GridPP
UK Computing for Particle Physics

Tier-2 optimisation of dCache and DPM

Greig A Cowan

University of Edinburgh

Graeme A Stewart, Jamie K Ferguson

University of Glasgow

GridPP
UK Computing for Particle Physics

Outline

1. Background: a Tier-2 perspective of storage
2. Objective of this work
3. Details of optimisation tests:
   - Pool filesystems
   - Linux kernels
   - Transfer parameter configuration
4. Results and additional work
5. Future work
6. Conclusions
7. Extra: SRM + NFS

GridPP
UK Computing for Particle Physics

A *typical* Tier-2

No such thing as **typical**, but there are some similarities.

- Limited hardware resources:
  - One or two nodes attached to a few TB of RAID'ed disk.
  - Some storage NFS mounted from another disk server.
  - No tape storage.
- Limited manpower to spend on administering/configuring an SRM.
- Choice of SRM applications (dCache, DPM, StoRM . . .)
- Require SRM to be optimised in order to handle the data flows from the LHC.
  - GridPP service challenge set target for T1$\rightarrow$T2 transfer rate of $\geq$ 300Mb/s.

GridPP
UK Computing for Particle Physics

Objective

To use gLite's File Transfer Service (FTS) to study a typical T2 SRM setup, looking at how changes in the:

- disk pool filesystems
- Linux kernels
- FTS transfer parameters

affect the **data transfer rate** when writing into the SRM.

- GridPP uses both dCache and DPM, so run tests for both.
- Want to be able to make recommendations to sites about the optimal setup to use.

Hardware used

- Representative of Tier-2 hardware.
- Single Node running both admin **and** pool services of dCache/DPM.
  - Dual core Xeon.
- 5TB RAID level-5 disk, 64K stripe. Partitioned into three 1.7TB filesystems.
- Source SRM was a local DPM, capable of reading data at a sufficiently high rate that it would not act as a bottleneck.
- Gb/s network between the two SRMs (no firewalls or other annoyances in the way).

Pool filesystems and kernels

- Four different filesystems on 2.4 and 2.6 series kernels.
- Could not run xfs under stock SL3.0.5 2.4.21 kernel - use CERN build of 2.4.21 with xfs support.
- Default mount options used.

OS	Kernel	Filesystem			
		ext2	*ext3*	*jfs*	*xfs*
SL 3.0.5	*2.4.21*	Y	Y	Y	N
SL 3.0.5	*2.4.21+xfs*	Y	Y	N	Y
SLC 3.0.6	*2.6.9*	Y	Y	Y	Y

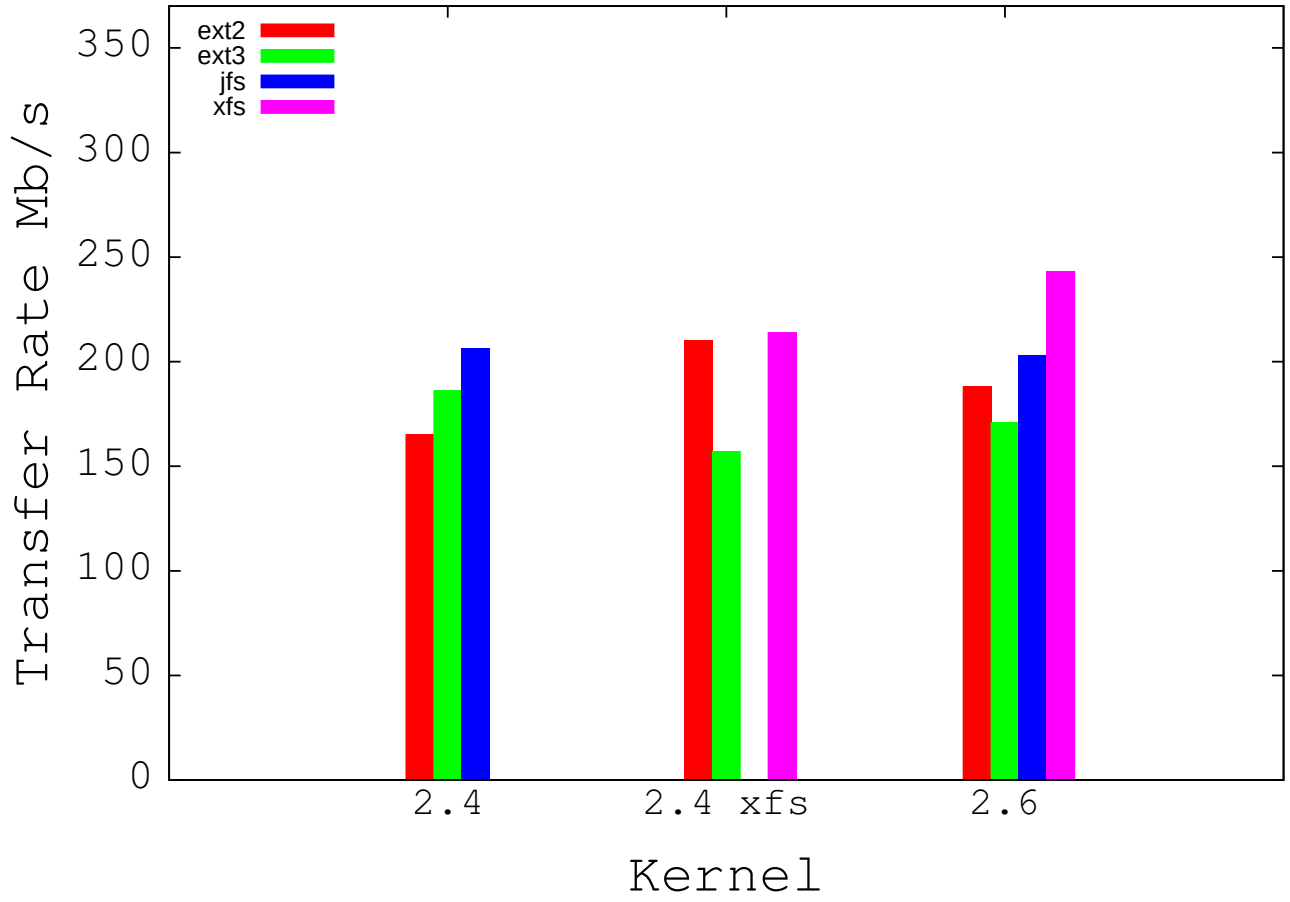
Transfer parameters

Many possibilities here, but we only looked at two that could be modified via FTS:

1. Number of concurrent files (i.e. number of files that FTS attempts to simultaneously transfer). $N_f \in \{3, 5, 10\}$
2. Number of parallel streams (i.e. number of GridFTP streams used per file transfer). $N_s \in \{3, 5, 10\}$

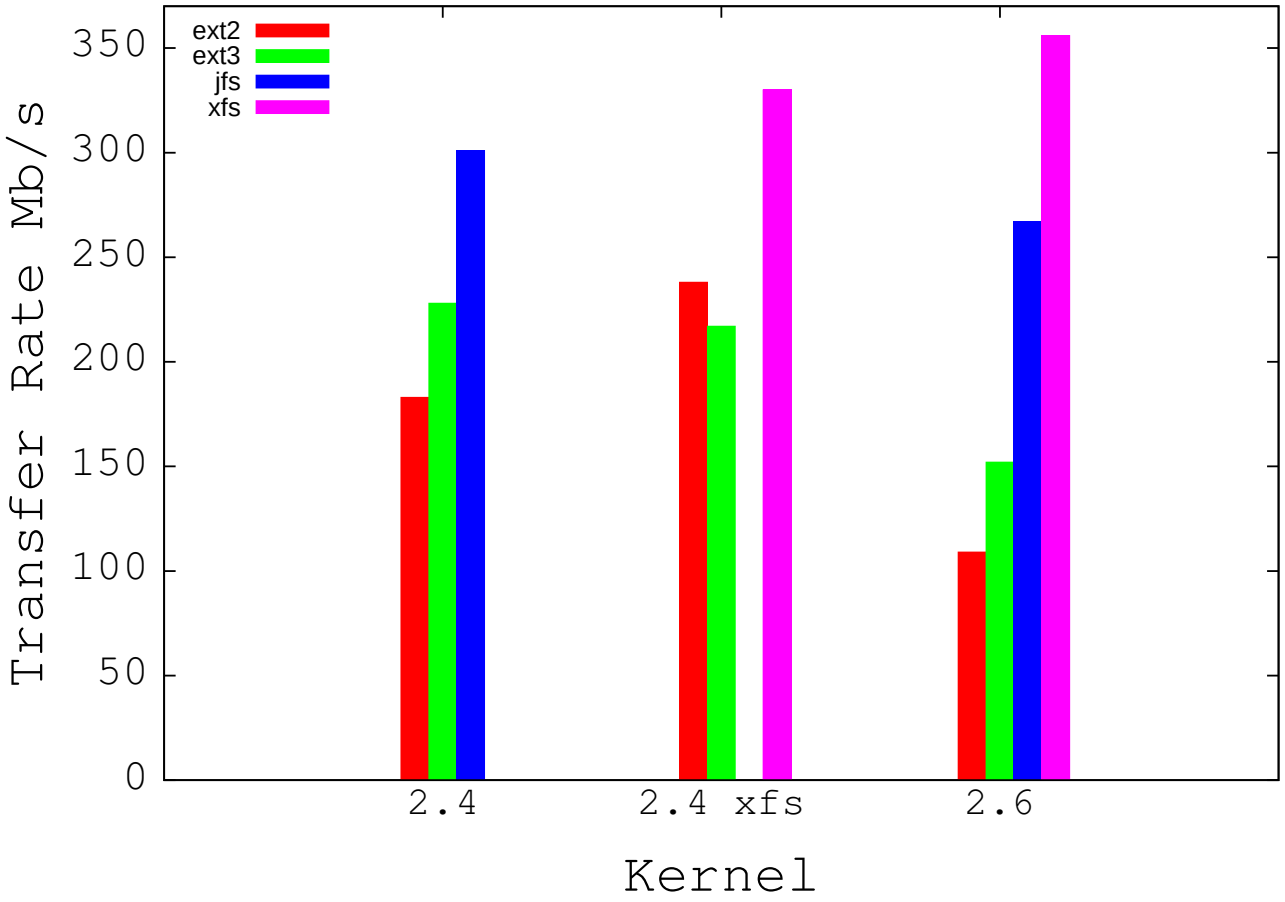
Submitted FTS job to transfer 30*1GB files from source DPM into our test SRM. Using FTS allowed us to monitor the status of the jobs (Done, Waiting. . .)

dCache: $N_f = N_s = 3$



No files failed to transfer.

DPM: $f = \quad s = 5$


ext2
ext3
jfs
xfs
Transfer Rate Mb/s
0
50
100
150
200
250
300
350
2.4
2.4 xfs
2.6
Kernel

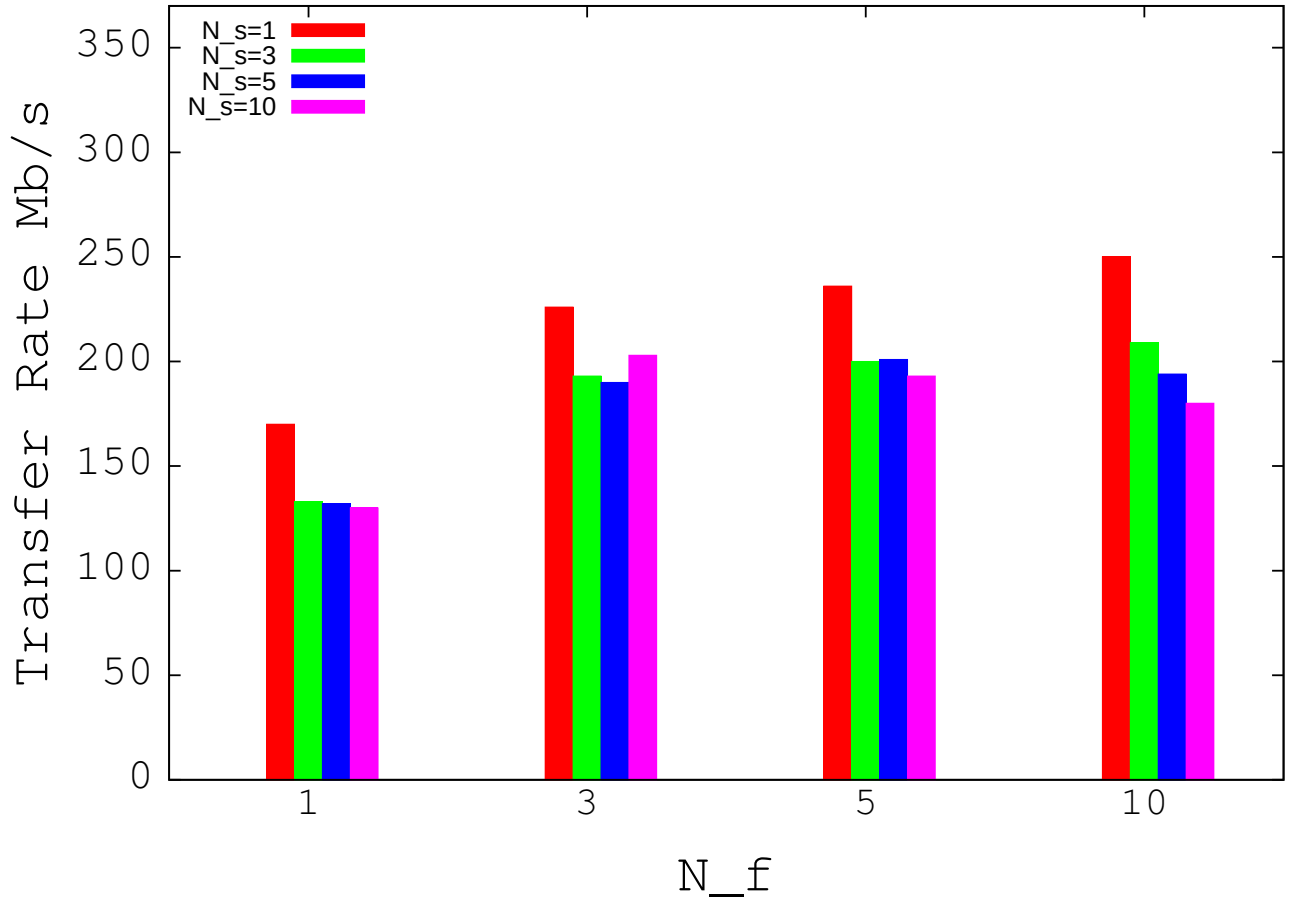
GridPP
UK Computing for Particle Physics

DPM: $f = \quad s = 5$

Percentage error rates for different filesystems and kernels with DPM.

Kernel	Filesystem			
	ext2	*ext3*	*jfs*	*xfs*
2.4.21	11	0	11	-
2.4.21+xfs	10	0	-	0
2.6.9	2.6	2.6	2.2	0

dCache, 2.6.9 kernel, jfs pool



Clearly using **single stream** leads to highest rate. For $N_f = 10$ there is a 20% improvement between $N_s = 3$ and $N_s = 1$.

dCache/DPM results

Observed highest transfer rates with the following setup:

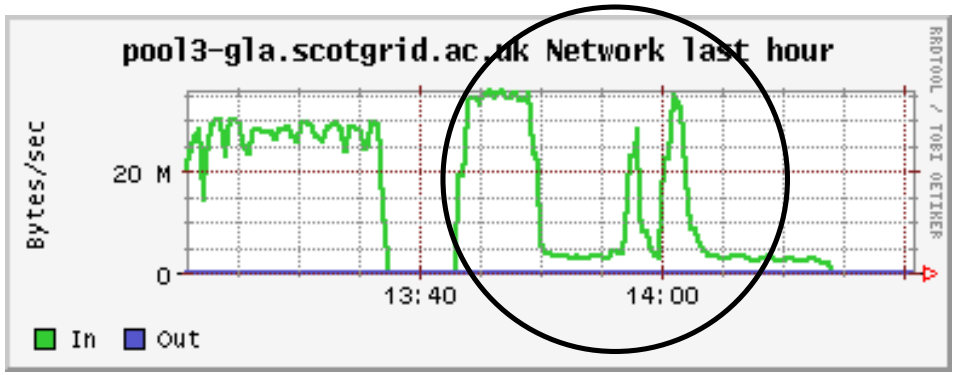
- Pool filesystem: xfs
- OS/Kernel: SLC 3.0.6, 2.6.9 kernel
- FTS parameters: N_s low, N_f high

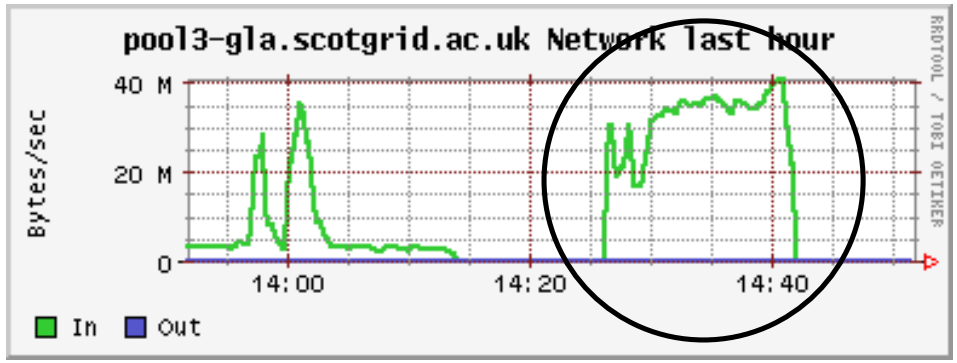
GridPP
UK Computing for Particle Physics

ADDITIONAL WORK

$f > 10$

- Initial tests show that problems occur if N_f is large since it leads to high load on machine after first batch of files transferred.
- Likely due to post-transfer SRM negotiation, leading to FTS requests timing out.
- For example 30*1GB files into 2.6.9 jfs dCache pool:





$N_f = 15$

142Mb/s, 15 failed

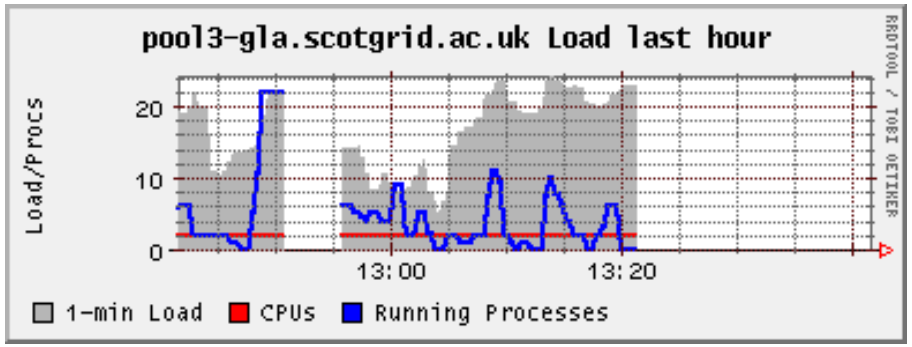
$N_f = 1 \rightarrow 15$

249Mb/s, 0 failed

- Would be better if FTS **staggered** the start times of transfers.

TCP buffer sizes

- Changed maximum TCP buffer sizes to match those set in the main dCache configuration file (1MB default).
- Led to $\geq$10% performance improvement with 2.4.21 kernel. No failed file transfers.
- Led to 20% performance improvement with 2.6.9 kernel running xfs. No failed file transfers.
- But, led to high machine load with 2.6.9 kernel running ext2 and ext3. Eventually caused the machine to crash:

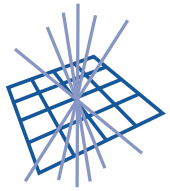


Future Work

- Using non-default mount options for each of the filesystems.
- Repeat with SL4 as base OS.
- Other filesystems i.e. ReiserFS, GPFS, Lustre
  - If looking at GPFS, then could make comparison to StoRM (another SRM).
  - Report at HEPiX suggested that ReiserFS is only optimal if used with small file sizes - is this a use case for GridPP/LCG?
- Further investigations of kernel-network tuning parameters since defaults typically un-suitable for HEP.
  - TCP BIC for 2.6 kernels (see T. Ferrari's talk at HEPiX)
- Repeat tests for different RAID stripe sizes.

Conclusions

- Era of SRM at Tier-2 sites is upon us.
- Sites need to deploy and configure their SEs hardware and software in order to meet the needs of the experiments computing models and to provide efficient service to users.
- Tier-2's typically do not have time to carry out this optimisation themselves. They need guidelines/recommendations.
- Tests have shown that using xfs with a 2.6.9 kernel leads to highest file transfer rate when used with dCache and DPM.
- Still further tuning work to be done to extract optimal performance from the SRMs.


GridPP
UK Computing for Particle Physics

EXTRA


ScotGrid
Scottish Grid Service

NFS + SRM

ADVANTAGES

- Easy way to get access to storage which would not be available otherwise.

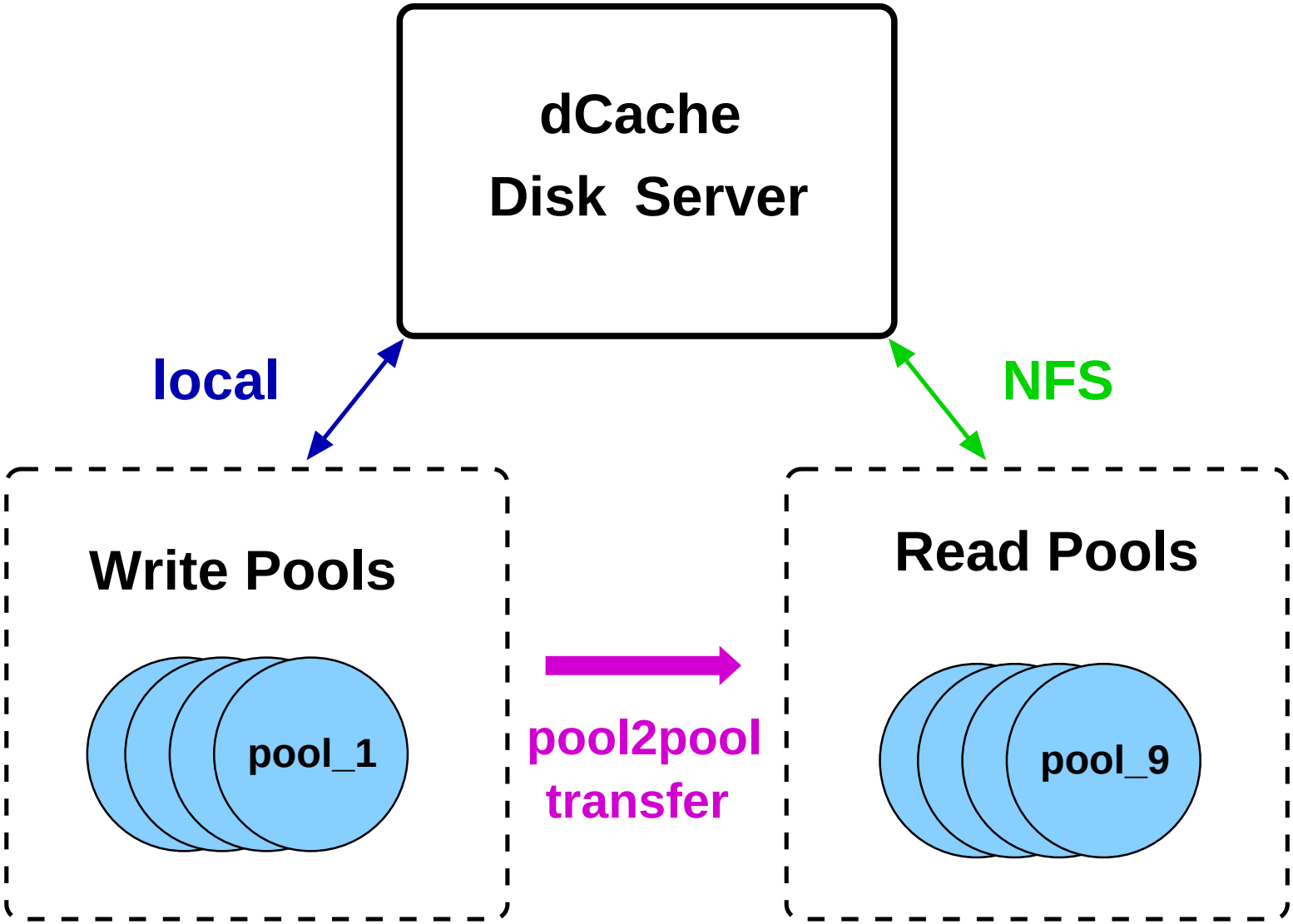
DISADVANTAGES

- Slow writes - limits transfer performance
- Stale NFS file handles keep cropping up - need to resovle these before storage can be used again.

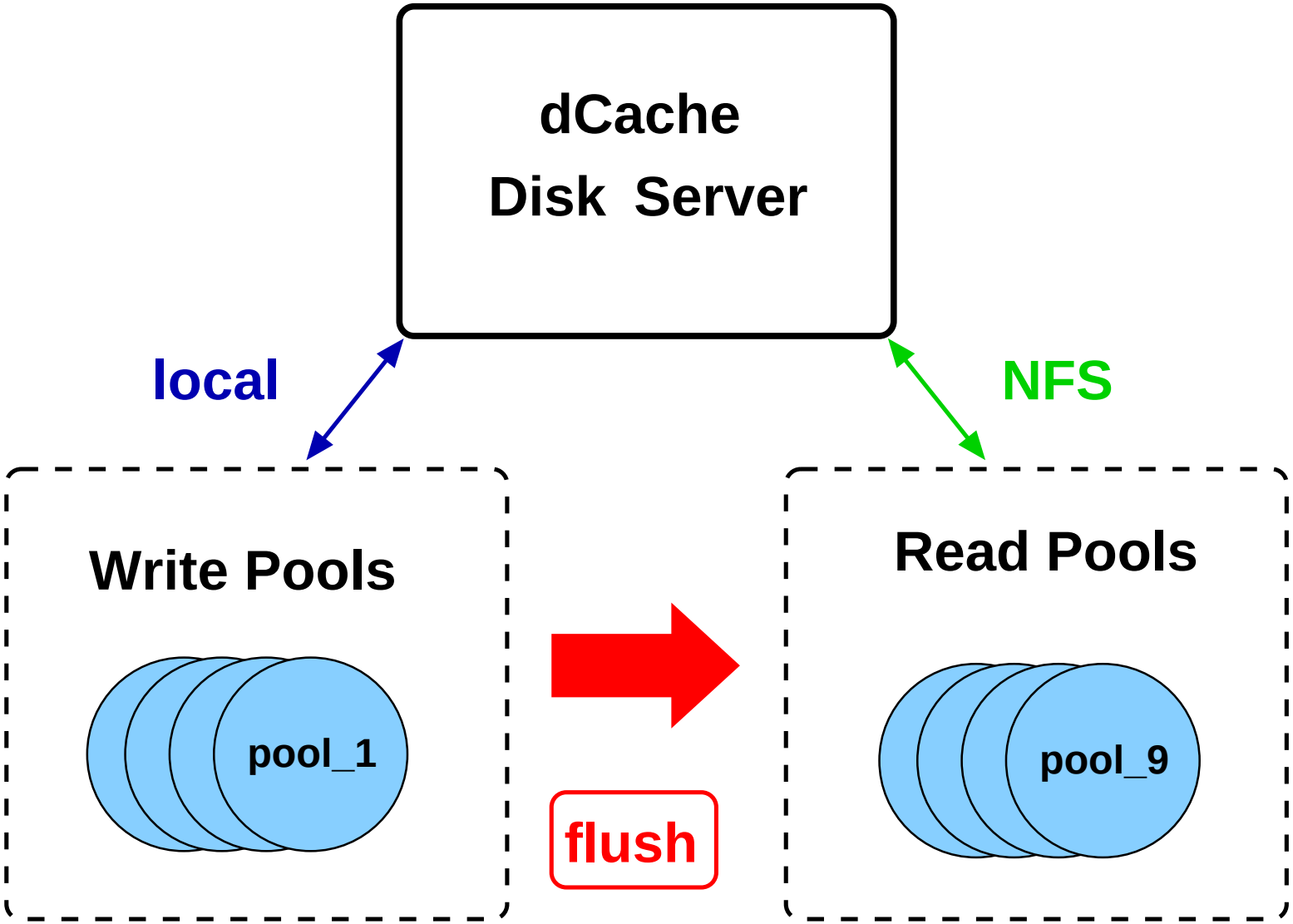
If you need to use it within your SRM, then the flexibility of dCache can help to improve performance:

- use local storage as **write pools**
- use NFS storage as **read pools**

dCache 1.6.6 (current)



dCache 1.6.7 (next release)



NFS + SRM

- Do you have to use NFS?
- Is it possible to run the dCache/DPM pool node software on the NFS server?
- What sbout using other access methods - would it be possible to directly attach to the server?

dCache configuration

- `parallelStreams` in dCacheSetup file had no effect on the FTS transfers. Only has effect when using `srmcp` to initiate transfer.