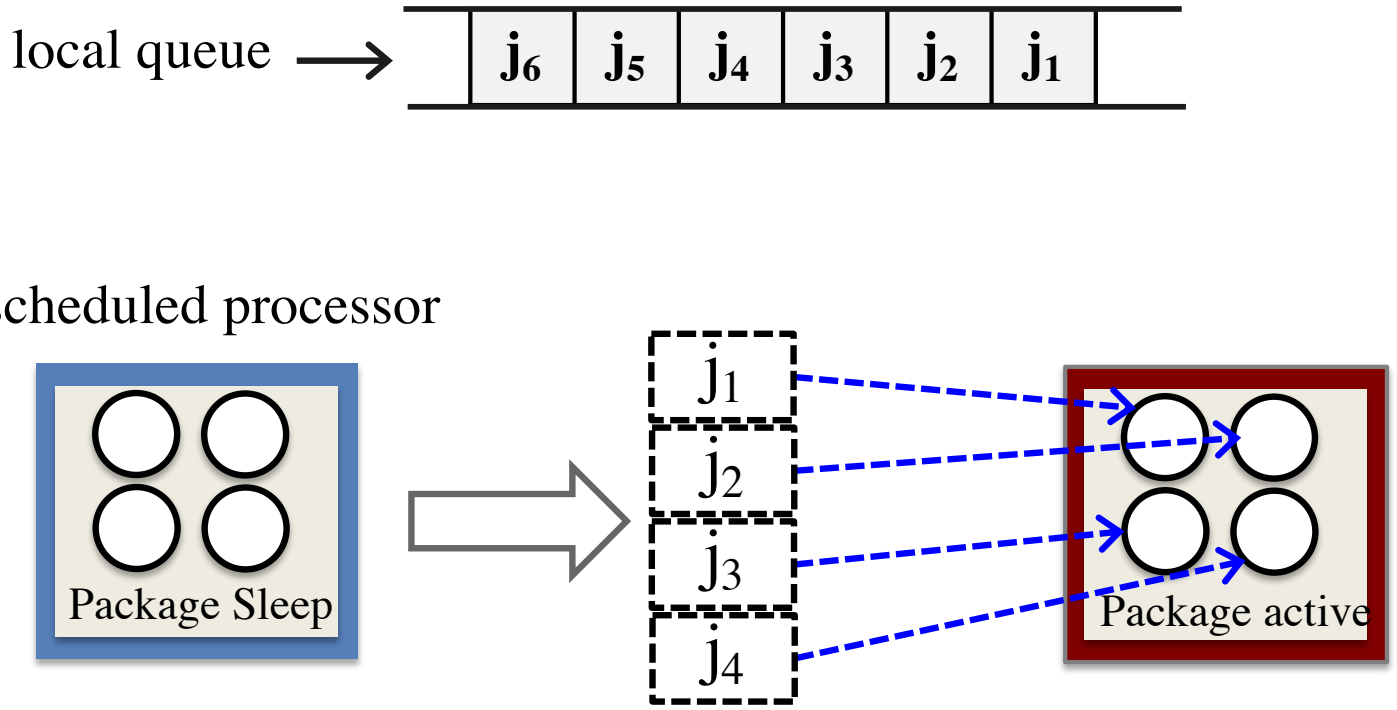
↗Either keep all the cores active ➔ ***highest power efficiency***

↗Or keep all the cores in idle state ➔ ***enables processor sleep***

Temporal Batching – Front End



Job Batch Processing – Back End



- Jobs are buffered in the local queue of the server
- For a *C* core processor, every *C* jobs can be executed simultaneously

Temporal Batching Problem Formulation

- ***Determine the max number of jobs to be batch***
- **For each job i**
  - Batching delay: $B_i = (K-i)/\lambda + \sigma$
  - Queuing delay: $U_i = S^{95} * (i-1)/C + W$
  - Total Delay: $D_i = B_i + U_i$
- **Find max *K* so that that** $\forall i \in [1, K]$:

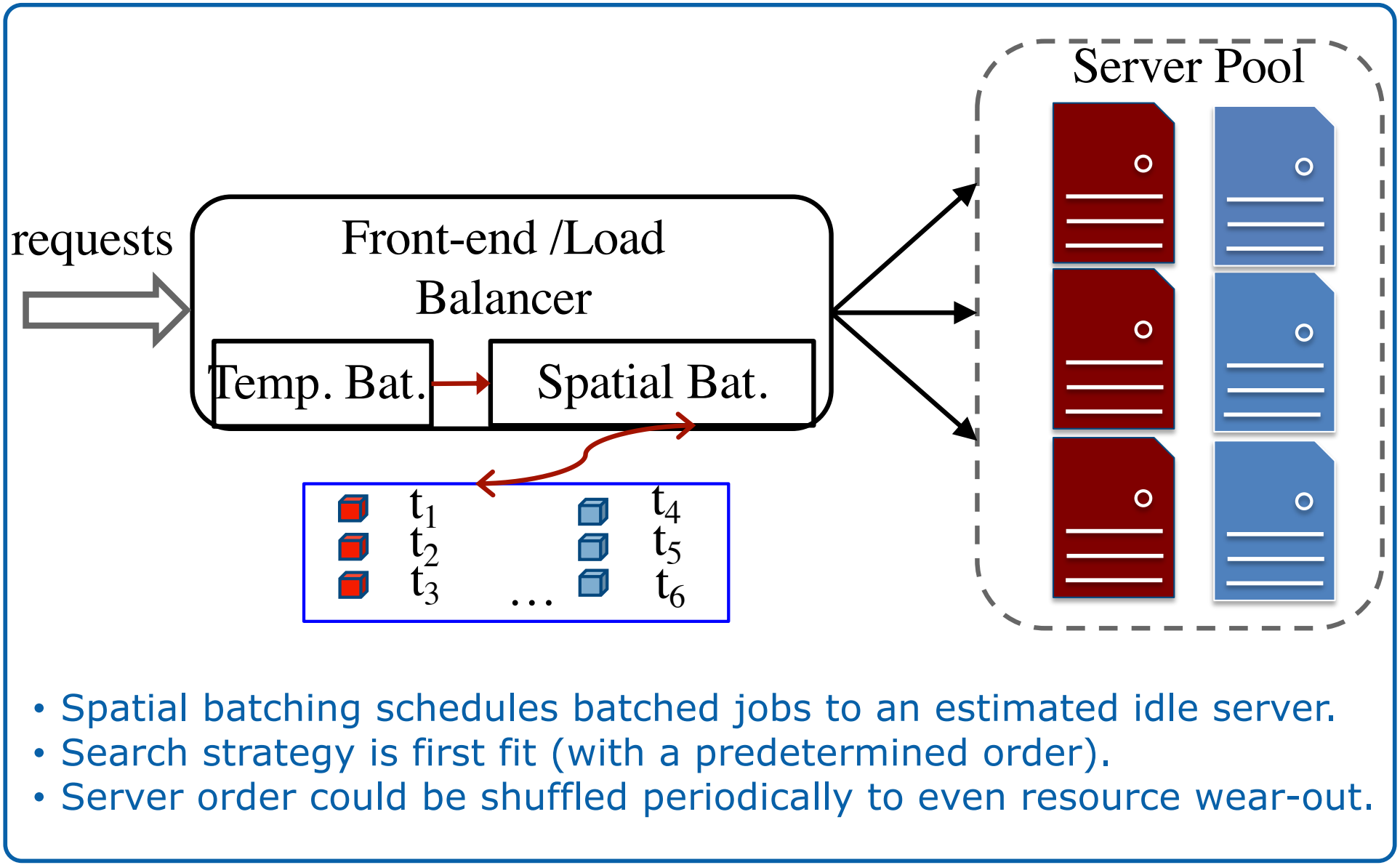
$$D_i + S^{95} \leq Q$$

λ : job arrival rate
C : number of cores per server
Q : target tail latency constraint
S : service time distribution,
S^{95}: 95th percentile service time
σ :processor wakeup latency

Integration with Spatial Batching

- **Group jobs onto specific servers**
- **Instead of evenly distributing load…**
  - ↗Keep a subset of CPUs in active state
  - ↗Create opportunities for other servers to enter deep sleep states
    - – Cuts down power significantly
- **Challenges: How to choose idle and active servers?**

Temporal + Spatial Batching – Scale-up



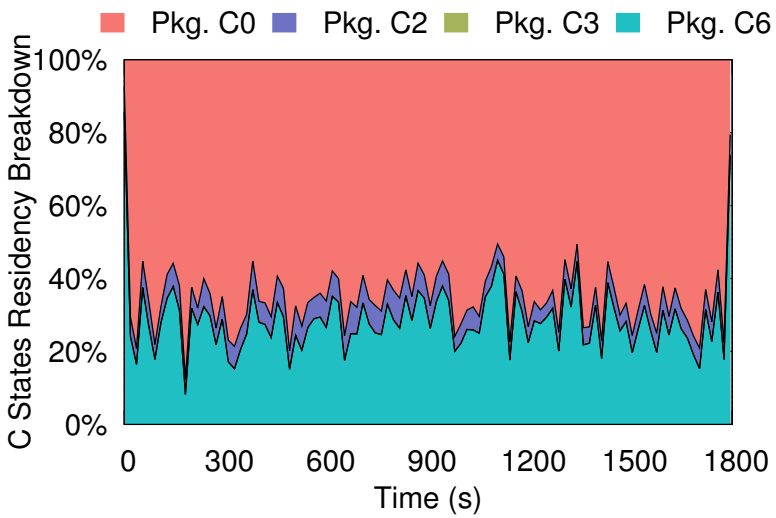
- Spatial batching schedules batched jobs to an estimated idle server.
- Search strategy is first fit (with a predetermined order).
- Server order could be shuffled periodically to even resource wear-out.

Experimental Setup

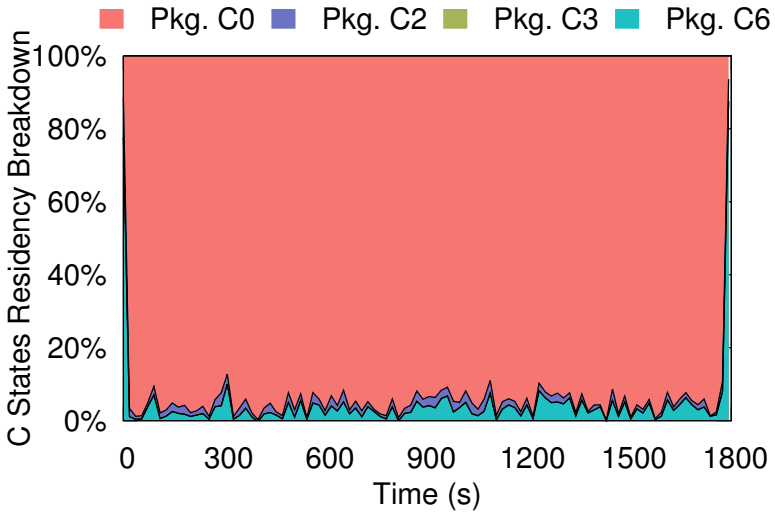
- **Testbed**
  - ➚Temporal Batching – single server
    - – Intel Xeon E5 10 core process
  - ➚Temporal Batching + Spatial Batching
    - – Cluster of 16 servers
    - – Each server equipped with Intel Xeon 5650 6-core server
    - – Network : Netgear 24-port Gigabit switch (start topology)
- **Software Framework**
  - ➚Load generator: *httperf*
  - ➚Application server: apache httpd
  - ➚Benchmark: CGI script running PARSEC-2.1 benchmarks at back-end
- **The targeted QoS constraint**
  - – 95^{th} percentile latency* is less than 2× job service time *

**95^{th} Percentile latencies are adopted as measurements of QoS constraints in Latency critical applications – Rubik, Micro 2014; PEGASUS, ISCA 2014; Tailbench, ASPLOS 2016*

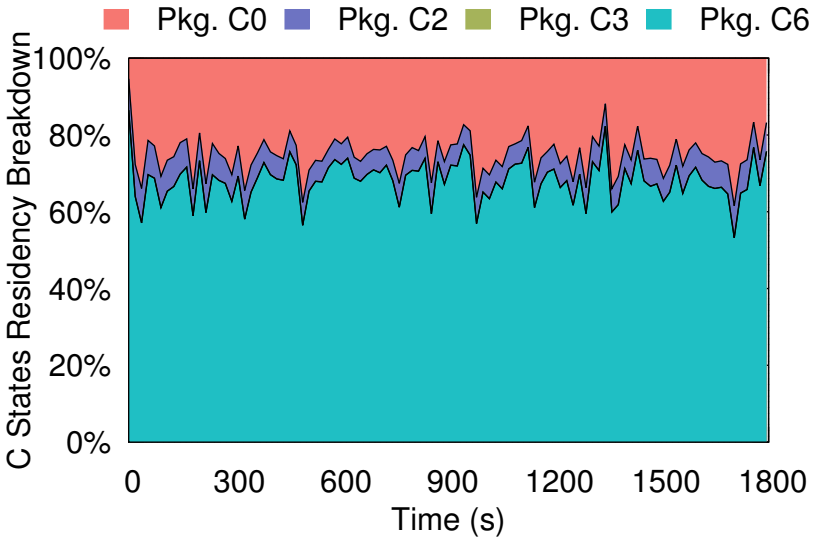
Temporal Batching – C State Residency



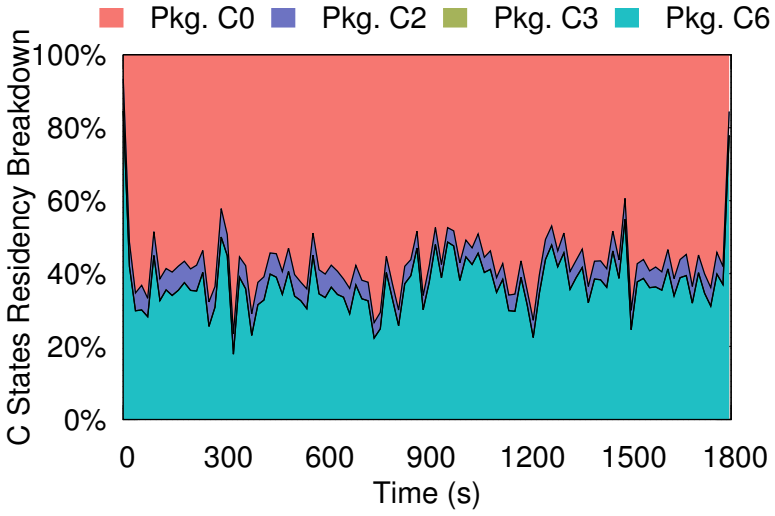
Bodytrack, without T. B. (Util. 10%)



Bodytrack, without T. B. (Util. 30%)

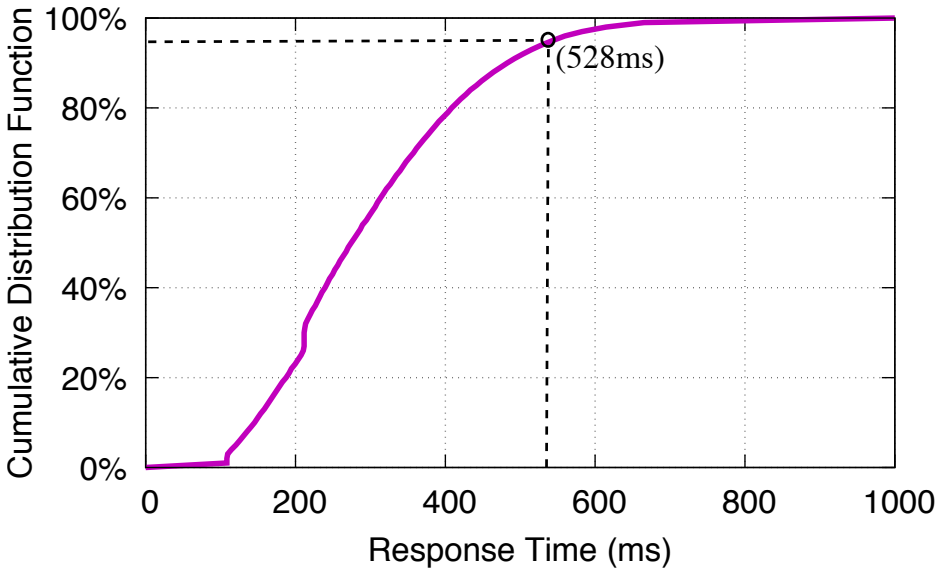


Bodytrack, with T. B. (Util. 10%)

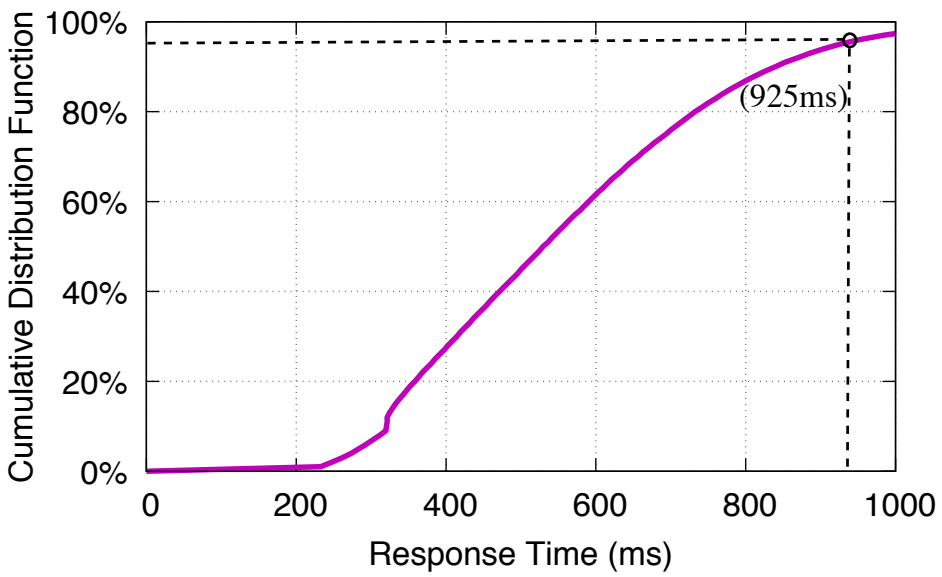


Bodytrack, with T. B. (Util. 30%)

Temporal Batching – Response Time



Bodytrack, Job Response time CDF (QoS 5x)



Bodytrack, Job Response time CDF (QoS 5x)

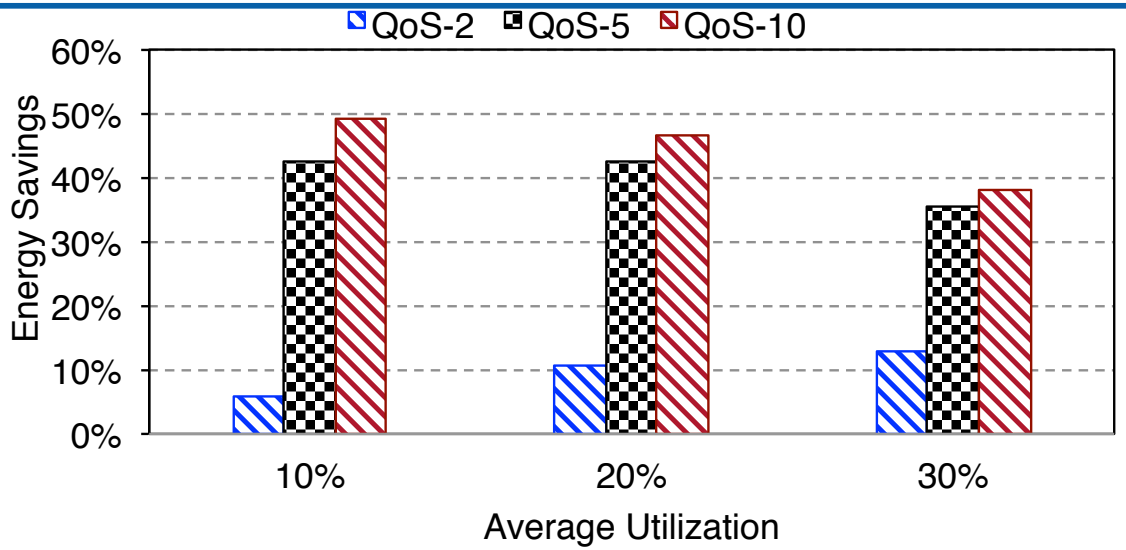
Bodytrack: average service time 108 ms

✓ For QoS 5x: 95th response time *528 ms* < target latency *540 ms*

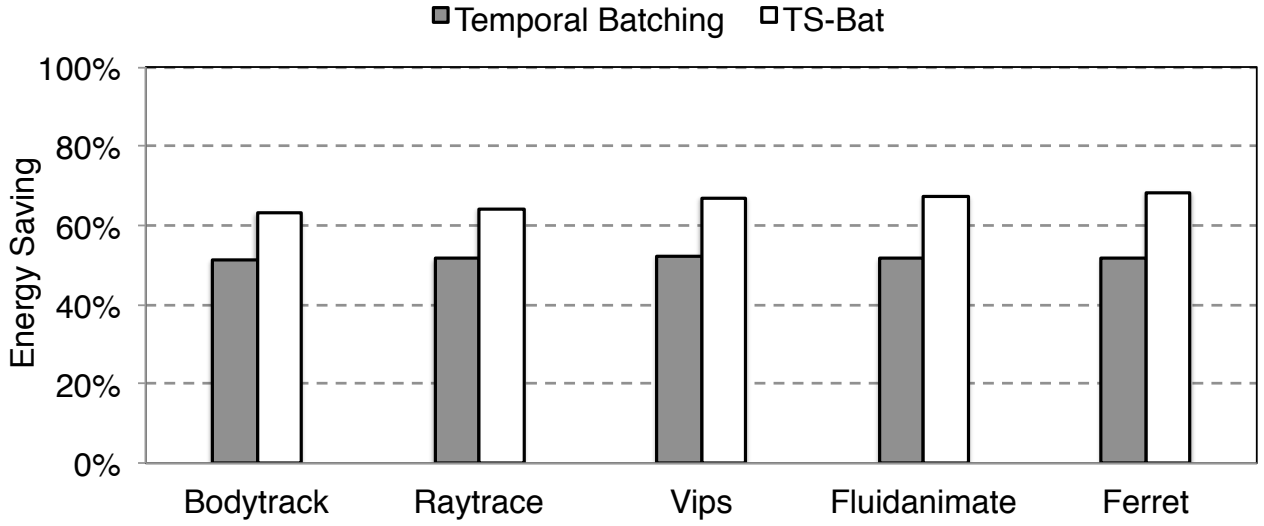
✓ For QoS 10x: 95th response time *925 ms* < target latency *1080 ms*

➔ Quality of Service constraints are effectively maintained

Energy Savings



Bodytrack, Energy saving of Temporal batching with different QoS



Energy saving of using Temporal and Spatial Batching, different workloads

Conclusion

- We characterized the power and performance of multi-core processor in data centers.

- We proposed TS-Bat, a QoS-aware energy optimization framework that combines temporal and spatial batching.

- TS-Bat achieved upto 68% processor energy saving under various QoS constraints.