Disk-to-Disk and Day-to-Day Placement Performance Metrics on a Trans-Pacific HTCondor Infrastructure

Philip Papadopoulos, Ph.D
University of California, San Diego
Greg Thain
HTCondor Maven
University of Wisconsin, Madison

NSF Award #ACI-1339508: EAGER: Fundamental Issues in International Data Placement for Data-Intensive Applications, a Laboratory Approach

UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Partners

- University of Wisconsin, Madison
  - Miron Livny
  - Greg Thain
- Beihang University, Beijing
  - Prof. Depei Qian
  - Dr. Hailong Yang
  - Guang Wei
  - Zhongzi Luan
- CNIC – Computer Network Information Center, Chinese Academy of Science, Beijing
  - Dr. Luo Ze
  - Dr. Gang Qin

UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Existential Problems

- We know the "speed of light" through the network via measurements tools/suites like perfSONAR
- Many researchers only really care about
  - Can I move my data reliably from point A to point B
  - Will it complete in a timely manner?
  - D2D: **Disk-to-disk** AND **Day-to-Day**.
- "Security" is more involved than memory-to-memory networking tests -- touching disks is inherently more invasive
- Is network measurement always a good proxy for disk-to-disk performance?

iDPL – Data Placement Lab

- Proof-of-principle project (EAGER, NSF ACI#1339508)
- Routinely measure end-to-end and disk-to-disk performance among a set of international endpoints
  - Compare performance of different data movement protocols
    - raw socket, scp, FDT, UDT, GridFTP, iRODs, …
  - Correlate to raw network performance
  - IPv4 and IPv6 whenever possible
- Re-use as much existing "command-and-control" infrastructure as possible
- Pay attention to some routine security concerns

HTCondor: Generic Execution Pool

"Where you care about every single host in your pool"

- Use HTCondor to assemble resources into a common execution environment
  - Must trust each other enough to support remote execution
  - NO common username is required
  - Pool password limits who can be added to global exec pool
- Current Configuration
  - Job submission at 4 sites
  - Host-based firewalls limits access

High Level Structure: Disk-to-Disk, Day-to-Day

Test Manifest

1. Network test (iperf)
2. Network test (iperfV6)
3. Move file via raw socket
4. Move file via FDTv6
5. Move file via UDT
6. Move file via GridFtp
7. Network test (iperf)

Submit Test as an HTCondor Job.

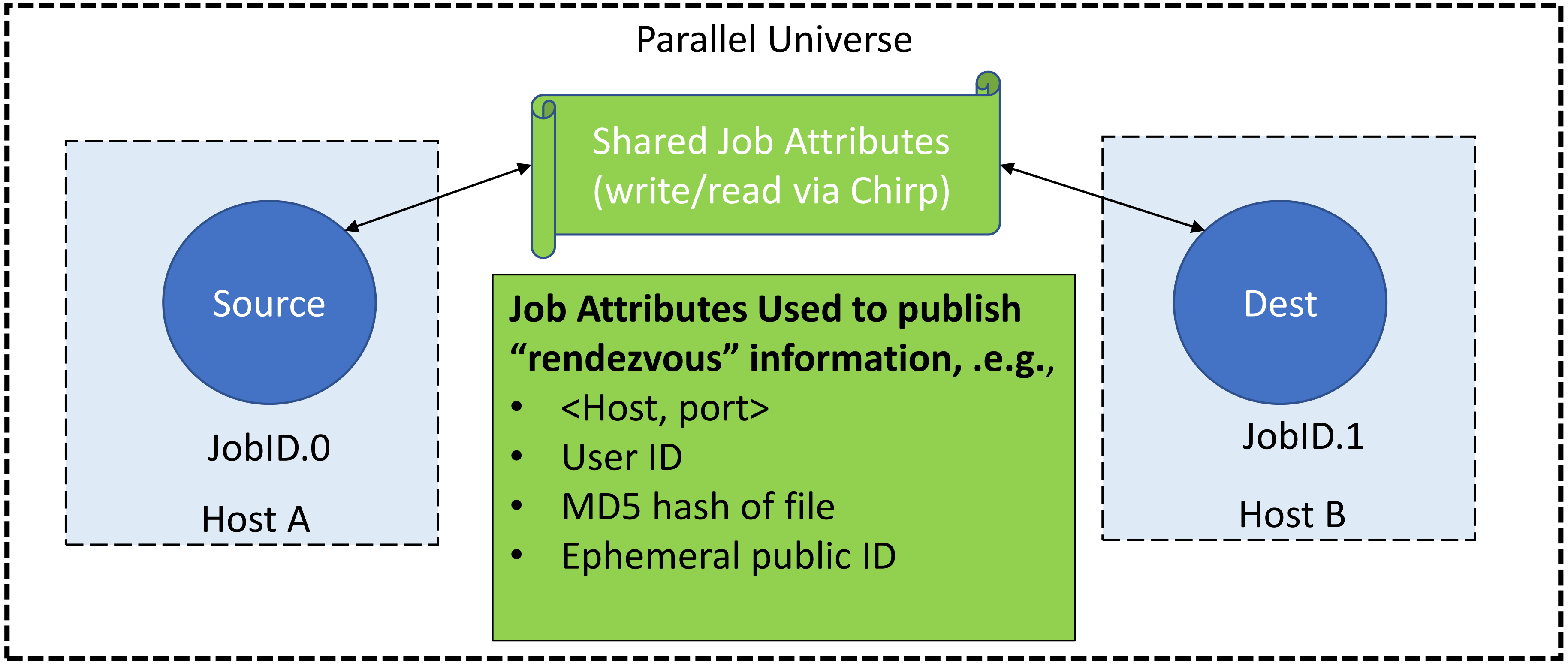
Let HTCondor handle errors, recovery, reporting, iteration

Repeat Test every N hours

- **Separate** concerns – Let Condor do what it does well.
  - Scheduling
  - Recovery
  - Output back to submitter
- Wisconsin → Beihang is a *different* experiment than Beihang → Wisconsin

UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Condor Parallel Universe



“Use FDT to place a file on Host B which is sourced on A”

- Server and Client Processes must be executed on two hosts at the same time

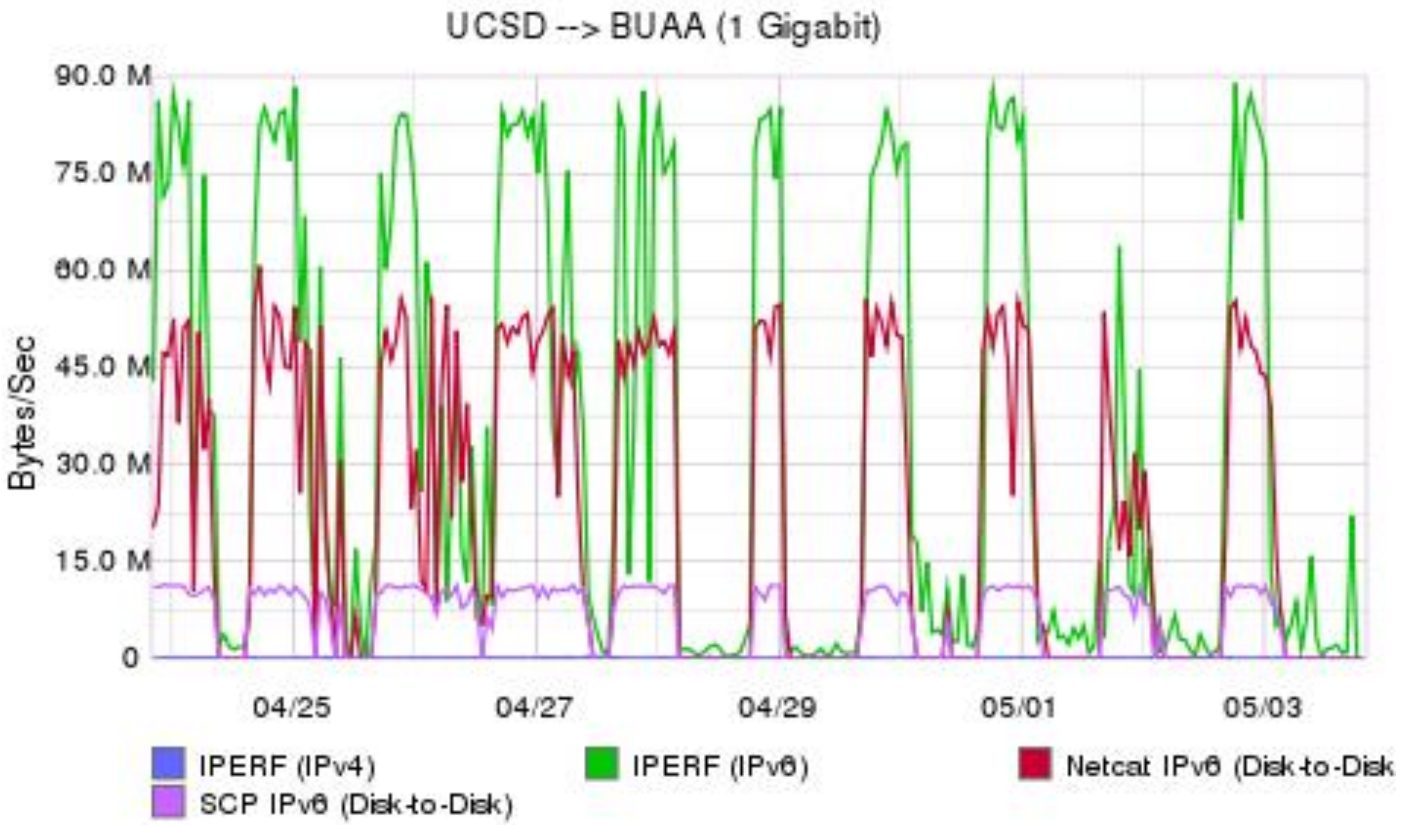
UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Things encountered trying to run day-to-day

- Public IP → Public IP mapping (e.g. 115.x.y.x → 210.p.q.r)
- IP address renumbering (4 different institutions: BUAA, CNIC, UWisc, UA)
  - Raw addresses live in a number of places: logs, iptables, condor config, etc.
- Stateless vs. Stateful firewall at University of Arizona
- perfSONAR installing its own set of FW rules, overwriting the system
- Defining a narrow range of ports (e.g. HTCondor Collector/Startd, 5000-5010 for ALL placement experiments)
- Remembering to be clueful sysadmins and installing v6 versions of everything.
- UCSD blackholed traffic to a particular host without notification to owner. Within UCSD was working, outside was not.
- Need to tune TCP params for 10G
  - OS updates wipe out tuned TCP parameters
- Git not performing automated garbage collection
- Chinese holidays shut down everything

UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

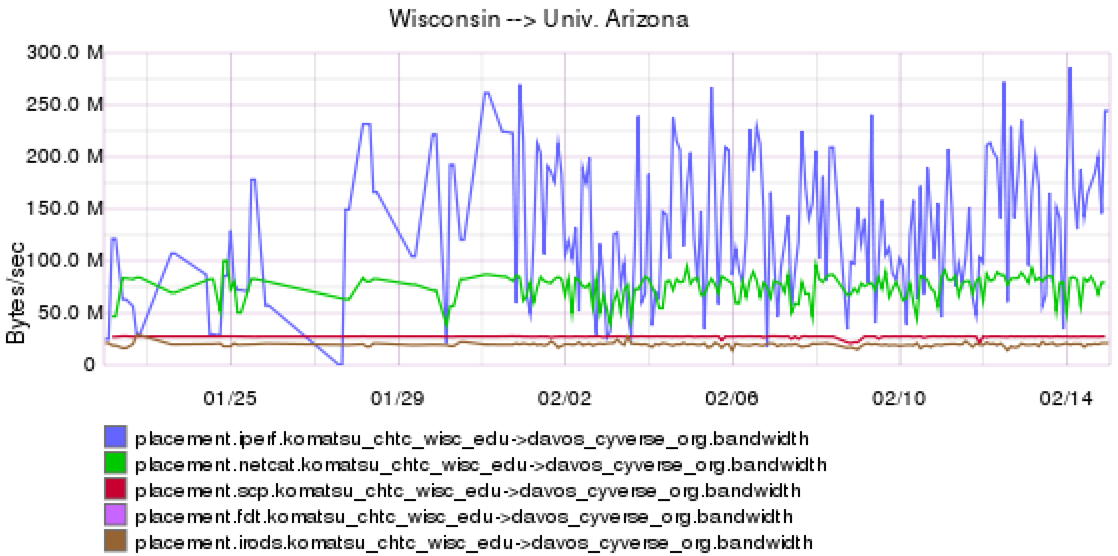
Sample Results: UCSD → BUAA (v4 & v6)

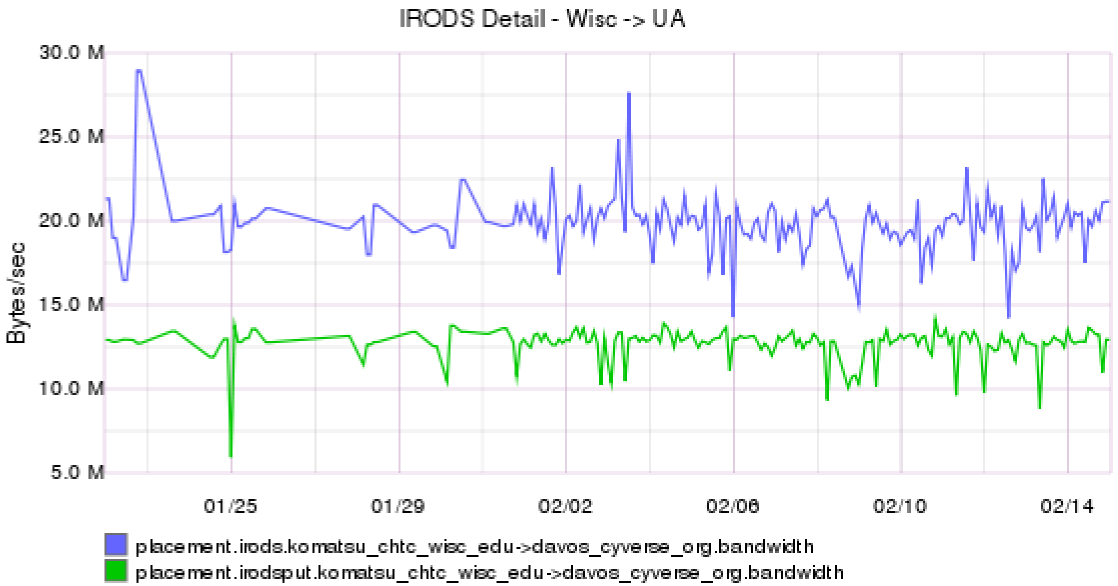


- **Highly variable raw network performance**
- **Netcat (Raw Socket) mirrors network – good correlation**
- **SCP, uniformly low**
- **IPv4 essentially flatlined (at the origin)**

iRODS Testing – 24 Day Trace Wisc -> UA

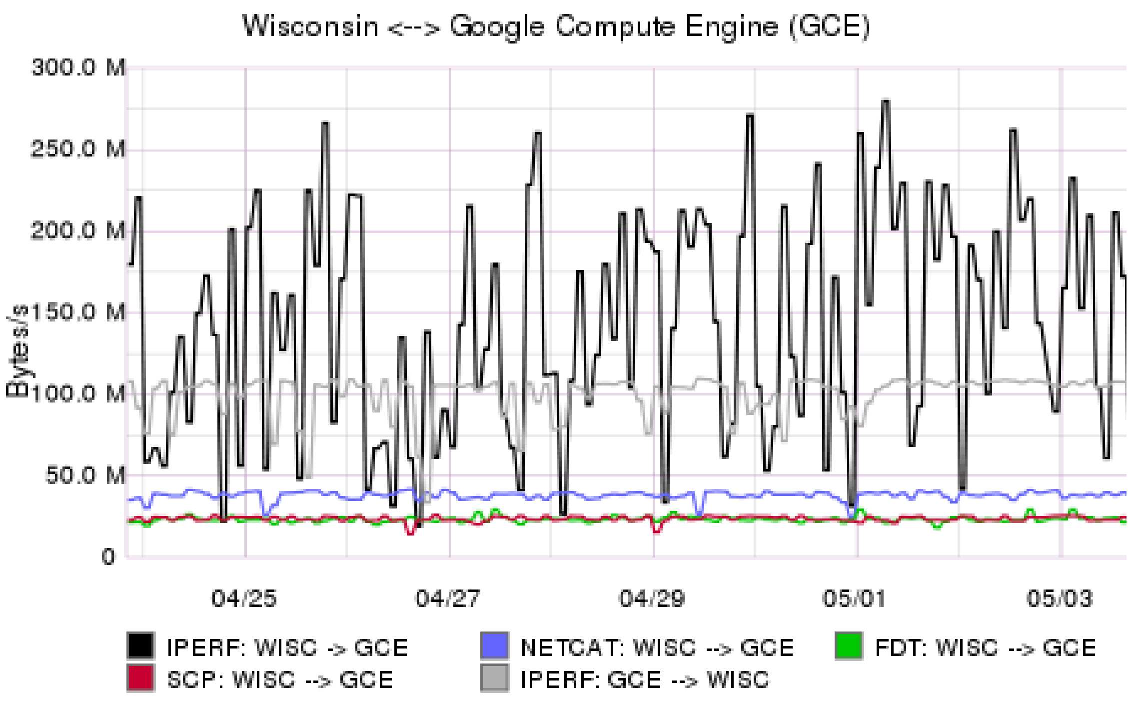
Y axis scale is 1/10th of left graph

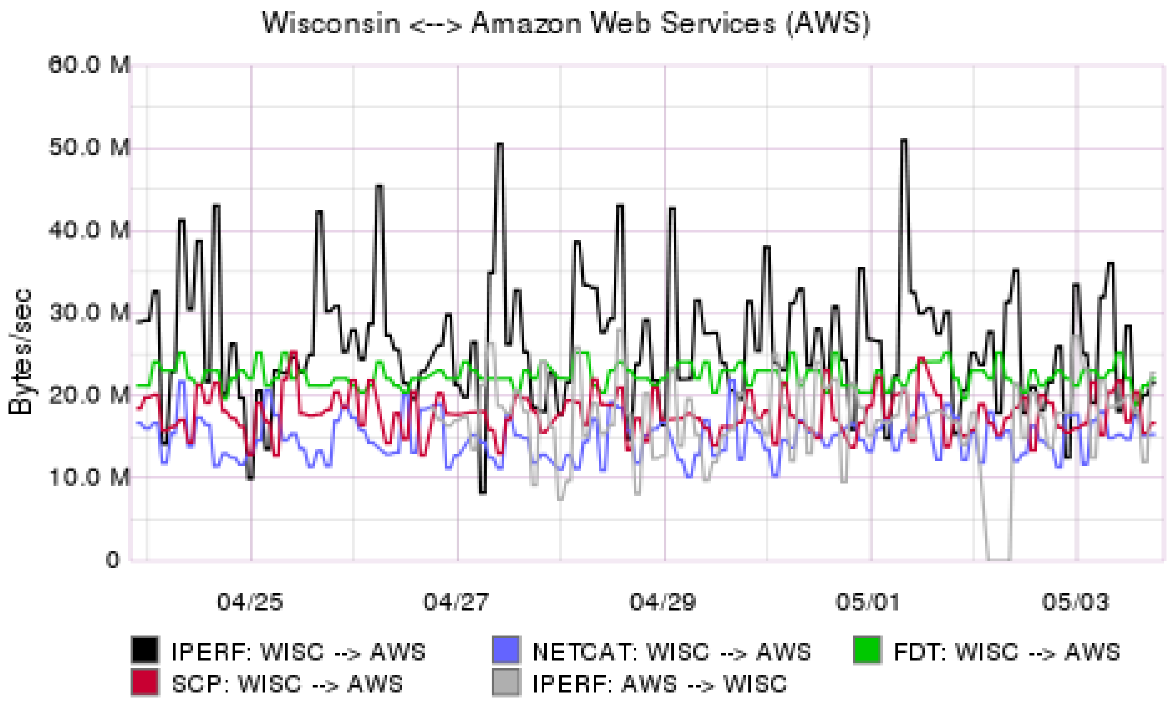




- **Highly variable raw network performance**
- **iRODS and iRODSPut 5-15 % of the raw network**
- **Raw socket ~ 90MB/sec 35-100% of network (Likely, this is disk performance limits)**

Testing performance to Cloud Instances



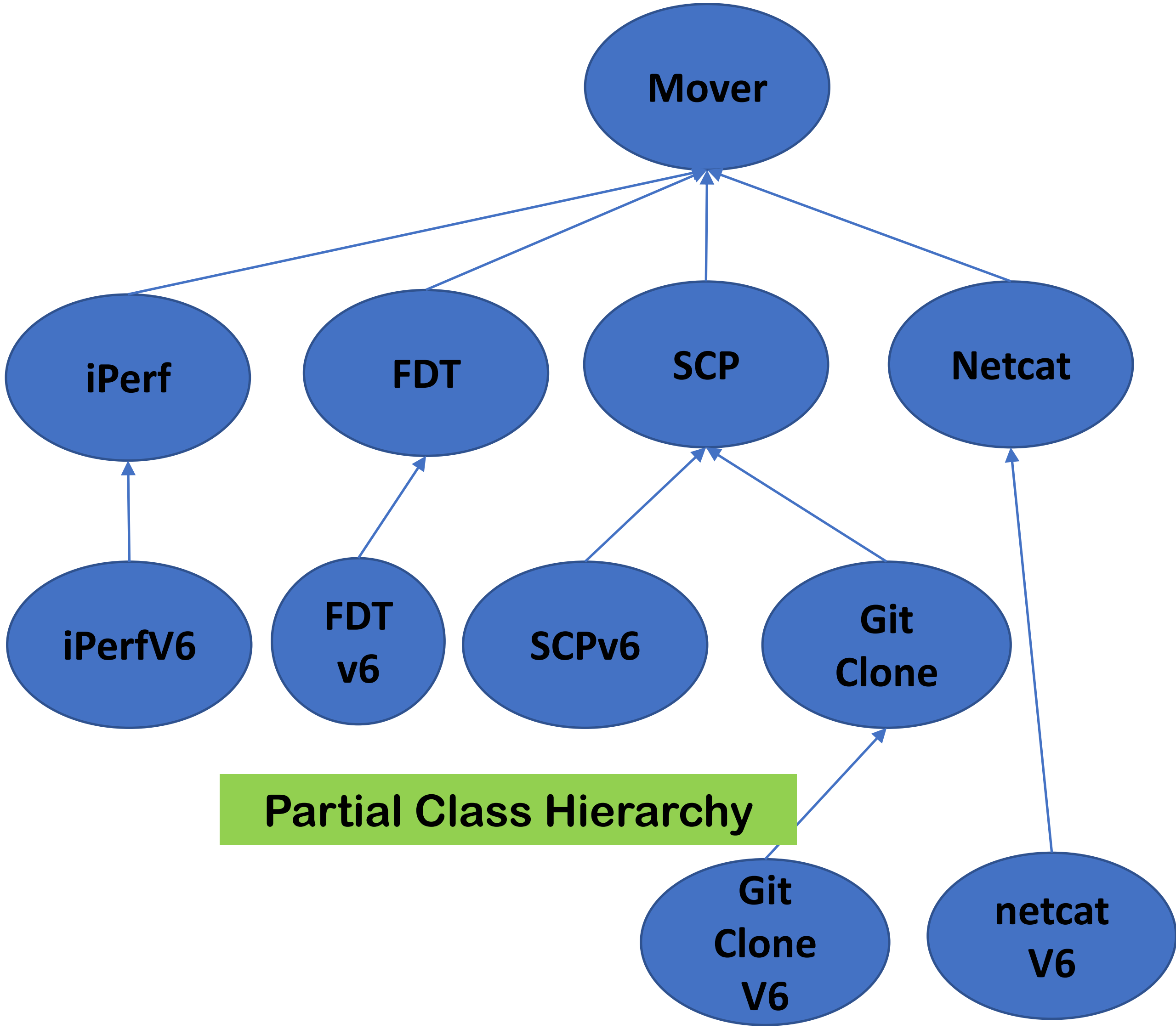


- **Both instances: "West" datacenters, 1-2 cores, On-Demand**
- **Significant network performance differences, Less so for disk-to-disk**

UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Custom software – Common Structure

- Observation: Different Placement Algorithms follow roughly the same pattern
  - Client (data flows from client), Server (data flows to server)
  - Setup exchange – via Chirp
    - Port utilized (e.g. iperf server is on port 5002)
    - Public key credentials (e.g. ssh daemon only supports connection with specific key)
  - Completion exchange
    - MD5 sum (or other hash) of sent file

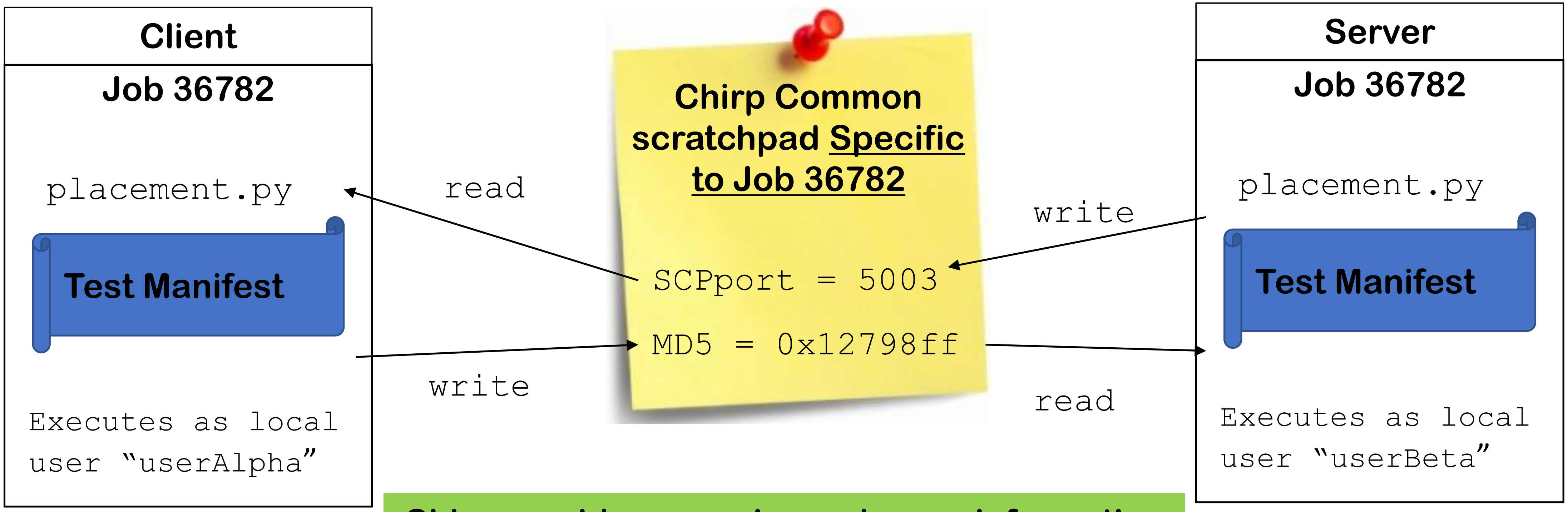


UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Custom Software – DataMover.py

- All movers are DataMovers (OO)
- All movers run in user space (including daemons like ssh,gridftp)
- Some only differ by v4 or v6
  - DataMover ← IperfMover ← IperfV6Mover
- Some require some complicated setup (e.g. SCPMover)
  1. Client creates public/private keypair. Public key written via Chirp
  2. Server configures user-level ssh daemon
     1. Uses public key as only accepted key
     2. Publishes port, full directory path, and name of user on server side
  3. Client connects to server and transfers file
- Some are pull-based movers
  - E.g. the “server” pulls from the client

HTCondor Chirp – Per Job Scratchpad



Client executing in an HTCondor slot

Chirp provides generic rendevous information
All Chirp information is logged, No private (e.g. temporary passwords) info written into the scratchpad

Server executing in an HTCondor slot

Recording and Displaying Experimental Data

- Each Placement Experiment (e.g. UCSD → CNIC)
  - Appends verbose information to a “job log” on the submit host
  - Each individual placement job has stdout,stderr recorded for client and server
    - 10-50Kbytes of text per iteration of the experiment
  - 1 Year of exeriments (hourly)
    - O(70MB) job log (1 log for all 8760 experiments, appended at each iteration)
    - O(500MB) stderr,stdout data (8760 jobs/experiment/year)
    - All highly-compressible text.
  - Use Git to record these
    - An interesting story (next slide)

We use Git to record each job within an experiment

- **Git add after each job within an experiment, replicate raw data (GitClone mover) across the iDPL**

```
[phil@murpa ucsd2wisc]$ du -sh .git placement*log detail
23G     .git
69M     placement4.log
32M     placement6.log
421M    detail
```

GIT does NOT automatically garbage collect

```
[phil@murpa ucsd2wisc]$ git gc
Counting objects: 63777, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (63772/63772), done.
Writing objects: 100% (63777/63777), done.
Total 63777 (delta 50503), reused 0 (delta 0)
Removing duplicate objects: 100% (256/256), done.
[phil@murpa ucsd2wisc]$ du -sh .git
107M    .git
```

~500MB of Experiment Data (Text)

Git Explodes to 23G in its repo

`git gc`
Repo is a svelte 107M

What has been good/not-so-good in HTCondor

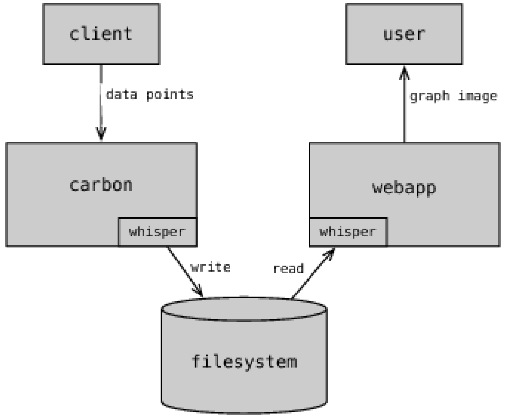
The Good:

- DAGman used for repetitive execution, Git logging of data
- "CRONDOR" used to execute job at specific time
- Per-job (iteration) stdout/stderr essential for debugging
- Reliability of Condor in the face of reboots of Master Collector, Schedd's , startd's, network partitioning, …
- Handles Cloud Networking (Non-routable/routable) split brain addressing

- The Not-So-Good
  - Parallel (Dedicated Scheduler) configuration limits scaling, execution within OTHER pools
  - I don't have a good understanding of more flexible ways inside of HTCondor to handle security/identity

Commonly Available Components

- Use HTCondor as job launching, job control
  - Directed Acyclic Graph Scheduler enables periodic submission
  - HTCondor's Chirp mechanism enables "rendevouz" for port advertisement
  - Well understood user/security model
  - Scales well, but iDPL uses it in a Non-HTC mode
- Graphite, Carbon and Whisper Database
  - Time-series display.
  - Open-sourced from Orbitz
  - Used at multiple large-scale data centers
- Python > 2.6.x
- Git – Software revision AND raw data stewardship



UCSD SDSC SAN DIEGO SUPERCOMPUTER CENTER QUALCOMM INSTITUTE

Sites

- Code: github.com/idpl/placement
- Graphite server (just a VM): http://vi-1.rocksclusters.org
- HTCondor - https://research.cs.wisc.edu/htcondor

Others

- FDT - http://monalisa.caltech.edu/FDT/
- Graphite/Carbon/Whisper - http://graphite.readthedocs.io/
- GridFTP - http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/
- iRODS https://irods.org/
- UDT - http://udt.sourceforge.net/