Grid Deployment Board

"SC3 Debugging"

James Casey, IT-GD, CERN

CERN, 5th September 2005

Overview

- Open Issues from SC3
  - Long latency network issues
  - Procedures
  - Transfer issues - best practices for #streams & rates
- First results from debugging phase
- Summary

Problem 1

- Performance on transatlantic networks
  - Very slow per-file transfer rate (~1-2MB/s)
    - Even when multi stream (10/20)
  - Solution is to put a lot of files onto the network at once
    - BNL achieved 150MB/s but with 75 concurrent files
  - We see a lot of timeouts happening
    - FTS retries and the transfers have a high success rate but we lose effective bandwidth
  - These sites have a lot of bandwidth that we don't use
    - e.g. ASCC have 2G/s but it's hard to fill even with TCP based iperf
- Q: How do we up the single file transfer rate on transatlantic sites?
  - Do we need to go back to per site network tuning?

Solution 1 (1/2)

- Sites have verbally reported that 2.6 kernels perform better for them than 2.4 kernels
  - FNAL, ...
- We are placing a SL4 node in the WAN area to start to test this
  - Initially IA32, and will then test with IA64 too
- We will test the new TCP stacks that come by default with the RHEL4 kernel
  - BIC, Westwood
- We will investigate the usage of FAST in this kernel
  - With support from the FAST team at CalTech and CERN CS and ADC groups

Solution 1 (2/2)

- Proposal is to place a “reference” node at each T-1 site
  - ‘standard’ OS installed – i.e. SL3/4
  - The DPM software would be installed on it
- This can be used for system debugging, network tuning and regression tests
  - This would be used to run background iperf tests and file replication for regression analysis
  - Can remove a link from the transfer chain to eliminate source or destination SRM as cause of problem
- Does not have to be most up-to-date hardware
  - But good network connectivity and NIC essential

Problem 2

- SRM cleanup procedures are not understood
  - Often we see something going wrong on the transfers and we diagnose and solve the problem e.g. all allocated transfers have timed but movers not cleaned up
  - But the effect tends to go on longer
    - We see degraded performance afterwards and often the sites ends up just rebooting everything

- Q: How can we create, document and share standard procedures, so we don't have to reinvent the wheel 11 times?

Solution 2

- dCache workshop held last week at DESY to share knowledge
  - Maarten Litmaath will report on it
- Plans to hold similar event at CHEP covering all the SRM systems

Problem 3

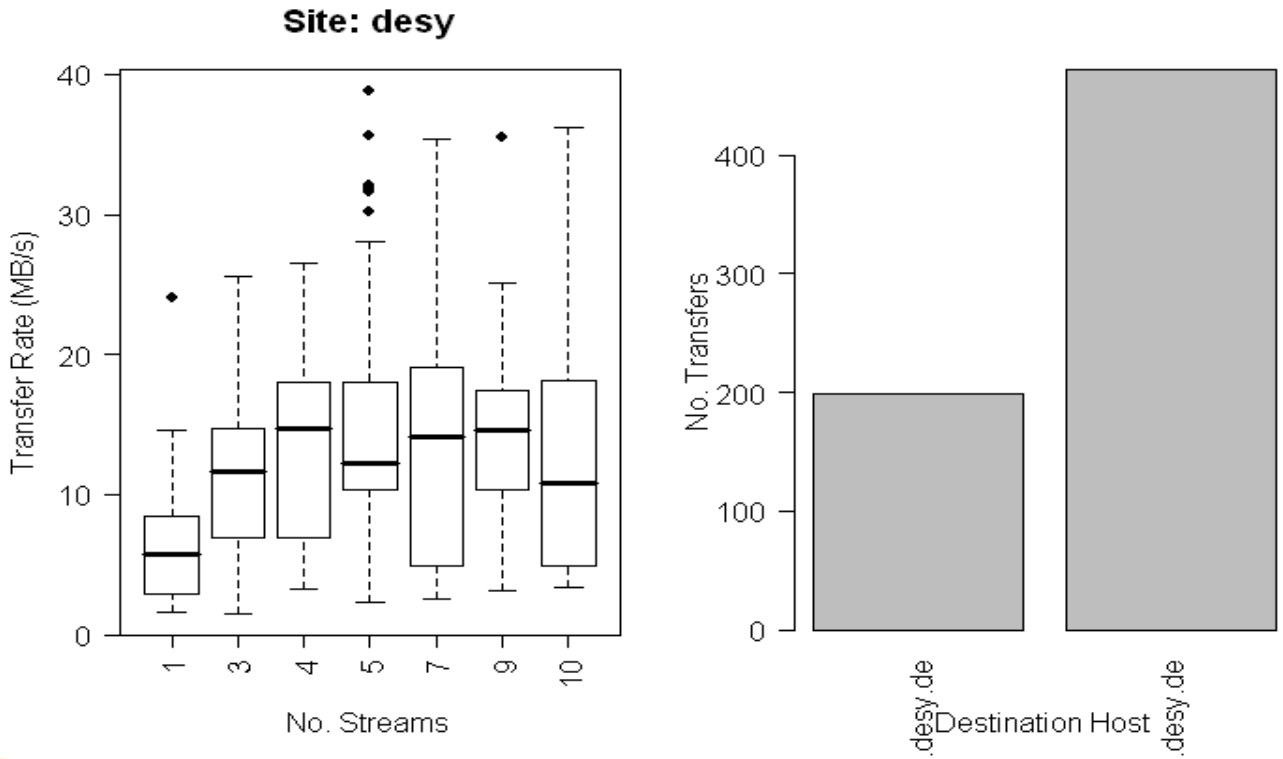
- During SC2, we tended to run with few transfers and a single stream per transfer
  - INFN – 10 single stream file transfers 100MB/s
  - FZK – 3 single stream file transfers – 150MB/s
- Now we don't see this
  - INFN has good file transfer rates (~10-15MB/s) but we only get 60% utilization of the network
  - FZK sees very low file transfer rate (~1-2MB/s) for many file transfers (but some seem to run much faster)
  - PIC (& IN2P3/SARA) work best when doing 10 concurrent streams
- Q: How can we reduce number of streams and get individual file rate higher (and more stable) ?

Debugging Phase

- Tackled the third problem:
  - How can we get higher and more reliable file transfer rates?
- Looked to answer several questions :
  - What is an ideal node kernel tuning?
  - How many streams are best?
  - What is effect of using SRM Copy?
- Restricted to low-latency sites
  - since network issues seem to play bigger role in high latency network routes
  - Tested with DESY to see how a well-tuned system should behave
  - Comparative results against INFN for CASTOR

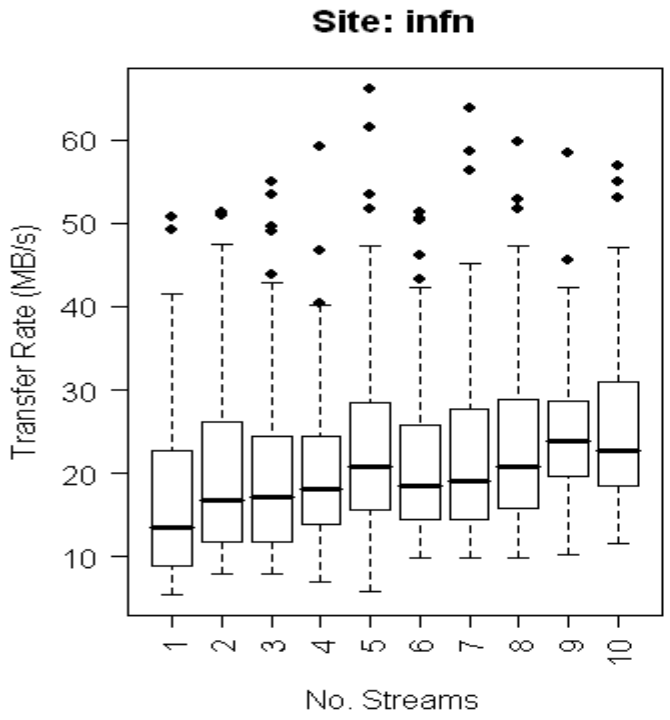
CERN-DESY with FTS (10 files)

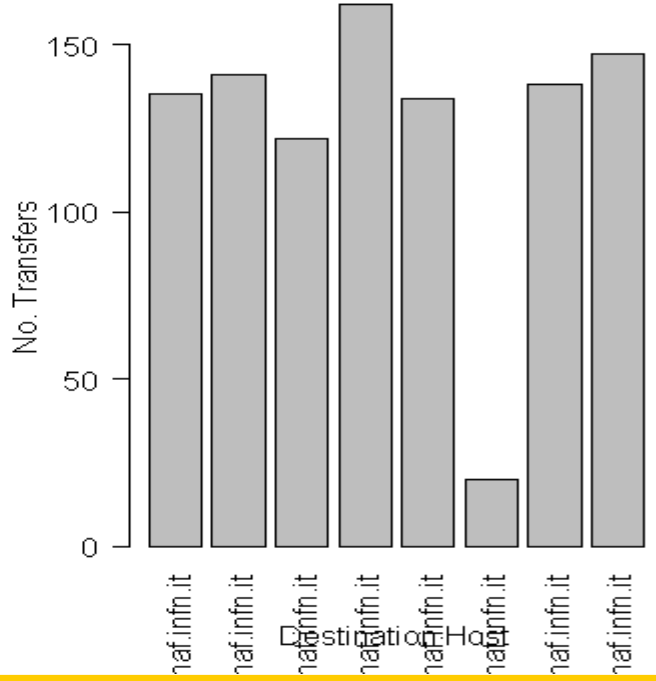
- With dCache transfer rate does not seem to scale with no. streams.
  - "# streams x #files ~ 50"



CERN-INFN with FTS (10 files)

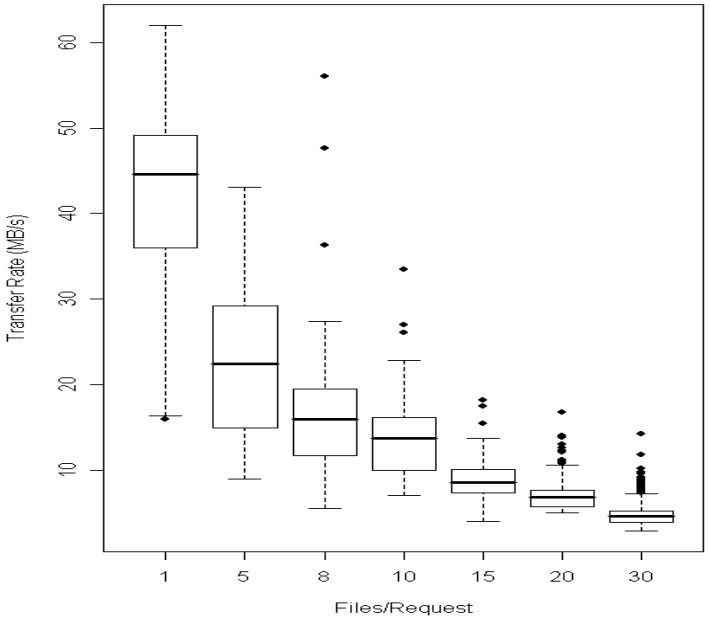
- Slight increase with no of streams (fixed to 10 concurrent files)
  - But total bandwidth did not translate to ~20MB x 10 – was in the range of 60-80MB/s.

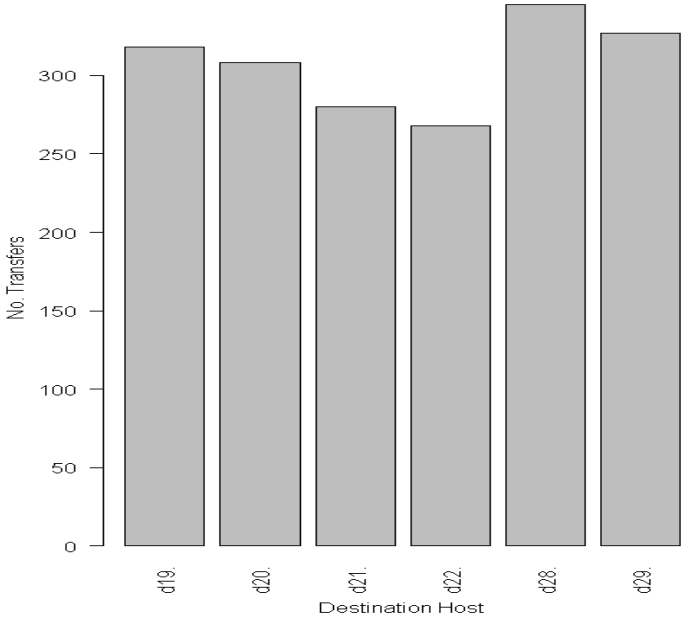




CERN-DESY with srmcp

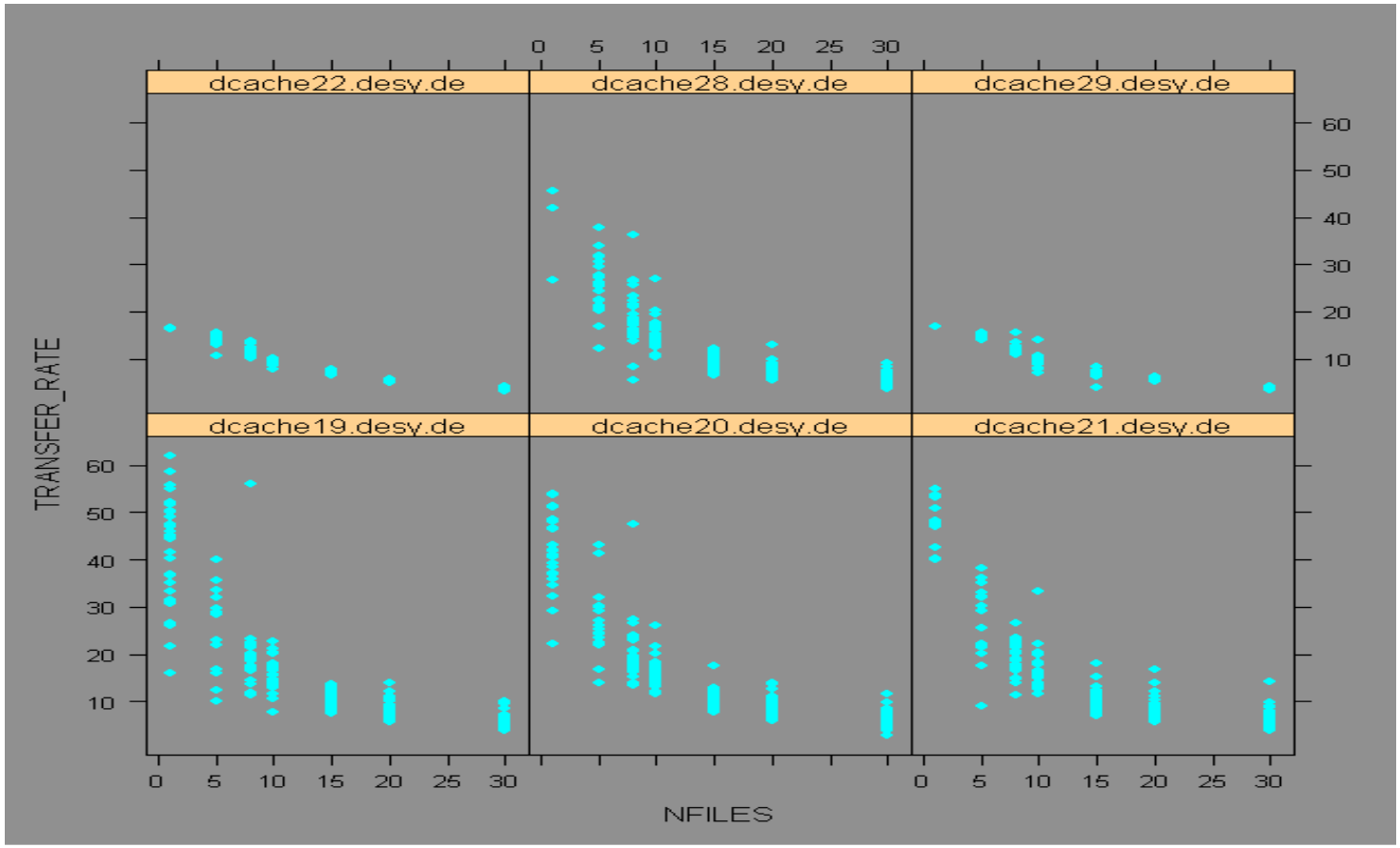
- We tended to fill bandwidth
  - but single file bandwidth inv. prop. to # streams
  - CASTOR returns all TURLs immediately, so dCache transfers them
    - Resource management needs to be done on both sides





Srmcp distribution by dest host

- Note effect of different TCP buffer sizes
  - 22+29 had 64K buffers, the rest had 2M buffers

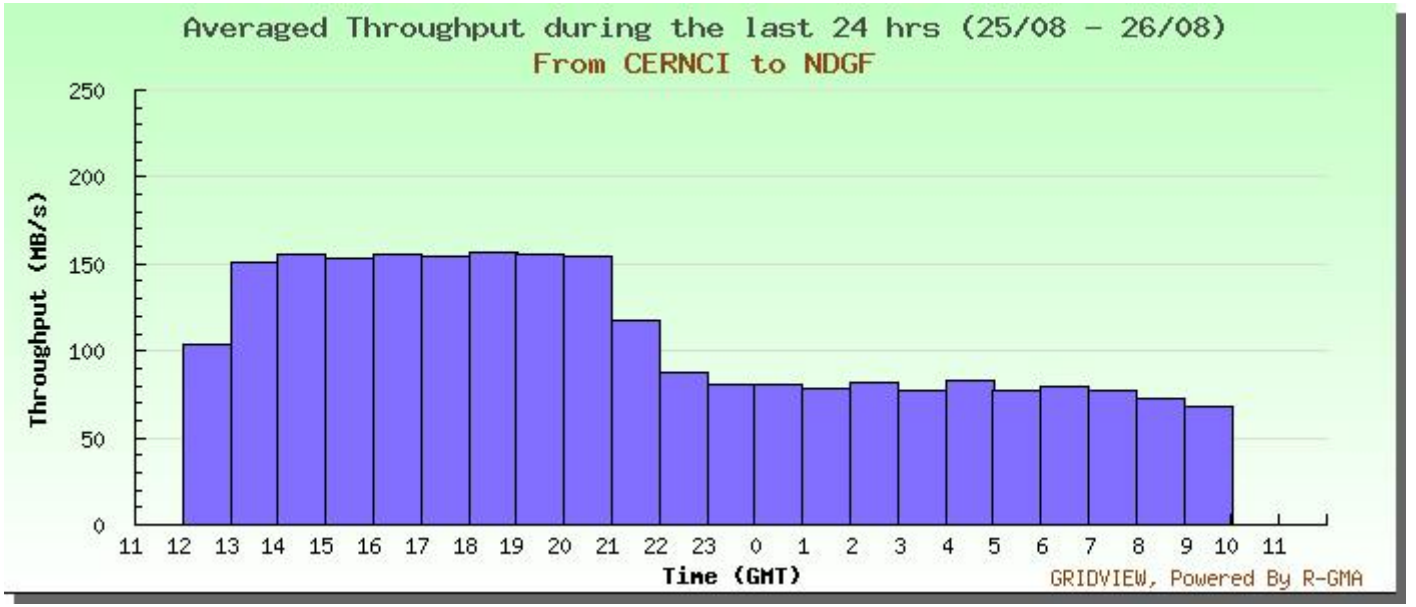


Monitoring ongoing transfers

- FTS used gridftp performance markers
  - Has 120 seconds marker-to-marker timeout
  - Has global transfer time set much higher (~1hr)
- dCache does not send the performance markers
  - This initially caused all long-hop transfers to time out
  - Have to disable this feature
- Had the effect of if any problem occurs, it takes 1hr to fail !
  - Bad for channel utilization
- dCache developers have promised to implement this feature

Failure of DPM pool node

- 1 DPM pool node out of 6 started to fail on gridftp
  - SRM kept scheduling to that node
  - Reminiscent of Globus gridftp black holes from SC2
- Rate drops from 150MB/s to 80MB/s

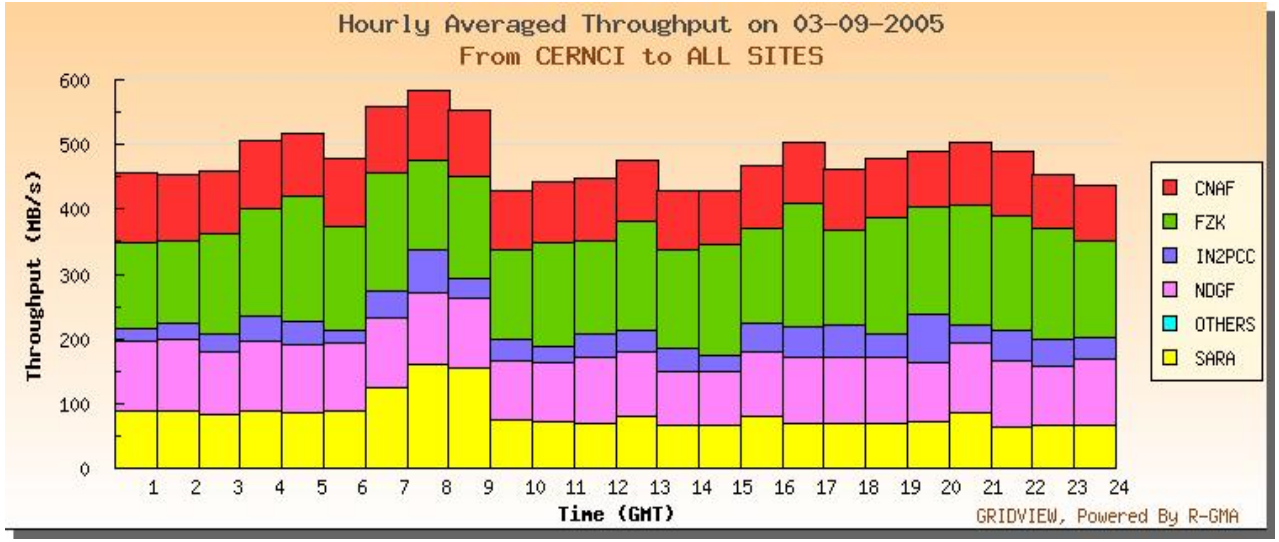


Misc Issues

- dCache 1.6.5 had a problem with pool balancing
  - Fixed in 1.6.6
  - This reduced rates, and explain some effects we saw at SARA, IN2P3
- FZK had problems with ext3 file systems
  - Moved to GPFS file systems – now can run at up to 250MB/s
- SARA incresed transfer rate to 160MB/s using 3 nodes by throttling #transfers in dCache
  - Allow a large number of FTS file transfers (~20) but throttle each pool node to a small number of movers (~3)
  - This leads to transfers “bunching up” before gridftp
  - Overcomes some of the latencies involved with SRM

Post-debugging

- Now can achieve same rate as before with fewer sites
  - Still need to add in other sites, and see how what the new upper limit it



Summary

- Started to tackle the problems
  - Especially in regards to rates to individual sites
- Added some knowledge
  - 5 streams is a good number for 10 concurrent files with FTS
    - But dCache does seem capable of running high speed single stream transfers
  - Srmcp gives better load balancing over door nodes
    - With FTS, all pool nodes were used for storage, but door node usage wasn't balanced
    - But throttling needed in other SRM implementations to stop dCache overloading them
  - #files x file transfer rate != throughput
    - Significant lossage, due to SRM overhead and FTS scheduling