Solving Hard Integer Programs with MW

JEFF LINDEROTH
ISE Department
COR@L Lab
Lehigh University
jtl3@lehigh.edu

2007 Condor Jamboree
Madison, WI
May 2, 2007

Thanks! NSF OCI-0330607, CMMI-0522796, DOE DE-FG02-05ER25694
Collaborators

\{ 
  François Margot  
  Carnegie Mellon  
\} 

\{ 
  Greg Thain  
  UW-Madison  
\}
In Our Last Episode...

The Design of a Gambling System

- Predict the outcome of $\nu$ soccer matches
- $\alpha = 3$
  - 0: Team A wins
  - 1: Team B wins
  - 2: Draw
- You win if you miss at most $d = 1$ games

The Football Pool Problem

What is the minimum number of tickets you must buy to guarantee that you hold a winning ticket?
How Many Must I Buy?

Known Optimal Values

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>C^*_n</td>
<td>$</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

The Football Pool Problem

What is $|C^*_6|$?

- Despite significant effort on this problem for $>40$ years, it is (was) only known that

$$65 \leq C^*_6 \leq 73$$
But It’s Trivial!

- There is a simple formulation of the problem as a reasonably-sized integer program (IP)
- For each \( j \in W \), let \( x_j = 1 \) iff the word \( j \) is in code \( C \)
- Let \( A \in \{0, 1\}^{|W| \times |W|} \)
  - \( a_{ij} = 1 \) iff word \( i \in W \) is distance \( \leq d = 1 \) from word \( j \in W \)

**IP Formulation**

\[
\begin{align*}
\text{min} & \quad e^T x \\
\text{s.t.} & \quad Ax \geq e \\
& \quad x \in \{0, 1\}^{|W|}
\end{align*}
\]
Football Pool Problem

Problem Description

Football Pool Covering Matrix
Beautiful But Deadly

- Workhorse algorithm is a tree-search procedure known as branch-and-bound.
- CPLEX: A commercial IP package that is putting integer programmers out of business.
- CPLEX _routinely_ solves 0-1 integer programs with (tens of) thousands of variables and constraints.
- Theorem: “Pretty” Matrices Make Hard IPs
- Many branches must be done to remove the symmetry
- Recognizing symmetry and designing algorithms to exploit the symmetry are _fundamentally important_
Grid Programmers Do It In Parallel

- Nodes in disjoint subtrees can be evaluated independently.
- But this is not an embarrassingly pleasantly parallel operation.
- We use the master-worker parallelization scheme.
Use Master-Worker!

Master:
- Send task (node) to workers

Worker:
- Evaluate node and send result to master

“Liberate” WMDs & People
As You Wish

Declare Mission Accomplished!

Huh?
Master-Worker is a flexible, powerful framework for Grid Computing

- It’s easy to be fault tolerant
- It’s easy to take advantage of machines whenever they are available
- You can be flexible and adaptive in your approach to computing
- Everyone can have access to a powerful computing platform

MW—We’re Here to Help!

- **MW** is a C++ software package that encapsulates the abstractions of the Master-Worker paradigm
- Allows users to easily build master-worker type computations running on Condor-provided computational grids
MW Classes

- MWMaster
  - getuserinfo()
  - setup_initial_tasks()
  - pack_worker_init_data()
  - act_on_completed_task()
- MWTask
  - (un)pack_work
  - (un)pack_result
- MWWorker
  - unpack_worker_init_data()
  - execute_task()

We’re Here To Help!
Please contact Greg or Myself if you need help getting set-up with MW
Condor + MW — Cobbling Together Resources

1. Condor Flocking
   - Jobs submit to local pool run in remote pools

2. Condor Glide-in (or manual “hobble-in”)
   - Batch-scheduled resources join existing Condor pool.

3. Remote Submit
   - Log-in and submit worker executables remotely
   - Can use port-forwarding for hard-to-reach private networks

Schedd-on-the-side

- A new Condor technology which takes idle jobs out of the local Condor queue, translates them into Grid jobs, and uses Condor-G to submit them to a remote Grid queue
- Perfect for OSG!
Deja Vu

- In 2006, we had almost established a lower bound of $70 \leq |C_6|^* \leq 73$

Statistics so far...

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall Time</td>
<td>47.15 days</td>
</tr>
<tr>
<td>CPU Time</td>
<td>57.31 years</td>
</tr>
<tr>
<td>Avg Workers</td>
<td>467.3</td>
</tr>
<tr>
<td>Max Workers</td>
<td>1253</td>
</tr>
<tr>
<td>Total Nodes</td>
<td>$8.37 \times 10^8$</td>
</tr>
<tr>
<td>Total LP Pivots</td>
<td>$6.53 \times 10^{11}$</td>
</tr>
<tr>
<td>Parallel Performance</td>
<td>94.9%</td>
</tr>
</tbody>
</table>
“Mission Accomplished”

- We were able to establish a lower bound of $70 \leq |C_6|^* \leq 73$

---

**Computational Statistics**

- Wall Time: 72.3 days
- CPU Time: 110.1 years
- Avg Workers: 555.8
- Max Workers: 2038
- # Different Workers: 4266
- Total Nodes: $2.85 \times 10^9$
- Total LP Pivots: $2.65 \times 10^{12}$
- Parallel Performance: 90.3%
Our Continuing Mission

- I couldn’t think of a good analogy, but it has been 312 days since we declared “mission accomplished”: $70 \leq |C_6|^* \leq 73$

But We Press On

- Can we improve the lower bound even more?
- We’ve ramped up the scale of resources available
## Resources Used in Computation

<table>
<thead>
<tr>
<th>Site</th>
<th>Access Method</th>
<th>Arch/OS</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wisconsin - CS</td>
<td>Flocking</td>
<td>x86_32/Linux</td>
<td>975</td>
</tr>
<tr>
<td>Wisconsin - CS</td>
<td>Flocking</td>
<td>Windows</td>
<td>126</td>
</tr>
<tr>
<td>Wisconsin - CAE</td>
<td>Remote submit</td>
<td>x86_32/Linux</td>
<td>89</td>
</tr>
<tr>
<td>Wisconsin - CAE</td>
<td>Remote submit</td>
<td>Windows</td>
<td>936</td>
</tr>
<tr>
<td>Lehigh - COR@L Lab</td>
<td>Flocking</td>
<td>x86_32/Linux</td>
<td>57</td>
</tr>
<tr>
<td>Lehigh - Campus</td>
<td>Remote Submit</td>
<td>Windows</td>
<td>803</td>
</tr>
<tr>
<td>Lehigh - Beowulf</td>
<td>ssh + Remote Submit</td>
<td>x86_32</td>
<td>184</td>
</tr>
<tr>
<td>Lehigh - Beowulf</td>
<td>ssh + Remote Submit</td>
<td>x86_64</td>
<td>120</td>
</tr>
<tr>
<td>TG - NCSA</td>
<td>Flocking</td>
<td>x86_32/Linux</td>
<td>494</td>
</tr>
<tr>
<td>TG - NCSA</td>
<td>Flocking</td>
<td>x86_64/Linux</td>
<td>406</td>
</tr>
<tr>
<td>TG - NCSA</td>
<td>Hobble-in</td>
<td>ia64-linux</td>
<td>1732</td>
</tr>
<tr>
<td>TG - ANL/UC</td>
<td>Hobble-in</td>
<td>ia-32/Linux</td>
<td>192</td>
</tr>
<tr>
<td>TG - ANL/UC</td>
<td>Hobble-in</td>
<td>ia-64/Linux</td>
<td>128</td>
</tr>
<tr>
<td>TG - TACC</td>
<td>Hobble-in</td>
<td>x86_64/Linux</td>
<td>5100</td>
</tr>
<tr>
<td>TG - SDSC</td>
<td>Hobble-in</td>
<td>ia-64/Linux</td>
<td>524</td>
</tr>
<tr>
<td>TG - Purdue</td>
<td>Remote Submit</td>
<td>x86_32/Linux</td>
<td>1099</td>
</tr>
<tr>
<td>TG - Purdue</td>
<td>Remote Submit</td>
<td>x86_64/Linux</td>
<td>1529</td>
</tr>
<tr>
<td>TG - Purdue</td>
<td>Remote Submit</td>
<td>Windows</td>
<td>1460</td>
</tr>
</tbody>
</table>
## OSG Resources Used in Computation

<table>
<thead>
<tr>
<th>Site</th>
<th>Access Method</th>
<th>Arch/OS</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSG - Wisconsin</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>1000</td>
</tr>
<tr>
<td>OSG - Nebraska</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>200</td>
</tr>
<tr>
<td>OSG - Caltech</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>500</td>
</tr>
<tr>
<td>OSG - Arkansas</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>8</td>
</tr>
<tr>
<td>OSG - BNL</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>250</td>
</tr>
<tr>
<td>OSG - MIT</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>200</td>
</tr>
<tr>
<td>OSG - Purdue</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>500</td>
</tr>
<tr>
<td>OSG - Florida</td>
<td>Schedd-on-side</td>
<td>x86_32/Linux</td>
<td>100</td>
</tr>
<tr>
<td><strong>OSG:</strong></td>
<td></td>
<td></td>
<td><strong>2758</strong></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td></td>
<td><strong>19,012</strong></td>
</tr>
</tbody>
</table>
Mission Accomplished-er

- We have been able to establish $71 \leq |C^*_6| \leq 73$

---

**Computational Statistics**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Workers</td>
<td>562.4</td>
</tr>
<tr>
<td>Max Workers</td>
<td>1775</td>
</tr>
<tr>
<td>Worker Time (years)</td>
<td>30.3</td>
</tr>
<tr>
<td>Wall Time (days)</td>
<td>19.7</td>
</tr>
<tr>
<td>Nodes</td>
<td>$1.89 \times 10^8$</td>
</tr>
<tr>
<td>LP Pivots</td>
<td>$1.82 \times 10^{11}$</td>
</tr>
</tbody>
</table>

"Mission Accomplished-est": Working on $72 \leq |C^*_6| \leq 73$

- Brings the total to $> 200$ CPU Years!
$M = 71$, Number of Processors (Slice)
Working Paper

Greatest Title Ever!

- With thanks to Greg Thain for the title (among other things)
- Submit to Teragrid ’07

Rejected

- Perhaps the program committee wishes to wait until the problem is finally solved.
- I have one thing to say to them...
You Can’t Legislate a Timeline for Completion!

Stupid Tax-o-crats!

- Having my fragile ego crushed, we may really declare “mission accomplished” and turn to the solution of other “important”, pretty integer programs
- Our approach this time is making use of new MWBlackBox Framework
Pretty IPs

- Steiner Triples
- Covering Designs
- Error Correcting Codes
MWBlackBox = Dynamic DAGMAN

- Each of the nodal subproblems is in-fact a smaller version of the original problem (with some variables fixed/removed).
- Can we use the exact same (blackbox) software that is used to solve the full-scale problem?
- Have user write no code for Task or Worker classes.
- Many people would like this functionality. Maybe you would too.

**Condor DAGMAN**
- Designed for static job dependencies
- Simple, Robust, Reliable
- Have to pay Condor executable startup overhead

**MW Blackbox**
- Designed for dynamic job dependencies
- C++ coding must be done. Less battle-tested
- Amortizes Condor executable startup overhead
BlackBox Snippets

MWTask_blackbox(const list<string> &args,
                 const list<string> &input_files,
                 const list<string> &output_files);

Creating Blackbox Tasks

1. List of arguments to the executable
2. List of input files (shipped to the worker executing the task)
3. List of output files: automatically shipped back to the master
MWBlackBox Snippets

BlackboxIPMaster::get_userinfo(int argc, char *argv[]) {
    // Set target number of workers
    RMC->set_target_num_workers(512);
    // Set name of blackbox executable
    set_executable(string("cplex"));
    // Can also stage files on the workers
    add_staged_file(param_.getProblemName());
}
BlackboxIPMaster::act_on_completed_task(MWBlackboxTask *bbt)
{
   // Can get file streams for standard output
   ifstream stdoutfile = bbt->getStdoutStream();
   // Or open streams from output file
   string ofname = bbt->getOutputFiles()[0];
   ifstream ofile = ifstream(ofname.c_str());
   // Then must parse output stream.
   // 1) Collect statistics or
   // 2) Create new tasks
}
Conclusions

- The Football Pool Problem is hard!, but now $71 \leq |C^*_6| \leq 73$
- The burning question: Will Miron let me speak about $72 \leq |C^*_6| \leq 73$ next year?
- Most important: Real large-scale applications can take advantages of all the computing power that’s out there – Local grids, Tera-grids, Open-Science Grids
- MW (via Condor) can help pull all of this together
- MWBlackBox: Write no code for worker and task. Why not try it out?
Any Questions

- www.cs.wisc.edu/condor/mw
- Please talk to Greg or I
- We’d be happy to help you get started with MW
- mailto:
  - jtl3@lehigh.edu
  - gthain@cs.wisc.edu
  - mw@cs.wisc.edu