Scheduling Algorithms for Dynamic Workload

Dalibor Klusáček (MU)
Hana Rudová (MU)
Ranieri Baraglia (CNR - ISTI)
Gabriele Capannini (CNR - ISTI)
Marco Pasquali (CNR – ISTI)
Outline

• Motivation & Problem description
• Applied techniques
  ▪ Queue-based (Backfilling)
  ▪ Schedule-based (Dispatching rules & Local Search)
• Simulation toolkit
• Preliminary results
• Future work
Job Scheduling on Computational Grids

• The general job scheduling problem includes:
  ▪ Selection of a processing resource for every job
  ▪ Selection of a job processing order/time for every resource

• Driven by different constraints:
  ▪ Job QoS requirements (e.g. deadline and sw licenses)
  ▪ Data/time dependencies between jobs
  ▪ Processing limitation of resources (e.g. sw licences), etc.

• Objectives:
  ▪ To optimize the system throughput maximizing the overall resource utilization
  ▪ To guarantee a maximum level of performance required from applications
Job Scheduling on Computational Grids

• In the past a lot of research effort was devoted to understand and develop job scheduling algorithms (e.g. FCFS, Backfilling, Gang scheduling, etc.)

• Nowadays many of these algorithms are exploited into commercial and open source job schedulers

• However, none of these scheduler capabilities deal with an entire wide range of constraints and requirements (e.g. job’s deadline) presented by the users
Examples:

Maui scheduler —> FCFS, Backfilling, EASY backfilling

Load Lever —> FCFS, backfilling, gang scheduling, external schedulers

Load sharing facility —> FCFS, fair-share, preemptive, backfilling, service Level Agreements

Portable Batch System —> FCFS, Shortest Job First, user/group Priorities, fair-share

Sun Grid Engine —> FCFS, job priorities and fair-share, migration support. Future version: backfilling
Current Problem

• Machines
  ▪ Parallel machines with different number of CPUs
  ▪ Different machines with different CPU speed

• Jobs
  ▪ *Dynamically arriving jobs*
  ▪ With/without deadline
  ▪ Job require >= 1 CPU
  ▪ Known job-execution time

• Objective function
  ▪ Maximize number of jobs that *meet their deadline*
Queue-based Algorithms

• Queues

• Basic methods
  ▪ Trivial queue-based technique used as a comparison with advanced techniques
  ▪ **FCFS**: First Come First Serve
  ▪ **EDF**: Earliest Deadline First

• Backfilling
  ▪ **Easy Backfilling**
  ▪ **Flexible Backfilling** (future work)
Backfilling

• Is an optimization of the FCFS algorithm
• Tries to balance the goals of utilization and maintaining FCFS order.
• Requires that each job also specifies its maximum execution time. While the job at the head of the queue is waiting, it is possible for other, smaller jobs, to be scheduled, especially if they would not delay the start of the job on the head of the queue.
• Several variants of backfilling algorithm were proposed.
Algorithms with Global Schedule

• Global schedule
  ▪ Both resource and time assignment considered

• Dispatching rules
  ▪ Used for initial schedule generation
  ▪ **MTEDF**: Minimum Tardiness Earliest Deadline First

• Local Search
  ▪ Local changes in schedule by movement of jobs
  ▪ **Tabu Search**: recent moves prohibited to avoid cycling
Simulation Toolkit

- **GridSim-based simulator**
  - Implemented at MU

- **Centralized scheduler**

- **Our extensions**
  - Simulated scenarios
    - Static vs. dynamic problems
    - Sequential/parallel jobs, chain of jobs
    - Total tardiness, makespan minimization
  - Different scheduling techniques
    - ERD, EDD, MTERD
    - Tabu search, Hill climbing, Simulated Annealing

- **Easy integration**
  - New scenarios, new scheduling techniques
Current Data Sets and Algorithms

- 1500 jobs
  - 1-16 CPU
  - 30% no-deadline jobs

- 150 machines (cca 1500 CPUs)
  - 2-16 CPU, different CPU rates

- 7 different types of problem with 20 data sets
  - Arrival time distribution (high/low frequency)

- FCFS, EDF
- Easy Backfilling
- MTEDF
- Tabu search
Preliminary Results

Arrival time distribution

Jobs not meeting the deadline (%)

Arrival time distribution

Total scheduling time (ms)
Future Work

• Problem extensions
  ▪ Estimated running time, SW licenses, RAM size

• New algorithms
  ▪ Flexible backfilling

• Extension of current algorithms
  ▪ Improvements in the data representation for global schedule
  ▪ More complex reasoning for parallel jobs
  ▪ Study of local search algorithms for dynamic problems
Thank you!