Trying out dCache for local use at Nikhef

Andrew Pickford
andrewp@nikhef.nl

Introduction

- Dutch National Institute for Subatomic Physics
  - research – LHC, gravitional waves, cosmic rays, neutrinos, and more ...
  - computing – part of the Netherlands Tier 1 for LHC
  - AND a batch system for local users: stoomboot

- stoomboot:
  - Dell PowerEdge M600
  - 90 nodes, 720 job slots, 1Gb/s NICs
  - SLC6
  - Bursty usage – idling then quickly full then idle again

Stoomboot Storage

- nfs - user home areas ~1.3 TB, data ~40 TB
- cvmfs – experiment software
- gluster – data ~150TB
  - 6 servers
  - native gluster client
  - stability issues
    - fails under high load
    - access limited to 24 out of 90 stoomboot nodes
    - storage is filling up

New Storage Hardware

- commodity hardware
  - 5 servers
    - Opteron 6320, 128 GB RAM, 36 x 4TB discs per server
    - 10 Gb/s NIC
    - 110 usable TiB per server
- file system requirements
  - stable
  - scalable
  - available on every compute node
  - easy to administer and maintain
    - data migration, data verification, quotas
  - traditional file system interface
- Try dcache via nfs

Initial Testing 1

- dcache setup
  - two control nodes
    - admin/nfs door, database
  - five storage nodes
  - monitoring node (carbon/grafana)
- test
  - repeatedly write randomised 4 GiB file
  - up to 20 client nodes (10 Gb/s NICs)
    - not stoomboot nodes, didn't want testing to disrupt batch system
  - client access via nfs v4.1

Initial Testing 2

Start with one client add additional client each hour

Reach ~ 3.5 GB/s summed across servers

Try increasing memory allocation to dcache

IO failures on clients unchanged

dCache server process failed

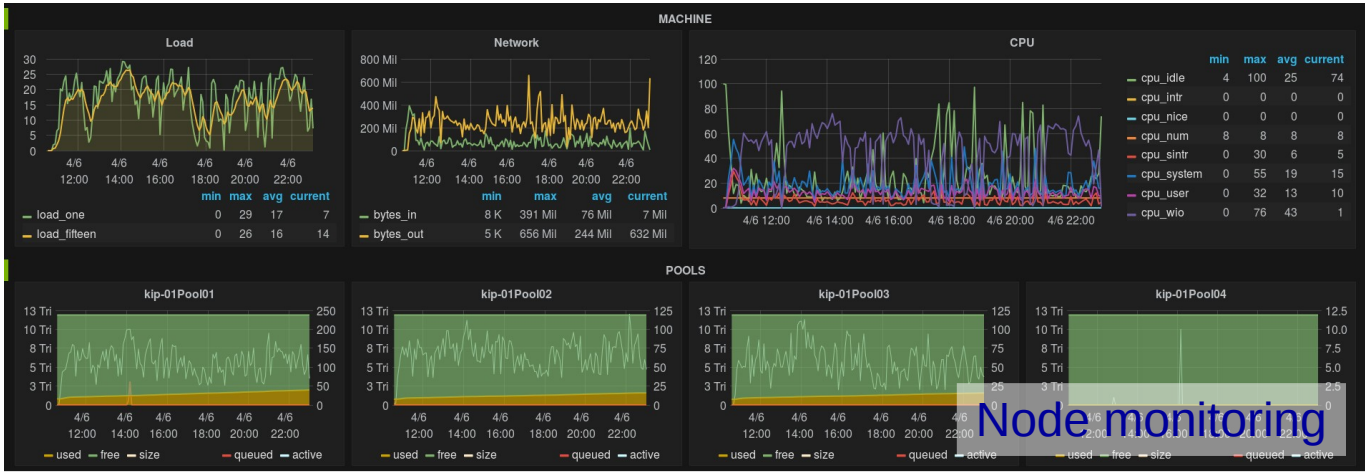
IO failures on clients

Observations

- client and server communication problems
  - client processes failed reporting errors
  - client processes locked in uninterruptible sleep state
  - occasionally clients completely unable to access dcache nfs mounts
  - we found bugs in dcache
  - reported and fixed
- changes
  - admin node – admin, info, httpd, statistics services
  - nfs door node – nfs door, pnfs, pool, pin and space manager services and chimera, space and pin manager databases
  - billing node – billing, alarms services, billing and other databases
  - storage servers – pool domains
  - increased heap and direct memory to 4GiB for pool and nfs door domains

Monitoring

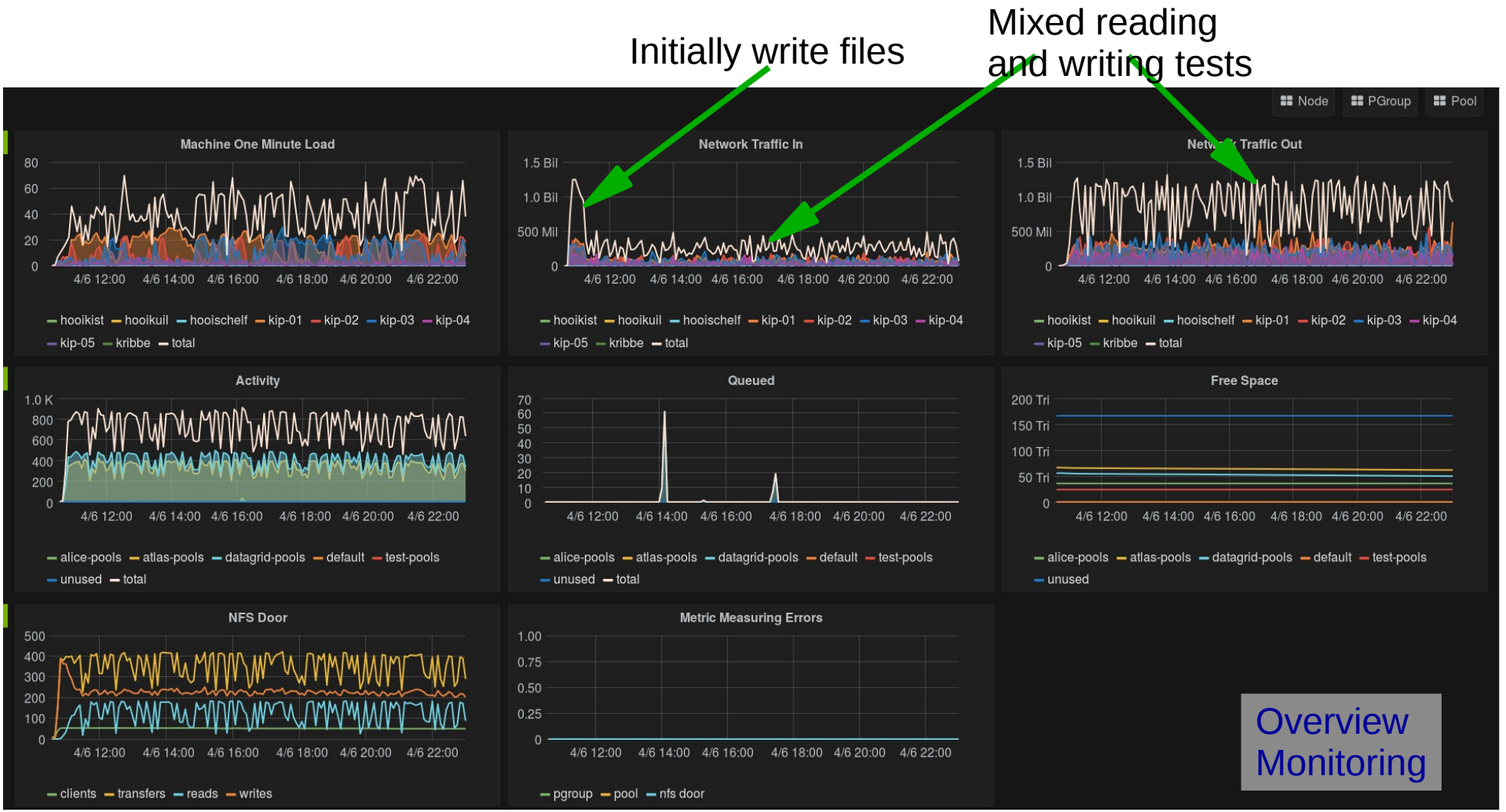
- carbon/grafana monitoring
- read xml output from info service
- push data to carbon server
- push cpu metrics from ganglia into carbon
- plot with grafana



Ongoing Work 1

- Testing / bug fixing over last 6 months
  - test → report bugs → test new version
  - fixed most (if not all) of the client/server communication issues
- more complex tests
  - read/write/copy/move/delete mixed testing
    - try to mimic demanding work flow
- nfs client / dcache server communication now stable
  - use stoomboot nodes without disrupting batch system
  - tests are now done through batch jobs
  - 200 testing jobs on ~40 clients
- still the odd problem
  - new clients can fail to connect to nfs door during high IO load
  - occasionally see double entries when listing files

Ongoing Work 2



Observations

- linux very aggressively buffers network traffic
  - large (GiB) file writes may be copied completely into client memory before any network data transfer starts
  - congestion/fair share issues with multiple processes per client
    - some processes can get starved of bandwidth
  - looked at starting network IO sooner
    - lowering dirty_bytes/dirty_background_bytes
    - might help, but nothing definite (maybe helped mitigate now fixed bugs)
  - looked at interface queuing: stochastic fairness queueing
    - but for us the client does not control the queue so doesn't help
- nfs mount options → rsize & wsize
  - tried 8kiB → 1 MiB, nothing definitive for stability
  - still to evaluate for performance

Conclusion / Future Plans

- Really only just started
  - we have a working system - but with some issues to work on
  - next step is to get our users fully involved and see what they can break
  - performance tweaking
    - so far only really looked at stability

- Many, many thanks to Tigran for all the bug fixes