JOB SCHEDULING
High Performance Computing Systems

Tianhe-I A, National Supercomputer Center, Tianjin China
Prof. Kai Wang, University of North Dakota (intro to HPC)
Prof. Henri Casanova, University of Hawaii (Job Scheduling)
Prof. Klara Narstadht, University of Illinois at Urbana-Champaign (Gang Scheduling)
Prof. Andy Chien, UCSD (Gang Scheduling Performance)
Overview

► High Performance Computing
► Hardware and Architecture
   Top 500 supercomputers
► Job Scheduling
   ► Batch scheduling
   ► Gang scheduling
   ► Migration
► Performance Evaluation
High Performance Computer 1991

- Cray C90 Machine
- Cost $26 million
- 8 Vector Processors
- Ethernet
- 8 gigaflops
- A problem that takes a PC 8 hours to solve, the CRAY C90 can do in 0.002 seconds
Tianhe 1A
(2010 fastest supercomputer)

- 2.507 petaflops on Linpack benchmark
- 14,336 Intel Xeon CPUs
- 7,168 Nvidia Tesla M2050 GPUs
- Chinese interconnect
- fabric, 2X InfiniBand
- 103 cabinets
- 155 tons
- consumes 4.04 megawatts
What is High Performance Computing

- High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems.

- **Supercomputer** [noun /ˌsoʊpərˈkəmpyətər/] – typically one-of-a-kind design with specialized hardware and interconnect

- **Compute cluster** – built from COTS (commodity off-the-shelf components)
HPC Applications

- **Aeronautics and astronautics**: Computational fluid dynamics simulations to test the stability of the wings at high speed (e.g. NASA)
- **Meteorology**: weather prediction using data gathered from stations/sensors
- **Insurance**: risk management and decision analysis (e.g. Cray CX1 and financial companies)
- **Scientific Visualization** of all kinds
- **Health care**: data analytics and speech recognition for medical diagnosis (IBM Watson and Nuance Communication)
What Does High Performance Computing Study

- Hardware
- Computer Architecture
- Networking
- Software
- Algorithms
- Language
- Operating system
- Scheduling and Resource Mgmt
- *Synchronization*
- Memory Mgmt
- *Communication*
Shared Memory Architecture

- Multiple processors
- Share memory
Distributed Memory Architecture

- Multiple processors
- Every processor has its own memory
Hybrid

A cross between shared and distributed computers
Grid Computing
Top Ten Most Powerful Computers
(http://www.top500.org)
Top 500 Computers--Countries (http://www.top500.org)
Top 500 Computers--Vendors (http://www.top500.org)
Top 500 Computers—Manufacturers Trend (http://www.top500.org)
Application Area / Systems
November 2010

- Research
- Finance
- Logistic Services
- Information Service
- Geophysics
- Aerospace
- Weather and Climate Res
- Information Processing Se
- Service
- Telecomm
- Energy
- Defense
- Others
- Not Specified
Top 500 Computers--Applications Trend (http://www.top500.org)
Job Scheduling

- A HPC is shared by many users. Each want the illusion of being on a dedicated machine.
- The Job Scheduler schedules the users jobs on the supercomputer/compute cluster.
- JOB is a collection of (communicating) parallel tasks but we will ignore communication issues
  - Communication performance dominated earlier supercomputer architectures; now not so critical
- Scheduler admits jobs to the system and decides when and where (on which processors) the tasks will run.
Assumptions

- We consider a single “job scheduler”
- The job scheduler “manages” some number of identical “nodes”
Space- or Time-sharing

- **Space-sharing**
  - a single task per node
  - “batch scheduling”

- **Time-sharing**
  - Multiple tasks on a nodes with synchronized context-switching
  - “gang scheduling”
Batch Scheduling

- A Batch scheduler maintains a queue of pending jobs
- Each job is defined as:
  - Number of nodes
  - Time
  - “I want 6 nodes for 1 hour”
  - Typically users are “charged” against an “allocation”
    - e.g., “You only get 100 CPU hours per week”
- There can be different queues, different priorities, etc.
- There can be limits on usage
  - number of jobs in the queue < X
  - number of jobs per day < X
  - job size < X
  - etc.
- Notions of user groups
  - power users
- These are complex systems with many config options
Graphical Representation of a Schedule
Graphical Representation of a Schedule

- **nodes**
- **time**
- **RUNNING**
- **WAITING**
- **NEW JOB**

max # of nodes
Graphical Representation of a Schedule

- **RUNNING**
- **WAITING**
- **NEW JOB**

Nodes vs. Time

- max # of nodes
Scheduling FCFS

- Simplest scheduling option: FCFS
  - First Come First Serve

- Problem:
  - Fragmentation:

![Diagram showing scheduling with running and stuck tasks]

- first job in queue
- running
- stuck
- stuck

Now
The Solution: Backfilling

- **running**
- **stuck**

Time progression diagram showing nodes running and stuck over time.
Backfilling: Question

- Which job(s) should be picked for promotion through the queue?
- Many heuristics are possible
- Two have been studied in detail
  - EASY Backfilling
  - Conservative Back Filling (CBF)
- In practice EASY (or variants of it) is used, while CBF is not as common
**EASY Backfilling**

- **Extensible Argonne Scheduling System**
- **Maintain only one “reservation”, for the first job in the queue**
- **Definitions**
  - Shadow time: time at which the first job in the queue starts execution
  - Extra nodes: number of nodes idle when the first job in the queue starts execution
- **Go through the queue in order starting with the 2nd job**
- **Backfill a job if**
  - it will terminate by the shadow time, or
  - it needs less than the extra nodes
EASY: Example

- extra nodes
- first job in queue
- running
- shadow time
EASY: Example

- First job in queue
- Extra nodes
- Second job in queue
- Running

Nodes vs. Time
EASY: Example

Nodes

Running

Extra nodes

First job in queue

Shadow time

Time
EASY: Example

- Running
- Second job in the queue
- First job in queue
- Third job in the queue
- Extra nodes
- Shadow time
- Time
EASY: Example

- First job in queue
- Second job in the queue
- Third job in the queue

Running nodes:

- Extra nodes
EASY: Properties

- Unbounded Delay
  - The first job in the queue will never be delayed by backfilled jobs
  - BUT, other jobs may be delayed infinitely!
EASY: Unbounded Delay

- Extra nodes
- First job in the queue
- Running
- Second job in queue
- Third job in the queue
- Shadow time
EASY: Unbounded Delay
EASY: Unbounded Delay

- running
- first job in the queue
- third job in the queue
- second job in the queue
- fourth job in the queue

Nodes

Shadow time

Time

Extra nodes
EASY: Unbounded Delay

And so on...
EASY: Properties

- **Unbounded Delay**
  - The first job in the queue will never be delayed by backfilled jobs
  - BUT, other jobs may be delayed infinitely!

- **No starvation**
  - Delay of first job is bounded by runtime of current jobs
  - When the first job runs, the second job becomes the first job in the queue
  - Once it is the first job, it cannot be delayed further
Conservative Backfilling

- EVERY job has a “reservation”
- A job may be backfilled only if it does not delay any other job ahead of it in the queue
- Fixes the unbounded delay problem that EASY has
- More complicated to implement
  - The algorithm must find holes in the schedule
- EASY favors small long jobs
- EASY harms large short jobs
When does backfilling happen

- Possibly when
  - A new job arrives
  - The first job in the queue starts
  - When a job finishes early

- Users provide job runtime estimates
  - Jobs are killed if they go over

- Trade-off
  - provide an aggressive estimate: you go through the queue faster (may be backfilled)
  - provide a conservative estimate: your job will not be killed

- Are estimates accurate?
User Estimate Accuracy

- One key issue in scheduling: how accurate is the information that the scheduler uses to make decision?