Parallel Job Scheduling Under Dynamic Workloads

Eitan Frachtenberg[1,2], Dror Feitelson[2], Juan Fernandez[1], Fabrizio Petrini[1]

[1] CCS-3 Modeling, Algorithms, and Informatics Group

Computer and Computational Sciences (CCS) Division

Los Alamos National Laboratory

`{fabrizio,juanf}@lanl.gov`

[2] School of Computer Science and Engineering

Hebrew University, Jerusalem, Israel

`{etcs,feit}@cs.huji.ac.il`

JSSPP 2003

Outline

- Background and methodology

Outline

- Background and methodology
- Effect of multiprogramming Level
    - What multiprogramming levels should we use?
    - What is the effect of using backfilling?

Outline

- Background and methodology
- Effect of multiprogramming Level
  - What multiprogramming levels should we use?
  - What is the effect of using backfilling?
- Effect of time quantum on Gang Scheduling
  - What are good values for the time quantum?
  - What is the effect of different architectures?

Outline

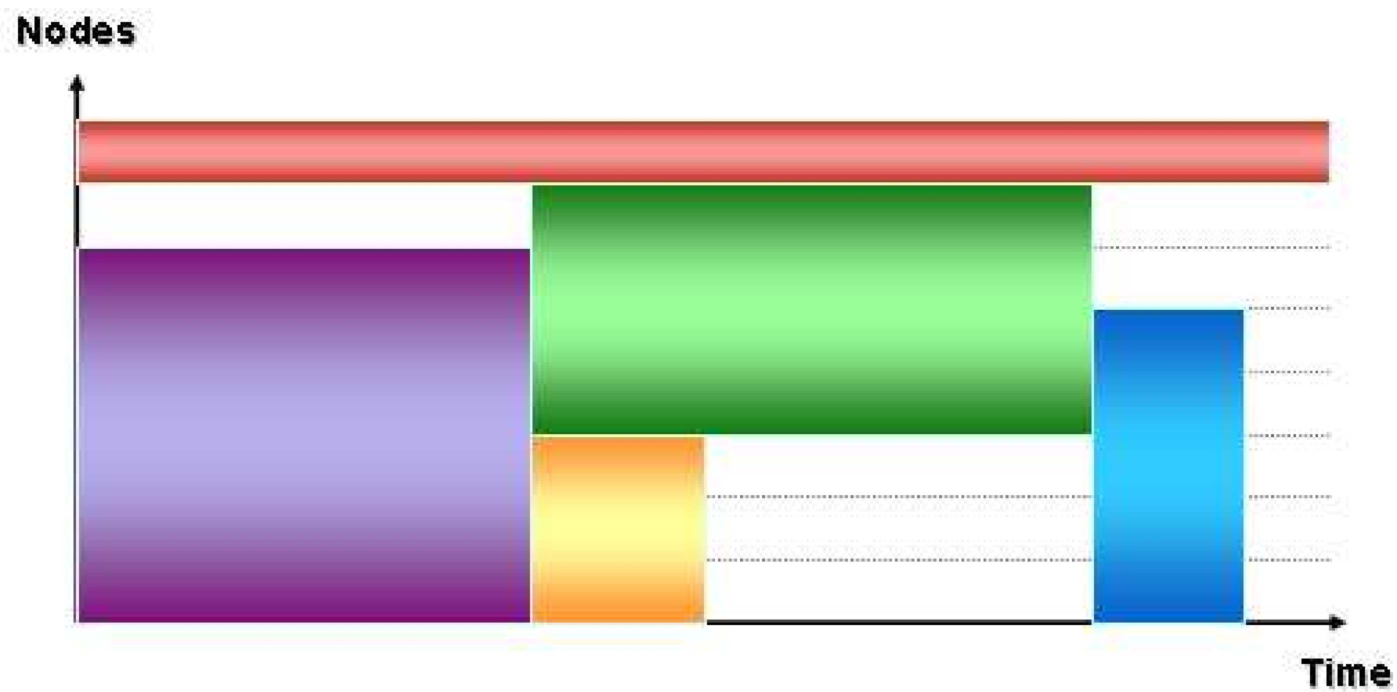
- Background and methodology
- Effect of multiprogramming Level
  - What multiprogramming levels should we use?
  - What is the effect of using backfilling?
- Effect of time quantum on Gang Scheduling
  - What are good values for the time quantum?
  - What is the effect of different architectures?
- Effect of load
  - How do different algorithms compare?
  - What type of jobs benefit from different algorithms?

Motivation

- An up-to-date and comparative evaluation of job scheduling algorithms
- Actual implementation on a modern cluster, with communication processes
- Focus on complex, dynamic workload, capturing feedback effects

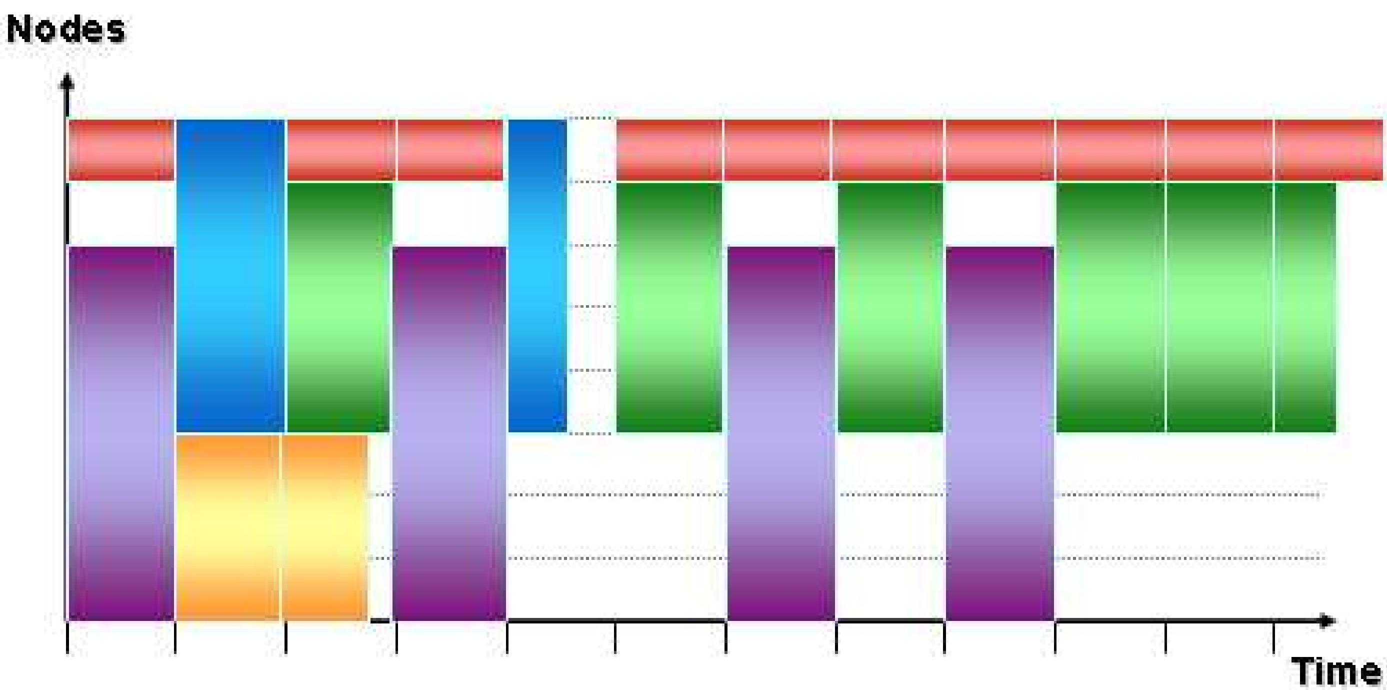
First-Come-First-Serve (FCFS) Scheduling

- Processors are divided to partitions
- Each job runs to completion in its dedicated partition
- Backfilling techniques for queue management



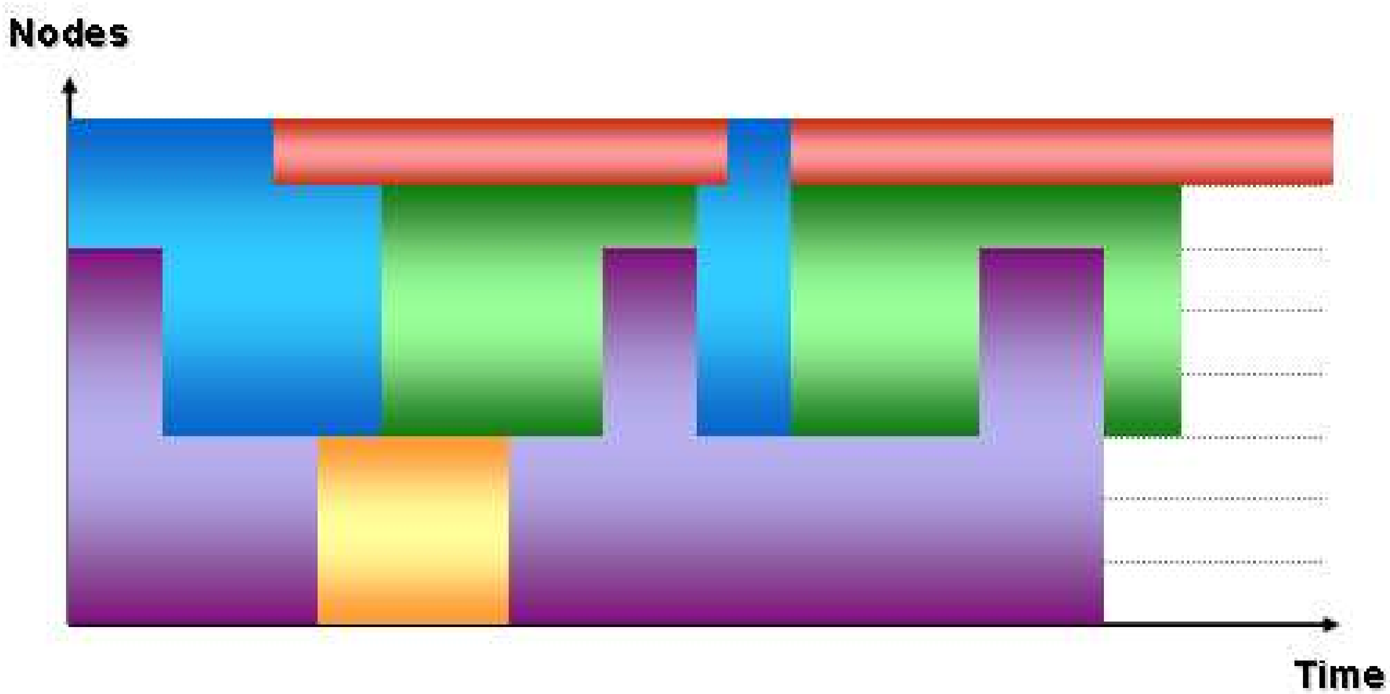
Explicit (Gang) Coscheduling

- Gang Scheduling (GS): coordinated context switching
- Context switch incurs overhead and cache pressure
- Scalability issues with global context switch



Implicit Coscheduling

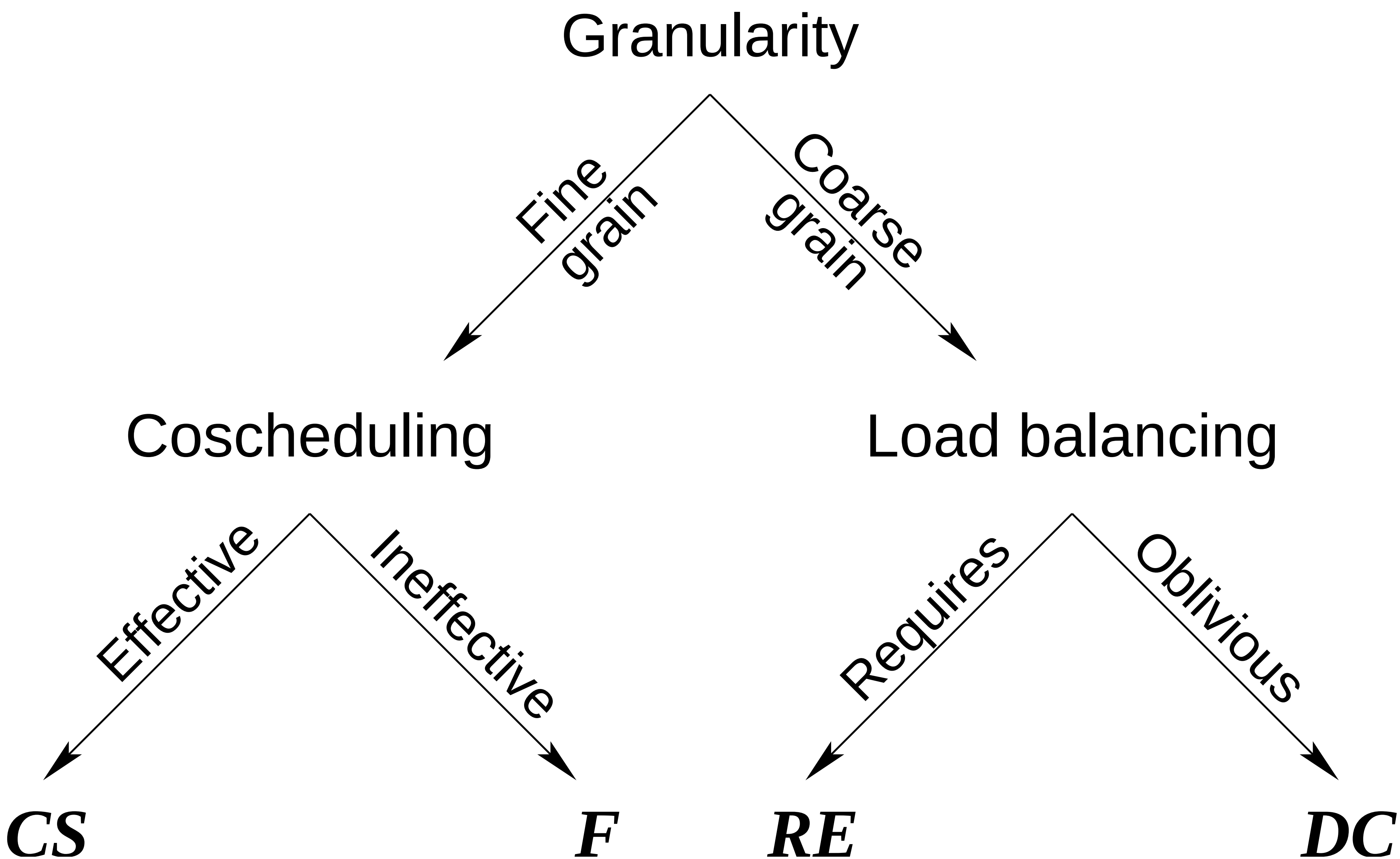
- Various methods: DCS, SB, PBT, ICS,...
- Use only local information for coordination
- Good for load-imbalance and utilization
- Not ideal for fine-grained jobs



Flexible Coscheduling (FCS)

- Use global coordination with local information
- Monitor processes' communication activity
- Classify processes based on communication
- Schedule processes according to their needs

FCS Decision Tree



FCS Scheduling

Use regular time-slices, but schedule processes based on classification:

- Fine-grained (CS) use explicit coscheduling
- Coarse-grained (DC) use no coordination
  - Local UNIX scheduler
- Load-imbalanced (F) use implicit coscheduling
  - Prioritized Spin-Block

Implementation Framework

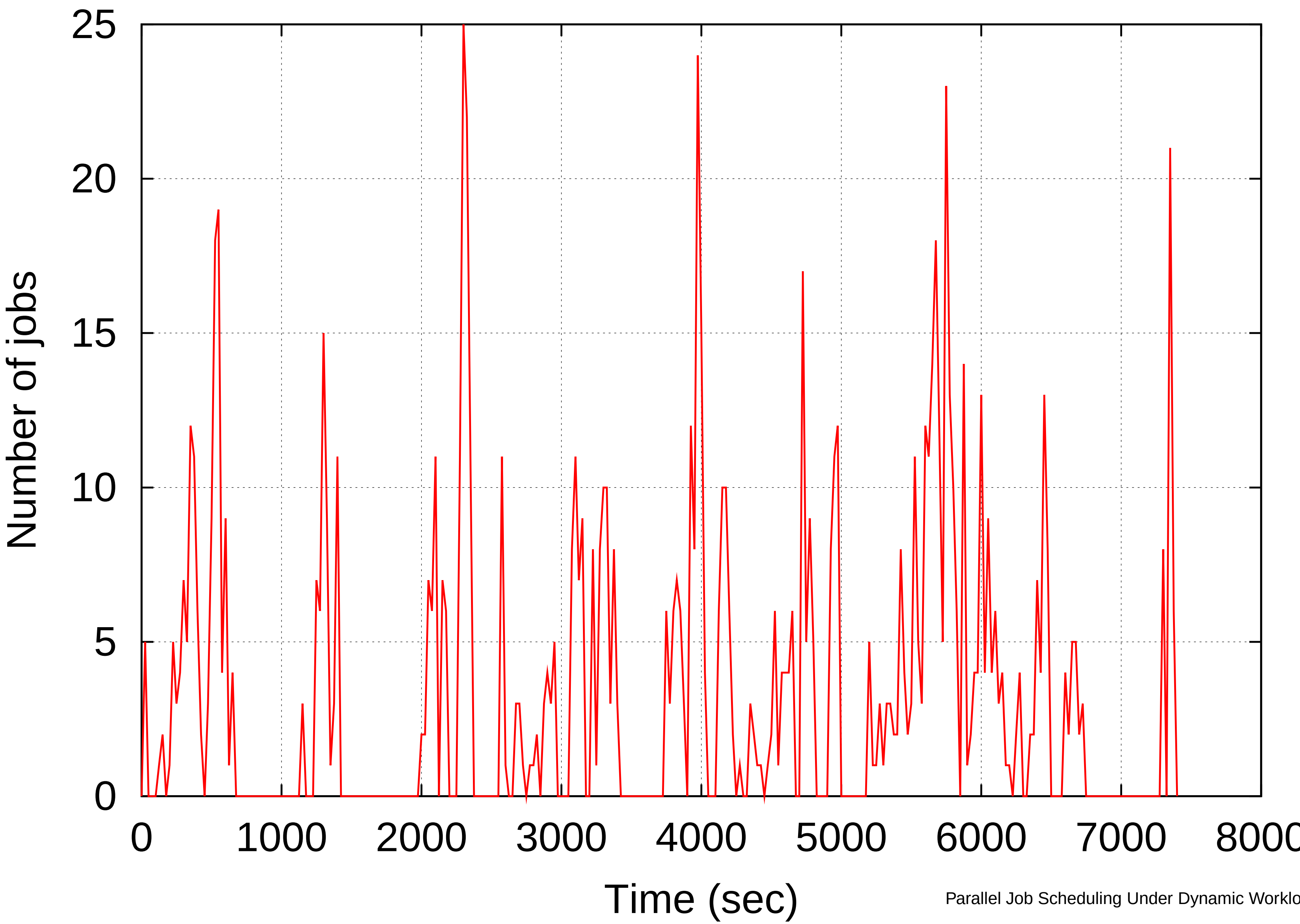
Fully implemented FCFS, GS, SB, FCS using STORM - Scalable Tool for Resource Management:

- Lightweight mechanisms, using HW collective communication primitives
- Scalable to thousands of nodes [SC02]
- “Pluggable” scheduling algorithms (a few more are implemented)
- Ported to x86, IA64 and Alpha architectures, Quadrics interconnect
- Most runs performed on a 16-node 2-way P-III cluster
- Queue management with EASY backfilling (w/all algorithms)

Dynamic Workload

- 1000 jobs with dynamic job arrivals, sizes and runtimes
- Based on detailed model of several traces [Lublin01]
- Synthetic BSP application with different granularities $5ms$, $50ms$, $500ms$
- Multiprogramming levels 1-6
- Timeslices of $50 - 2000\,ms$
- Offered load altered by factoring job run-times

namic Workload Characteristics (75% loa



Effect of Multiprogramming Level

What is a good MPL value?

- ▷ Tradeoff between overhead and utilization
- ▷ Relative effect of backfilling

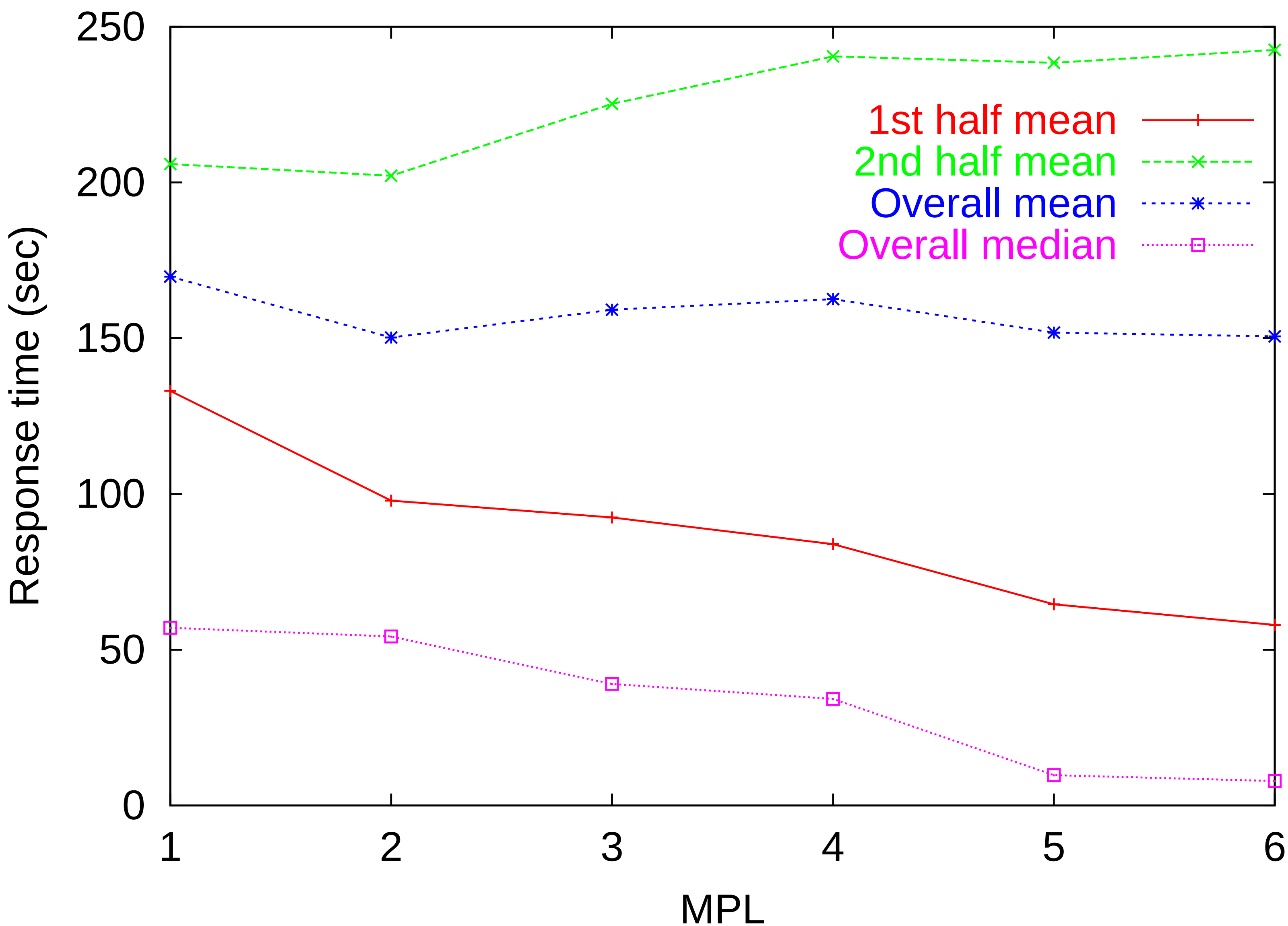
Effect of Multiprogramming Level

What is a good MPL value?

- ▷ Tradeoff between overhead and utilization
- ▷ Relative effect of backfilling

In most real scenarios, MPL is limited by memory

MPL - Response Time

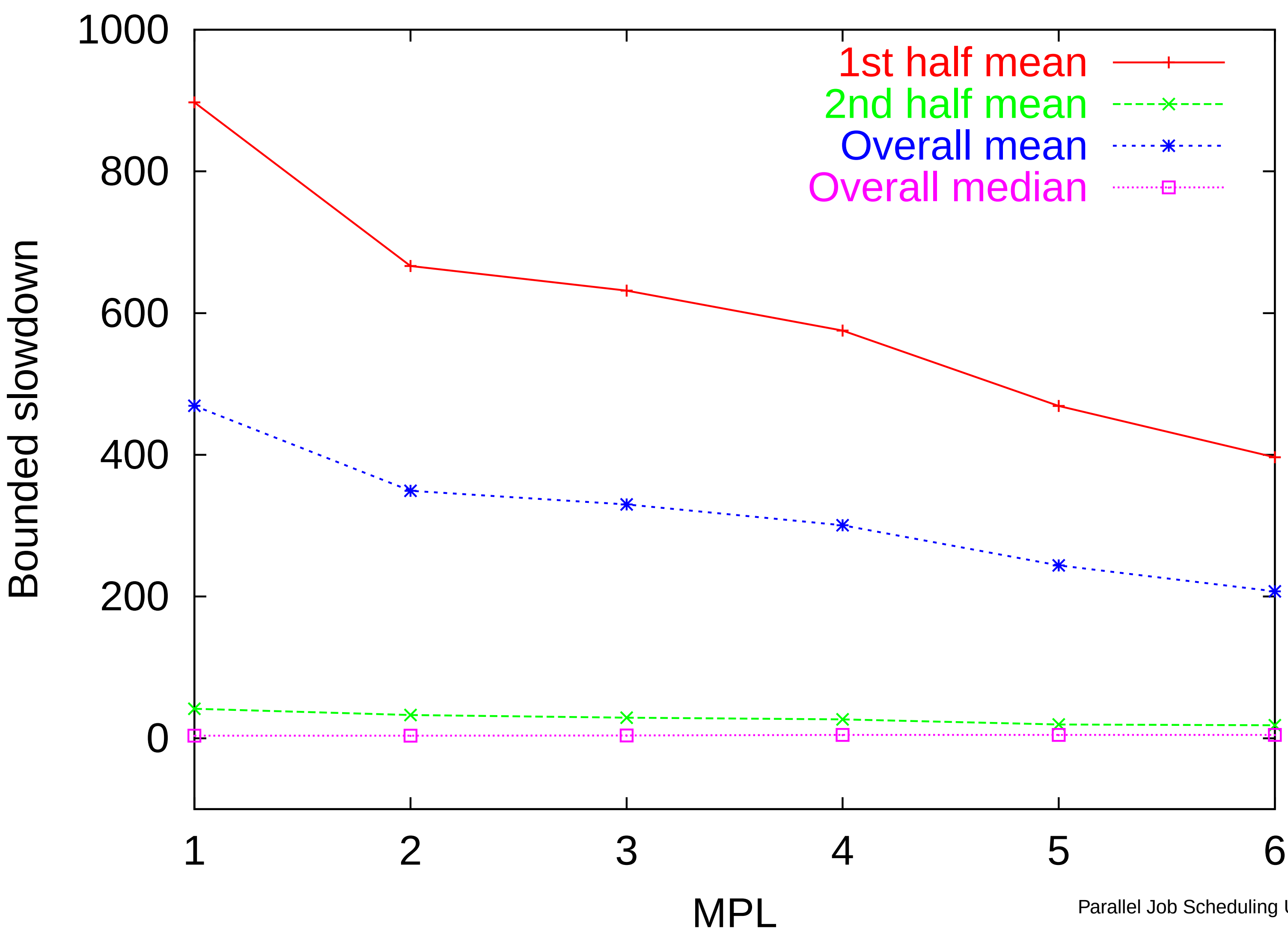


MPL - Bounded Slowdown

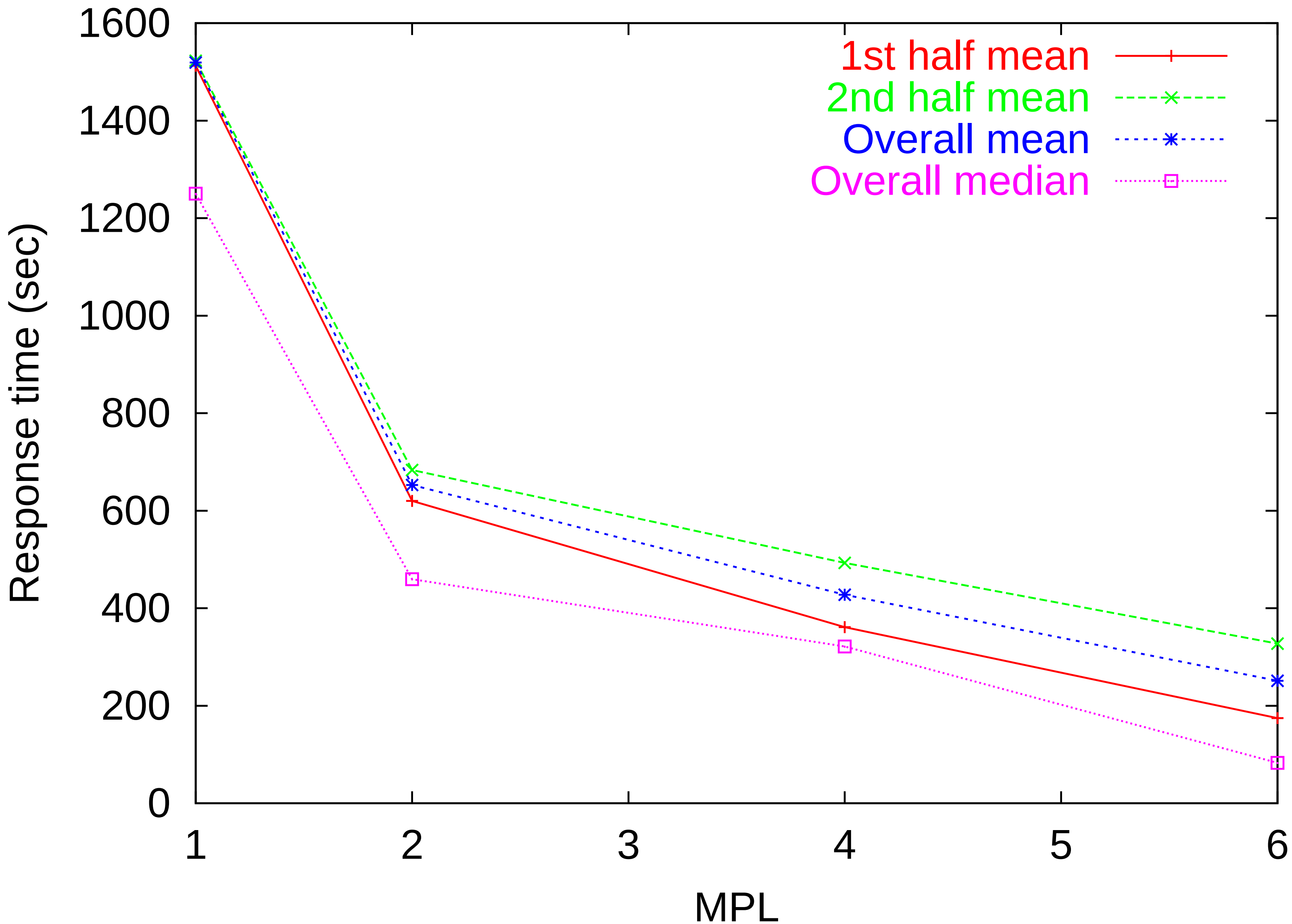
$$Bounded\,Slowdown = \max\left\{\frac{T_w + T_r}{\max\{T_d, \tau\}}, 1\right\}$$

MPL - Bounded Slowdown

$$Bounded\,Slowdown = \max\left\{\frac{T_w+T_r}{\max\{T_d,\tau\}},1\right\}$$



MPL - Response Time with no Backfilling



Effect of Time Quantum

What is a good time quantum value?

▷ Tradeoff between overhead and responsiveness
▷ Some networks allow for some interleaving of communication and computation
▷ Different architectures have different overheads
▷ Cache pressure depends on application

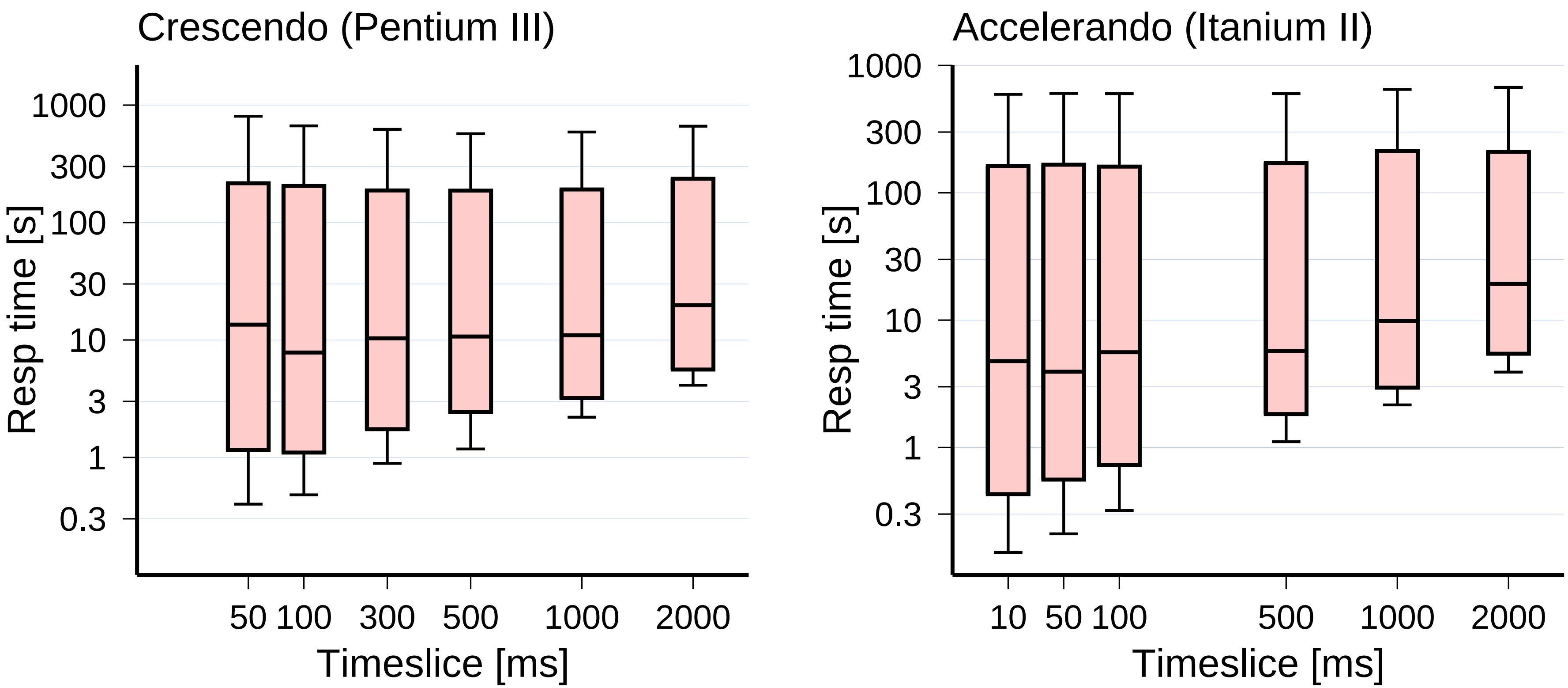
Effect of Time Quantum

What is a good time quantum value?

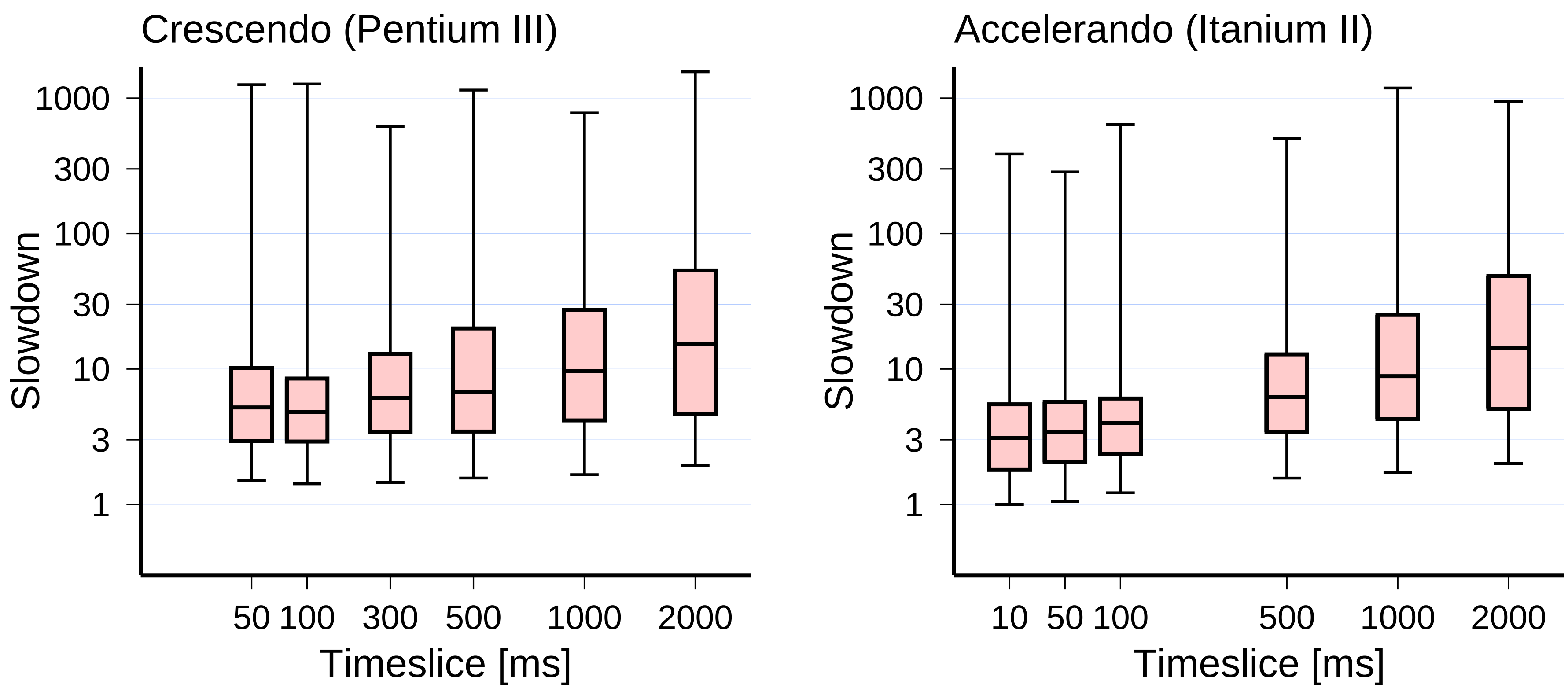
- ▷ Tradeoff between overhead and responsiveness
- ▷ Some networks allow for some interleaving of communication and computation
- ▷ Different architectures have different overheads
- ▷ Cache pressure depends on application

Lower (sustained time quantum) is better (utilization, responsiveness)

Time Quantum vs. Response Time



Time Quantum vs. Slowdown



Effect of Offered Load

What is the effect if increasing load in a dynamic workload?

▷ Different offered load values obtained by factoring run times algorithms compare?
▷ Comparison of Batch, Gang Scheduling, Two-Phase Spin-Block and Flexible Coscheduling
▷ Analysis of different types of jobs

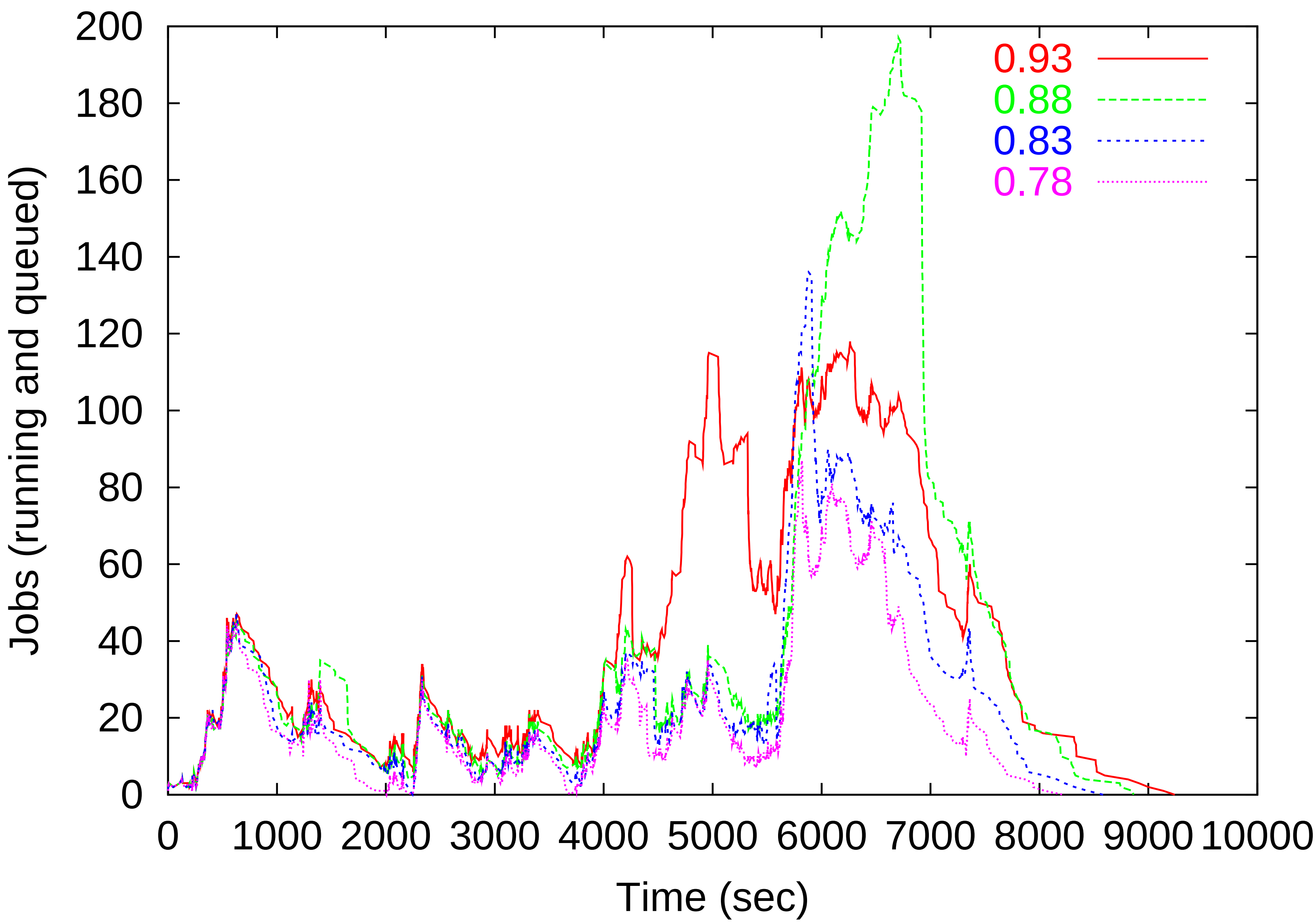
Effect of Offered Load

What is the effect if increasing load in a dynamic workload?

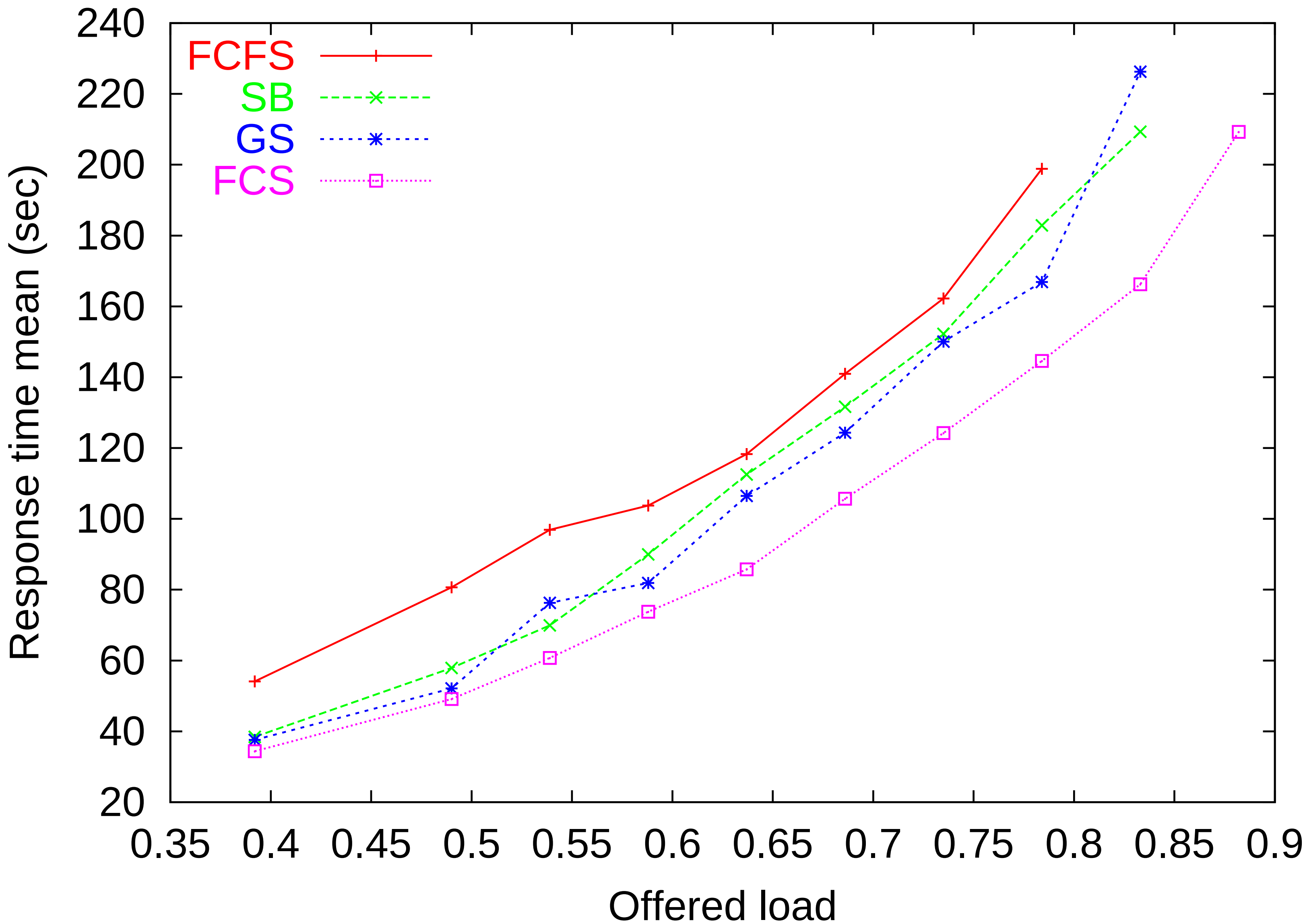
▷ Different offered load values obtained by factoring run times algorithms compare?
▷ Comparison of Batch, Gang Scheduling, Two-Phase Spin-Block and Flexible Coscheduling
▷ Analysis of different types of jobs

Caveat:
Finite workload hides saturation point

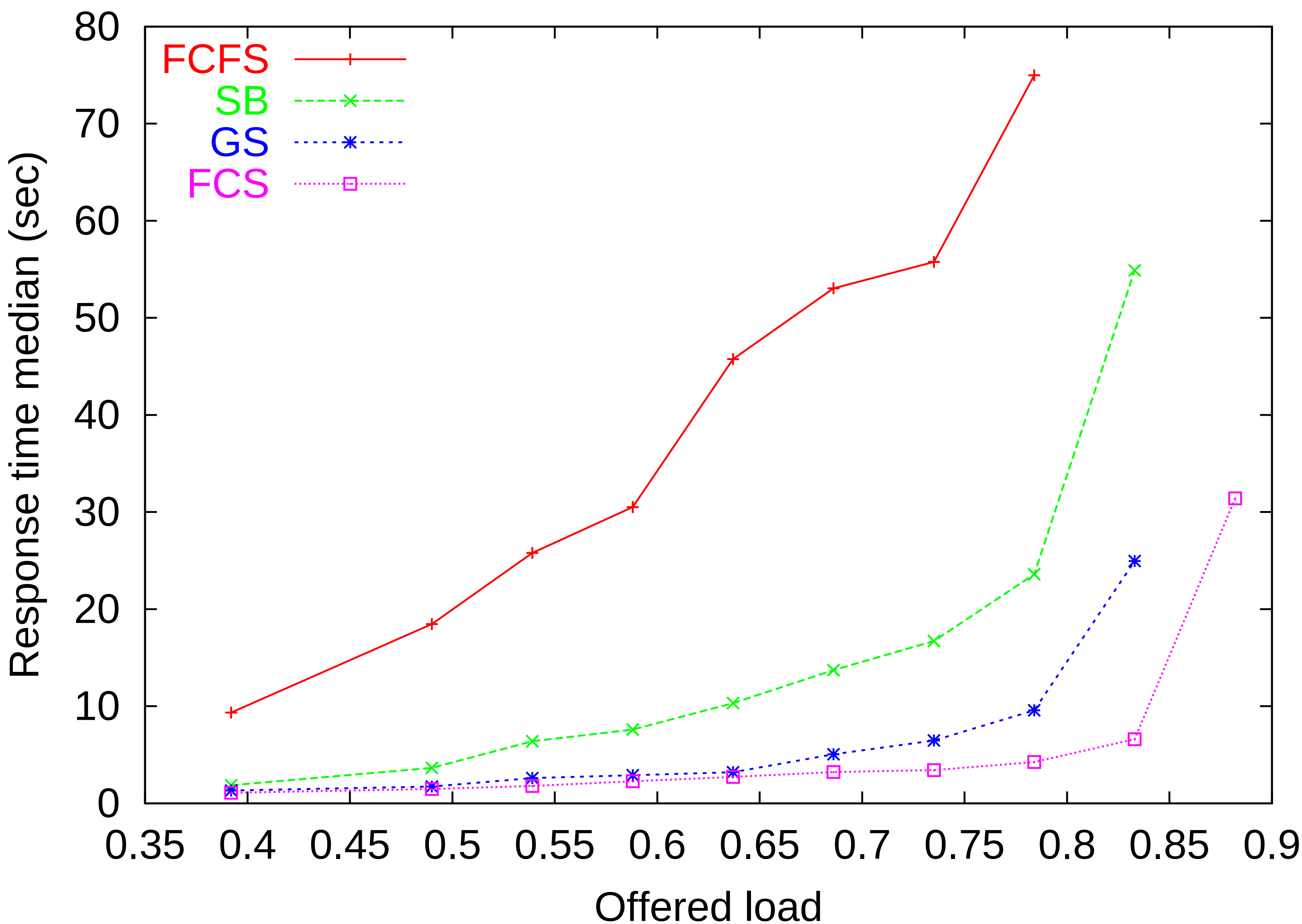
Determining Saturation (GS example)



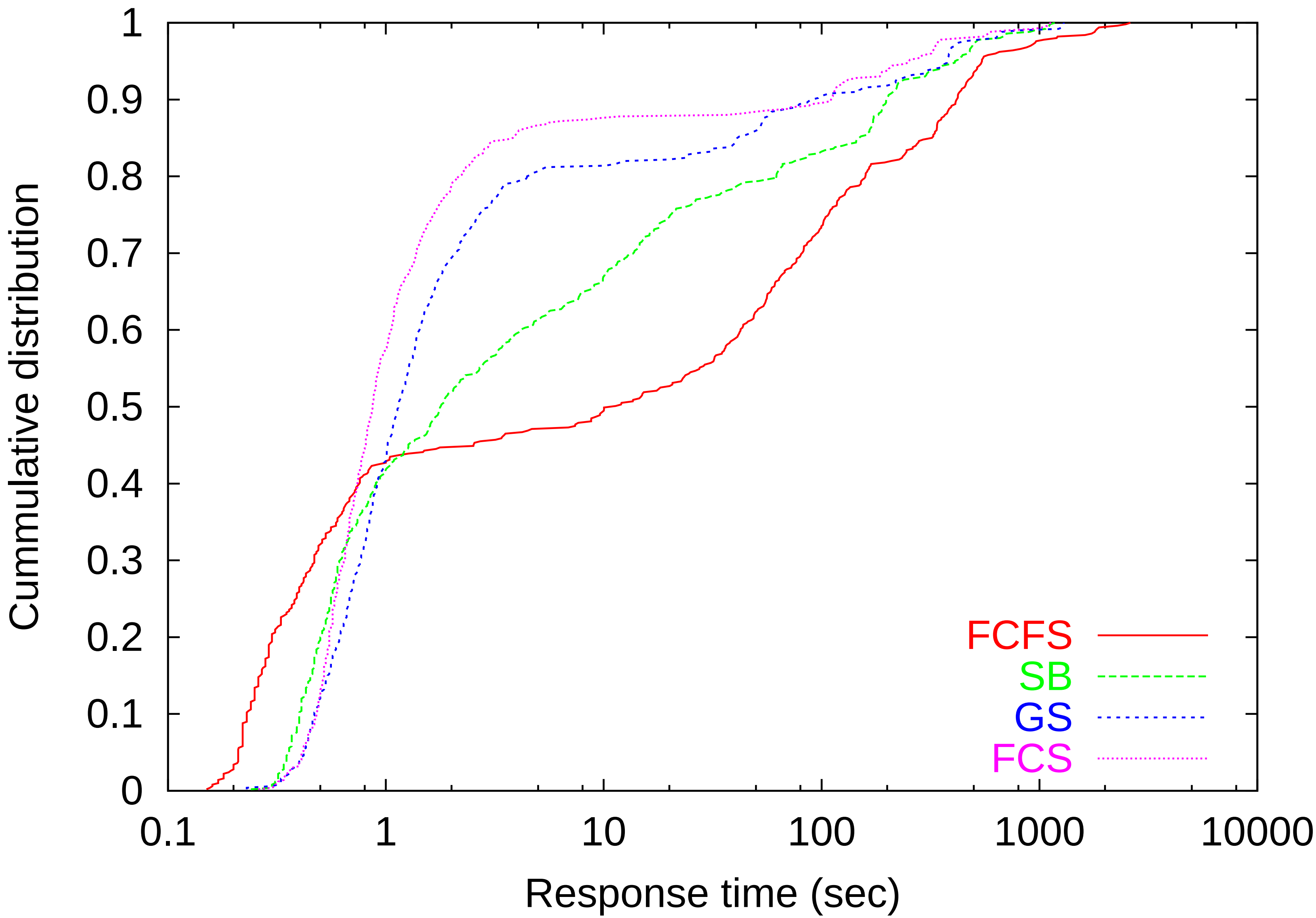
Mean Response Time



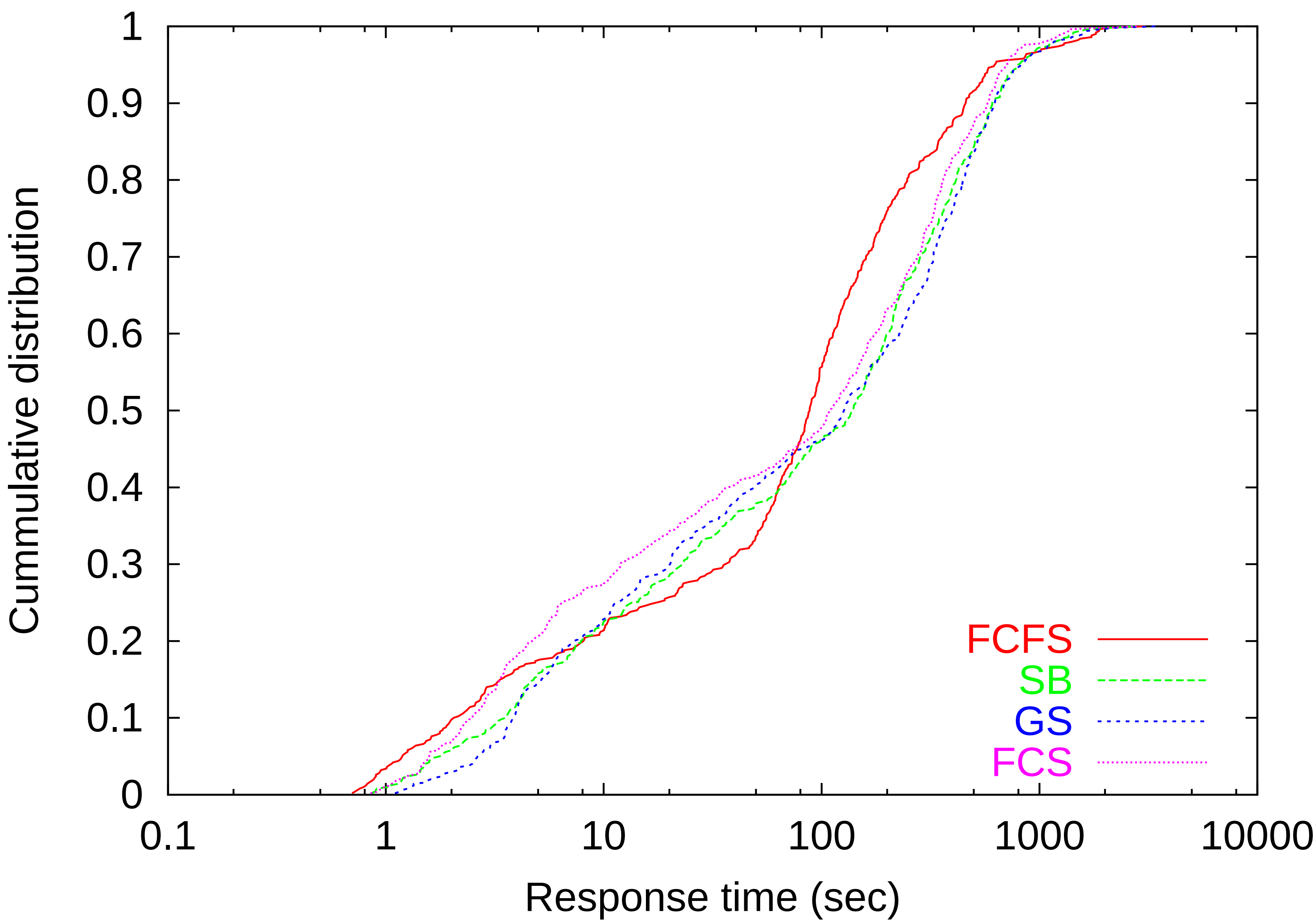
Median Response Time



Short Jobs CDF



Long Jobs CDF

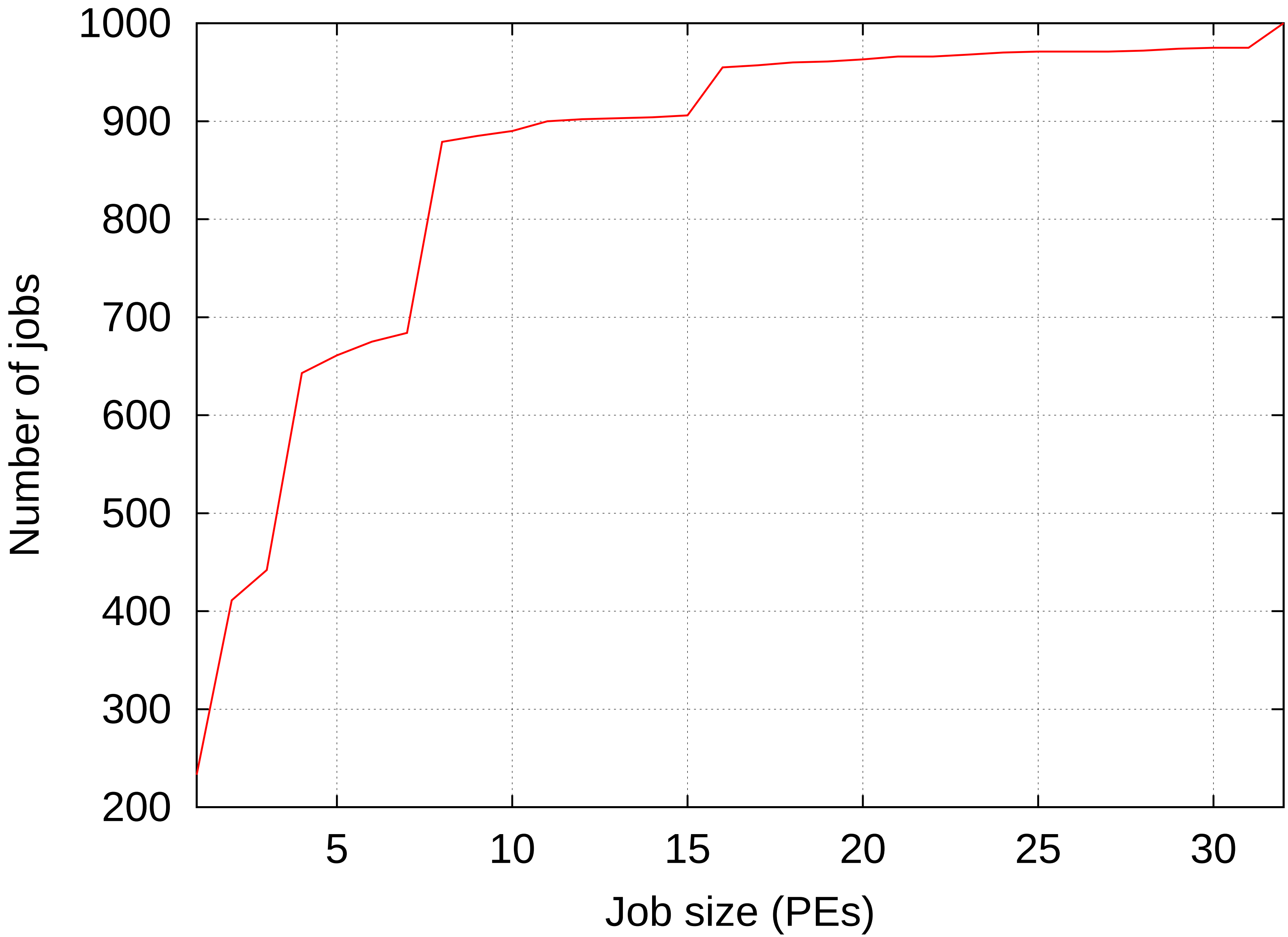


Conclusions

- Preemptive (coscheduling) techniques improve responsiveness and utilization over non-preemptive scheduling.
- Combining backfilling (knowledge of the future) with preemptive scheduling is indeed effective, even at low multiprogramming levels.
- Not all techniques are equal under dynamic workloads: The more flexible the scheduler, the denser the packing and the better the response time and utilization.

For more information:
`http://www.cs.huji.ac.il/~etcs`
email: etcs@cs.huji.ac.il

Some More Workload Properties...



FCS Phase Diagram