Improving Parallel Job Scheduling by Combining Gang Scheduling and Backfilling Techniques

Presenter: Benjie Miao

Overview

- IPDPS 2000
- Author: Y. Zhang, H. Franke, J. E. Moreira, A. Sivasubramaniam
- Institution: Pennsylvania State University, IBM T.J. Watson Research Center

Background: Job scheduling

- In an HPC platform, jobs are requested from time to time
  - The platform needs to distribute its resources (cores, networks, etc.) wisely for a better overall performance
- *"Parallel job scheduling consists of at least two interdependent steps: **the allocation of tasks to the processors** (space-sharing) and **the scheduling of the tasks over time** (time-sharing)"*
- Space-sharing policy: static, adaptive, dynamic
- Time-sharing policy: independent local scheduling (ILS), dynamic-based co-scheduling (DCS)

Abawajy J H , Dandamudi S P . Time/space sharing distributed job scheduling policy in a workstation cluster environment[C]// International Conference on Parallel Computing in Electrical Engineering. IEEE, 2000:116-120.

Background: Space-sharing Policy

- Trivial policy/Baseline: Each node is exclusively assigned to a job
  - A ‘static’ policy
  - Cons:
    - poor utilization: nodes can be left empty despite a waiting queues of job
    - High wait and response time
- Approaches to improve performance:
  - Backfilling: assign unutilized nodes to jobs that are behind in the priority queue of waiting jobs, rather than keep them idle
  - Gang scheduling/coscheduling: add a time-sharing dimension to space sharing by slicing time axis into multiple space-shared virtual machine
- Both requires an estimate of job execution time

Main idea of this paper

- Question: Can we combine gang scheduling and backfilling in some sophisticated way for a better scheduling performance?

- The answer: **Yes!** A combined strategy is always better than the individual gang scheduling or backfilling policy.

- This paper uses simulation and synthetic workload to evaluate the performance

Modeling parallel job workloads

- Use stochastic-model-based simulation to evaluate policies
  - Need a characterization technique and a procedure to synthetically generate the expected workloads

- Methodology:
  - fit a typical workload with mathematical models
  - generate synthetic workloads based on the derived mathematical models
  - simulate the behavior of the different scheduling policies for those workloads
  - determine the parameters of interest for the different scheduling policies

Workload model

- Hyper Erlang Distributions of Common Order
- Using a parameter Ω to indicate the uncertainty of the estimation on job execution time
- Baseline workload: 10000 jobs, size from 1 to 256 nodes

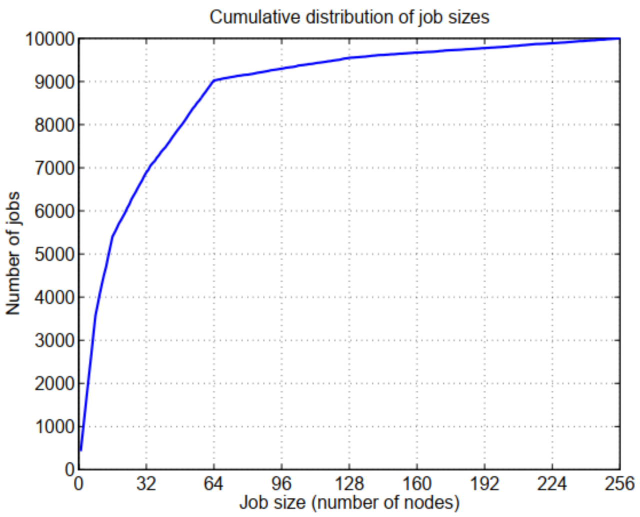


Figure 1. Distribution of job sizes in workload.

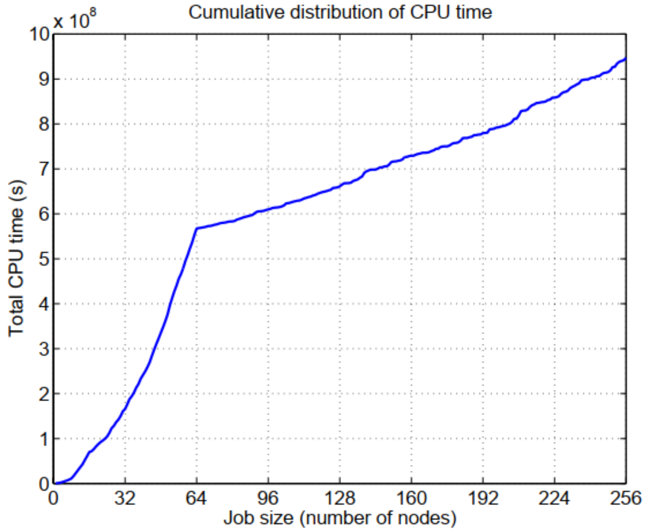


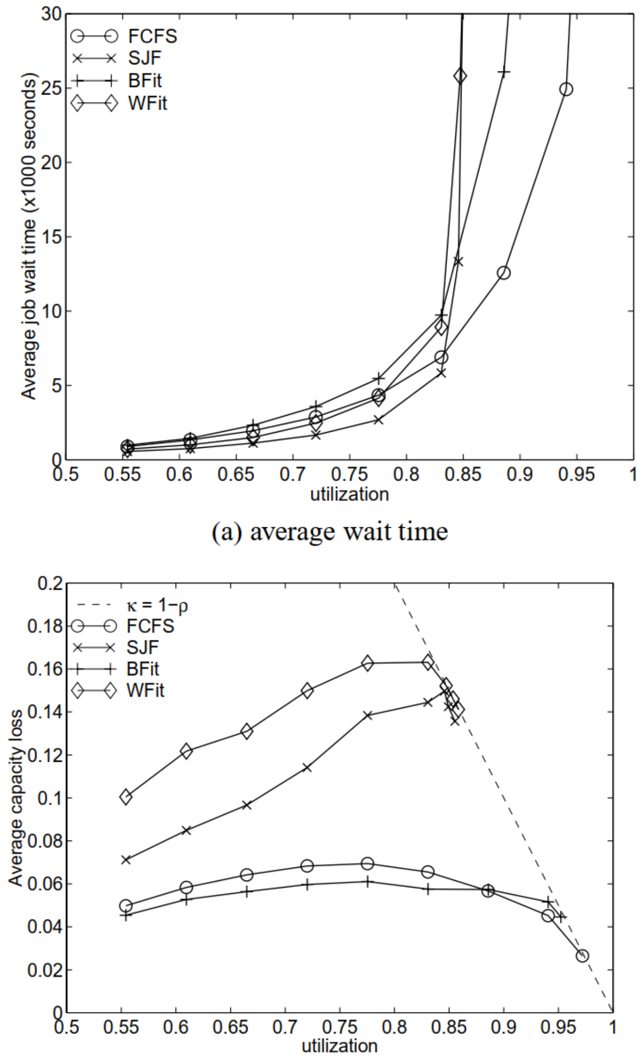
Figure 2. Distribution of cpu time in workload.

Metrics for Performance Evaluation

- $Response\ time\ = finish\ time\ - arrival\ time$
- $Wait\ time = start\ time\ - arrival\ time$
- $Slowdown\ = \frac{\max(response\ time, 10)}{\max(response\ time\ in\ a\ dedicated\ settings, 10)}$
- User's perspective:
  - Quality of service: average job slowdown & average job wait time
  - Fairness: average/std. dev. of slowdown/wait time for small/large/all jobs
- System's perspective:
  - Utilization: fraction of total system resources in use
  - Capacity loss: #idle nodes when some other jobs are waiting

Queueing policies with backfilling

- Queueing policy: a set of rules that give priority to some jobs
- Four baseline queueing policy
  - FCFS, Shortest job first, Best fit, Worst fit
- Each can be combined with backfilling
- When estimation of execution time is accurate, **FCFS** is the best when workload is high

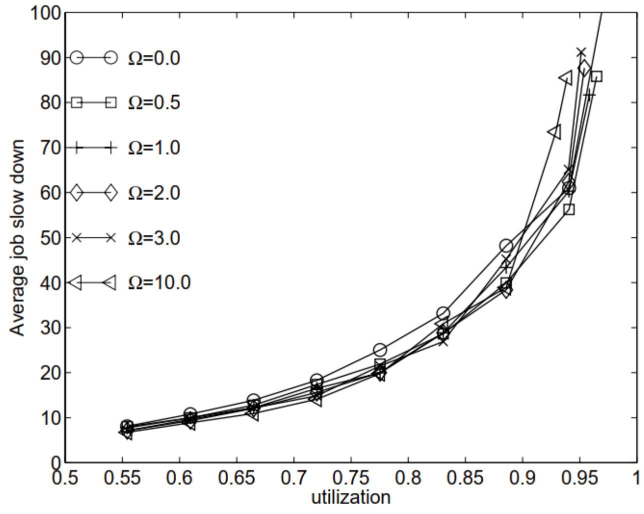


(a) average wait time

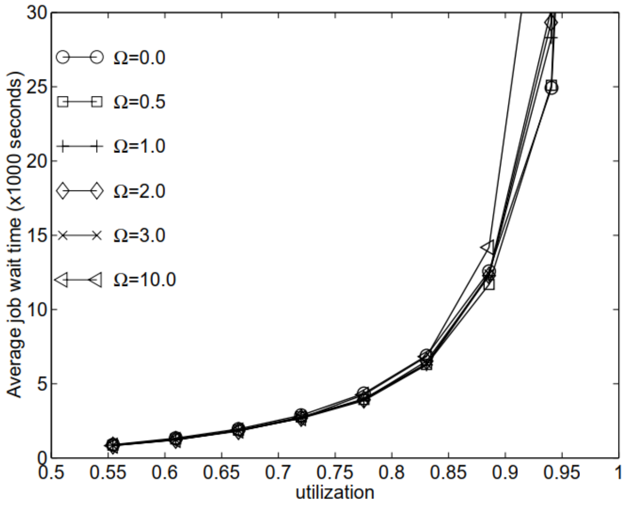
(b) average capacity loss

Impact of overestimation on backfilling

- Literature shows that there is little correlation between estimated (provided by users) and actual execution time
- Jobs are killed when the estimated time is reached, users have an incentive to **overestimate** the execution time



(a) average slowdown



(b) average wait time

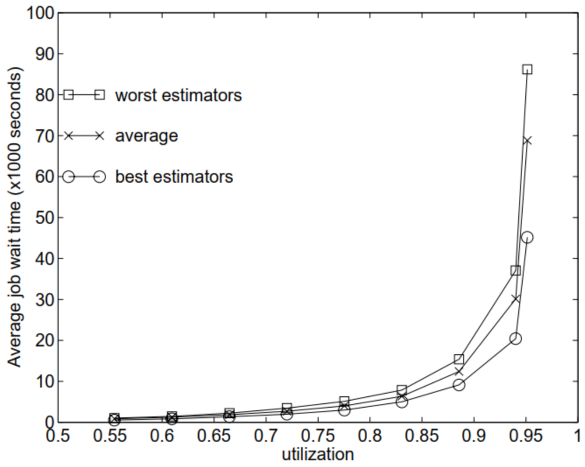
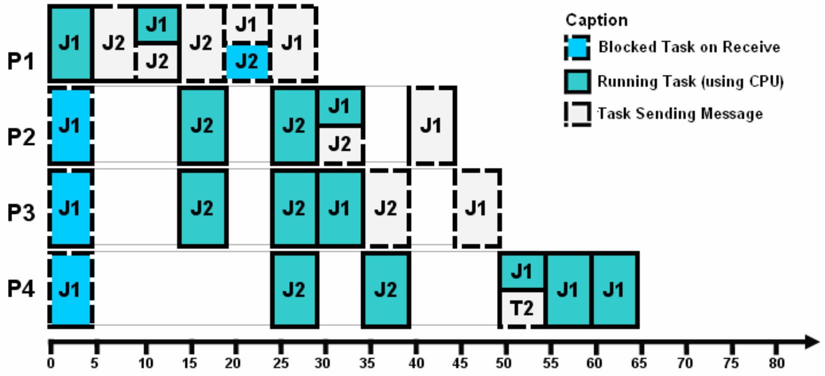


Figure 5. The impact of good estimation from a user perspective.

Backfilling gang scheduling

- Schedules for space/time-sharing can be represented by an Ousterhout matrix
  - Rows represent time slices and the columns represent processor
- Gang scheduling: optimization on time axis
  - Orthogonal to backfilling: can be combined
- Proposed some policies that combines backfilling and gang scheduling

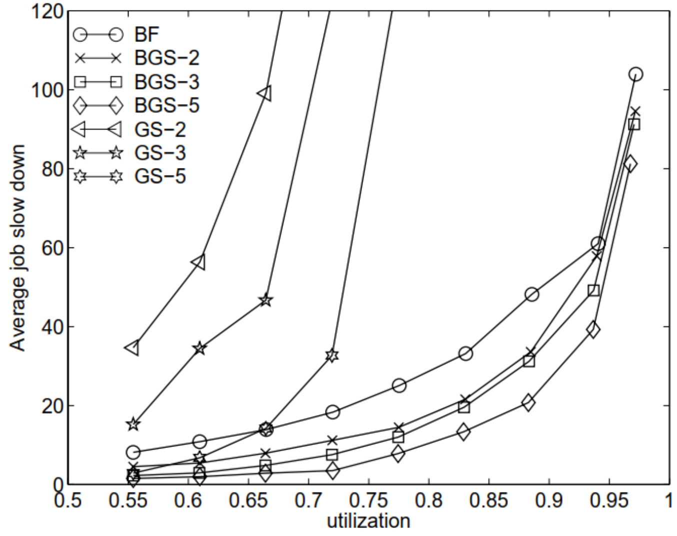
	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
time-slice 0	J_1^0	J_1^1	J_1^2	J_1^3	J_1^4	J_1^5	J_1^6	J_1^7
time-slice 1	J_2^0	J_2^1	J_2^2	J_2^3	J_2^4	J_2^5	J_2^6	J_2^7
time-slice 2	J_3^0	J_3^1	J_3^2	J_3^3	J_4^0	J_4^1	J_5^0	J_5^1
time-slice 3	J_6^0	J_6^1	J_6^2	J_6^3	J_4^0	J_4^1	J_5^0	J_5^1



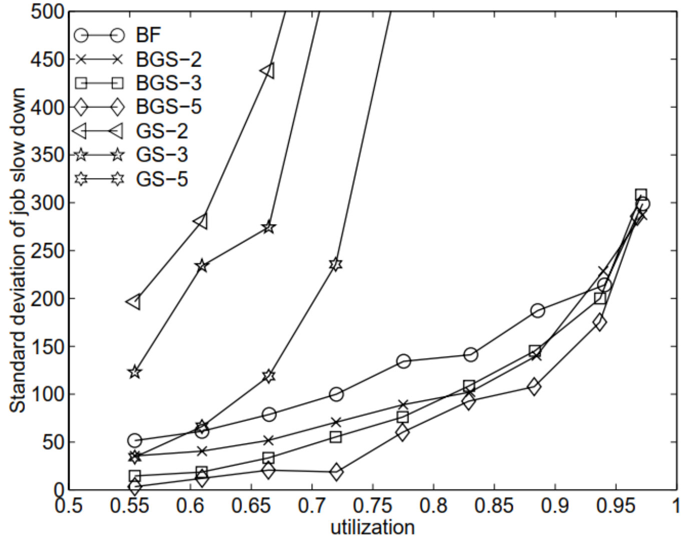
Backfilling gang scheduling

- BF: Baseline backfilling
- GS-2, GS-3, GS-5: different variations of gang scheduling
- BGS-2, BGS-3, BGS-5: variations of backfill gang scheduling

- Result shows that BGS is always better than pure BF or pure GS

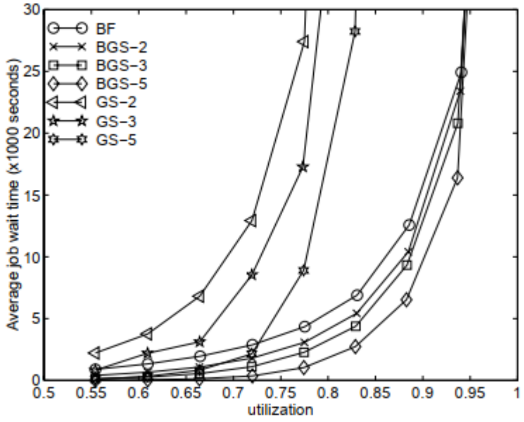


(a) average slowdown

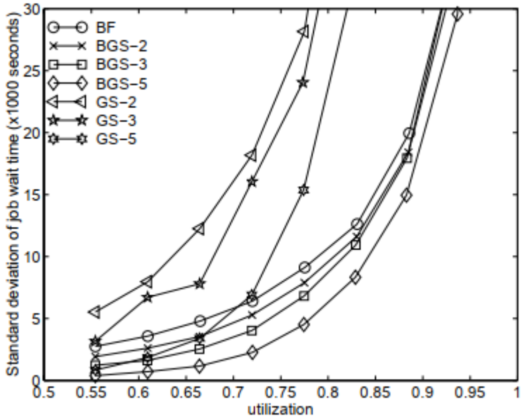


(b) standard deviation of slowdown

Backfilling gang scheduling (BGS)

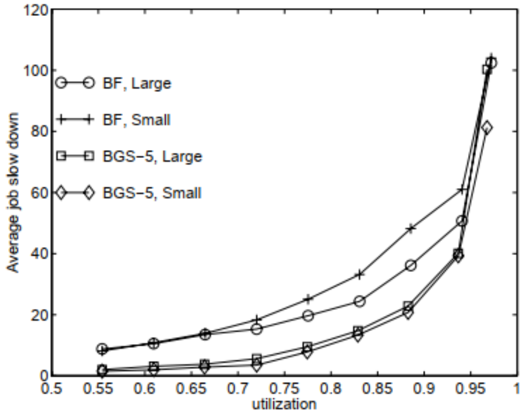


(a) average wait time

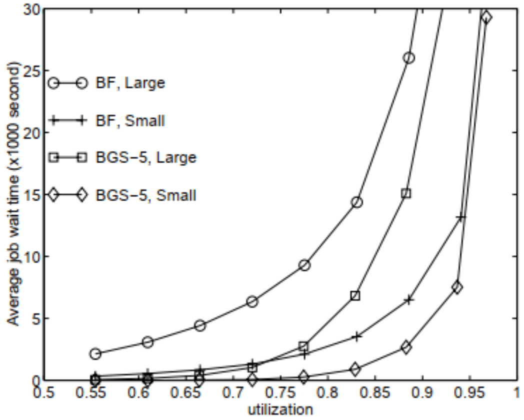


(b) standard deviation of wait time

Figure 7. Comparing job wait time for BGS **with** BF **and** GS**.**



(a) slowdown of large and small jobs



(b) wait time of large and small jobs

Figure 8. Comparing the behavior of large and small jobs for BGS **with** BF**.**

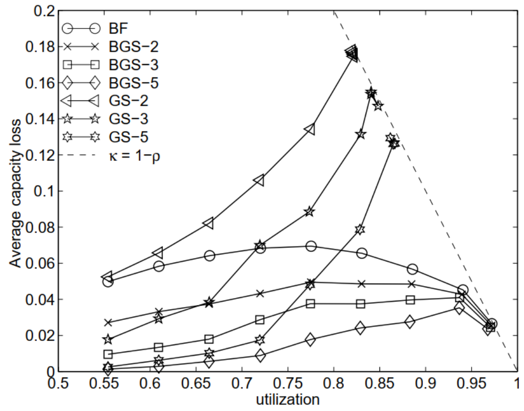
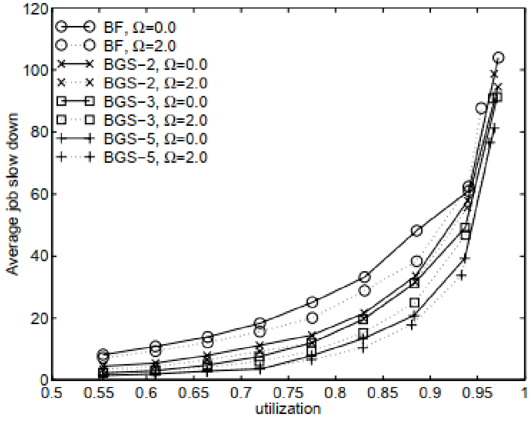
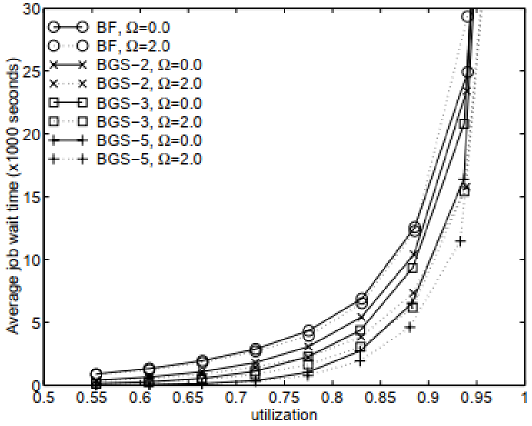


Figure 9. Comparing capacity loss for BGS **with** BF **and** GS**.**



(a) slow down



(b) wait time

Figure 10. Comparing performance parameters for BGS **with** BF **and** GS**.**

Conclusion and future work

- Valuable insights:
  - FCFS policy + backfilling does as well as other policies such as SJF, BF, WF since it avoids starvation.
  - Overestimation of execution time has minimal impact on resulting system behavior, but better estimation can enable users to shorten wait time
  - Effective combination of gang scheduling and backfilling can perform better than any individual policy
- Future works:
  - Consider the impact of context switching costs
  - Examine issues related to migration in BGS, with respect to different performance criteria
  - Compare the pros and cons of different time-, space-sharing strategy