*Selective Preemption Strategies for Parallel Job Scheduling**

Rajkumar Kettimuthu, Vijay Subramani, Srividya Srinivasan, Thiagaraja B Gopalasamy, DK Panda and P Sadayappan

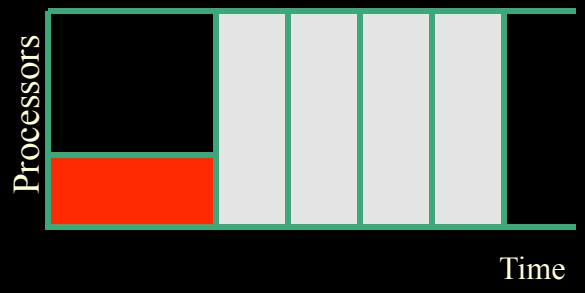
*Supported in part by a grant from Sandia Laboratories

Outline

- Background
- Motivation
- Related work
- Job Categorization
- Preemption Strategies
- User Estimates
- Overhead Modeling
- Conclusion

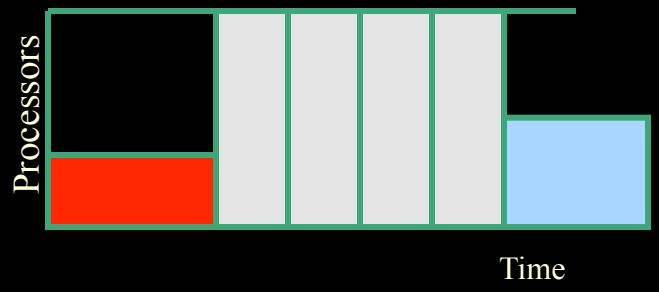
Background

- Backfilling
  - A later arriving job is allowed to leap frog previously queued jobs



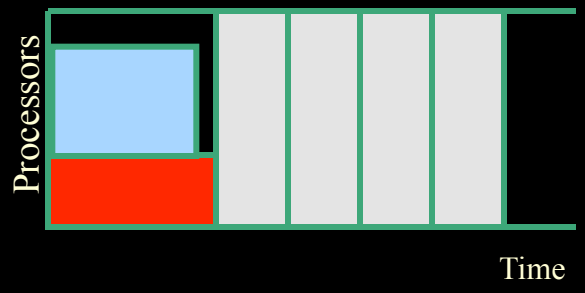
Background

- Backfilling
  - A later arriving job is allowed to leap frog previously queued jobs



Background

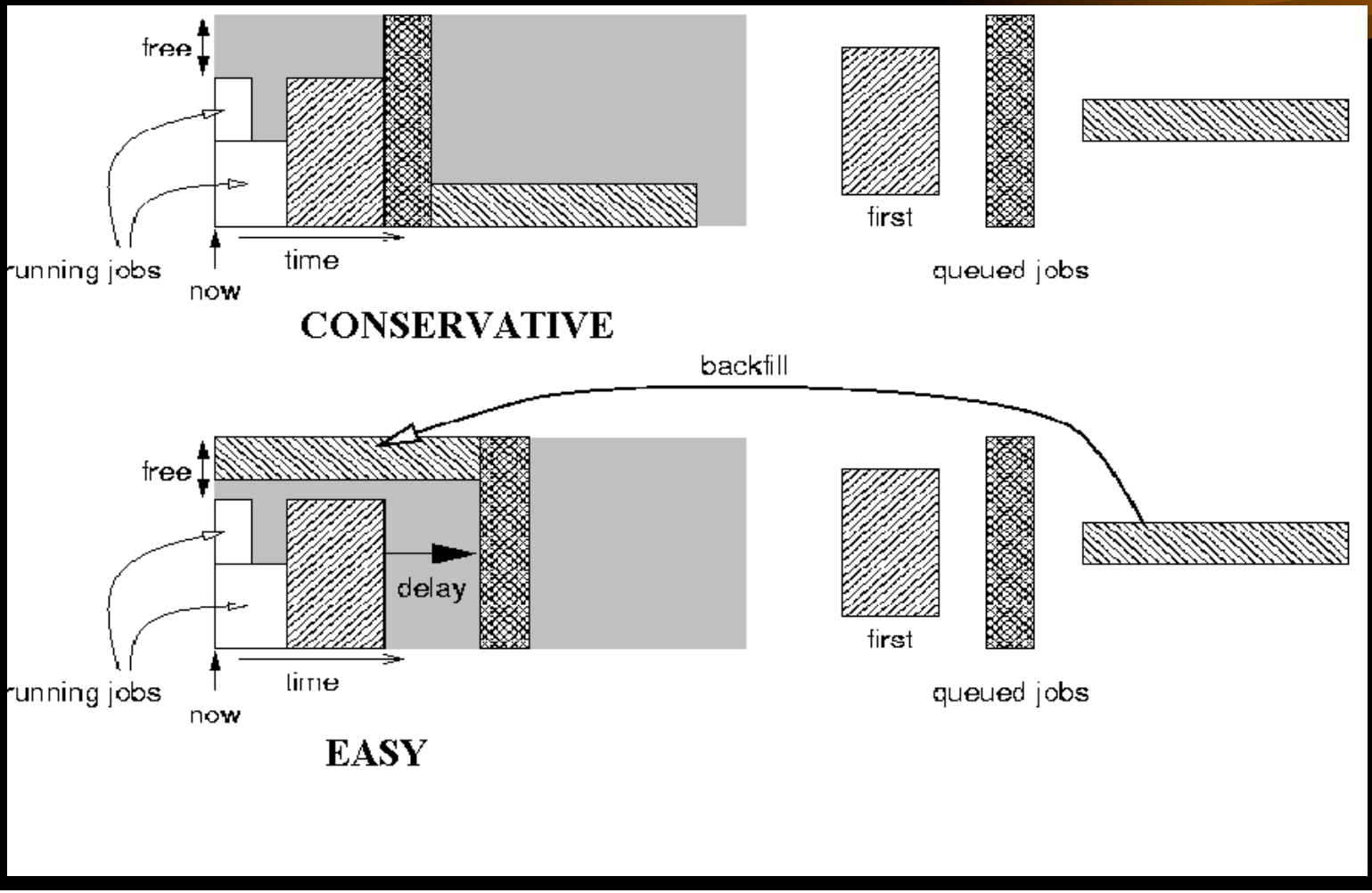
- Backfilling
  - A later arriving job is allowed to leap frog previously queued jobs



Background

- Conservative
  - Every job is given a reservation when it enters the system and a job is allowed to backfill only if it does not violate any of the previous reservations.
- EASY
  - Only the job at the head of the queue is given a reservation and a job is allowed to backfill if it does not violate this reservation

Background


free
running jobs
now
time
first
queued jobs
CONSERVATIVE
backfill
free
delay
running jobs
now
time
first
queued jobs
EASY

Motivation

- Temporarily suspend a long running job and allow a waiting short job to run to completion first
- Wait time of the short job is significantly decreased, without much fractional increase in the turn-around time of the long job

Motivation ... contd

- Consider a long job with runtime T_l. If after time t, a short job arrives with runtime T_s
- If the short job were run after completion of the long job, the average job turnaround time would be $(T_l + (T_l + T_s - t))/2$, or $T_l + (T_s - t)/2$.
- Instead, if the long job were suspended when the short job arrived, the turnaround times of the short and long jobs would be T_s and $(T_s + T_l)$ respectively, giving an average of $T_s + T_l /2$.

Motivation ... contd

- The average turnaround time with suspension is less if $T_s < T_l - t$
  - if the remaining runtime of the running job is greater than the runtime of the waiting job
- Suspension criteria simply based on remaining runtime may result in starvation
- Suspension strategy should bring down the average slowdown without increasing the worst case slowdowns.

Primary Contributions

- Selective-suspension strategy for pre-emptive scheduling of parallel jobs
- Characterization of the significant variability in the average slowdown for different job categories
- Impact of suspension on worst case slowdowns of various categories
- A tunable scheme to improve worst case slowdowns

Metric

- Bounded slowdown =

  Wait time + Max(run time,10) /

  Max(run time, 10)
- The threshold of 10 seconds - to limit the influence of very short jobs on the metric

Related work

- Most of the work on pre-emptive scheduling of parallel jobs considers the jobs to be malleable
- "Immediate Service (IS)" scheme - each arriving job given an immediate time-slice of 10 minutes, by suspending one or more running jobs if needed
- Selection of jobs for suspension was based on their instantaneous-xfactor {(wait time + total accumulated run time)/ (total accumulated run time)}

Immediate Service

- IS strategy significantly decrease the average job slowdown for the traces simulated.
- A potential shortcoming of the IS scheme is that its preemption decisions are not in any way reflective of the expected runtime of a job.
- IS can provide significant improvement to the slowdown of aborted jobs in the trace

Job Classification

- The CTC workload trace was used to evaluate the proposed schemes
- Previous works used overall average slowdown / turnaround time as metric to evaluate the performance
- But often the effect on one or more job categories is unacceptably negative
- Analyze the impact of preemptive scheduling strategies with respect to different job classes

Job Classification …contd

- To analyze the performance of jobs of different sizes and lengths
- Jobs were classified into 16 categories:
  - Four based on their run time – very short, short, medium and long
  - Four based on the number of processors – sequential, narrow, wide and very wide

Job Classification …contd

- Overall average slowdown - 3.58

	1 Proc	2-8 Procs	9-32 Procs	>32 Procs
0-10 min	2.6	4.76	13.01	34.07
10min - 1Hr	1.26	1.76	3.04	7.14
1Hr-8Hrs	1.13	1.43	1.88	1.63
>8Hrs	1.03	1.05	1.09	1.15

Selective Suspension

- An idle job can preempt a running job only if its preemption threshold is sufficiently higher than that of the later
- Suspension factor (SF) is used to control the rate of suspensions
- SF specifies the minimum ratio of the suspension threshold of idle job to that of running job for preemption to occur

Suspension Threshold

- Pre-emptive scheduling aims at providing lower delay to short jobs relative to long jobs
- Suspension criteria used is *xfactor*, which increases rapidly for short jobs and gradually for long jobs
- Xfactor =

  (Wait time + estimated run time) / estimated run time)

Selective Suspension

- Let T_1 and T_2 be two identical tasks submitted to the scheduler at the same time
- Both tasks require the entire system for execution
- Initially, both tasks have a suspension threshold of 1
- T_1 starts instantaneously
- Suspension threshold of a task remains constant when the task executes and increases when the task waits

Selective Suspension

- T_1 and T_2 suspend each other repeatedly until they complete
- Number of suspensions is related to the suspension factor
- $s = 2^{1/(n+1)}$; s - suspension factor, n - maximum suspensions
- In this scenario, for no suspension to occur (n=0) - suspension factor has to be set to 2

Selective Suspension

- In a workload, there will be jobs of varying length and width
  - With s=1, the number of suspensions is very large
- To reduce the number of suspensions, different suspension factors between 1.5 and 5 were used
- Backfill scheduling schemes use job reservations for one or more jobs in the queue to avoid starvation
- But the start time guarantees do not make much sense in the presence of preemption

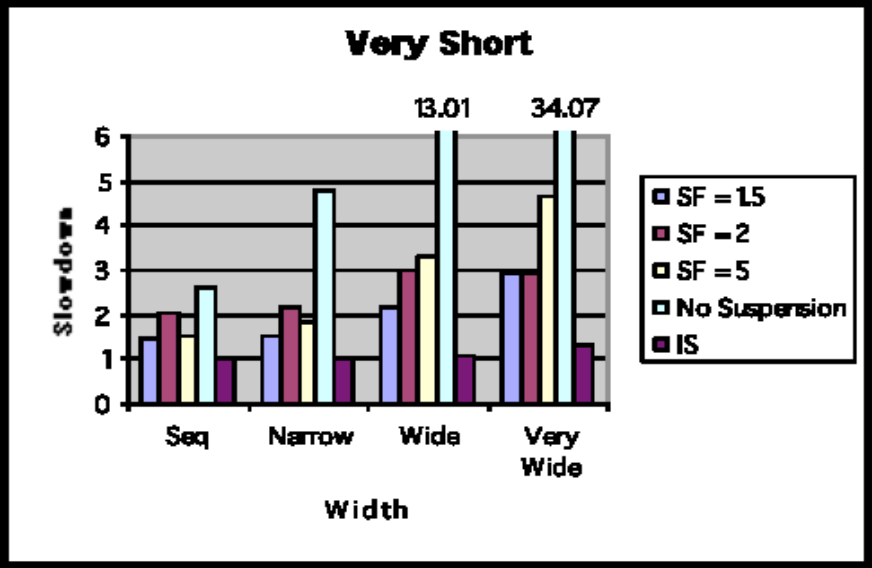
Selective Suspension

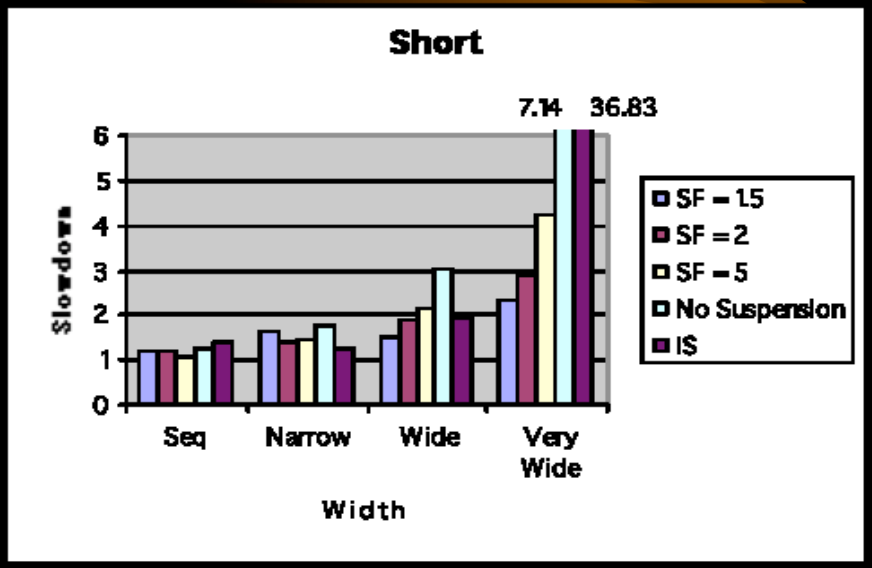
- Since the SS strategy uses the expected slowdown as the suspension threshold, there is an automatic guarantee of freedom from starvation
- Job's expected slowdown factor will get large enough that it will be able to preempt some running job and begin execution
- No start time guarantees in our preemption schemes

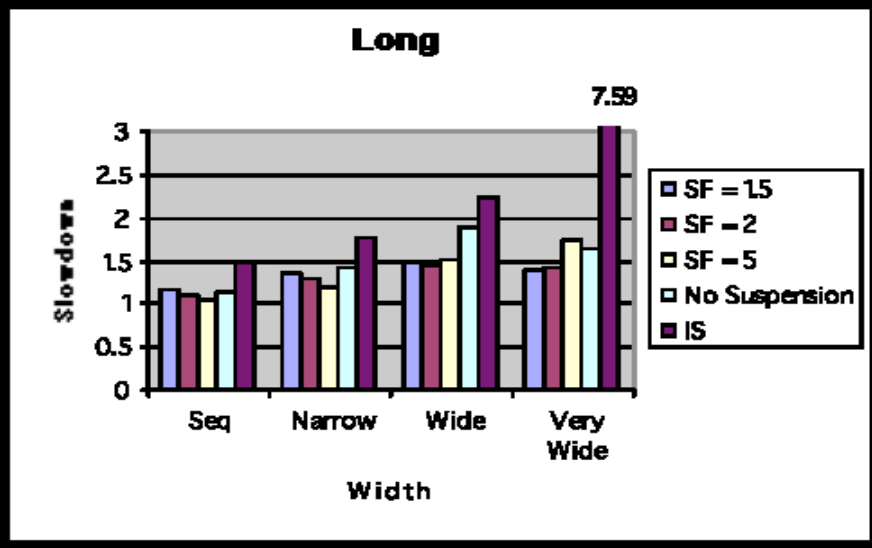
Selective Suspension

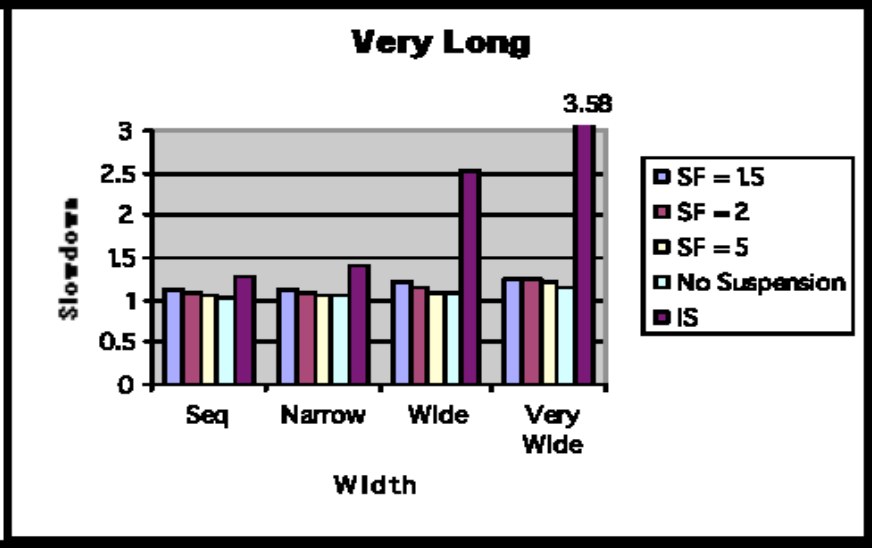
- Wide jobs have a higher probability of waiting longer in the queue than narrow jobs with comparable xfactor
- Number of nodes requested by a suspending job should be at least half of the number of nodes requested by the job that it suspends

Results

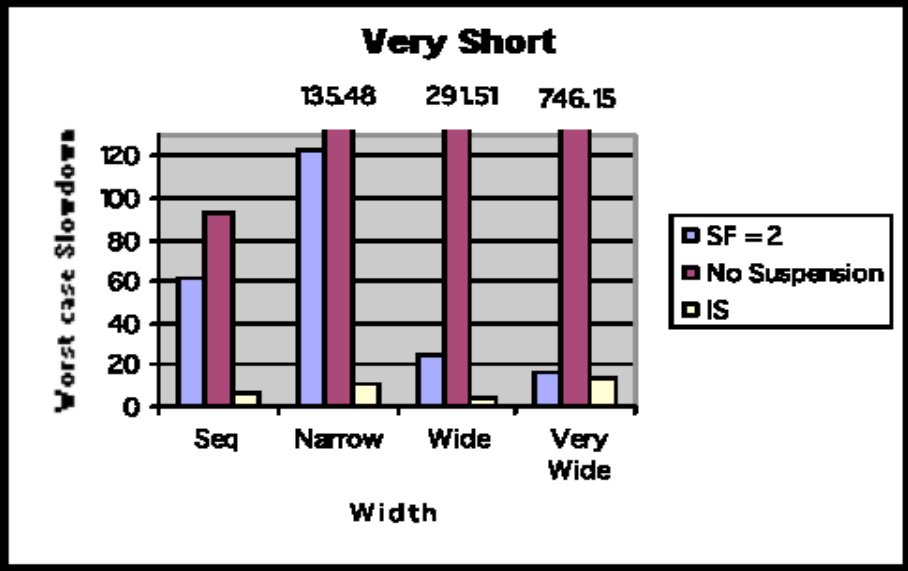

Very Short
13.01
34.07
Slowdown
0
1
2
3
4
5
6
Seq
Narrow
Wide
Very Wide
Width
SF = 1.5
SF = 2
SF = 5
No Suspension
IS

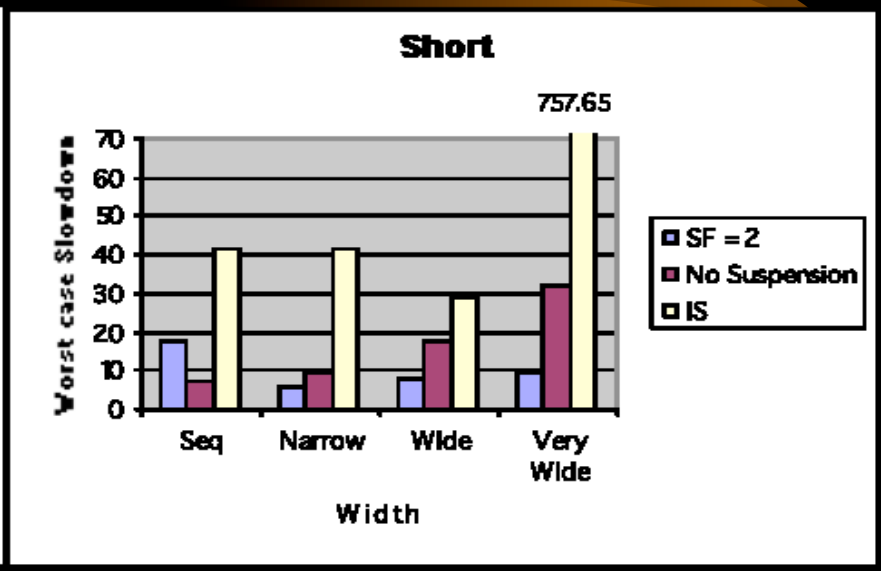

Short
7.14
36.83
Slowdown
0
1
2
3
4
5
6
Seq
Narrow
Wide
Very Wide
Width
SF = 1.5
SF = 2
SF = 5
No Suspension
IS

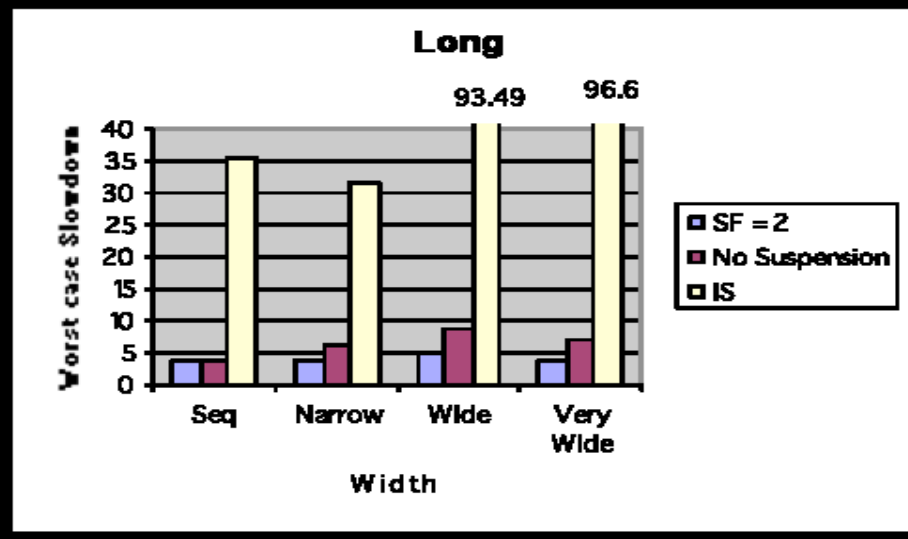

Long
7.59
Slowdown
0
0.5
1
1.5
2
2.5
3
Seq
Narrow
Wide
Very Wide
Width
SF = 1.5
SF = 2
SF = 5
No Suspension
IS

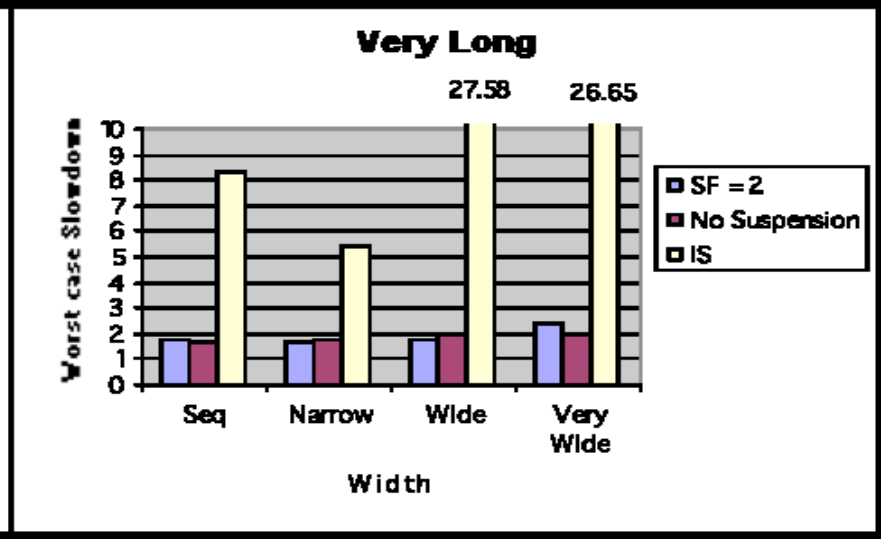

Very Long
3.58
Slowdown
0
0.5
1
1.5
2
2.5
3
Seq
Narrow
Wide
Very Wide
Width
SF = 1.5
SF = 2
SF = 5
No Suspension
IS

Worst Case Results





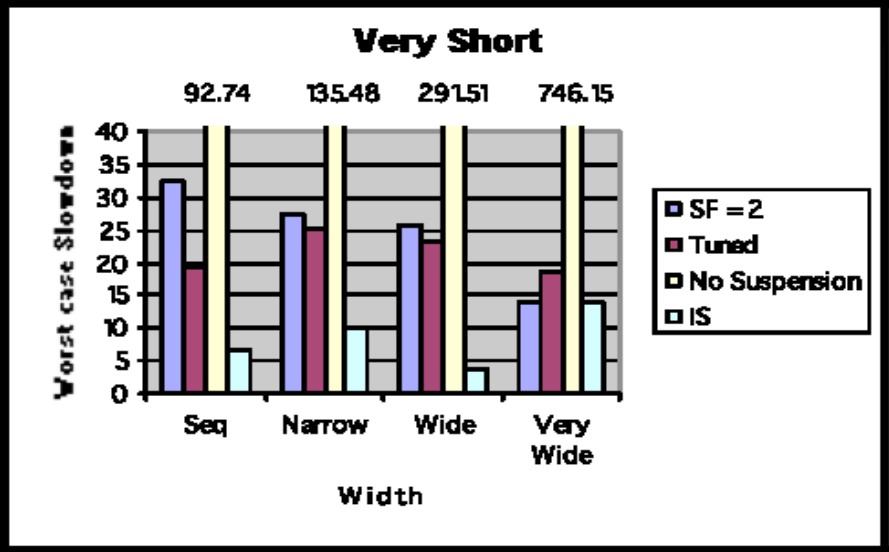


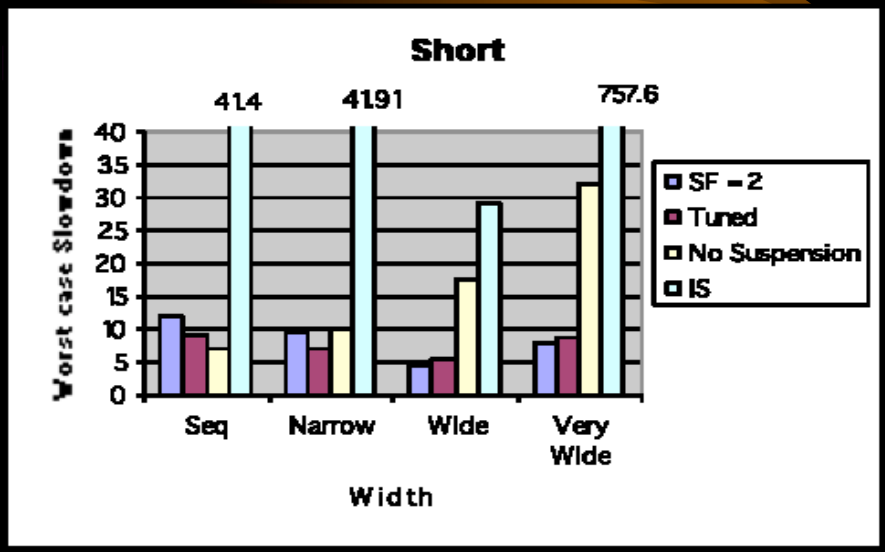


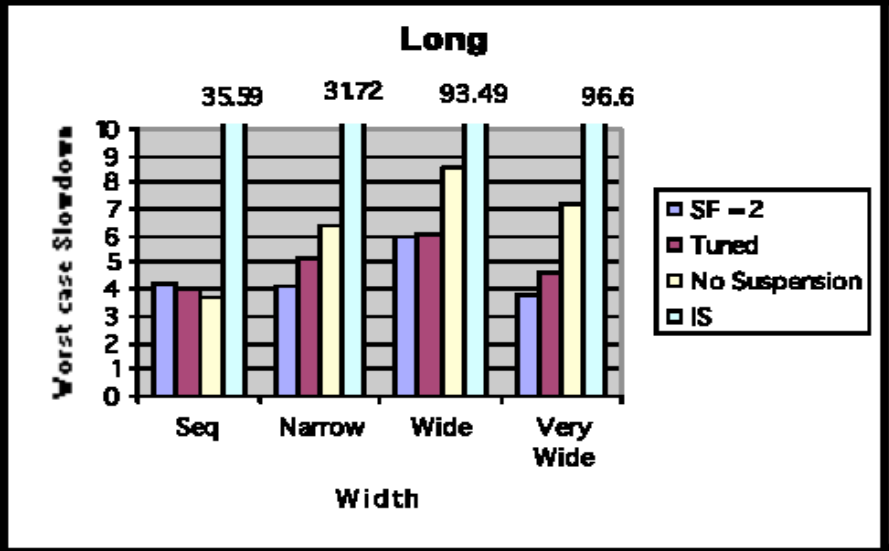
Tunable Selective Suspension

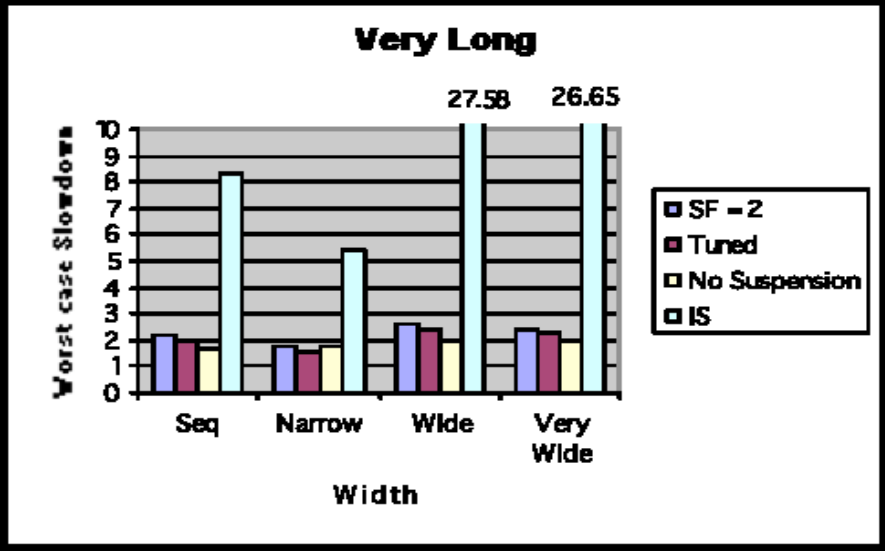
- This is done by controlling the variance in the slowdowns by associating a limit with each job.
- Preemption of a job is disabled when its threshold exceeds this limit. This limit is set to 1.5 times the average slowdown of the category that the job belongs to.

Worst Case

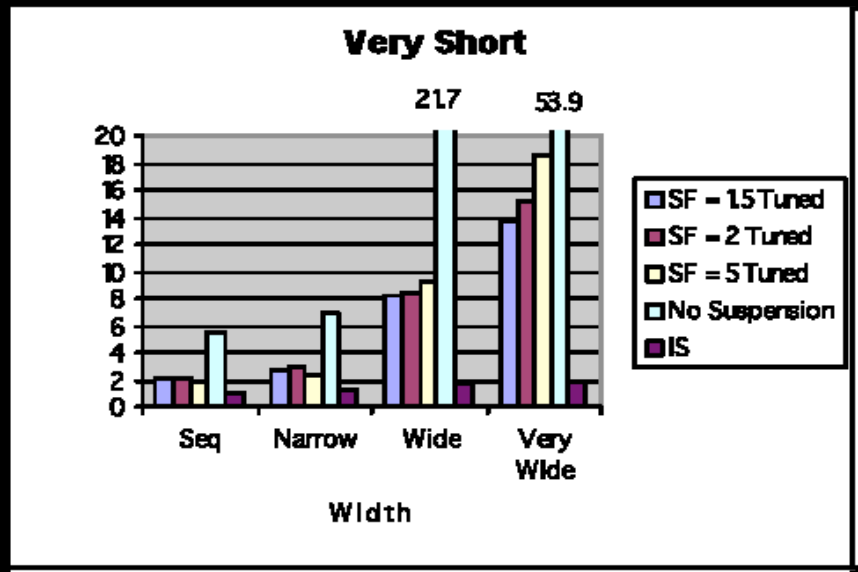

Very Short
92.74
135.48
291.51
746.15
Worst case Slowdown
40
35
30
25
20
15
10
5
0
Seq
Narrow
Wide
Very Wide
Width
SF = 2
Tuned
No Suspension
IS

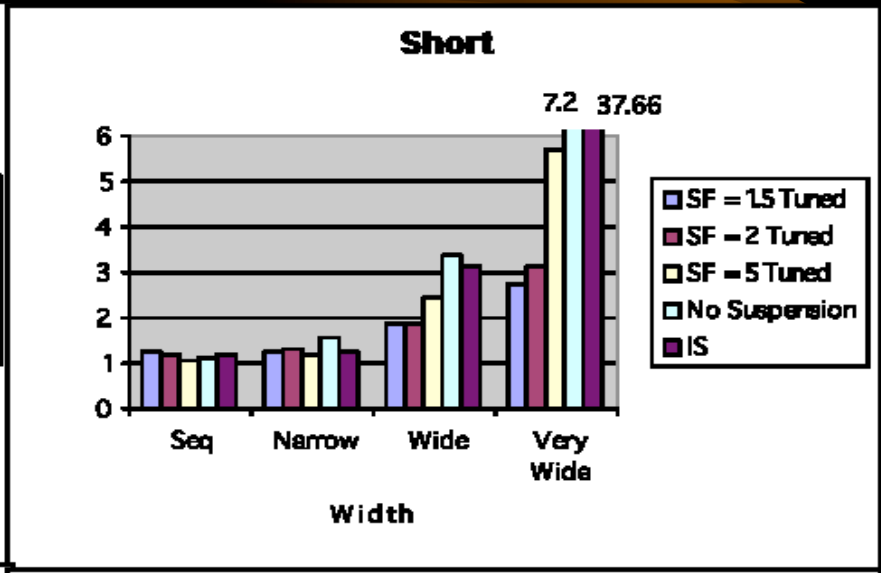

Short
414
4191
757.6
Worst case Slowdown
40
35
30
25
20
15
10
5
0
Seq
Narrow
Wide
Very Wide
Width
SF = 2
Tuned
No Suspension
IS

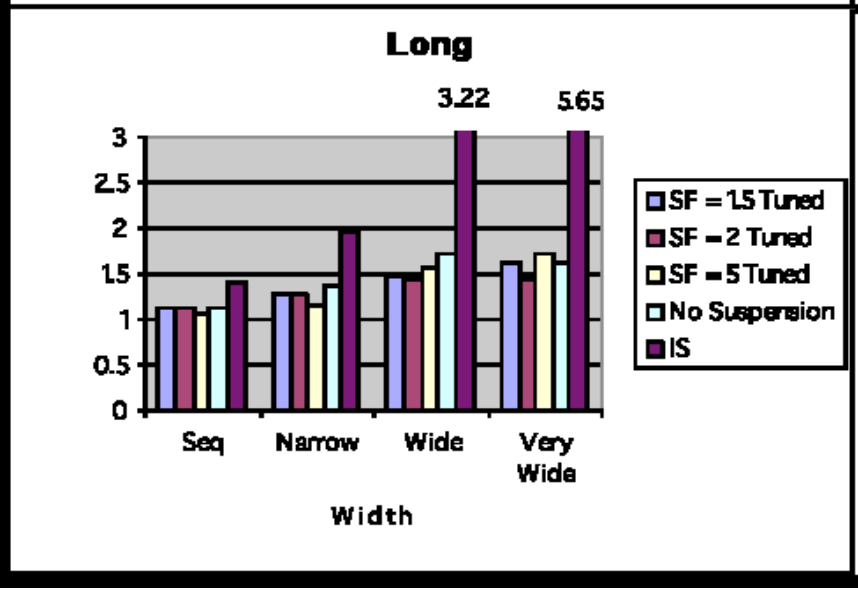

Long
35.59
31.72
93.49
96.6
Worst case Slowdown
10
9
8
7
6
5
4
3
2
1
0
Seq
Narrow
Wide
Very Wide
Width
SF = 2
Tuned
No Suspension
IS

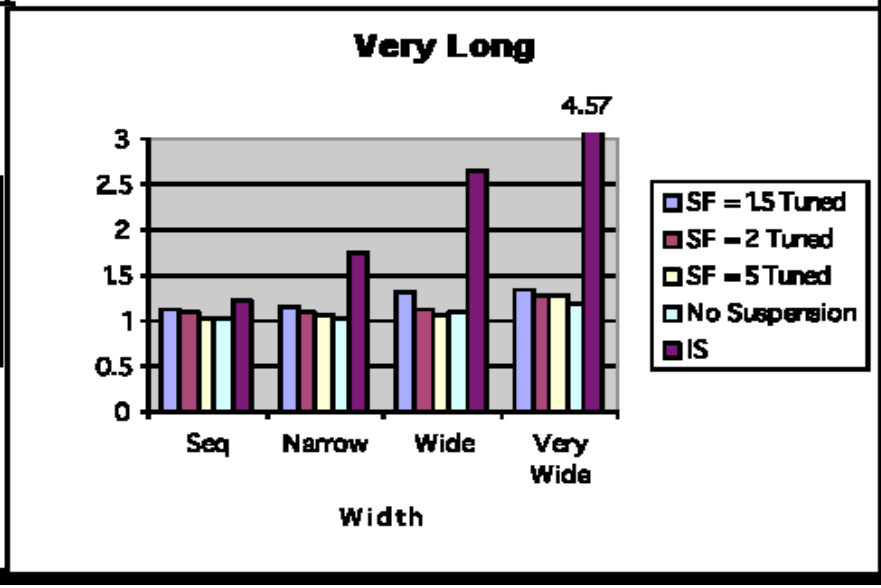

Very Long
27.58
26.65
Worst case Slowdown
10
9
8
7
6
5
4
3
2
1
0
Seq
Narrow
Wide
Very Wide
Width
SF = 2
Tuned
No Suspension
IS

User estimates

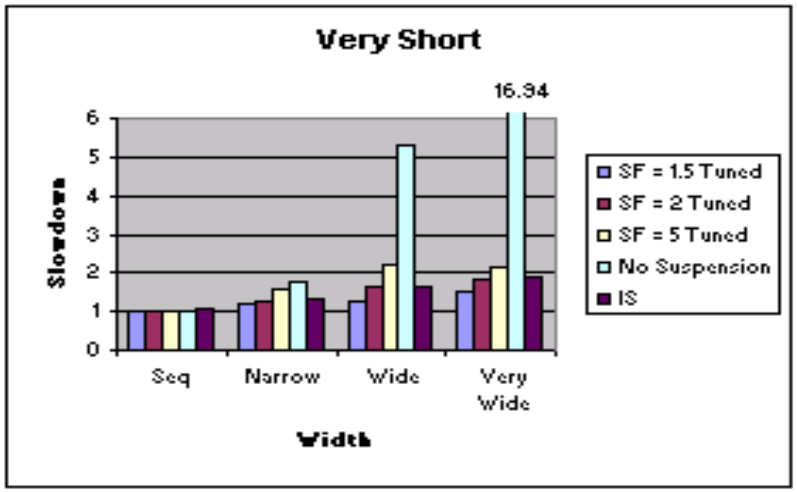


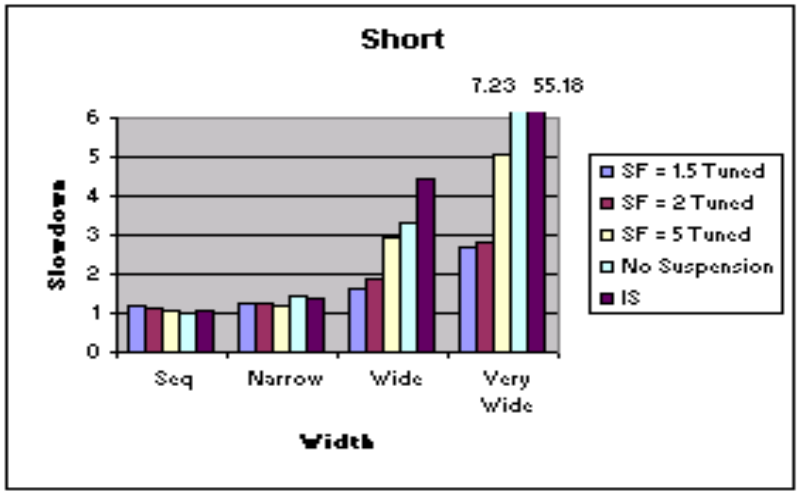


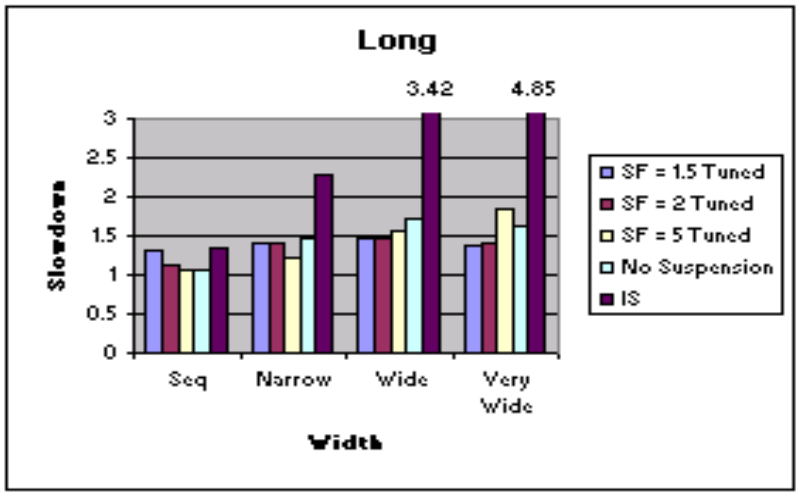


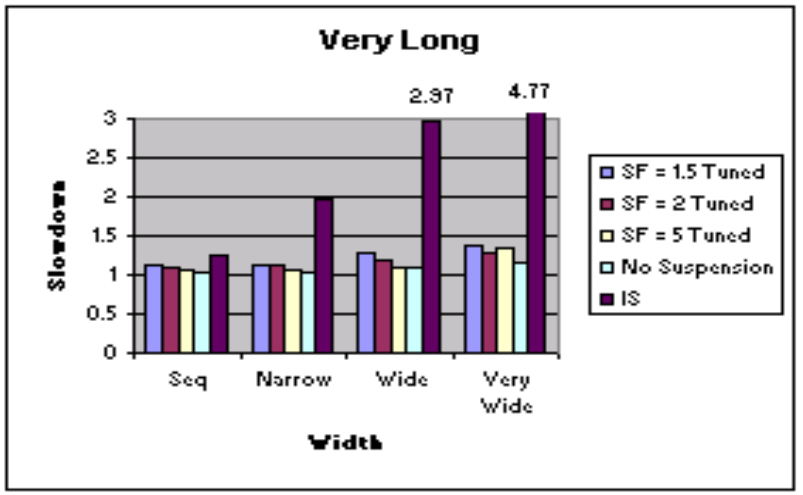


Good Estimators

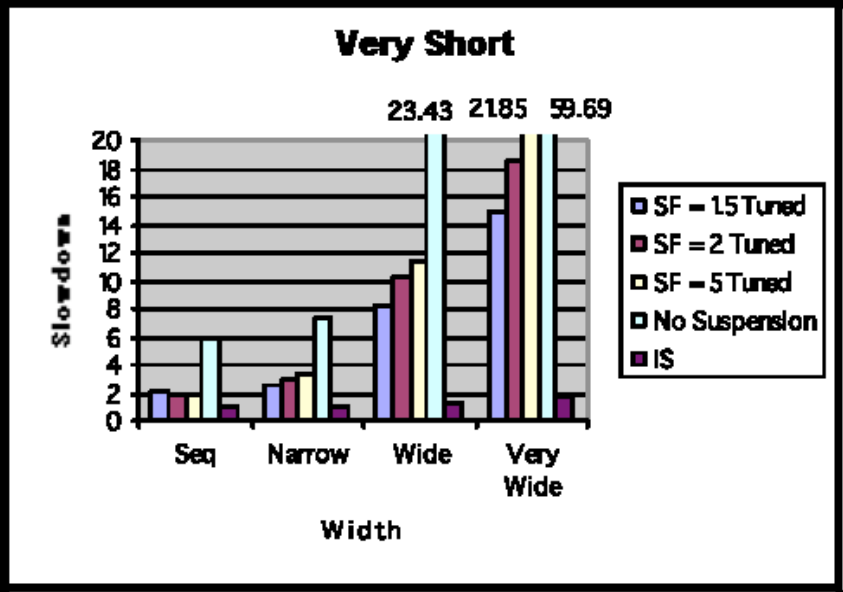


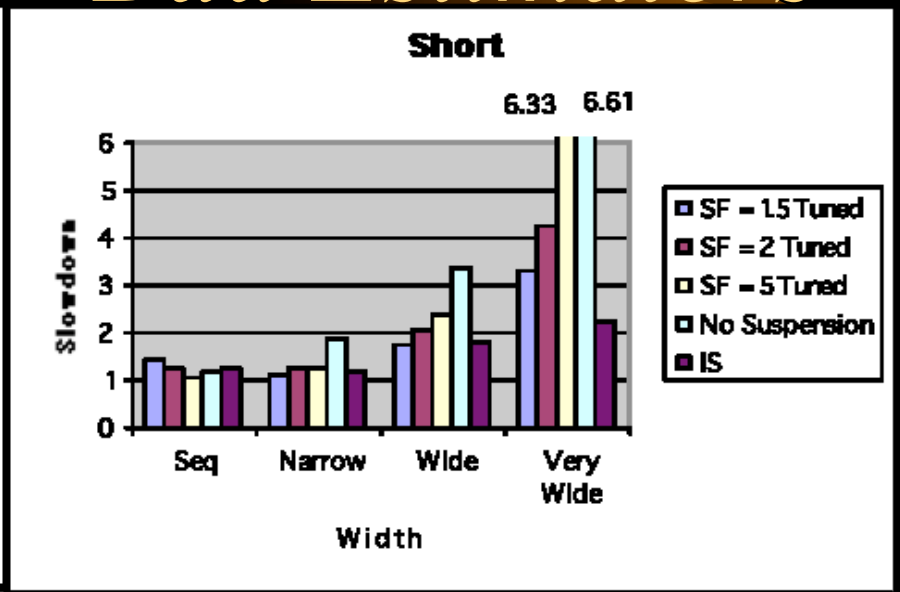


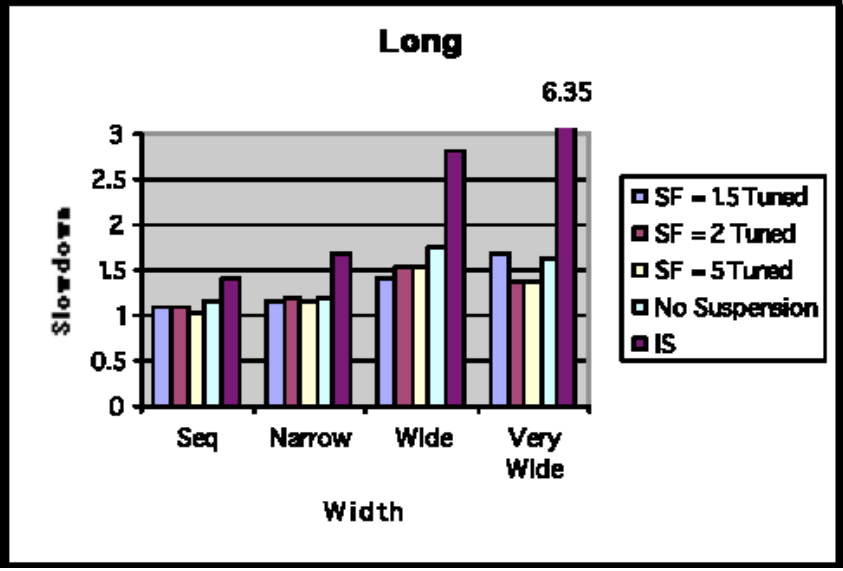


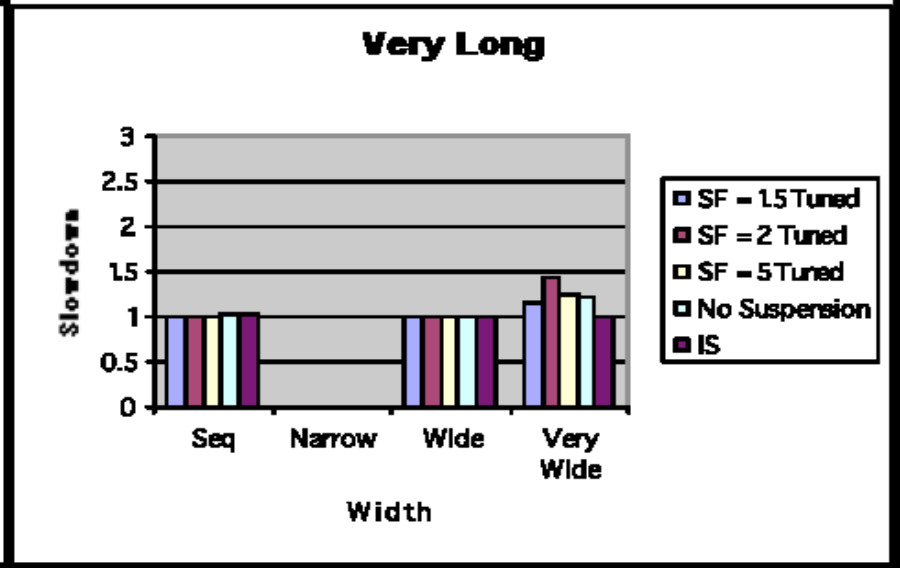


Bad Estimators





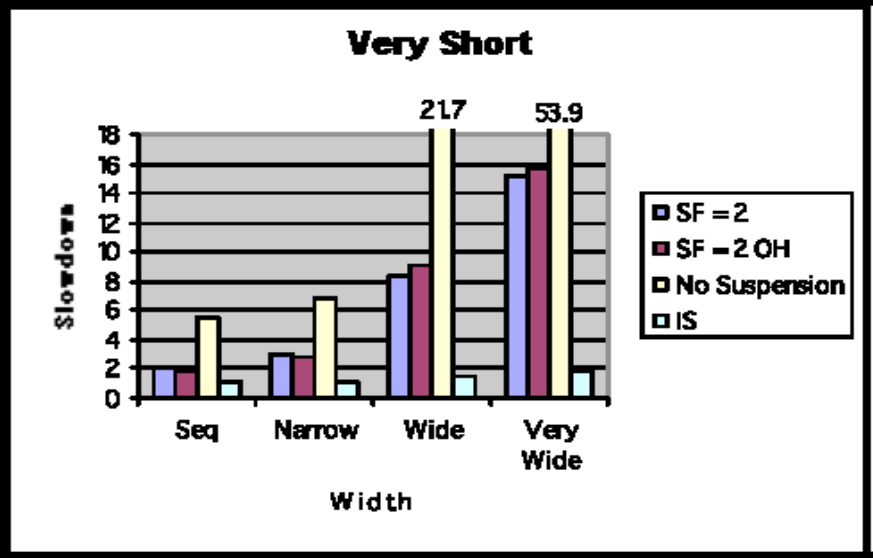


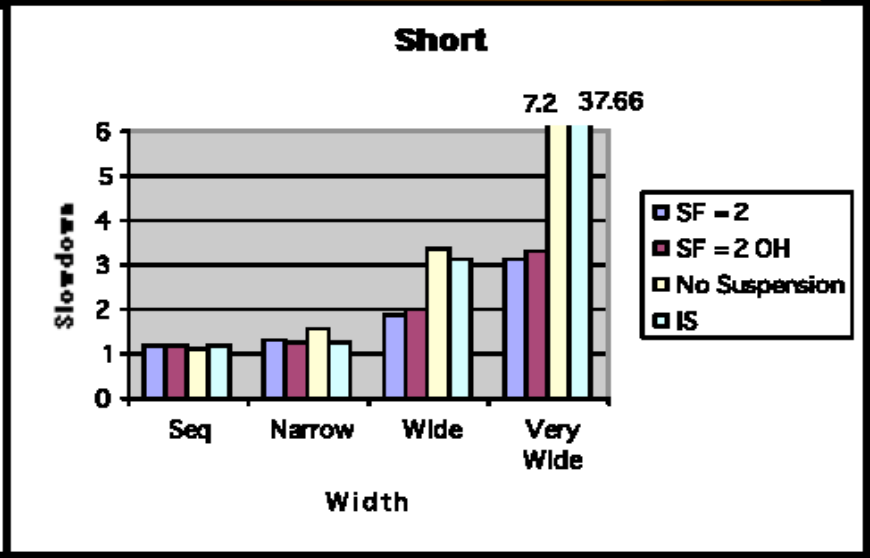


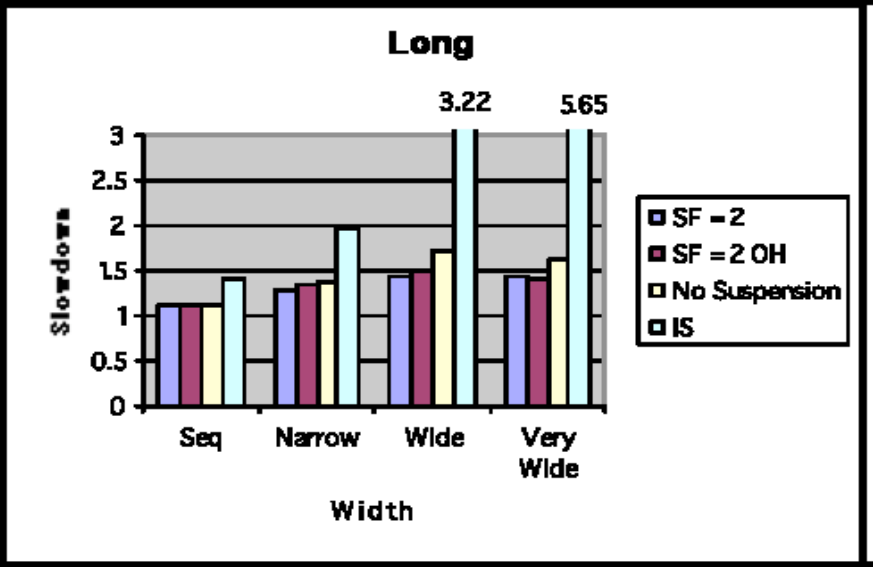
Job Suspension Overhead

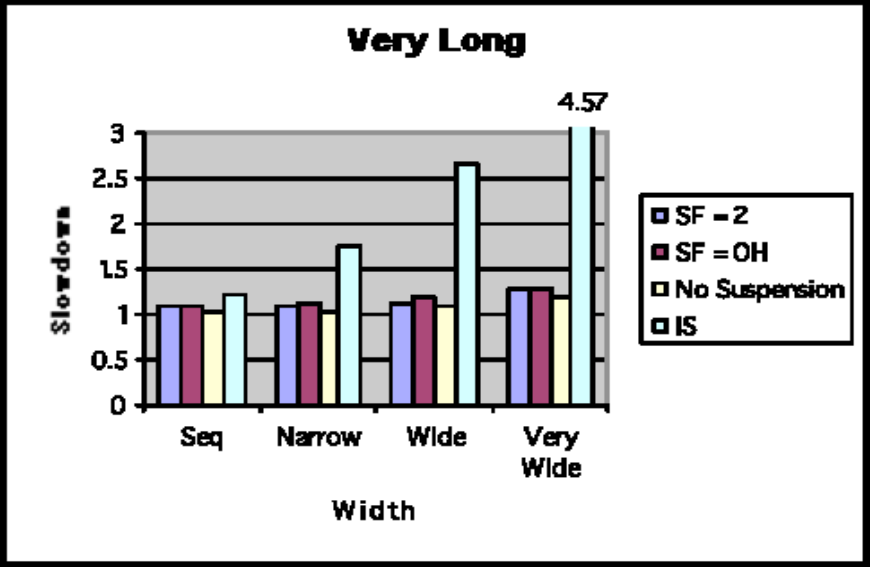
- Job trace did not have information about memory requirements
- Random and uniformly distributed between 100 MB and 1 GB
- Overhead - time taken to write main memory used by job to disk
- Transfer rate – 2 MBps.

Overhead


Very Short
21.7
53.9
18
16
14
12
10
8
6
4
2
0
Slowdown
Seq
Narrow
Wide
Very Wide
Width
SF = 2
SF = 2 OH
No Suspension
IS


Short
7.2
37.66
6
5
4
3
2
1
0
Slowdown
Seq
Narrow
Wide
Very Wide
Width
SF = 2
SF = 2 OH
No Suspension
IS


Long
3.22
5.65
3
2.5
2
1.5
1
0.5
0
Slowdown
Seq
Narrow
Wide
Very Wide
Width
SF = 2
SF = 2 OH
No Suspension
IS


Very Long
4.57
3
2.5
2
1.5
1
0.5
0
Slowdown
Seq
Narrow
Wide
Very Wide
Width
SF = 2
SF = OH
No Suspension
IS

Conclusion

- Proposed a tunable, selective suspension scheme
  - Significant improvement in the average and worst case slowdowns of several job categories
- Shown to provide better slowdown for most job categories over a previously proposed Immediate Service scheme
- Evaluated the schemes in the presence of over estimations
- Proposed schemes provide greater benefits to well estimated jobs