Building the International Data Placement Lab

Greg Thain
Center for High Throughput Computing

Overview

What is the IDPL?

How we built it

Examples of use

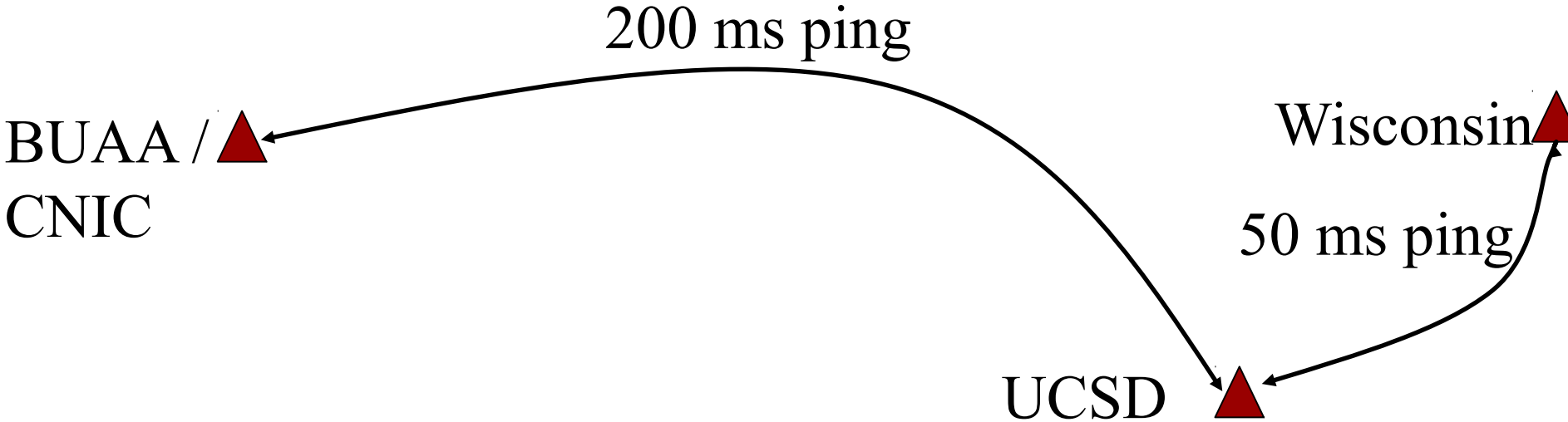
Who is the IDPL?

Phil Papadapolus -- UCSD
Miron Livny -- Wisconsin

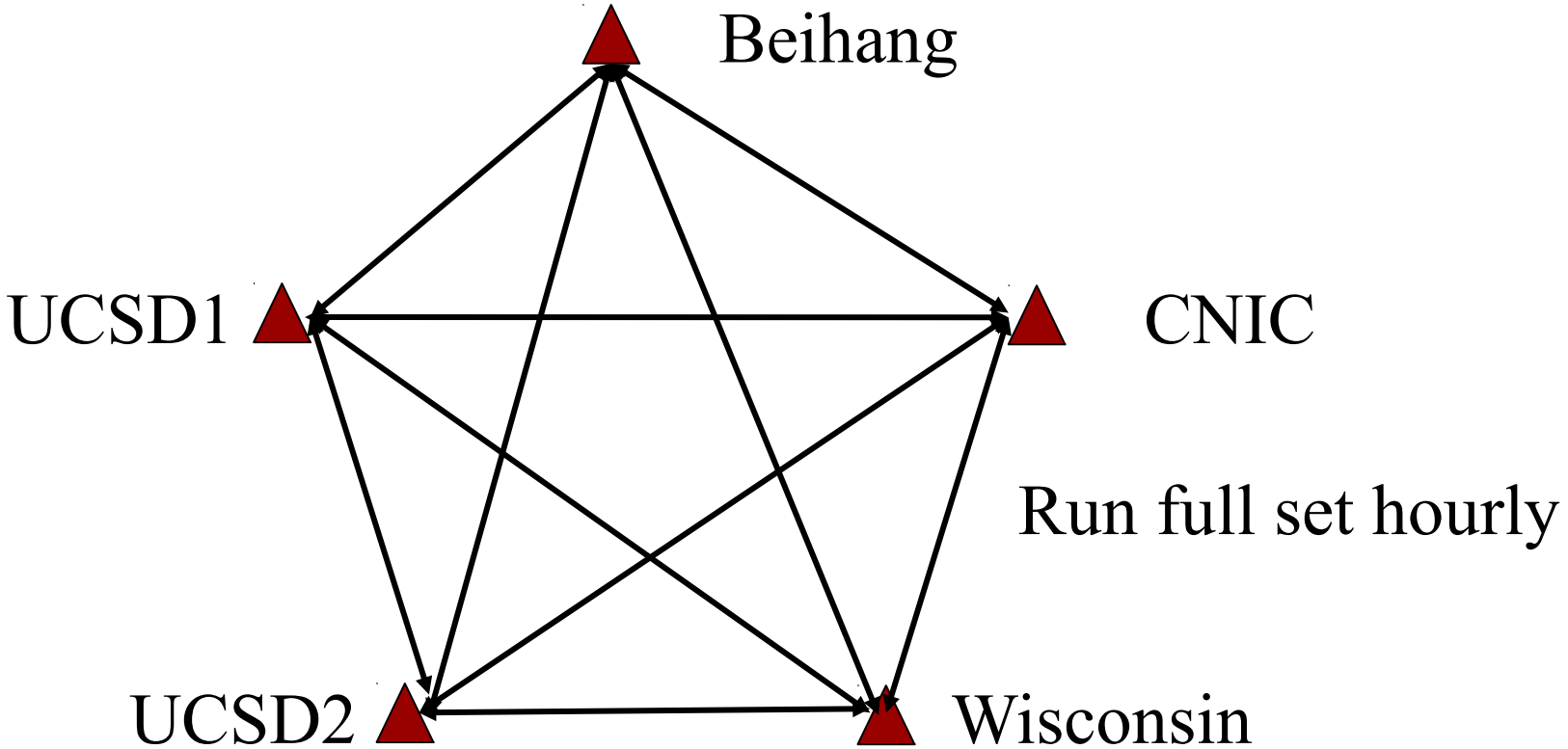
Collaborators in
China:Beihang // CNIC


Beihang
Wisc
CNIC
UCSD (2x)

A network engineer worldview:



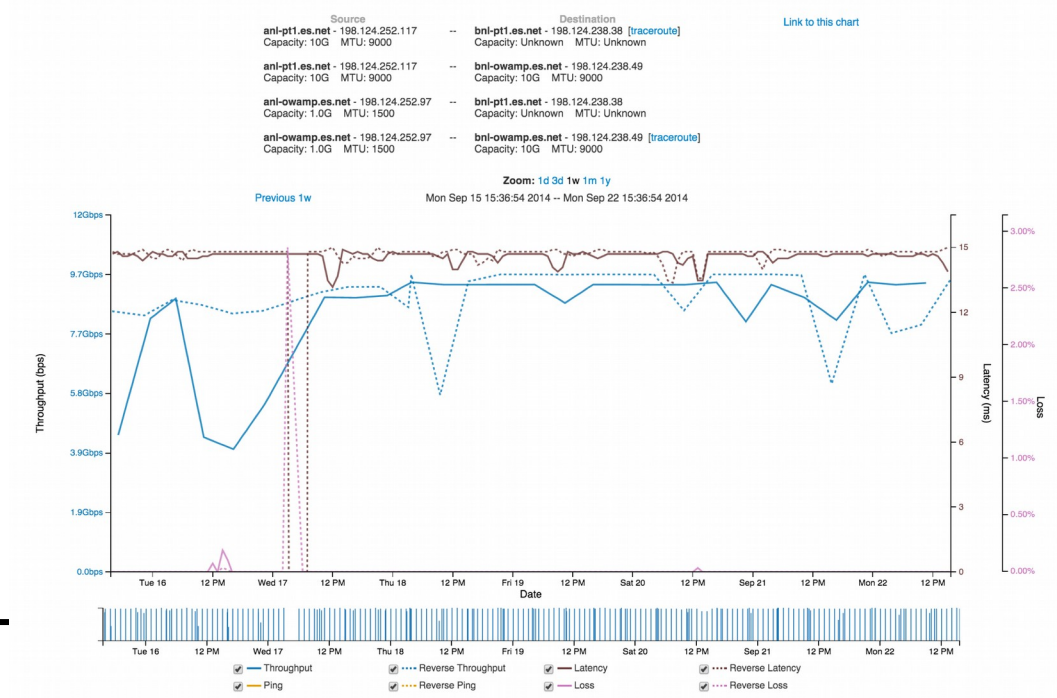
A Topologist's World View



Wait? What about PerfSonar?

perfSONAR

www.perfsonar.net



CENTER FOR HIGH THROUGHPUT COMPUTING


NO PACKET LOSS AT EDGE ROUTER
THERE'S NO PROBLEM HERE
memegenerator.net

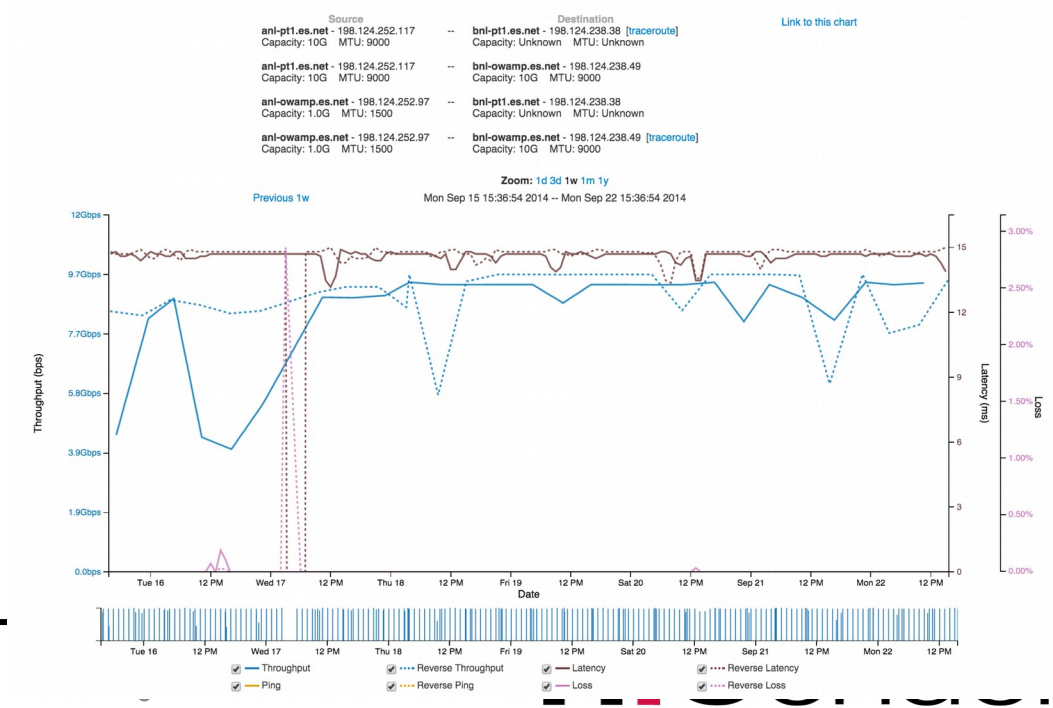
Wait? What about PerfSonar?

perfSONAR

www.perfsonar.net

Necessary, not sufficient

Network only



Requirements for data placement

1. Co-scheduling of jobs

   Start client and server simultaneously

2. Timed start of jobs

   To prevent overlap and race conditions

3. Returning and managing of results

Co-Scheduling with Parallel Universe

- Using static slots
- One slot per host
- Submit two-host job
  - Proc 0 is client
  - Proc 1 is server
    - (just a convention)

Example Submit file

```
Universe = parallel
Executable = startup_script.sh
+ParallelShutdownPolicy = “WAIT_FOR_ALL”
<usual file transfer stuff>
Requirements = (machine == “client-machine-name”)
Queue
Requirements = (machine == “server-machine-name”)
Queue
```

Startup Script (mark 1)

```
#!/bin/sh
if [ $_CONDOR_PROCNO = 0 ]
then
   do_client_stuff
else
   do_server_stuff
fi
```

But this doesn't work…

- Synchronization problem
  - Not actually co-scheduled
  - Servers need to tell clients their port numbers

Unix Process :: Condor Job

Unix Process	Condor Job
fork/exec	condor_submit
kill -9	condor_rm
Environment variables	Job Attributes
Standard error file	Condor job log

condor_chirp to the rescue!

- condor_chirp set_job_attr
- Uses job ad as a ~~blackboard~~ whiteboard
- Always read/writes to Proc 0
  (in parallel universe)

Simplified Workflow

Server Side

```
Start Server
Listen on ephemeral port
condor_chirp set_job_attr Port #
```

Client Side

```
condor_chirp get_job_attr Port #
repeat as needed
run test
```

That works – once

- Need to do this periodically
- Condor cron – two features

```
on_exit_remove = false
cron_minute = 23
cron_hour = 0-23/2
cron_month = *
cron_day_of_week = *
```

Problems with condor cron

`on_exit_remove = false`

- Means one job for all runs
- stale set_job_attr's
- held jobs a headache

Really want one condor job per run

Enter…

“CronMan”

The Idea

- Have dagman itself be the restarter
  - Creates a new job every time
- With a one-node dag
- Whose one node has a delayed start time
  - Gives the placement job the job-nature

The dag file

```
JOB A placement4-submit
SCRIPT POST A /bin/false
RETRY 1000000
```

Modified Submit file

```
Universe = parallel
Executable = startup_script.sh
+ParallelShutdownPolicy = "WAIT_FOR_ALL"

cron_minute = 30

cron_window = 400

Requirements = (machine == "client-machine-name")
Queue
Requirements = (machine == "server-machine-name")
Queue
```

condor_q output

```
$ condor_q
ID        OWNER      SUBMITTED     RUN_TIME ST PRI SIZE CMD
26795.0   gthain     1/20 23:50  11+12:26:06 R  0   0.3  condor_dagman
27720.0   gthain     5/11 10:22   0+00:00:00 I  0   0.0  wrapper_script
27720.1   gthain     5/11 10:22   0+00:00:00 I  0   0.0  wrapper_script
```

Two of three problems solved

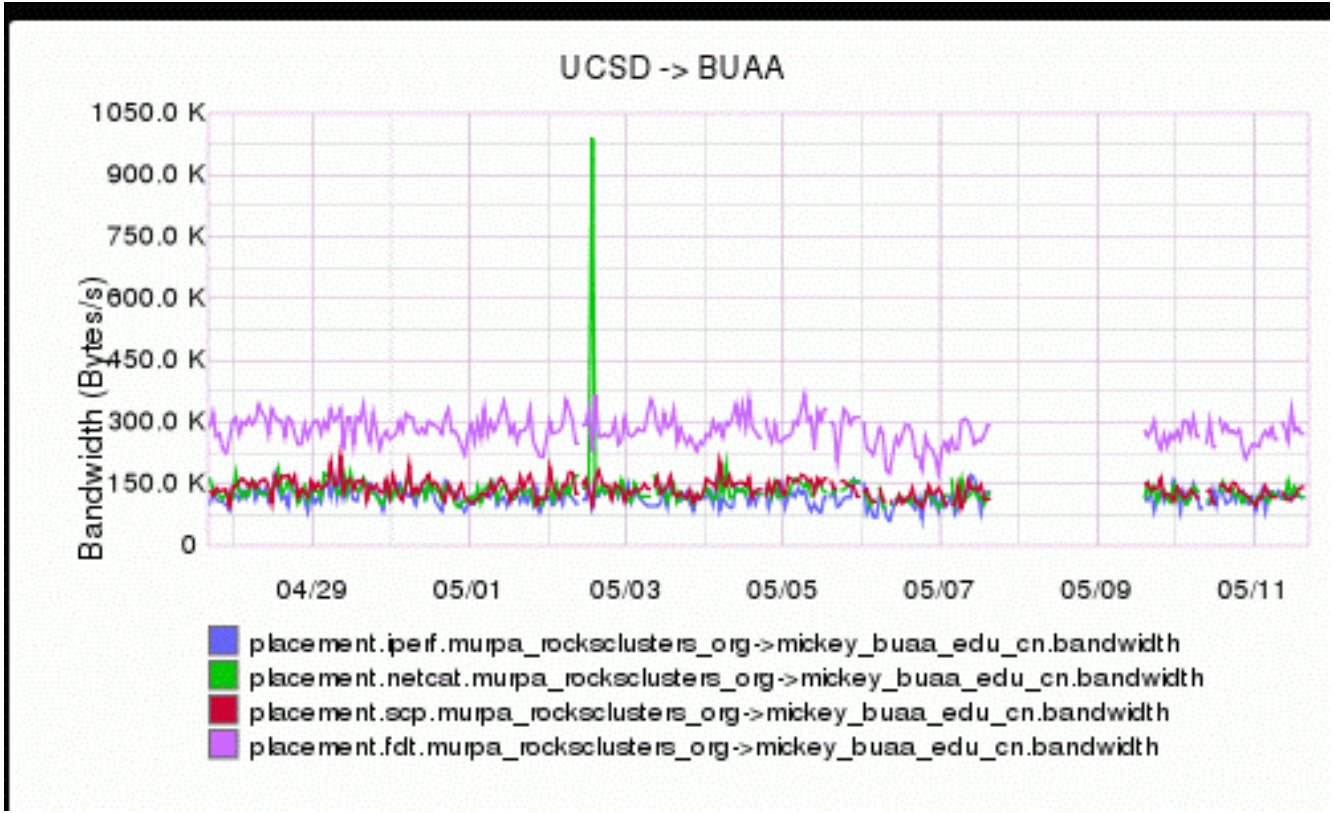
- What about reporting results?

- condor_chirp ulog “server start”
- condor_chirp ulog “client results x y z”

Job log output

```
000 (27720.000.000) 05/11 10:22:48 Job submitted from host:    DAG Node: A
014 (27720.000.000) 05/11 11:17:00 Node 0 executing on host: client
008 (27720.000.000) 05/11 11:17:00 'client:start'
014 (27720.000.001) 05/11 11:17:04 Node 1 executing on host:
001 (27720.000.000) 05/11 11:17:04 Job executing on host: MPI_job
008 (27720.000.000) 05/11 11:17:06 'server:start'
033 (27720.000.000) 05/11 11:17:08 Setting job attribute iperfServer to '20650'
008 (27720.000.000) 05/11 11:17:31
'results,1462979830.739976,1462979851.114580,1,20.100000,5736960'
008 (27720.000.000) 05/11 11:17:31 'komatsu.chtc.wisc.edu(iperf) client:end'
008 (27720.000.000) 05/11 11:17:31 'komatsu.chtc.wisc.edu(irods) client:start'
008 (27720.000.000) 05/11 11:17:31 'davos.cyverse.org(iperf) server:end'
```

Then forward to Graphite



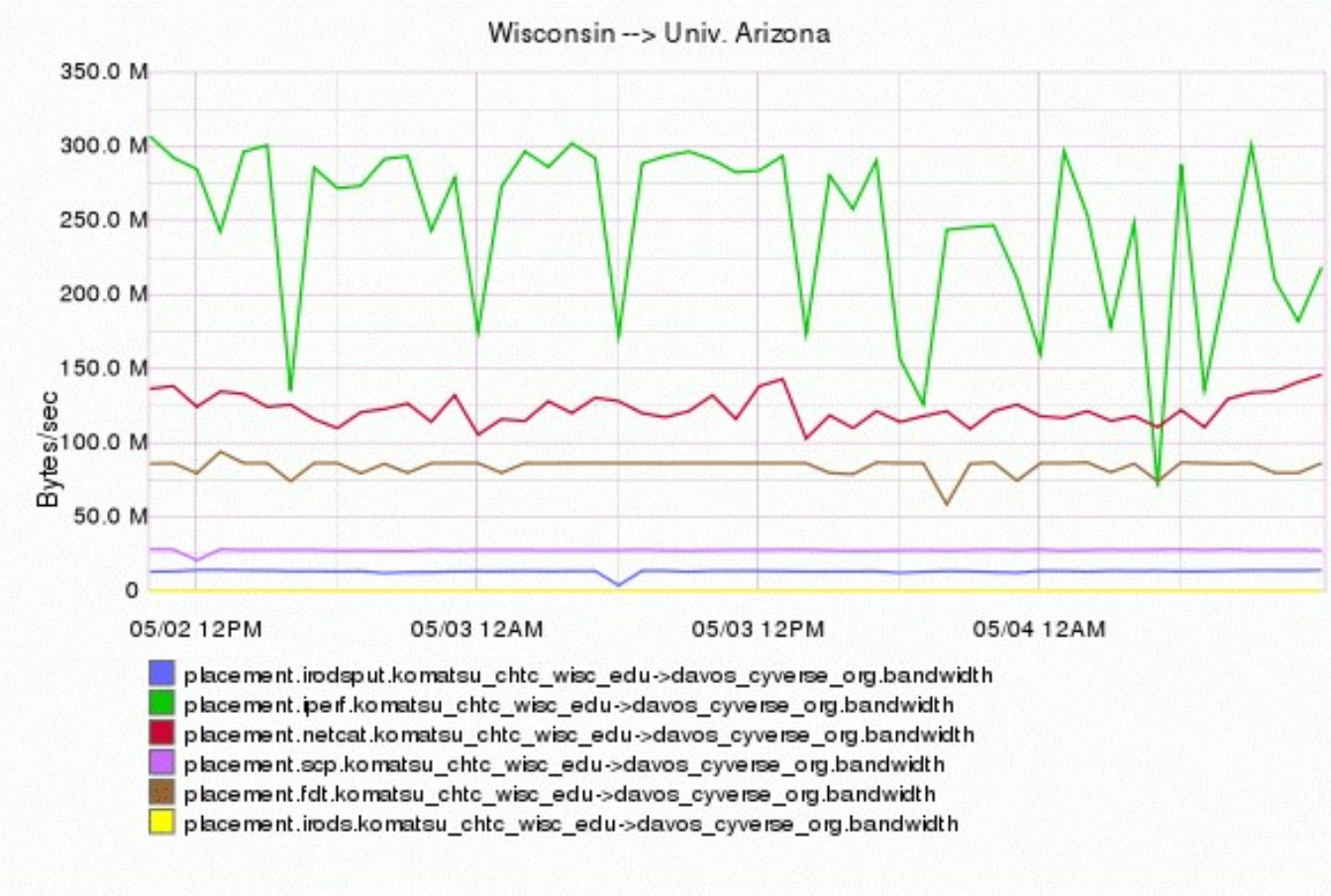
Application of IDPL: The Phytomorph problem

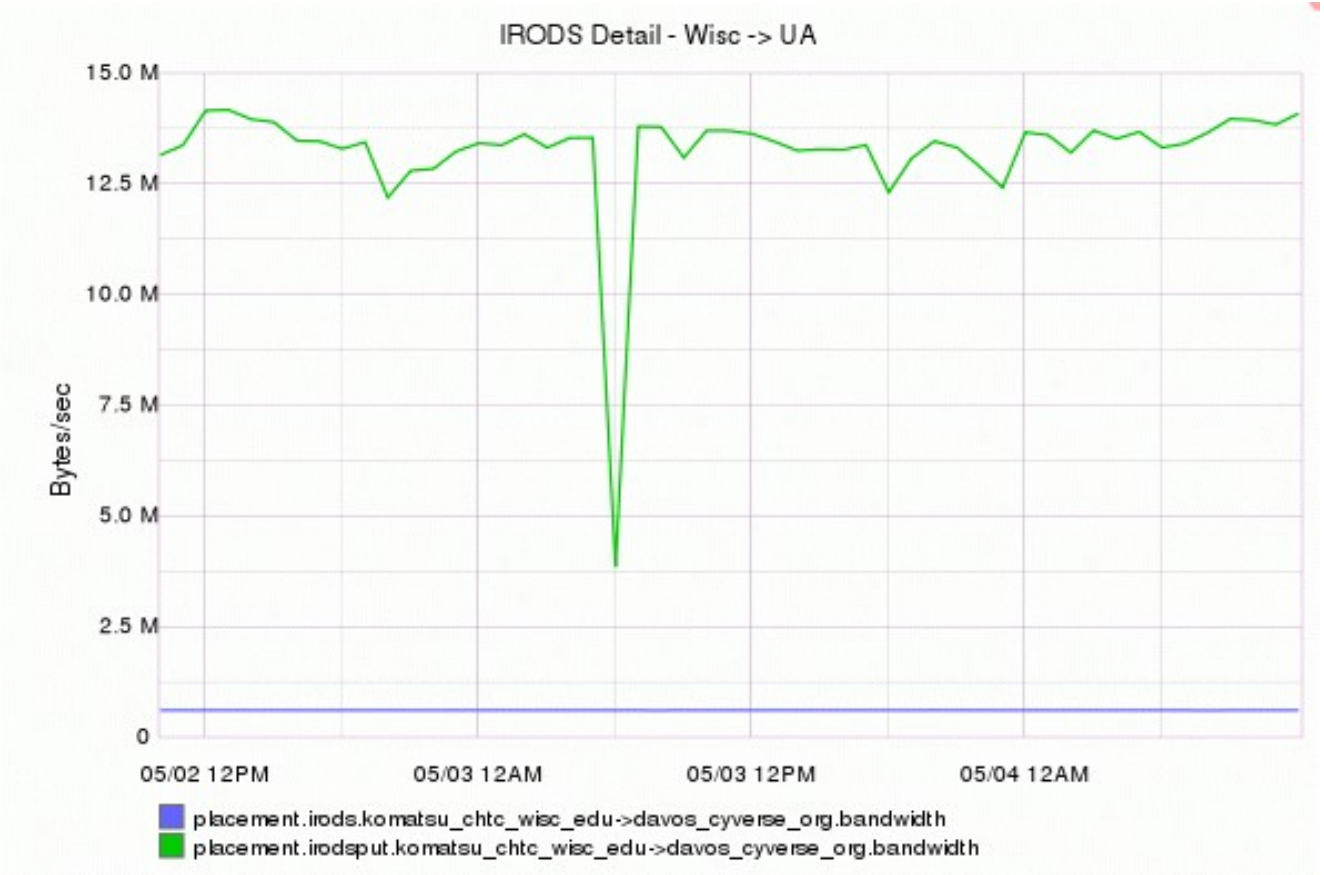
PI in the Spalding lab researching corn seedlings

Runs analysis jobs at Wisconsin HTCondor pool

Pulling data from Cyverse (nee iPlant) in Arizona

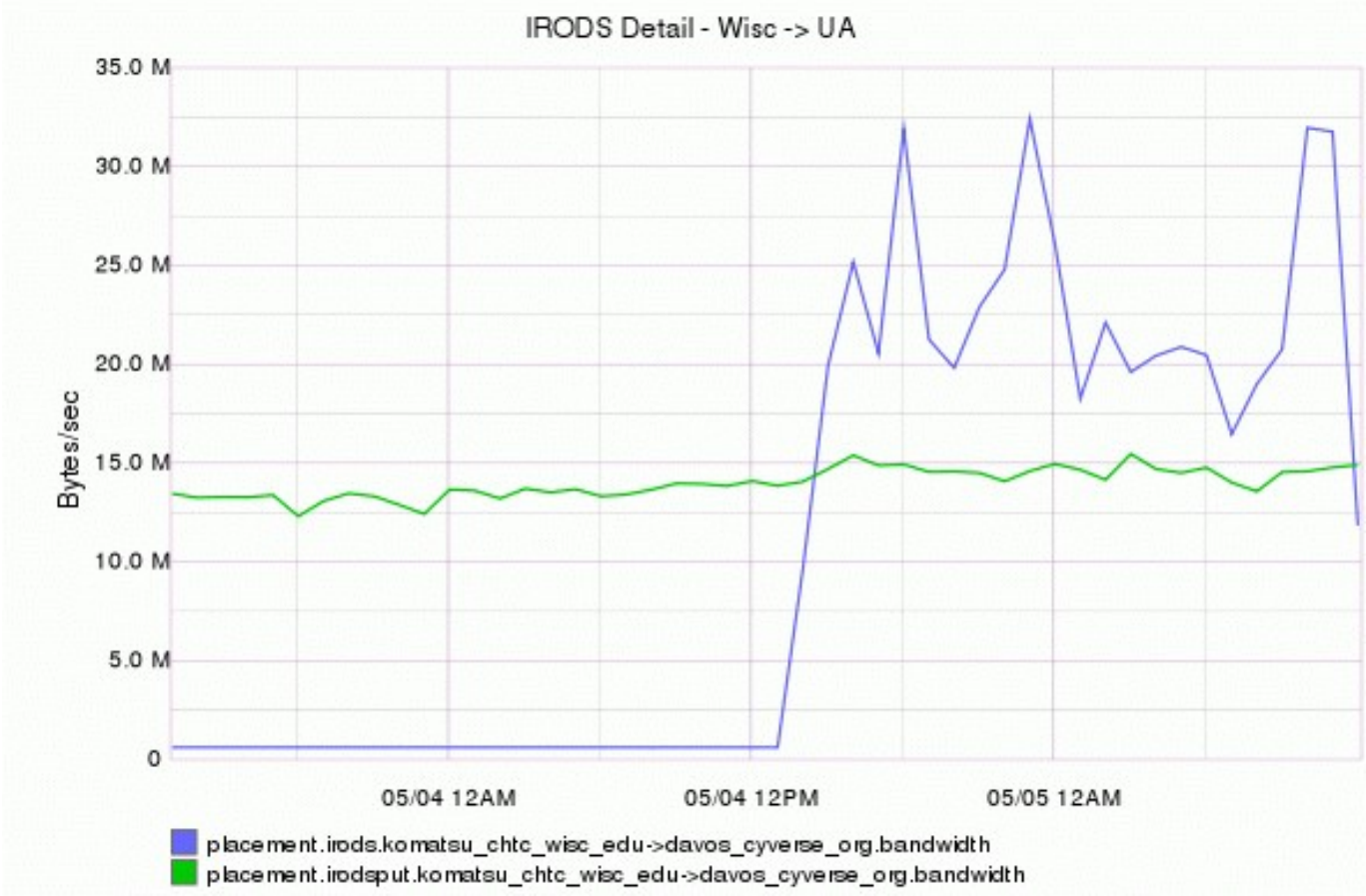
Claims data xfer rates slow


Wisconsin --> Univ. Arizona
350.0 M
300.0 M
250.0 M
200.0 M
150.0 M
100.0 M
50.0 M
0
Bytes/sec
05/02 12PM
05/03 12AM
05/03 12PM
05/04 12AM
placement.irodsput.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.iperf.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.netcat.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.scp.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.fdt.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.irods.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth


IRODS Detail - Wisc -> UA
15.0 M
12.5 M
10.0 M
7.5 M
5.0 M
2.5 M
0
Bytes/sec
05/02 12PM
05/03 12AM
05/03 12PM
05/04 12AM
placement.irods.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.irodsput.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth

After talking with iRODS devs

- All performance testing had been LAN
- Our Networking folks ID'd problems
- Made one big fix
  - Don't set TCP_SNDBUF explicitly!
- Gave us new clients & servers


IRODS Detail - Wisc -> UA
35.0 M
30.0 M
25.0 M
20.0 M
15.0 M
10.0 M
5.0 M
0
Bytes/sec
05/04 12AM
05/04 12PM
05/05 12AM
placement.irods.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth
placement.irodsput.komatsu_chtc_wisc_edu->davos_cyverse_org.bandwidth

Future Work

Add more protocols

More sites

Work with clouds

Thank you!

- Try CronMan pattern yourself!
  - Even with vanilla universe jobs
- Think about Unix process patterns
- Example on slides simplified!
- Real code on git hub at
  - https://github.com/iDPL/placement