SCOJO – Share-Based Job Coscheduling with Integrated Dynamic Resource Directory in Support of Grid Scheduling

Angela C. Sodan and Xuemin Huang
University of Windsor, Canada
HPCS 2003
Outline

- **Introduction**
  - Scheduling on the Grid
  - Local Support for Grid Scheduling
- **SCOJO Scheduler**
  - Reservation and QoS Scheduling
  - Dynamic Resource Directories
  - Experimental Results
- **Summary and Future Work**
Computational Grids – Our Goal

- Make resources from different sites jointly available and run application across multiple sites
- Requires middleware like Globus for bridging heterogeneous software and hardware, providing security, and QoS
- Needs to prepare local scheduling
Problem: Scheduling Jobs Simultaneously on Different Sites

Site A: job queue, local scheduler, \( t_x \)

Site B: \( t_x \)

Site C: network, communication

Site D: \( t_x \)
Possible Approaches to Local Scheduling

- Immediate scheduling for flexible time-shared execution
  - On most shared-memory machines (e.g. IRIX switched back to it as default in 6.5)
  - In NOWs

- Batch scheduling with job queues
  - In most clusters
  - In most cases exclusive resource assignment (space sharing), possibly gang
Job Scheduling

Space Sharing

Batch – Entry Control

Time Sharing
Loosely-Coordinated Co-scheduling

Motivation:

- Close to standard time sharing
  - Permits latency hiding (communication, I/O)
  - Better exploitation of hyperthreading
  - Better resource utilization
- Multiple time-shared applications mean multiple virtual machines and thus scheduling options (esp. important if no preemption / adaptation)
  - Better response times
**SCOJO: QoS Loosely Coordinated Coscheduling**

- Exploits loose coscheduling
- Adds again entry control (which, how many)
- Permits start-time and share reservations for simultaneous grid applications
- Guarantees certain time shares
Taking a Closer Look at Co-Scheduling

- Requires loose coordination and limited use of polling → change in MPI implementation and OS support
- Speedup: I/O and long communication latencies can be hidden
- Slowdown: medium-range communication - slowdown increasing with multiprogramming level
Some Special Features in SCOJO

- Exploits application characteristics
- Guarantee effective shares
- Can easily exploit all free shares
- Can deal with wrong estimates
- Uses backfilling and priorities (aging)
- Envisions (distributed) protocol to find all possible matching slots at different places (master per application)
Co-Scheduling with Guaranteed Shares and Full Utilization

- Time sharing
- Exclusive assignment (standard space sharing)
Our Findings

- Slowdowns differ – patterns, granularity and combination matter

<table>
<thead>
<tr>
<th></th>
<th>Grid-300</th>
<th>Grid-1200</th>
<th>Grid-2400</th>
<th>Central</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid-300</td>
<td>1.2</td>
<td>1.4 / 1.1</td>
<td>2.2 / 0.9</td>
<td>0.9 / 1.6</td>
<td>1 / 1.3</td>
</tr>
<tr>
<td>Grid-1200</td>
<td>1.2</td>
<td>1.4 / 0.8</td>
<td>1.1 / 1.7</td>
<td>1 / 1.4</td>
<td></td>
</tr>
<tr>
<td>Grid-2400</td>
<td></td>
<td>1.1</td>
<td>1 / 3.1</td>
<td>1.5 / 2.3</td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td></td>
<td></td>
<td>1.3</td>
<td>1.3 / 0.9</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
**Effective Shares**

- Use application characteristics to determine slowdown/speedup factor
- Determine coschedule yes/no
- Calculate *effective shares* to provide stability and keep (remote) guarantees
- If slowdown, increase actual share
  
e.g. if requested and guaranteed share is 30% and slowdown 1.2, then actual share of 30%*1.2 = 36% would be reserved
SCOJO – Our Approach for Share-Based Scheduling (cont.)

Additional measures/benefits:
- Reserve only up to $R_{\text{max}}$ share and priorities
  - Enable short jobs to get through quickly (if long one running), better response times
  - Give the possibility to schedule other work if start-time reservation for remote job
  - Use for overrun, can do without preemption
- Free shares easy to use (fragmentation only if cross-site extra shares not usable)
**SCOJO – The Overall System**

- User or scheduler agent
- Network
- Applications
- Grid/local job scheduler:
  - Model and scheduling plan
  - Job queue
- Dynamic directory:
  - Model, decision system
  - Application infos
  - Co-scheduling estimator
- Application infos
The Dynamic Directory

- Stores application characteristics
  - Runtime, granularities, patterns, owner etc.

- Problem:
  - Exposes details of site and applications

- Solution:
  - Multiple views / access on data with different access permissions / keys
  - Private data of application not exposed to other applications
The Dynamic Directory

dynamic directory

- general infos – accessible to every legal user
- application infos – accessible to owner and system (semi-static)
- registration-time part
- execution-time part (fully dynamic)

standard directory service

model and estimator
Experimental Results

- Enabling cross-site scheduling with start-time and share reservation, overrun
  - Demonstrate feasibility
- Response-time improvement
  - Quantification
- Used simulations and test runs on SMP server (Sun Enterprise 6500 / Solaris 8, SUN MPI)
SCOJO – Results of Simulation Study

avg. response time, Mlevel = 2, equal shares, real program runs (test overrun)

2 workloads:
Case 1: similar to NASA Ames, iPSC/860
Case 2: similar to NCSA, Origin 2000
SCOJO – Results of Simulation Study

average relative response time, same setting
**SCOJO – Results of Simulation Study**

average and average relative response time, flexible shares, slowdown 1.2, C2: Mlevel = 2, C3: Mlevel = 3
Summary

- **SCOJO**
  - QoS share reservation in batch/time-scheduling (with backfilling and priorities)
  - Dynamic directory with application infos

- **Benefits**
  - Stable/QoS scheduling
  - More flexible scheduling across sites
  - Better response times
  - Better resource utilization
Future Work

- Integration into existing job schedulers
- Combination with space sharing, adaptation
- Test I/O- and memory-bound applications
- True local share allocation
- Integration into grid software like Globus
- Dynamic Directory: Globus OGSA which permits stateful dynamic services, database
- Improvements of co-scheduling, model