Cloud Computing for Optimization

Jeff Linderoth

Dept. of ISyE
Univ. of Wisconsin-Madison
linderoth@wisc.edu

Informs Compoting[1] Society 2013 Santa Fe January 8, 2013

[1] Steve Dirkse's Joke

Dateline August 7, 2012...

```
Hi Jeff,
We are trying to do some new things at ICS 2013, and the notion
of "advanced tutorials" is one of these things.  Is there any
chance you would be interested in presenting an advanced tutorial
on "cloud computing for OR?" We would like to see something
involving cloud computing, and the list of experts in that area
is rather short...
Thanks!  jpw
```

- Jeff delays....
- He doesn't really know very much modern information about Cloud Computing

Joint Project Meeting, September 13, 2012

"Jeff, I never heard from you about whether you would be willing to give an Advanced Tutorial"

"That's because I don't think I know enough about that to give a good tutorial."

"I will buy you a beer"

I can definitely do the tutorial

Not All Were Happy With This Decision...

On 9/23/2012 9:42 PM, Ted Ralphs wrote:
Well, if I had known Jeff was going to undermine the credibility of the Advanced Tutorials by talking about some "Cloud" BS, I wouldn't have agreed. But I guess it's too late to withdraw now.

- I honestly don't think "the cloud" is BS.
- However, there is a lot of hype around the idea

It's a *BAD* Title

Advanced Tutorial: Cloud Computing for Optimization

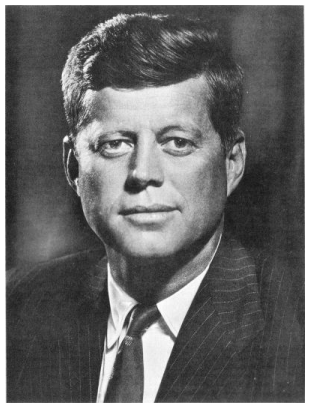
1. Advanced? – The things I am talking about are relatively basic
2. Tutorial – Implies that I know what I am talking about.
   - I don't actively work in this area anymore.
3. Cloud Computing for Optimization
   - Will be biased (heavily) towards the cloud computing modes and types that I know about.

The Goal

- I hope this tutorial raises more questions than answers
- I also don't have the energy to talk for 90 minutes, so I am hoping that we will have lots of time for questions

Obviously...

$\neq$



This Talk is The Opposite of the one JFK Would Give

- Ask not what you can do for the cloud
- Ask what the cloud can do for you...

- A discussion of how Operations Research can help solve decision/resource allocation problems that cloud computing deployments face:
  - I. Iyoob, E. Zarifoglu, and A. B. Dieker "Cloud Computing Operations Research"
    `www2.isye.gatech.edu/~adieker3/publications/cloudOR.pdf`

Outline

- What is Cloud Computing?
- A Little "History" of Cloud Computing

Cloud Computing Tools

- HTCondor
- NEOS
  - Using NEOS via XMLRPC
- Others—Solver Studio, Optimization Services, Gurobi Cloud

Big Computing!

- Federating resources on the cloud
- MW—Building cloud-enabled parallel applications
- Experiences in solving the Football Pool Problem

The $64 Question

What Is Cloud Computing?

- I (honestly) didn't know!
- Being a good Madison, WI liberal, I turned to the one source that I was sure could help me...

The Government!!

The NIST Definition of Cloud Computing

Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interation.

`http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`

More about Cloud Computing

Five Essential characteristics

1. On-demand self-service
2. Broad network access
3. Resource pooling
4. Rapid elasticity
5. Measured service

- If you have these five characteristics, then this is a cloud infrastructure

Cloud Characteristics

1. On-demand self-service – Need no human interaction to obtain resources
2. Broad network access – Available through standard network protocols
3. Resource pooling – Provider has large pool of computing resources to serve multiple requests
4. Rapid elasticity – Capabilities can be automatically provisioned and released, to meet demand
5. Measured service – Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Service Models for Cloud Computing

1. Software as a service (SaaS):
   - Use provider's applications running on cloud infrastructure
   - Google Apps, NEOS, Gurobi Cloud
2. Platform as a Service: (PaaS)
   - Allow the user to deploy onto cloud infrastructure user-specific applications
   - Google App Engine, Condor, MW
3. Infrastructure as a Service (IaaS).
   - Can actually provision and control certain aspects of cloud infrastructure. (Operating systems)
   - Google Compute Engine... Dropbox, iCloud (Cloud Storage)

The Service Models

IaaS	PaaS	SaaS
Applications	Applications	Applications
Data	Data	Data
Runtime	Runtime	Runtime
O/S	O/S	O/S
Virtualization	Virtualization	Virtualization
Servers	Servers	Servers
Storage	Storage	Storage
Networking	Networking	Networking

- Highlighted text describes which aspects user controls in different cloud deployment models

`http://venturebeat.com/2011/11/14/cloud-iaas-paas-saas/`

- Why did JP and Bill think I knew anything about Cloud Computing?
- Because I used to do something called metacomputing

Way Back in 1995 – Metacomputing was born!

- (http://archive.ncsa.illinois.edu/Cyberia/MetaComp/WhatIs.html)

‘‘When you switched on your razor or blow dryer or toaster this morning, you probably gave no thought to the original source of your electricity. You didn’t have to select a generator with adequate capacity; nor did you consider the gauge of wire used to connect the outlet or whether the power lines are underground or on poles. You were using a network of electric power sophisticated enough to route the electrons across hundreds of miles, and easy enough for a child to use.’’

‘‘In theory, a metacomputer is a similarly easy-to-use assembly of distinct computers or processors working together to tackle a single task or set of problems... The machines and/or instruments may be located the same building, or separated by thousands of miles, yet their capabilities appear on the user’s workstation screen as one system.’’

Quote By Larry Smarr (NCSA,UCSD)

"Beneath the calm and the deference, Larry is an intellectual pitchman of the first order"

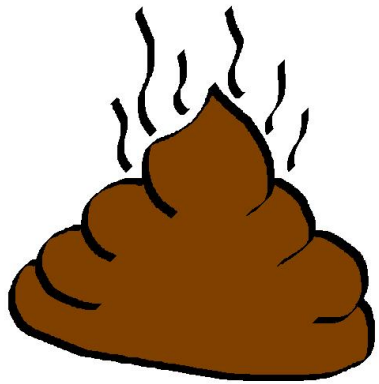
http://www.theatlantic.com/magazine/archive/2012/07/the-measured-man/309018/

Larry is now VERY into quantified health

- Keeps meticulous notes on his diet, biochemical markers, exercise, and supplemental vitamins, saying, "I routinely use food and supplements to tune my numbers to more optimal levels."

EWWWW!!!!!!

"human stool has a data capacity of 100,000 terabytes of information stored per gram. That's many orders of magnitude more information density than, say, in a chip in your smartphone or your personal computer. So your stool is far more interesting than a computer."



What's in a name?

It's All About the Benjamins

- In the never-ending quest to obtain sustainable funding, "distributed systems" invents many new names for itself...

- Metacomputing: circa 1995-1999
- The Grid: circa 2000-2005.
  - A form of distributed and parallel computing, whereby a "super and virtual computer" is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks.
  - "Computational Grid" $\approx$ Electricity Grid

More Names

- Cyberinfrastructure: 2005—
  - "Cyberinfrastructure describes the melding of tools and capabilities—hardware, middleware, software applications, algorithms, and networking—that are now transforming research and education, and are likely to do so for decades to come. There is a gathering avalanche of demand for cyberinfrastructure to suit a wide variety of needs in the open science community." (Arden Bement Jt, Director, NSF)
  - National Science Foundation (NSF) creates Office of Cyberinfrastructure (OCI)
- Cloud Computing: 2006?—
  - Eric Schmidt (Google) describes their approach to SaaS as "cloud computing" at a search engine conference.
- I honestly hope that Larry Smarr does not invent a new type of "biological computing"

Cloud Questions

- How can optimizers use "the cloud?"
- How should optimizers use "the cloud?"
- Why aren't more people using the cloud for optimization?

Using The Cloud

- An easy way to use the cloud is for "pleasantly parallel" applications
  - Require no communication/synchronization between tasks
  - Monte Carlo simulations
  - Solving different instances for testing purposes

Platform as a Service

- HTCondor is a software package from which one can build a cloud computing platform

Some of you may recall...

- Miron Livny gave the plenary at ICS2009: "Harnessing High Throughput Computing for Compute Intensive Science"

HTCondor

THE UNIVERSITY of WISCONSIN MADISON

Miron Livny
Todd Tannenbaum
Greg Thain

http://research.cs.wisc.edu/htcondor/

What's With the HT?

- The software used to be called Condor
- HT: High-Throughput
- Late 2012, Reached an out of court settlement with Phoenix Software (`http://www.phoenixsoftware.com/`): a developer of software libraries of Mainframe machines

---

```
Date: Wed, 24 Oct 2012 18:17:44 -0500
From: Todd Tannenbaum <tannenba@xxxxxxxxxxx>

Subject: [Condor-users] "Condor" name changing to "HTCondor"
In order to resolve a lawsuit challenging the University of
Wisconsin-Madison's use of the "Condor" trademark, the University
has agreed to begin referring to its Condor software as "HTCondor".
```

What is HTCondor?

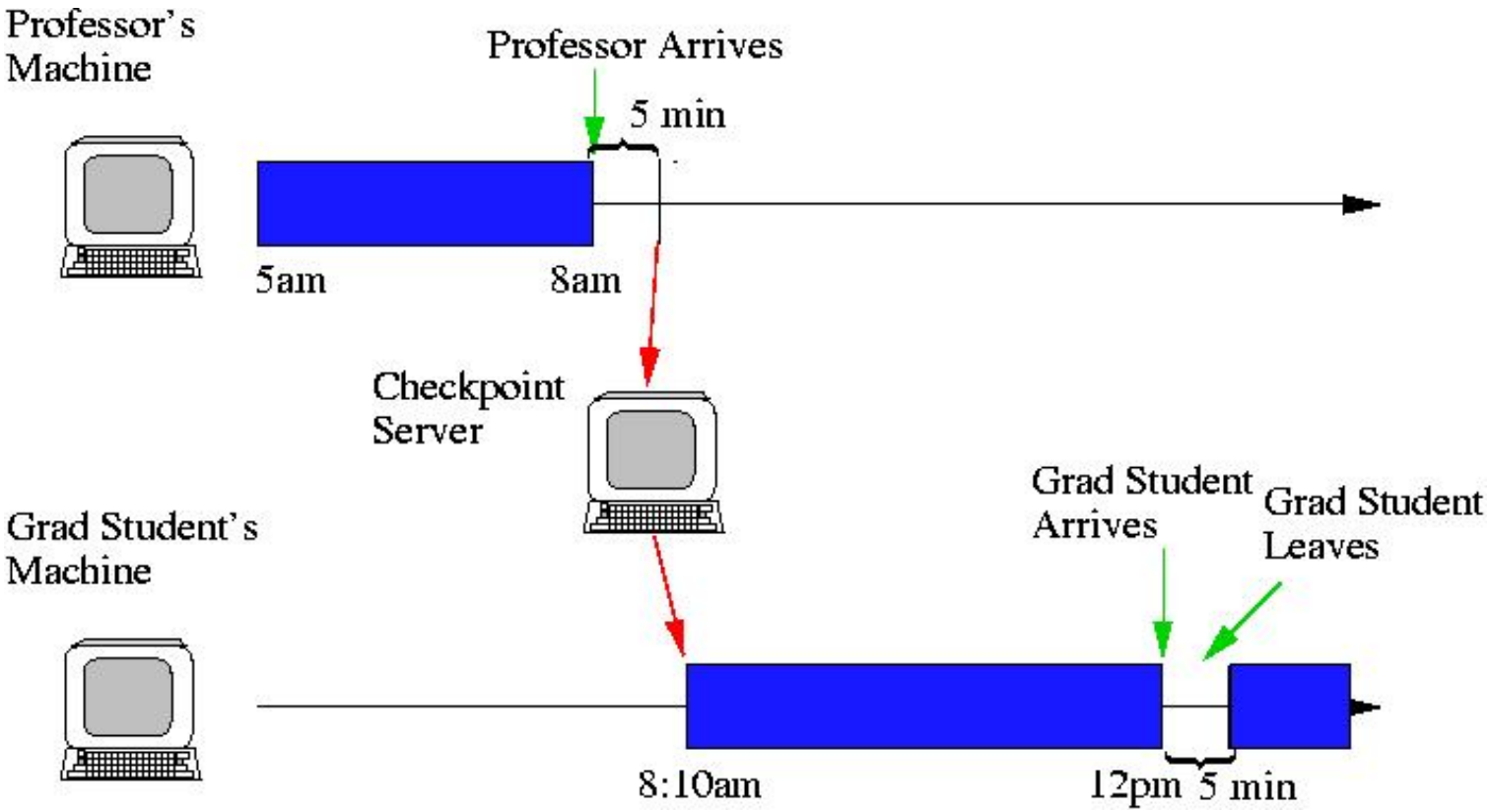
- Manages collections of "distributively owned" workstations
  - User need not have an account or access to the machine. Jobs submit to a central location.
  - Workstation owner specifies conditions under which jobs are allowed to run
  - All jobs are scheduled and "fairly" allocated among the pool
- How does it do this?
  - Scheduling/Matchmaking
  - Jobs can be checkpointed and migrated
  - Remote system calls provide the originating machines environment
  - Pecking Order: Users are assigned priorities based on the number of CPU cycles they have recently used. If someone with higher priority wants a machine, your job will be booted off

Matchmaking – Classified Ads

MyType = Job
TargetType = Machine
Owner = ferris
Cmd = cplex
Args = seymour.d10.mps
HasCplex = TRUE
Memory $\geq$ 512
Rank = KFlops
Arch = SUN4u
OpSys = SOLARIS27 || SOLARIS28

MyType = Machine
TargetType = Job
Name = nova9
HasCplex = TRUE
Arch = SUN4u
OpSys = SOLARIS27
Memory = 1024
KFlops = 53997
RebootedDaily = TRUE

Checkpointing/Migration



Recent/Modern Condor Features

Support for virtualization

- Has a `vm` universe for matching and landing a disk image on an execute machine within an HTCondor pool. This disk image is intended to be a virtual machine.
- Has support for `vm_type = vmware`, `vm_type = xen`, or `vm_type = kvm`

Cloud Service

- HTCondor jobs may be submitted to clouds supporting Amazon's Elastic Compute Cloud (EC2) interface.
- Amazon's EC2 is an on-line commercial service that allows the rental of computers by the hour to run computational applications.
- Condor can run virtual machine images that have been uploaded to Amazon's online storage service (S3 or EBS).

Using The Cloud

Testing Algorithms

- Why test on 10 instances, when you could be testing on 100?
- It is a requirement that all my students use Condor during development and testing

---

- Submitting many instances is as "simple" as creating the appropriate job submission file.
- I often use a python script to generate these files

A (Complicated) Sample Condor Job Submission File

```
universe = vanilla
executable = bcps.$$(Opsys).$$(Arch)
rank = kflops
requirements = ((Arch=="X86_64") && (Opsys=="LINUX")) ||
               ((Arch=="INTEL") && (Opsys== "LINUX"))
should_transfer_files = yes
notification = never
transfer_input_files = moa5637_26_71/bidon.mylp,moa5637_26_71/bidon.grp
input = moa5637_26_71/bidon.inp
arguments = -O2 -x
output = moa5637_26_71.out
queue
```

```
transfer_input_files = moa4034_23_101/bidon.mylp,moa4034_23_101/bidon.grp
input = moa4034_23_101/bidon.inp
arguments = -O2 -x
output = moa4034_23_101.out
queue
```

- I typically write a python script to generate job description files like this

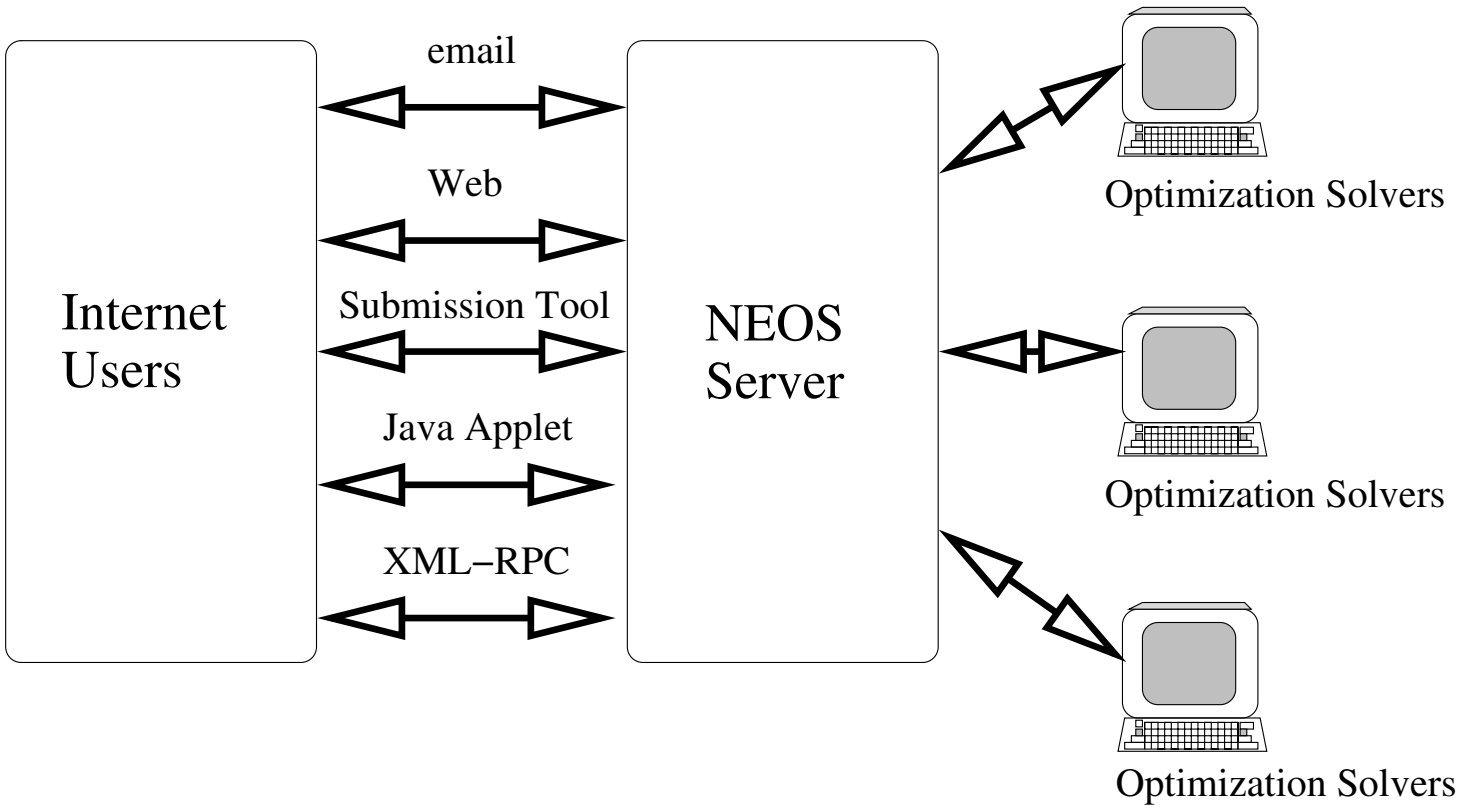
Cloud Computing for Optimization (Before it was cool)

Optimization SaaS

- NEOS—Network Enabled Optimization Server (`neos-server.org`)
- An easy interface to allow users to solve their numerical optimization problems with remote resources.

- Started in 1994. MCS Division, Argonne National Lab. email interface to Server, based on netlib
- February 2002, Version 4. Kestrel interface – allows remote solution from within modeling language environments
- April 2005. Version 5. XML-RPC API Interface.
- Summer, 2010. NEOS Moves to Wisconsin

The NEOS System



It Keeps Going

- 1999: 1400 jobs/month
- 2001: 6200 jobs/month
- 2005: 12000 jobs/month
- 2012: 29000 jobs/month

Top Solver Use in 2012

Gurobi	59248	XpressMP	17426
MINOS	31191	MINTO	15525
CONOPT	27640	SNOPT	14738
SBB	20082	Bonmin	12928
KNITRO	19145	Ipopt	12558

One Use of Optimization in the Cloud

- Distributing/building optimization software is hard
- This (honestly) was one of the primary original motivations of NEOS
- In the 1990s many people wanted to be able to try many different (NLP) solvers for their problem, without having to download/build/install the software

---

- Even now, if you want to test how a specific solver may work on your class of problems, it may take some significant startup effort
- As an example, if you want to use our super-duper new open-source solver

GAE—(Greatest Acronym Ever)

Mixed
I nteger
Nonlinear
Optimization
Toolkit:
Algorithms,
Underestimators,
Relaxations.

Building Running Minotaur Requires

- AMPL Solver Interface,BOOST Headers,CLP,`Cppunit`,FILTER-SQP,IPOPT,MUMPS,Fortran,Blas,Lapack

- Making software available as a service allows people to use it/test it

NEOS API

`http://www.neos-server.org/neos/NEOS-API.html`

- NEOS runs an XML-RPC server to communicate with clients for submitting and retrieving jobs.
  - `http://www.xmlrpc.com`
- The server can communicate with clients written in Python, Perl, PHP, C, C++, Java, Ruby, ...

Submitting a job in 3 easy steps

1. Create a xmlrpc server object:
   ```
   neos = xmlrpclib.Server("http://%s:%d" %
   ("neos-server.org",3332))
   ```
2. Form XML string in format required by desired solver
3. (jobId, Password) = `neos.submitJob(xmlstring)`

NEOS API

Information

- `getSolverTemplate(category, solvername, inputMethod)`— Returns a template for a given solver.
- `listAllSolvers()`—Lists all solvers available on NEOS.
- `printQueue()`—Returns a string containing the current NEOS jobs.

Job Handling

- `submitJob(xmlstring)`—Submit an optimization job to NEOS. Returns tuple of (jobID, Password)
- `getJobStatus(jobNumber, password)`—Get the current status of your job. ("Done", "Running", "Waiting", "Unknown Job", or "Bad Password")
- `getFinalResults(jobNumber, password)`—Gets results of job from NEOS.

A TUTORIAL!!!!

- I was recently fascinated by Matteo's Fischetti colorful talk about the role that randomness can play in the performance of MIP solvers
- See Mike's blog post: (`http://mat.tepper.cmu.edu/blog/?p=1695`)

Use The Cloud!

- Optimizers should make use of additional resources to quantify the impact of randomness on search
- We'll do it on NEOS

Code Outline

Submit Jobs

1. Read an MPS file
2. For $i = 1, \ldots, N$
   - Scramble the columns
   - Submit scrambled instance to SYMPHONY and CBC
   - Write jobID and Password to a file

Retrieve Results

- Read IDs and Passwords from jobfile
- Check status of each job
- If all done, Retrieve results (string)
- Parse string for information (in this case number of nodes)

submit-jobs.py

```
f = open(sys.argv[1],"r")
cols = readMPS(f)
f.close()

neos = xmlrpclib.Server("http://%s:%d" % ("neos-server.org",3332))

jname = jobfile_name()
sys.stdout.write("Running jobs for batch %s\n" % jname)
jfile = open(jname,"w")
xml = ""
for i in xrange(5):

    f = open(sys.argv[1],"r")
    mps = scrambleMPS(f, cols)
    f.close()

    xml = write_symphony_header()
    xml += "<MPS><![CDATA["
    xml += mps
    xml += "]]></MPS>\n"
    xml += write_symphony_footer()

    (jobNumber,password) = neos.submitJob(xml)
    sys.stdout.write("Submit job # = %d\n" % jobNumber)
    jfile.write(str(jobNumber) + " , " + password + " , SYMPHONY\n")

    xml = write_cbc_header()
    xml += "<MPS><![CDATA["
    xml += mps
    xml += "]]></MPS>\n"
    xml += write_cbc_footer()
```

gather-results.py

```
neos = xmlrpclib.Server("http://%s:%d" % ("neos-server.org",3332))

alldone = True
status = ""
for (jobId,pw,solver) in jobs:
    print "Checking id: " + str(jobId) + "  PW: " + pw
    status = neos.getJobStatus(jobId,pw)
    if status != "Done":
        alldone = False

msg = ""
symnodes = []
cbcnodes = []
if alldone:
    for (jobId,pw,solver) in jobs:
        msg = neos.getFinalResults(jobId,pw).data
        sList = (line for line in msg.split(os.linesep))
        for line in sList:
            if symnodesreg.match(line):
                a = line.split()
                nodes = int(a[4])
                symnodes.append(nodes)
            if cbcnodesreg.match(line):
                a = line.split()
                nodes = int(a[2])
                cbcnodes.append(nodes)
    print "Symphony Nodes: " + str(symnodes)
    print "Cbc Nodes: " + str(cbcnodes)
```

The Latest with NEOS

- Server moved to Wisconsin in 2010. (Michael Ferris is in charge)
- Has some (small amount) of dedicated development and support
- Supported by Wisconsin Institutes of Discovery

Currently

- Most solvers hosted at Wisconsin
- Four 32 core machines
- Solvers are also hosted at Arizona State (Hans Mittelmann)

Next Generation

- We have an alpha version of NEOS that now uses HTCondor for local scheduling.
- Should be rolled out in 3 months or so

SolverStudio

Oops!

```
LINDEROTH:13Jan-ICS-SantaFe linderoth$ pdftk A=cloud.pdf
B=SolverStudioOverview-v4.pdf cat A1-A40 B A41-end output
test.pdf


pdftk(3953,0x7fff705b0cc0) malloc:  *** error for object
0x10418c880:  pointer being freed was not allocated *** set
a breakpoint in malloc_error_break to debug Abort trap
```

Optimization Services

- `http://www.optimizationservices.org/`
- "Next-generation NEOS"

OS is Many things...

- XML-Based standard for problem representation and an API for modifying instances
- Client side software that is used to create Web Services (SOAP) packages with OSiL instances and contact a server for solution.
- Standards that facilitate the communication between clients and solvers using Web Services.
- Server software that works with Apache Tomcat. This software uses Web Services technology and acts a middleware between the client that creates the instance and solver on the server that optimizes the instance and returns the result.

Gurobi Optimizer - Gurobi Cloud

`http://www.gurobi.com/products/gurobi-cloud`

- Run Gurobi using Amazon EC2. No need to purchase license/install software.

---

1. Create an Amazon Web Services (AWS) account.
2. 'Purchase' Gurobi. (Free – adds Gurobi Cloud as option for AWS)
3. Log into AWS Management Console
4. ssh to client
5. Use Gurobi

Combining Local Grids

More, More, More!

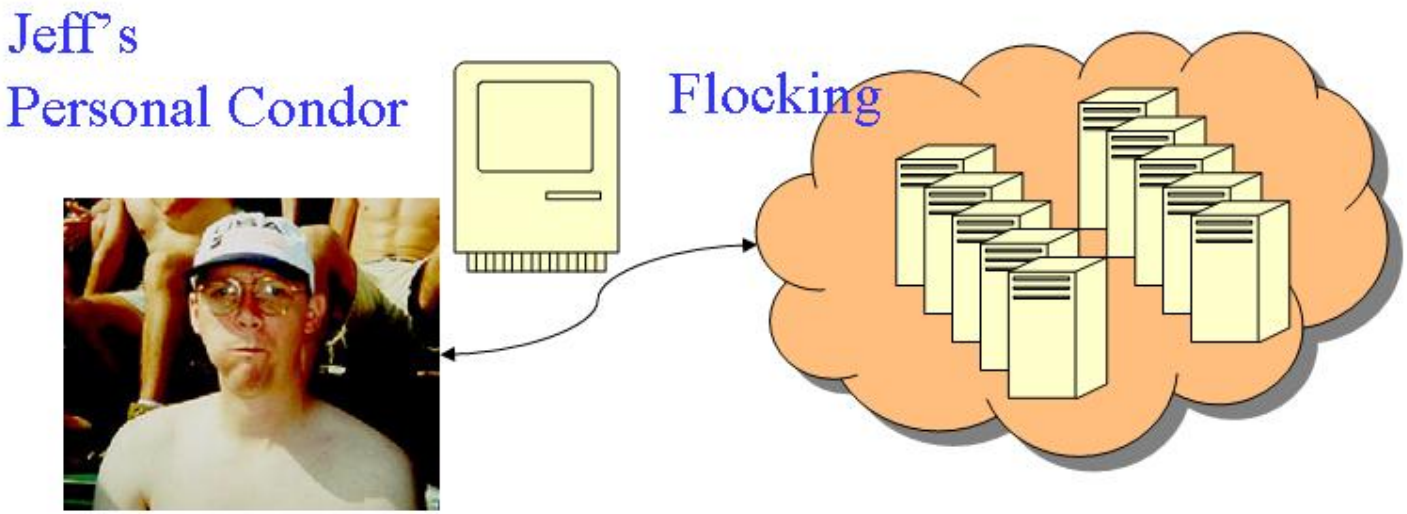
Condor has mechanisms for bringing grids together.

1. Flocking
   - Condor jobs can negotiate to run in other Condor pools.
2. Glide-in
   - Globus (Grid computing toolkit from Argonne) provides a "front-end" to many traditional supercomputing sites.
   - Submit a Globus job which creates a temporary Condor pool on the supercomputer, on which users jobs may run.
3. Schedd-on-the-side
   - A "secondary" scheduler that can transmogrify idle jobs in the local Condor queue to valid jobs on remote (Grid) resources, like the (NSF) Teragrid (Now XSEDE `https://portal.xsede.org`), or the Open Science Grid. `https://www.opensciencegrid.org`

Building a Big Pool—Mechanism #1—Flocking

Building My Grid Via Flocking

- UW-Madison Condor Pool in CS Dept.
- Large Condor Pool at NCSA
- Small pool lab at Lehigh: COR@L

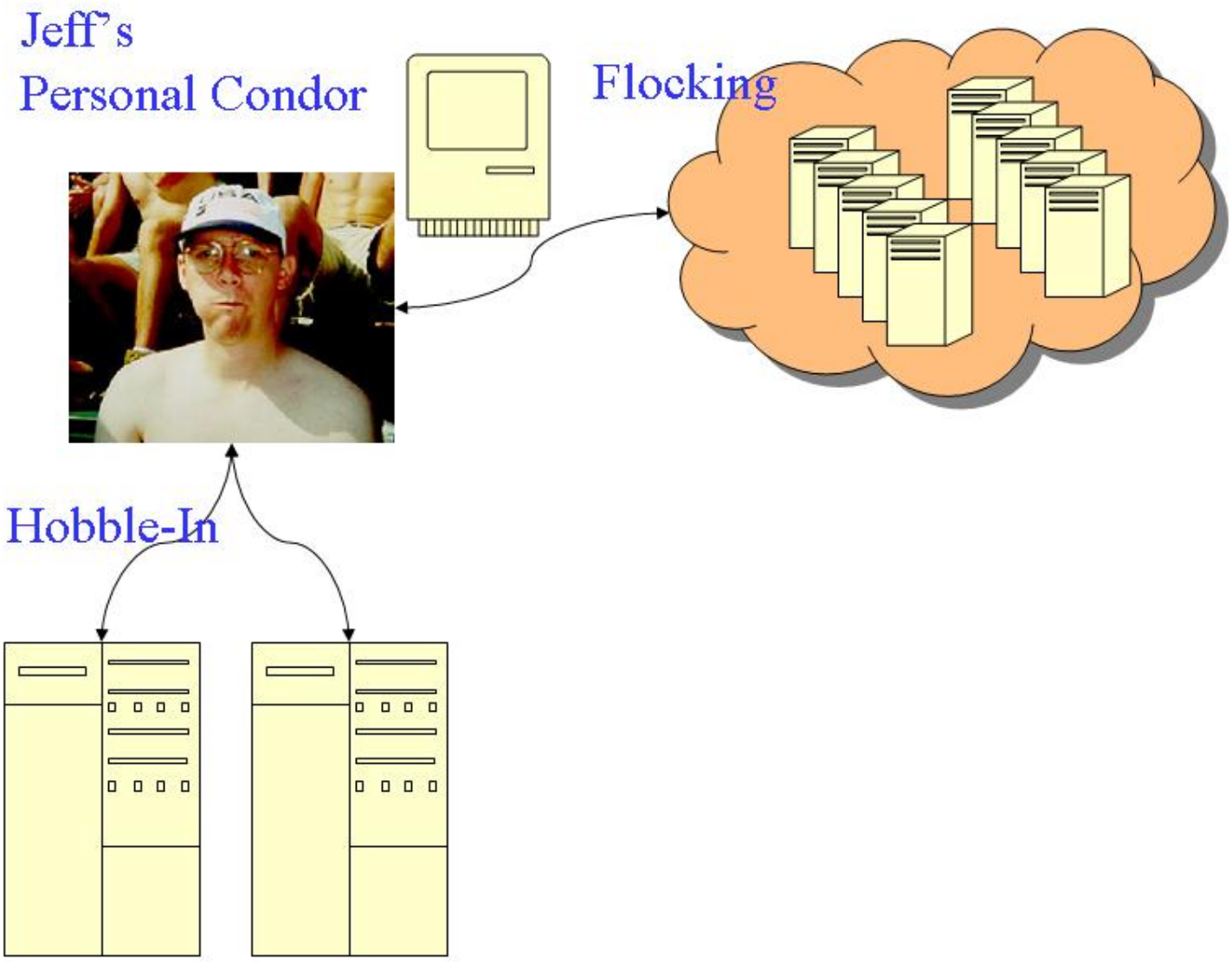


Mechanism #2: Hobble-In

1. Log in to Teragrid (XSEDE) site
2. Install `condor_startd` and `condor_starter`
3. Configure condor to report to your condor master
4. Write PBS or LSF scripts that run "preconfigured" condor startd's for limited about of time.
5. Submit away. Machines will appear in your personal pool when local scheduler deigns to run them.

- Condor glide-in tool does this all automatically – but didn't back when I was building my super-grid in 2006/2007

Building Your Grid: Flocking + Hobble-In



Mechanism #3: sshIdle-In

- 300 Machines at Lehigh were on running Condor on a private network: And I wanted to use them!

sshIdle-In

- A (hacky) mechanism for running on private networks
- Works directly with master-worker toolkit MW (explained in a bit)

1. Look up port number where master is listening: default 8997.
2. Log into "landing point" – machine with connections to both private network and outside world
3. Transfer (or build) worker executables on landing point
4. Forward traffic:

```
ssh -l fmargot -g -N -f -L 8997:0.0.0.0:8997 meisterbrau.cs.wisc.edu
```

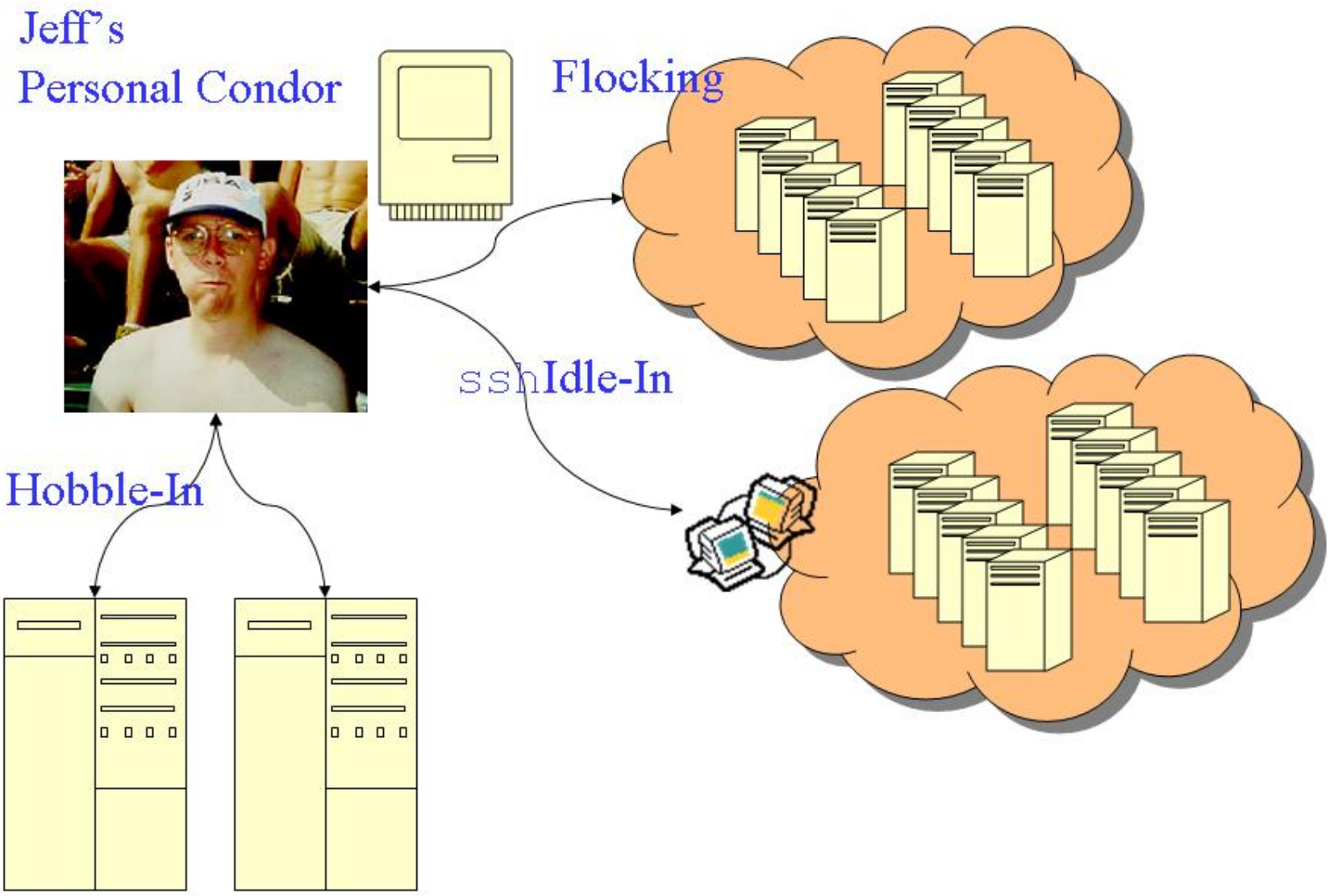
5. Submit worker executables in private network (via condor job submission)

Sample sshIdle-In Script

```
Universe = Vanilla
Executable = mw_exec0.$$(Opsys).$$(Arch).exe
arguments = 1512 8997 8997 192.168.1.1
should_transfer_files = Yes
when_to_transfer_output = ON_EXIT
rank = Mips
on_exit_remove = false
nice_user = true
queue

arguments = 1513 8997 8997 192.168.1.1
queue
arguments = 1514 8997 8997 192.168.1.1
queue
arguments = 1515 8997 8997 192.168.1.1
queue
```

Flocking + Hobble-In + sshIdle-In



Mechanism #3: IceCream-In

Stardate: Date: Thu, 20 Apr 2006 16:22:28 -0500

A Conversation Over Ice Cream...

"I really like all these mechanisms, Miron, but where and how can I get some more workers?"

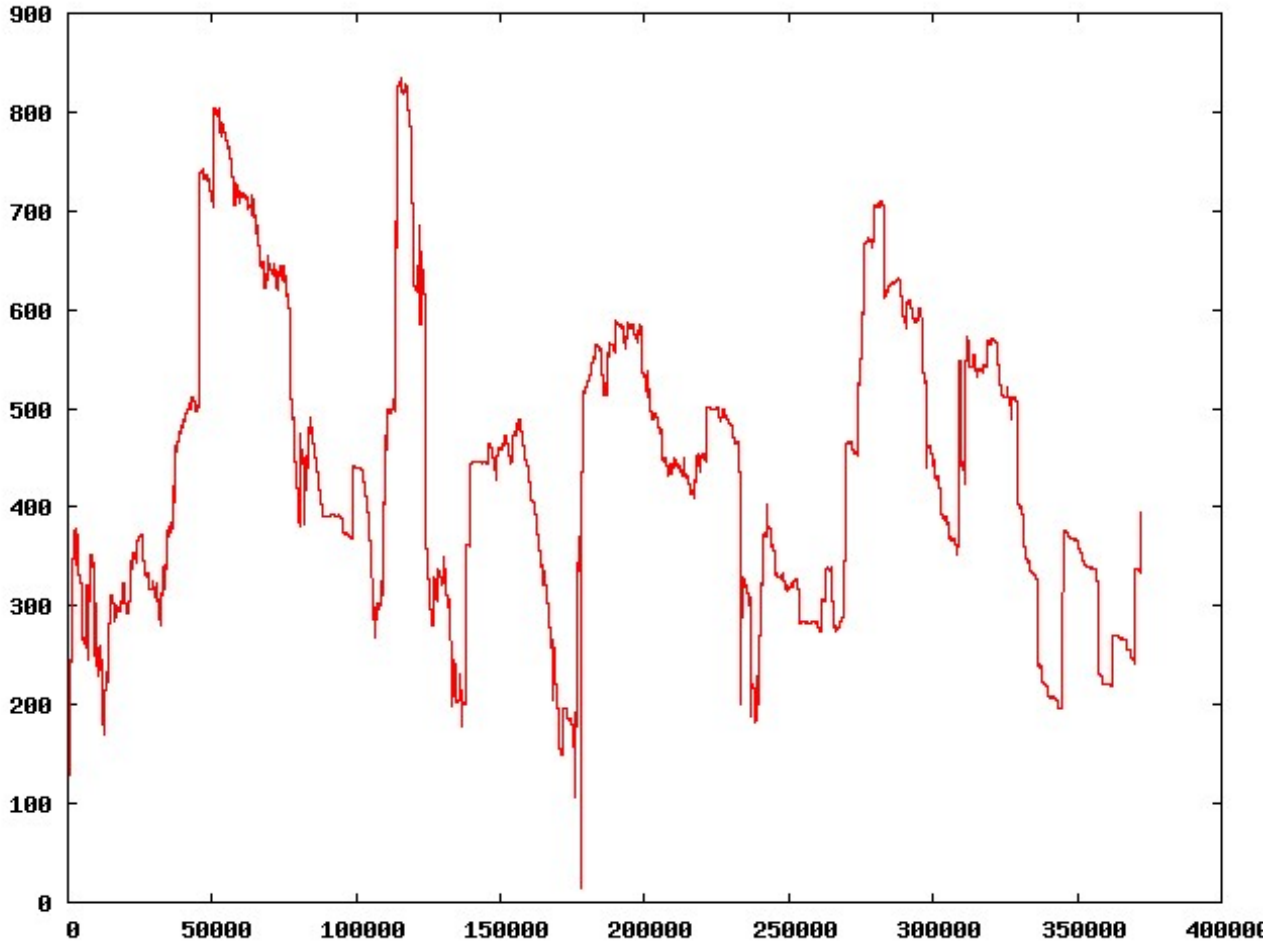
"There's this thing called the Open Science Grid. I'll see what I can do..."

Greg is My Hero!

Stardate: Date: Thu, 20 Apr 2006 17:15:47 -0500

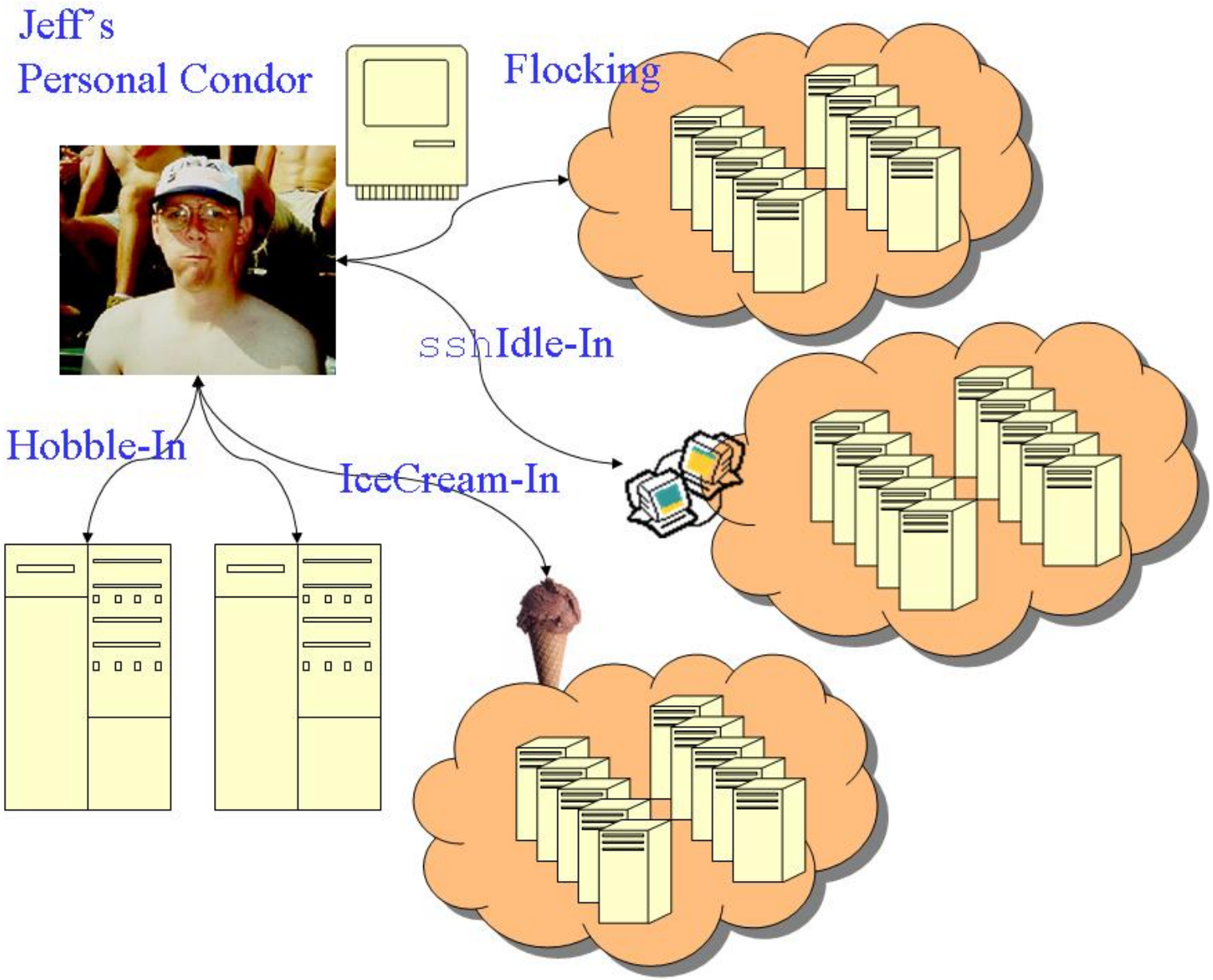
Jeff:
The Open Science Grid has asked us for some more jobs to demonstrate some of our Grid technologies, so I'm working on getting some of the football pool jobs routed over there. Hang on for some Super Grid Power! I need to reboot the personal condor first, so don't worry if the ff3 output looks a little wonky.
-greg

Workers in Solving One Batch of Subproblems



- This is the closest thing I've seen to a true "Cloud" reality: I'm eating an ice cream cone, and I get 300 new machines...

Jeff's Super-Duper Computational Grid



You Are Jealous

Site	Access Method	Arch/OS	Machines
Wisconsin - CS	Flocking	x86_32/Linux	975
Wisconsin - CS	Flocking	Windows	126
Wisconsin - CAE	Remote submit	x86_32/Linux	89
Wisconsin - CAE	Remote submit	Windows	936
Lehigh - COR@L Lab	Flocking	x86_32/Linux	57
Lehigh - Campus desktops	Remote Submit	Windows	803
Lehigh - Beowulf	ssh-Idle in	x86_32	184
Lehigh - Beowulf	ssh-Idle in	x86_64	120
OSG - Wisconsin	Schedd-on-side	x86_32/Linux	1000
OSG - Nebraska	Schedd-on-side	x86_32/Linux	200
OSG - Caltech	Schedd-on-side	x86_32/Linux	500
OSG - Arkansas	Schedd-on-side	x86_32/Linux	8
OSG - BNL	Schedd-on-side	x86_32/Linux	250
OSG - MIT	Schedd-on-side	x86_32/Linux	200
OSG - Purdue	Schedd-on-side	x86_32/Linux	500
OSG - Florida	Schedd-on-side	x86_32/Linux	100

Computational Grid, cont.

Site	Access Method	Arch/OS	Machines
TG - NCSA	Flocking	x86_32/Linux	494
TG - NCSA	Flocking	x86_64/Linux	406
TG - NCSA	Hobble-in	ia64-linux	1732
TG - ANL/UC	Hobble-in	ia-32/Linux	192
TG - ANL/UC	Hobble-in	ia-64/Linux	128
TG - TACC	Hobble-in	x86_64/Linux	5100
TG - SDSC	Hobble-in	ia-64/Linux	524
TG - Purdue	ssh-idle in	x86_32/Linux	1099
TG - Purdue	ssh-idle in	x86_64/Linux	1529
TG - Purdue	ssh-idle in	Windows	1460
			19,012

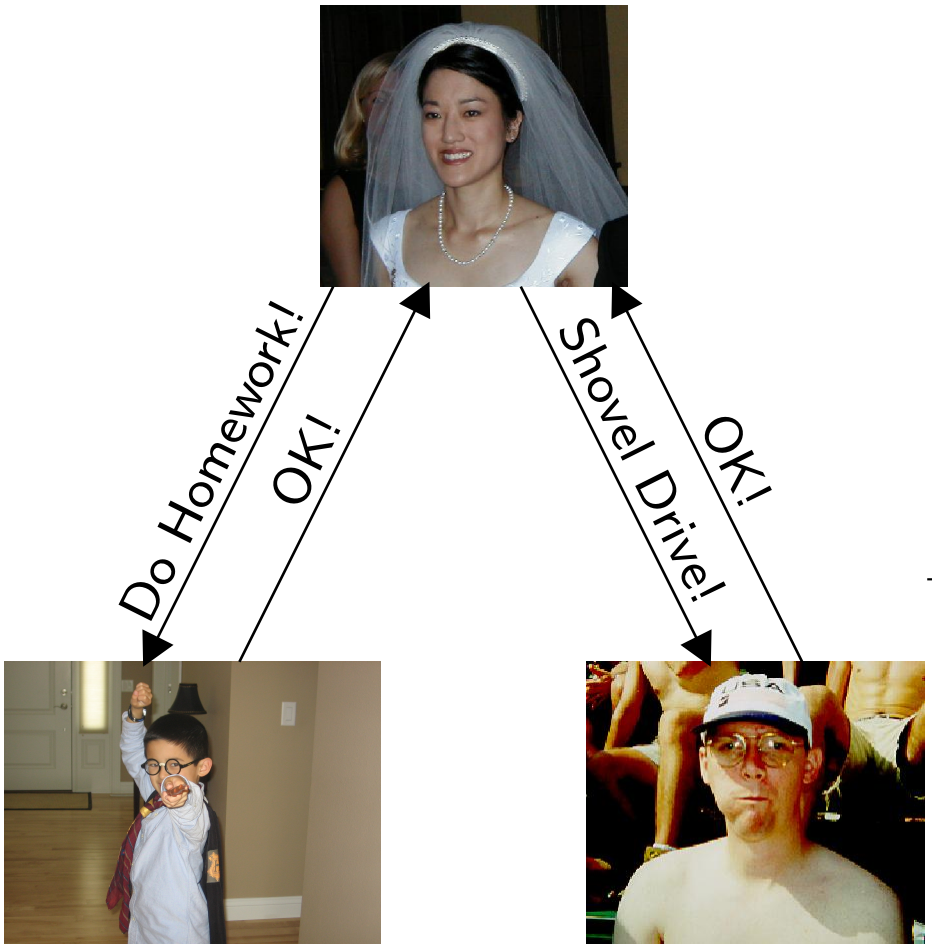
Grid-Enabling Algorithms

- Condor and Cloud-flocking mechanisms gives us the infrastructure from which to build a grid (the spare CPU cycles),
- We still need a mechanism for controlling the algorithm on a computational grid
- No guarantee about how long a processor will be available.
- No guarantee about when new processors will become available

---

- To make parallel algorithms dynamically adjustable and fault-tolerant, we could (should?) use the master-worker paradigm
- What is the master-worker paradigm, you ask?

Master-Worker!



- Master assigns tasks to the workers
- Workers perform tasks, and report results back to master
- Workers do not communicate (except through the master)

- Simple!
- Fault-tolerant
- Dynamic

Tool II – MW

Sanjeev Kulkarni
Greg Thain
Mike Yoder

Jean-Pierre Goux
Jeff Linderoth

`http://research.cs.wisc.edu/htcondor/mw/`

MW : A Master-Worker Grid Toolkit

- There are three abstraction in the master-worker paradigm: Master, Worker, and Task.
- MW is a software package that encapsulates these abstractions
  - API : C++ abstract classes
  - User writes 10 methods
  - The MWized code will transparently adapt to the dynamic and heterogeneous computing environment
- MW also has abstract layer to resource management and communications packages (an Infrastructure Programming Interface)
  - Condor/Sockets, Condor/PVM, Condor/Files, Condor/MPI, Single processor
- It's Free!, like free beer:
  `http://research.cs.wisc.edu/htcondor/mw/`

MWInfo

MW is middleware that (attempts to) make it easy to write large-scale parallel applications that can be flexible and agile, thus allowing ordinary users (like me) to run extraordinarily large computations.

The MW API

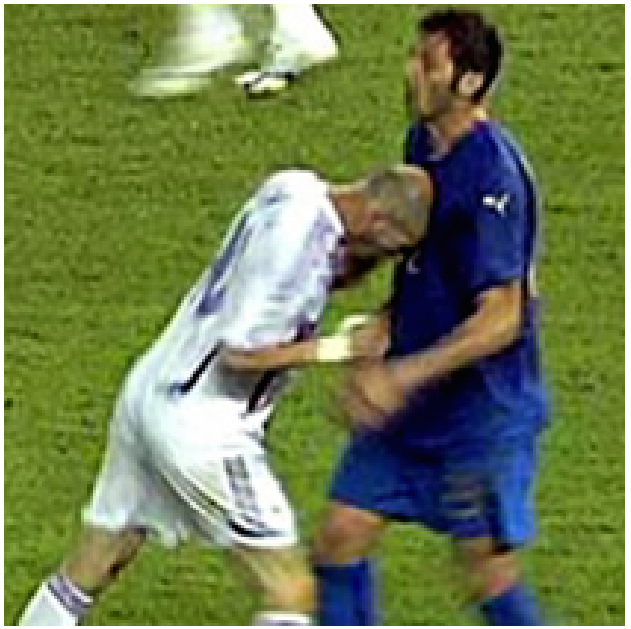
- MWMaster
  - `get_userinfo()`
  - `setup_initial_tasks()`
  - `pack_worker_init_data()`
  - `act_on_completed_task()`
- MWTask
  - `pack_work()`, `unpack_work()`
  - `pack_result()`, `unpack_result()`
- MWWorker
  - `unpack_worker_init_data()`
  - `execute_task()`

A smattering of MW Applications (for Optimization)

- MWFATCOP (Chen, Ferris, L) – A branch and cut code for linear integer programming
- MWMINLP (Goux, Leyffer, Nocedal) – A branch and bound code for nonlinear integer programming
- MWQAP (Anstreicher, Brixius, Goux, L) – A branch and bound code for solving the quadratic assignment problem
- MWATR (L, Shapiro, Wright) – A trust-region-enhanced cutting plane code for linear stochastic programming and statistical verification of solution quality.
- MWCouenne (Nannicini *et al.*) – Couenne, wrapped in MW
- MWSYMCOP (L, Margot, Thain) – A branch-and-bound code designed to solve symmetric integer programs

Motivation: Gambling—Football Pool Problem

- Predict the outcome of ν soccer matches
- You win if you miss at most $\mathrm{d} = 1$ games



The Football Pool Problem

What is the minimum number of tickets you must buy to assure yourself a win?

Football IP

- Let N be the set of possible outcomes (tickets) ($|N| = 3^v$)
- Binary variables: $x_j = 1$ if I purchase ticket $j \in N$
- Let $A \in \{0,1\}^{|N| \times |N|}$ with $a_{ij} = 1$ iff ticket $j \in N$ is a winner for outcome $i \in N$

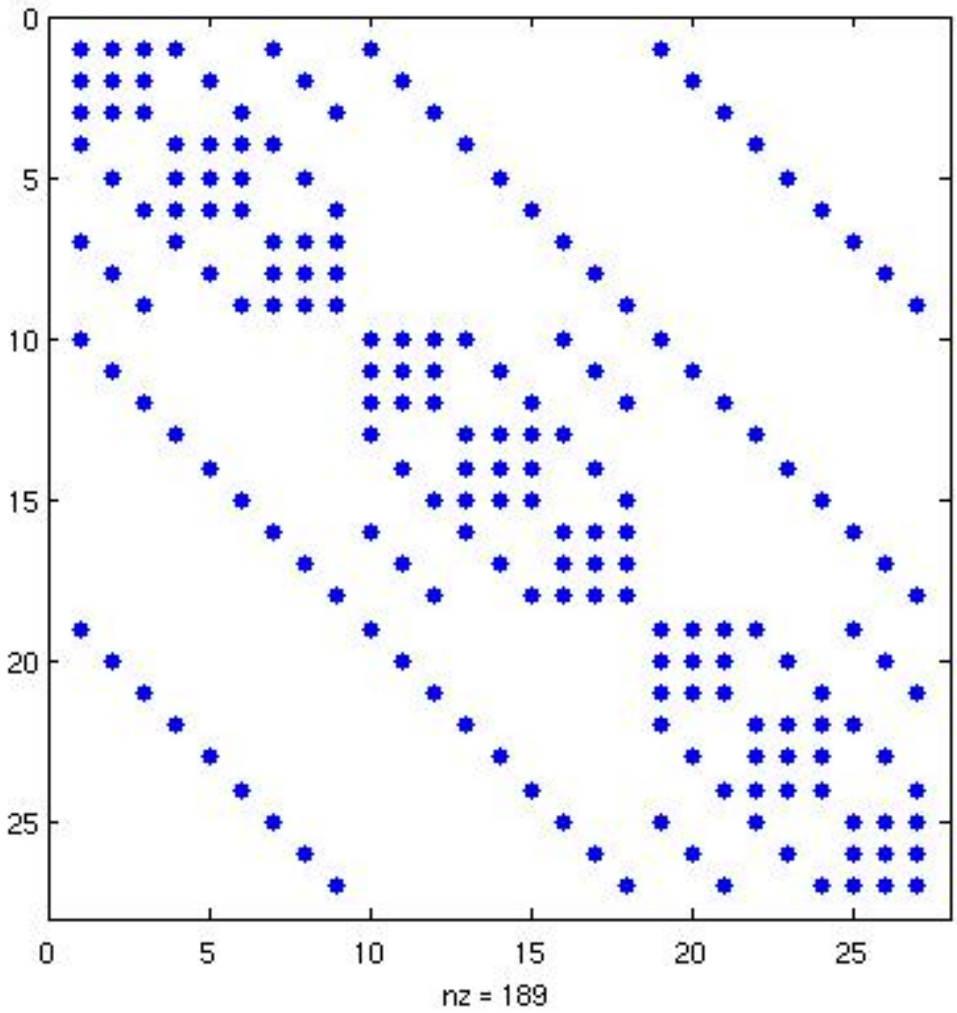
IP Formulation

$$\begin{aligned} \min \quad & 1^T x \\ \text{s.t.} \quad & Ax \geq 1 \\ & x \in \{0,1\}^{|N|} \end{aligned}$$

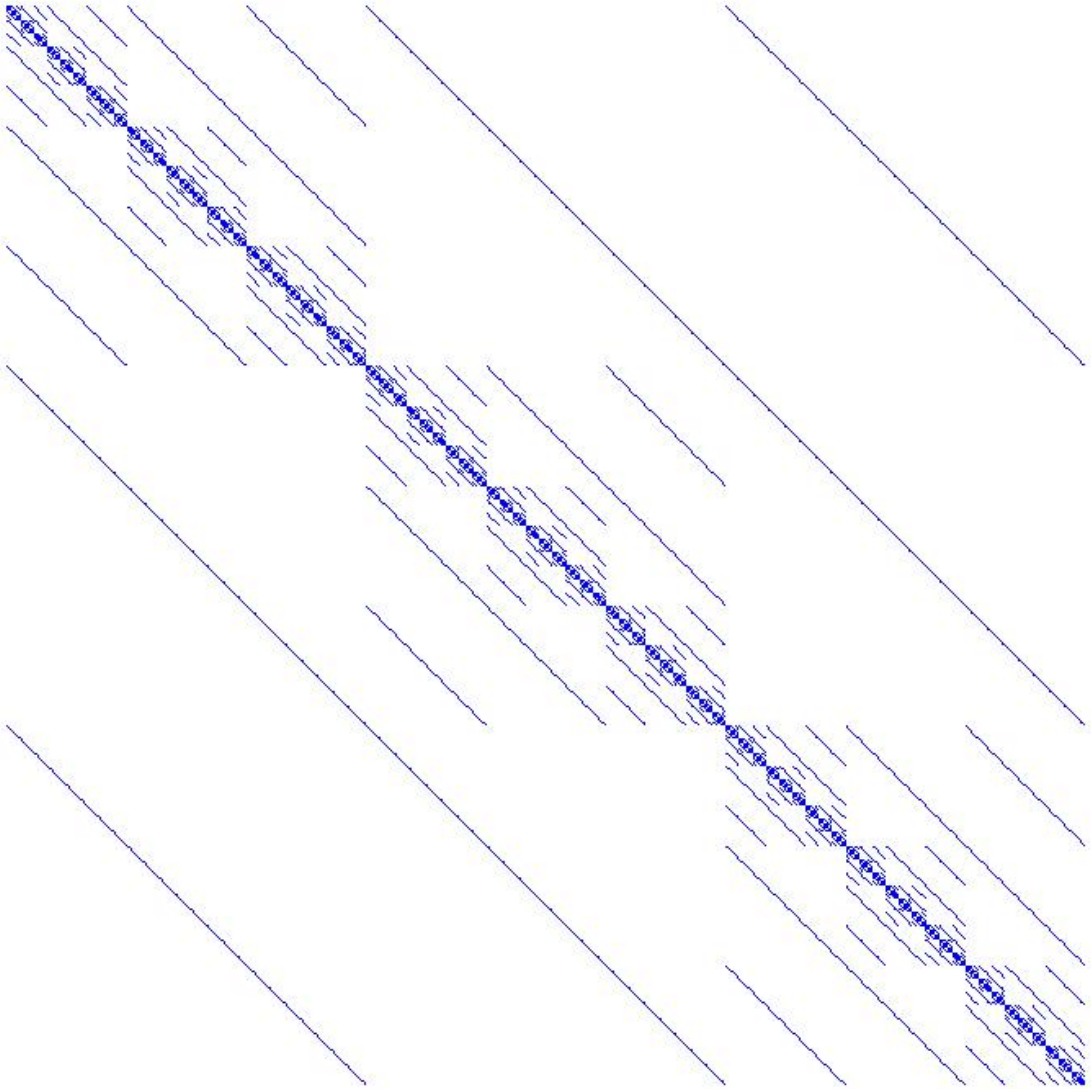
Football Matrix, $\nu = 3$

Playing Football

Chose columns to cover all rows



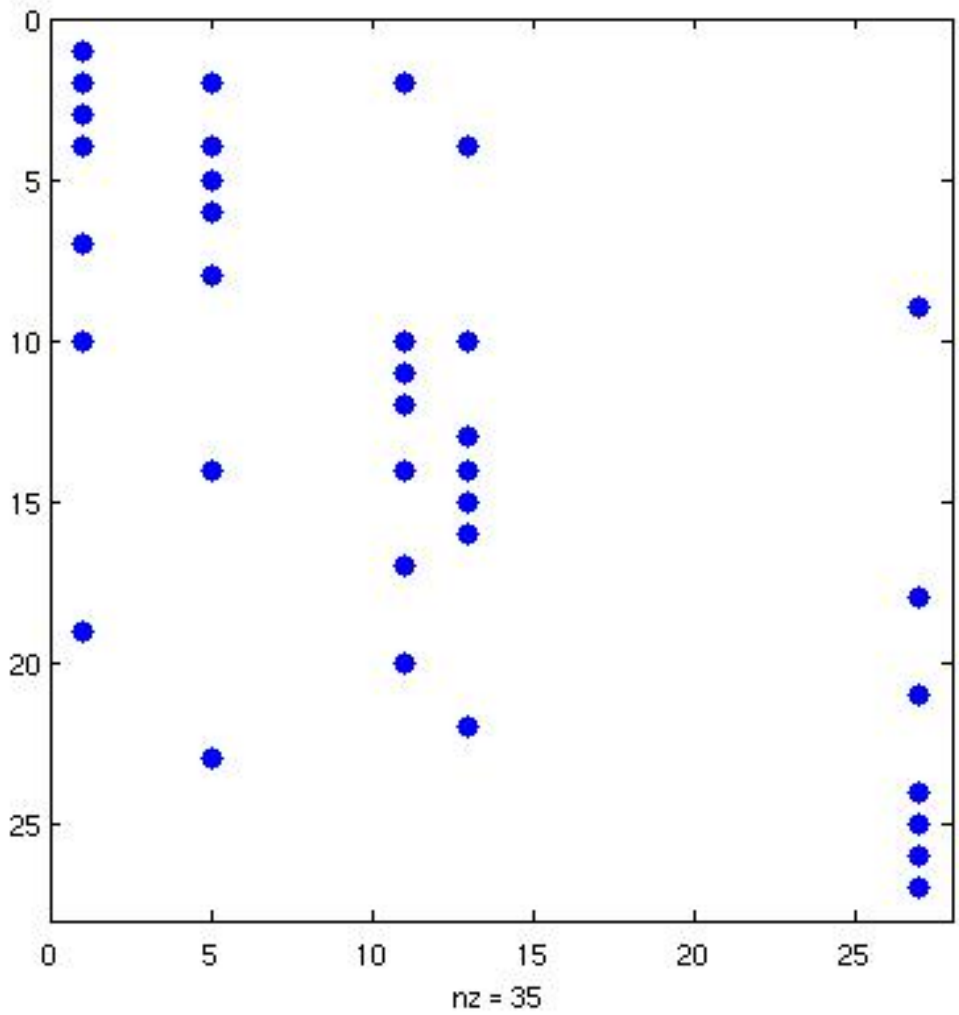
Football Matrix, $v = 6$



$v = 3$, Solution #1

Answer #1

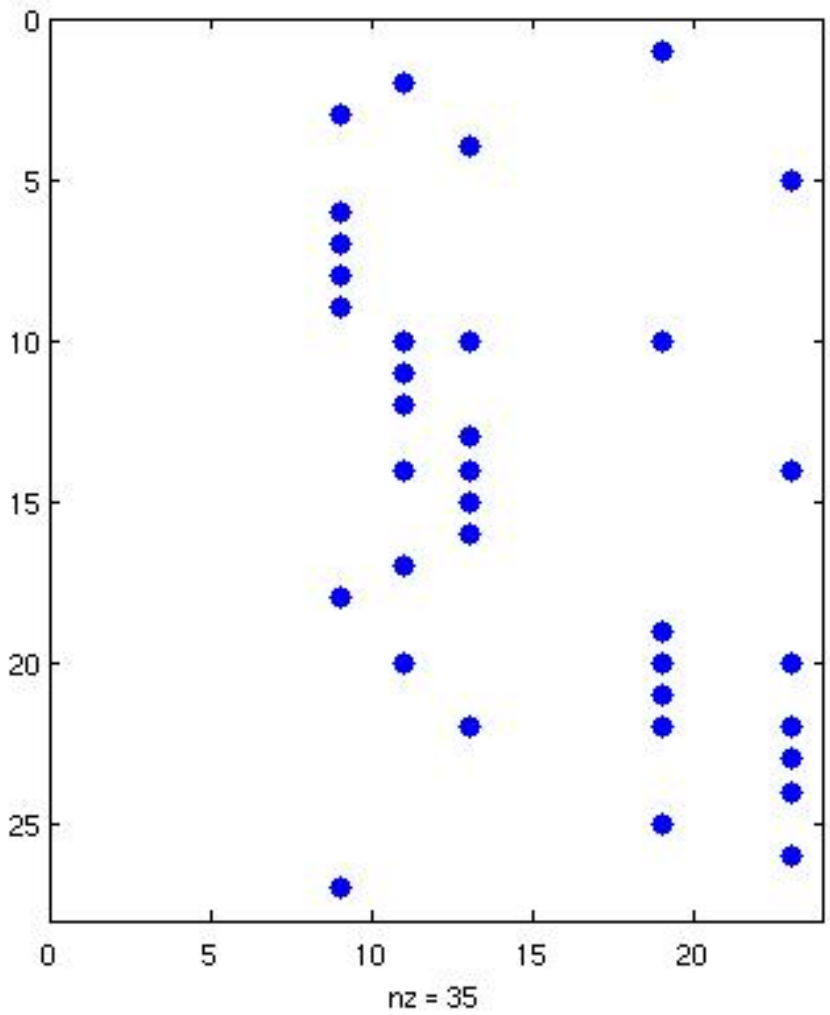
M1	M2	M3
W	W	W
L	L	W
L	W	L
W	L	L
D	D	D



$v = 3$, Solution #2

Answer #2

M1	M2	M3
D	W	W
L	L	W
L	W	L
D	L	L
W	D	D



Solutions for $\nu = 3$

Answer #1

M1	M2	M3
W	W	W
L	L	W
L	W	L
W	L	L
D	D	D

Answer #2

M1	M2	M3
D	W	W
L	L	W
L	W	L
D	L	L
W	D	D

- These solutions are isomorphic.
  - Swap W $\leftrightarrow$ D in the first match
- There are LOTS of isomorphic solutions:
  1. "Rename" W,L,D for any subset of the matches: $(3!)^\nu$
  2. Reorder the matches: $\nu!$
- There are $(3!)^3(3!) = 1296$ equivalent solutions for $\nu = 3$
- There are $(3!)^6(6!) = 33,592,320$ equivalent solutions for $\nu = 6$

How Many Must I Buy?

Known Optimal Values

v	1	2	3	4	5
$\lvert C_v^* \rvert$	1	3	5	9	27

The Football Pool Problem

What is $|C_6^*|$?

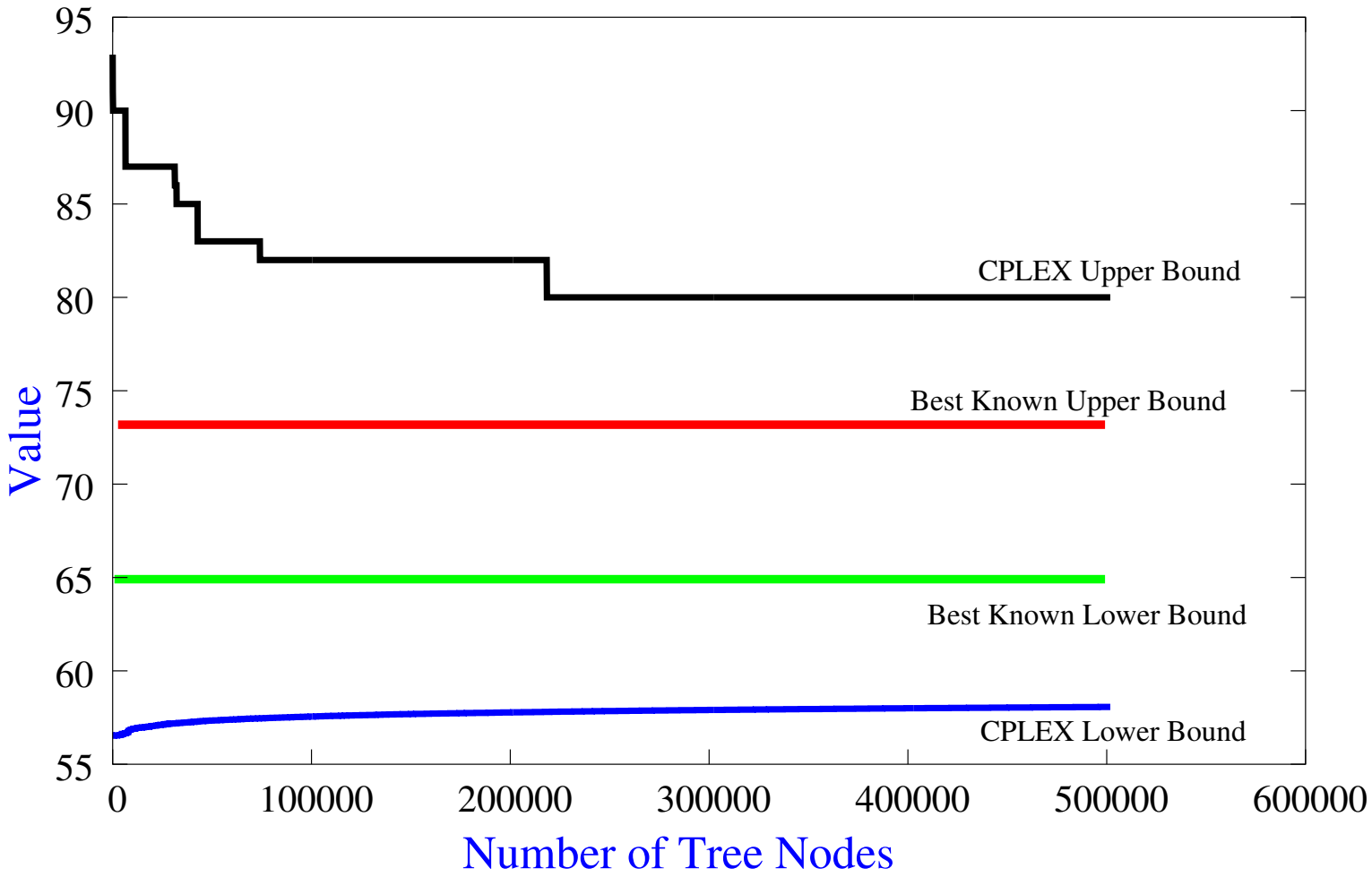
- Despite significant effort on this problem for > 40 years, it is only known that

$$65\textcolor{red}{7}1 \leq C_6^* \leq 73$$

- Now we know this!

CPLEX Can't Solve Every IP

- Roughly 10^8 universe lifetimes in order to establish that $|C_6^*| > 72$



Football Fail!

Methods for Symmetric IPs

- Margot—Isomorphism Pruning
- Ostrowski, Linderoth, Rossi, Smriglio—Orbital Branching

- These methods (by themself) fail to make progress on the football pool problem

Key Idea!

- Enumerate "necessary conditions" for there to exist an optimal solution (code) of value/cardinality M
- If for each "necessary" condition, no such code of value M exists...
- The smallest code must be of cardinality at least $M + 1$

Results of Preprocessing/Enuemration

- Checking if a code of size M exists for each "necessary condition" is a symmetric integer program that typically can be solved by isomorphism pruning
- In the end, by enumerating, preprocessing we are left with the following amount of work to do:

M	#seq	Prep. #seq
65	0	0
66	797	7
67	1,723	13
68	3,640	45
69	7,527	102
70	13,600	176
71	24,023	264
72	40,431	393
		1000

- Solving $M = 66, 67, 68$ IPs takes less than a week on a single CPU with isomorphism pruning.
- Other instances are (quite) difficult $\Rightarrow$—Use the Cloud!

$|C_6^*| \geq X???$

- François and Jeff implemented these ideas using the old MW-FATCOP framework for MILP.

Algorithm Engineering Is Important!

- Task: A time-limited subtree (1800 seconds)
- Workers search subtrees in depth-first fashion
- Send easy/hard tasks to workers depending on # of active nodes at master
- Adjust task time limit when # active nodes $<$ # available workers

Woo Hoo!!!!!

Breaking News

Deal colleagues, we are pleased to annouce that we have improved the lower bound on the cardinality of a covering code of radius one for the Hamming Space $\mathbb{Q} = \mathrm{F}_3^6$ to 70.

Or Have We!?!?!?!

Stardate: Tue, 11 Apr 2006 12:41:29 -0400 (EDT)

From: Francois Margot <fmargot@andrew.cmu.edu>
To: Jeff Linderoth <jtl3@Lehigh.EDU>
cc: gthain@cs.wisc.edu
Subject: Re: BooYah!

Now, remove all sharp objects from your desk. I noticed a mistake in the codbt06_fix2_reg01236_s.mylp.gz file (a permutation of the inequalities). I fixed the file, but I am afraid that we will have to rerun the *69s_*.seq problems. However, I can reduced the number of sequences in the files, taking into account the fact that every 5-subcode must have at least 22 words. I will let you steam a little bit before doing so ...

Doh!


```
jeff@pbr:  diff correct.lp incorrect.lp
1c1
< 738 729 1
---
> 739 729 1
```

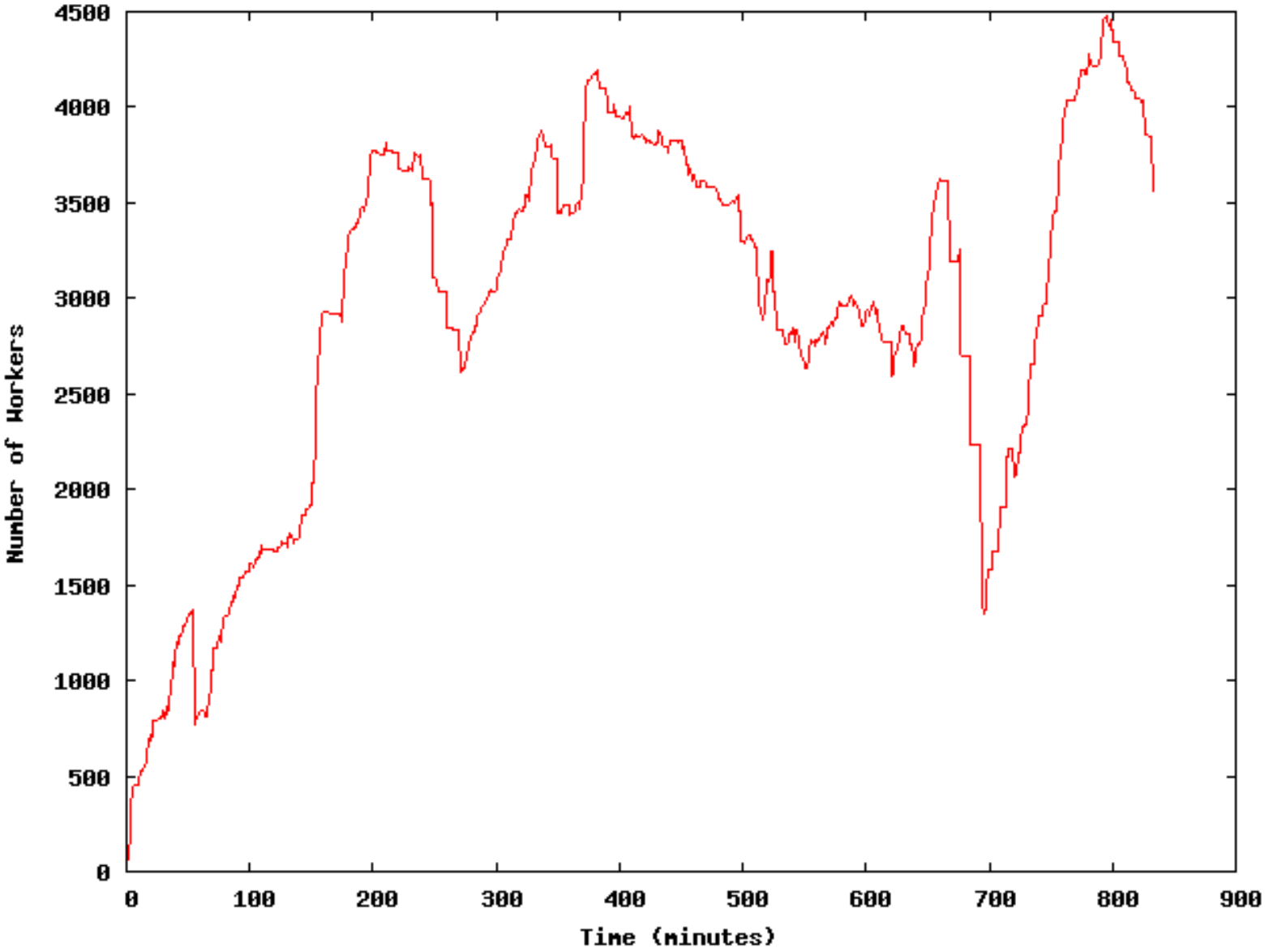
"The Cloud" Allows You To Make Mistakes

- We (read François) wasted "only" 19.76 CPU Years.
- Since we are flexible in our use of cycles, we can scavange the ones that would have otherwise been wasted

If At First You Don't Succeed...

	$M = 69$	$M = 70$
Avg. Workers	555.8	562.4
Max Workers	2038	1775
Worker Time (years)	110.1	30.3
Wall Time (days)	72.3	19.7
Worker Util.	90%	71%
Nodes	2.85×10^9	1.89×10^8
LP Pivots	2.65×10^{12}	1.82×10^{11}

Simultaneous Workers (Trying for $M = 71$)



Who's the King?

- Computations (if you believe them) establish that $|C_6^*| \geq 71$
- In all we used more than 200 CPU years!

- I should be at the top of the pyramid—Of Optimizers who have wasted the most CPU time
- The solution of TSPlib instance `pla85900` took "only" around 136 CPU years.

People Often Ask Me

- Why did I stop doing these large-scale computations?

Global Warming Is All My Fault

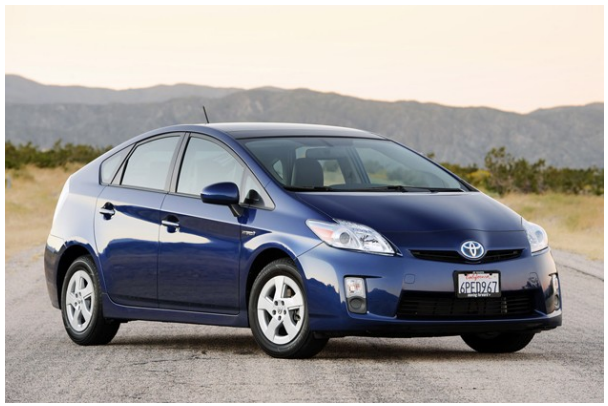
- 200 CPU Years = 1.752M CPU Hours
- CPU uses $\approx$ 500 W per hour
- $\Rightarrow$ 876 MWH for the calculation.
- This is 1.1388 million pounds (569 tons) of CO2

Air Travel

- "Standard" carbon charge of 2062 miles/ton of CO2
- $\Rightarrow$ 1.17 million flight miles

Car Travel

- Prius produces around one ton of CO2 for 5988 miles
- $\Rightarrow$ I could dive my Prius about 3.4 million miles.



Conclusions

- Jeff has contributed significantly to the global warming problem

Optimizers Should Use The Cloud

- Faster and more robust testing of algorithmic ideas
  - (Hopefully leads to Better algorithms)
- As a mechanism for bringing optimization tools to a wider range of people. Provide the software as a service
- To tackle larger problems than we could before.
  - (As long as you don't mind heating up a few machine rooms)

Cloud Computing for Optimization

- We all need to be adapting to the way in which computing resources will be provided
- We are missing an opportunity by not exploiting resources provided on the cloud