Scheduling: today and tomorrow

Larry Carter
Apologies for

- Misrepresenting your work
- Not knowing very much
- Taking extreme positions
- ...

Scheduling in Aussois workshop
Market-based systems are inevitable

A “convergence of technologies” is needed:

- Electronic money
- Allocatable resources
- Trust
- ...

Once this happens, compute power will either be:

- Abundant
- Scarce

Moore’s law will eventually fail; people’s imagination won’t.
Issues for market-based scheduling

User’s quality measure:
- Makespan, stretch, or steady-state throughput
- User-specific “utility”

Server’s goal:
- Maximize throughput
- Maximize profit ("structural unemployment" may be beneficial)

System wide goal:
- Fairness
  - Equal access to market, but “money talks”

User’s motivation:
- Altruism, Profit

New opportunities:
- Brokers, publicists, insurers, …

from 2005 workshop
Towards more generality

Computational platform
- Homogeneous $\rightarrow$ heterogeneous platform
- Processor time $\rightarrow$ communication, memory, ... times
- Centralized $\rightarrow$ distributed decision making
- Reliable $\rightarrow$ unreliable or collusive processors
- One $\rightarrow$ multiple administrative domains

Application model
- Independent tasks $\rightarrow$ Job = DAG of tasks
- Constant $\rightarrow$ changing resource needs
- Uniform tasks $\rightarrow$ multiple bags of tasks

Objective of optimization
- single goal $\rightarrow$ individual utility functions
Combating NP-completeness

NP complete result $\rightarrow$ adaptive approximation algorithms $\rightarrow$ (simgrid) simulations

Makespan $\rightarrow$ Steady state $\rightarrow$ Trim analysis
## Non-traditional features

<table>
<thead>
<tr>
<th></th>
<th>Platform model</th>
<th>Application model</th>
<th>Objective function</th>
<th>Other features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rosenberg</td>
<td></td>
<td>DAG</td>
<td>maximize</td>
<td></td>
</tr>
<tr>
<td>Weinberg</td>
<td>multi-level memory</td>
<td>(real)</td>
<td>user-specified</td>
<td>symbiosis</td>
</tr>
<tr>
<td>Lee</td>
<td>supercomputer</td>
<td>(real)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeannot</td>
<td></td>
<td></td>
<td>reduce errors</td>
<td>colluding users</td>
</tr>
<tr>
<td>Dongarra</td>
<td>supercomputer</td>
<td>(real)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trystram</td>
<td>multiple domains</td>
<td></td>
<td>multiple</td>
<td></td>
</tr>
<tr>
<td>Agrawal</td>
<td>varying allocation</td>
<td></td>
<td></td>
<td>adversarial allocator</td>
</tr>
<tr>
<td>Beaumont</td>
<td>heterogeneous</td>
<td>multiple, divisible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detti</td>
<td></td>
<td></td>
<td>reliability</td>
<td>crashes possible</td>
</tr>
<tr>
<td>Marchal</td>
<td>heterogeneous</td>
<td>multiple bags</td>
<td>stretch</td>
<td>on-line versus off-line</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>ET CETERA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What we’re accomplishing

Breadth-first search of new models

- Driven by technology changes
  - Multicore, unreliable processors, ...
- Improving constant from 9/7 to 5/4

Introducing (potentially important) new paradigms, e.g.

- IC-optimality
- Symbiosis
- Collusion-resistance
- Nash equilibrium
- …
What we’re not accomplishing

New algorithms implemented in “real” system

(Perhaps if we were that successful, we wouldn’t be attending this workshop)
What should we be accomplishing??

Computer science is not a natural science

- We get to invent our own models
- Discovering properties of random models isn’t nearly as interesting as discovering “nature”

We should work towards having an influence

(Well, that’s my opinion)
How to have an influence

“Throw great idea over the wall”

i.e. publish paper
If it’s good enough, people will pick it up
example: randomized routine

But what’s on the other side?

“Not invented here”

Usually, we must do (much) more
A not-yet influential idea

Bandwidth-centric scheduling

“A parent node responding to requests from multiple children should give first priority to child with highest bandwidth.”

Why hasn’t this been adopted by BOINC??

Bandwidth isn’t following one-port model
no parent node has knowledge of children
A not-yet influential idea

Bandwidth-centric scheduling

“A parent node responding to requests from multiple children should give first priority to child with highest bandwidth.”

Why hasn’t this been adapted by BOINC??

Bandwidths don’t follow one-port model

BOINC doesn’t even know the bandwidths
The other side of the wall

“Not invented here”

Learning our language and sifting many papers is very difficult

People have their own ideas they think are good

To overcome these barriers, we need to

Learn about their world

Demonstrate effectiveness on their data

My experience: this effort benefits me

new problems

new ideas
Other ways to have influence

Often, our ideas are discovered independently by others

At best, our theory can help assure others that the ideas are valid (a constructive interaction)

At worst, we can get into big fights over who deserves the credit or patents

Our work can suggest what general directions are more or less promising (if we can get ourselves in an advisory position).

Perhaps we can demonstrate value of:

- Collecting extra information
- Providing new capability
Influence-aware research

Suppose we want to do research on desktop grids for DAG applications

What is a potential application?

What information would be readily available to scheduler in such an application.

Can we argue that our technique is so good it’s worth the effort to collect needed parameters?

Can we envision a path towards implementation

Possible target: Chess or Go on BOINC
Multi-core: a new opportunity

Jack Dongarra’s problem (LAPACK)

Even easily parallelized applications will need to tolerate variable execution times and failures

Scheduling for more general programs (e.g. threaded programs) will be needed

Adaptive, self-scheduling techniques are easiest to get adapted

Locality will be very important in future

The cost of moving data is MUCH more than the cost of computation

We must learn to live with unreliable cores
Conclusion

We’re doing excellent work

I’m not suggesting you totally change your research

Perhaps we could do more to get work used

Discussion
Backup slides
Symbiotic Scheduling

Symbiosis: Two applications run concurrently take less time than running one then the other.
  - “Timesharing” on uniprocessors usually isn’t symbiotic.
  - Symbiosis has been demonstrated for multithreaded processors (Snavely)

Typical node:
  - Multiple processors
  - Shared memory (but separate caches)
  - Communication network to other nodes
  - I/O channel to disks

Opportunity for symbiosis when different applications have different bottlenecks.
Symbiosis

Processor ops/sec
DRAM bytes/sec
Network bytes/sec
Disk bytes/sec

Application #1 (compute-bound)
Application #2 (I/O-bound)

bottleneck
Symbiosis

Symbiotic schedule

bottleneck

savings

Processor ops/sec
DRAM bytes/sec
Network bytes/sec
Disk bytes/sec

Application #1
Application #2

Application #1
Application #2

Scheduling in Aussois workshop