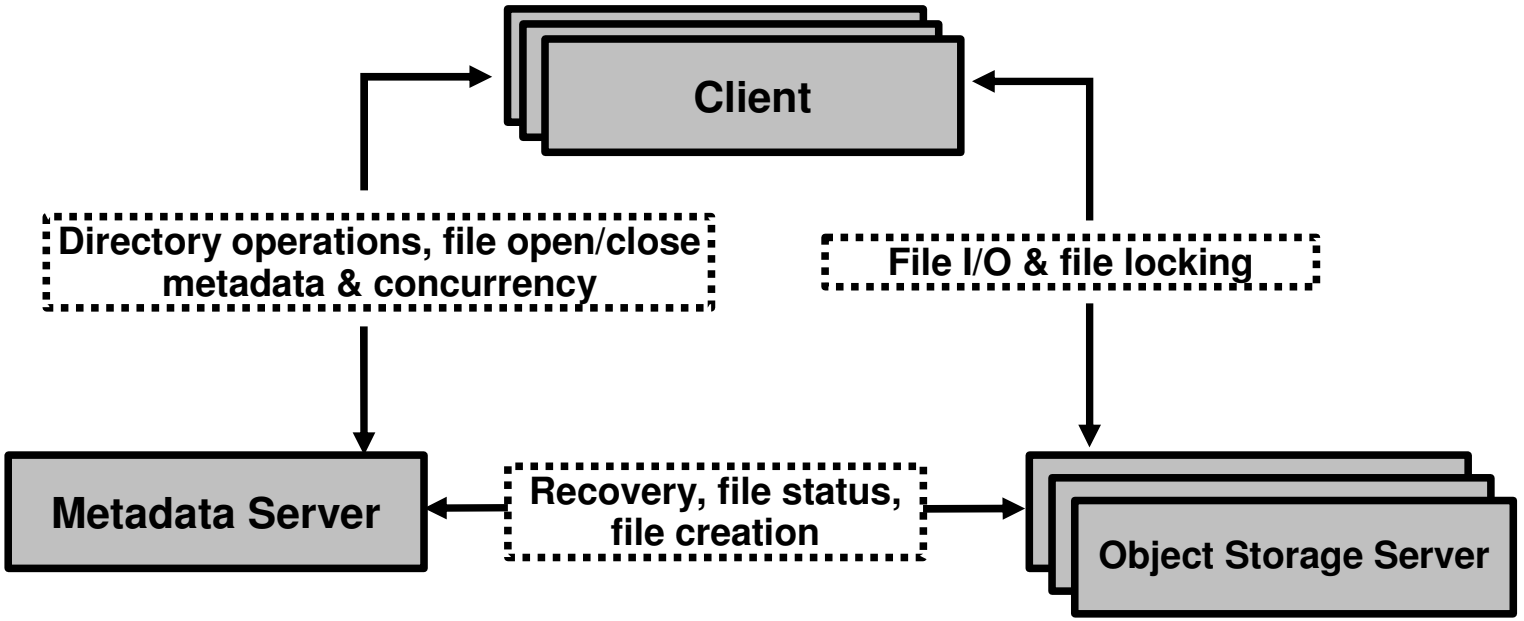
Karlsruher Institut für Technologie

Using file systems at HC3

Roland Laifer

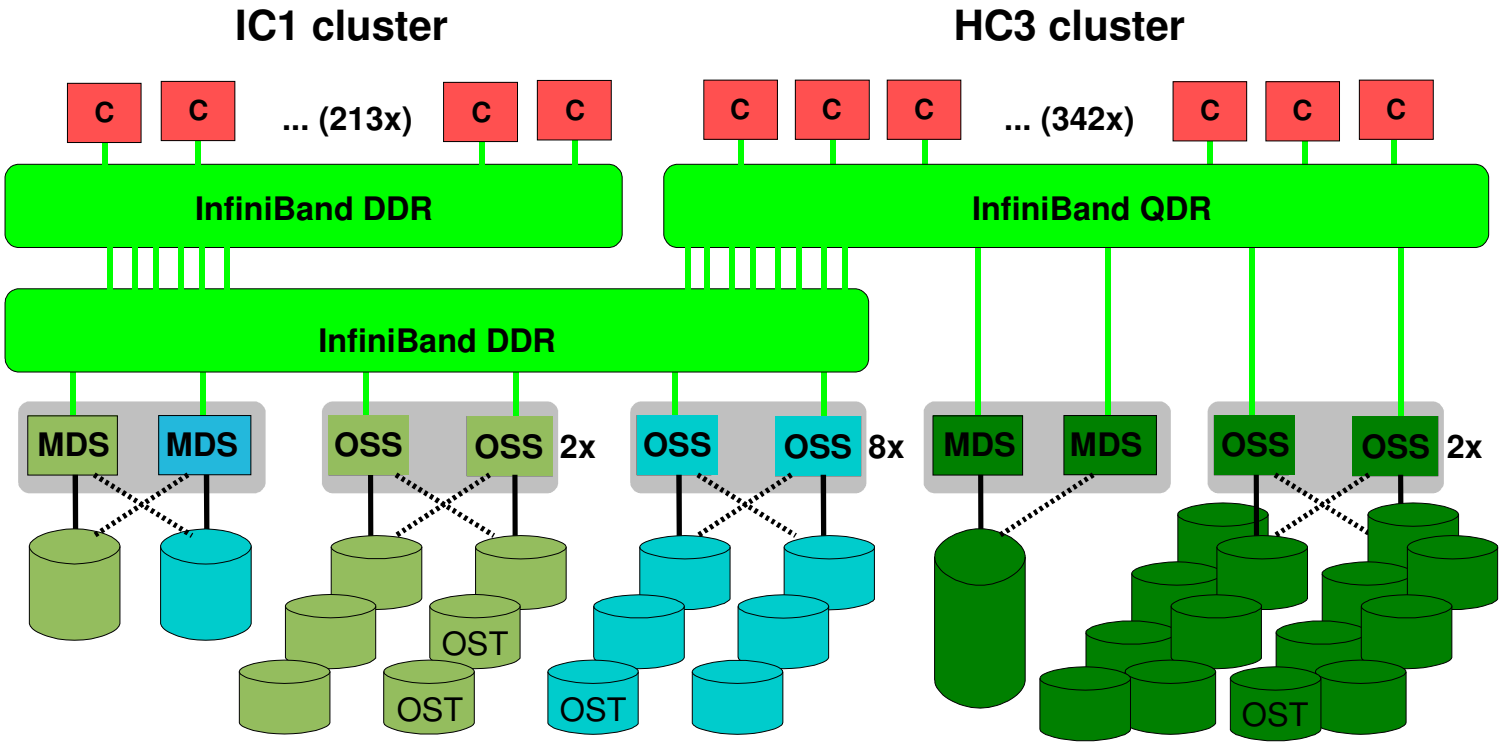
STEINBUCH CENTRE FOR COMPUTING - SCC

Basic Lustre concepts



- Lustre componets:
  - Clients (C) offer standard file system API
  - Metadata servers (MDS) hold metadata, e.g. directory data
  - Object Storage Servers (OSS) hold file contents and store them on Object Storage Targets (OSTs)
  - All communicate efficiently over interconnects, e.g. with RDMA

Lustre file systems at HC3



File system	$HOME	$PFSWORK	$WORK
Capacity (TiB)	76	301	203
Storage hardware	transtec provigo	transtec provigo	DDN S2A9900
# of OSTs	12	48	28
# of OST disks	192	768	290

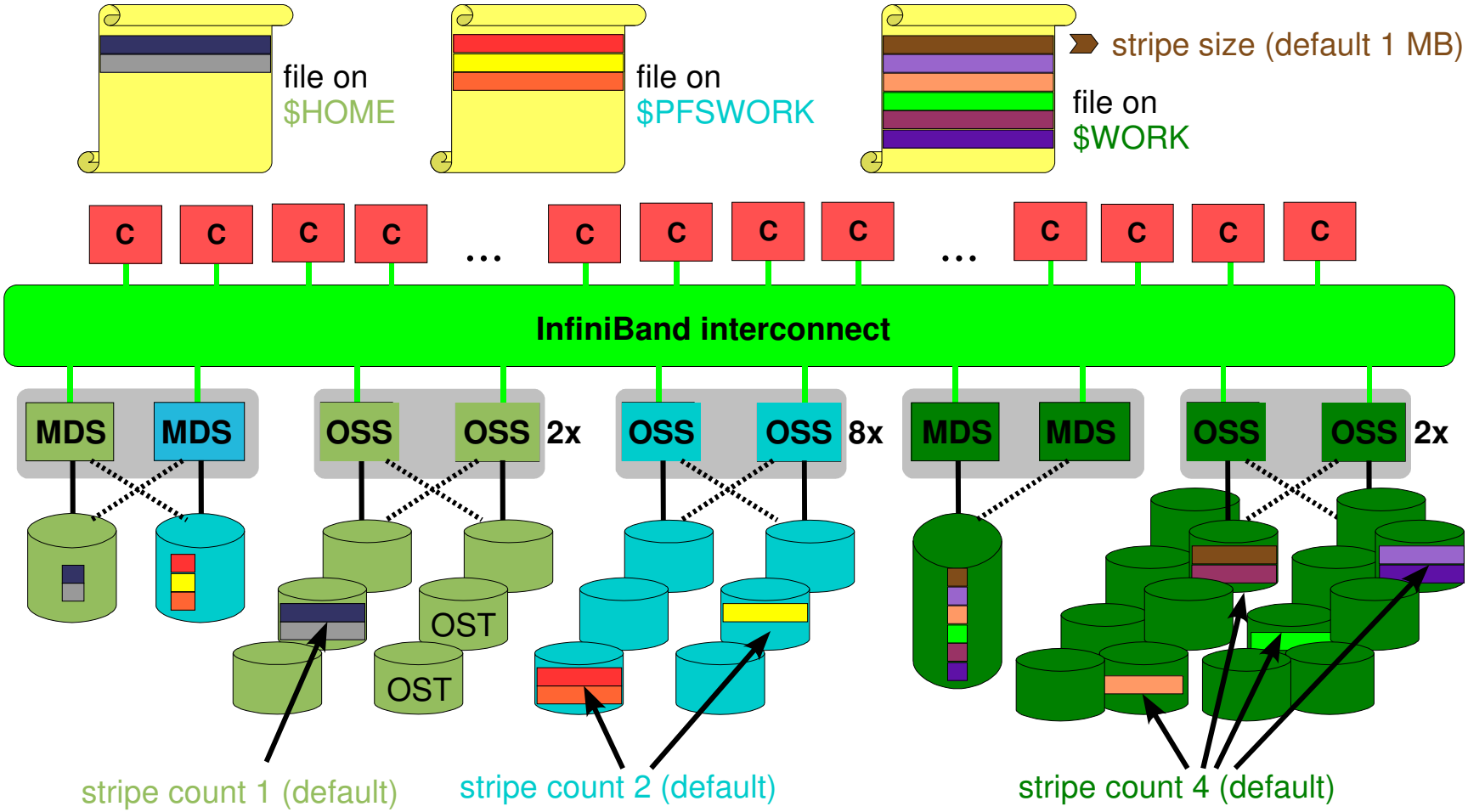
File system properties

Property	$TMP	$HOME	$WORK	$PFSWORK
Visibility	local node	HC3, IC1	HC3	HC3, IC1
Lifetime	batch job	permanent	> 7 days	> 7 days
Capacity	thin/med. | fat | login 129 | 673 | 825 GB	76 TB	203 TB	301 TB
Quotas	no	not enforced	not enforced	not enforced
Backup	no	yes	no	no
Read perf. / node	thin/medium | fat 70 | 250 MB/s	for IC1 nodes 600 MB/s	1800 MB/s	for IC1 nodes 600 MB/s
Write perf. / node	thin/medium | fat 80 | 390 MB/s	for IC1 nodes 700 MB/s	1800 MB/s	for IC1nodes 800 MB/s
Total read perf.	n*70 | n*250 MB/s	1700 MB/s	4800 MB/s	6200 MB/s
Total write perf.	n*80 | n*390 MB/s	1500 MB/s	4800 MB/s	5500 MB/s

Which file system to use?

- Follow these recommendations:
  - Whenever possible use $TMP for scratch data
  - Use $WORK for scratch data and job restart files
  - Use $PFSWORK to share scratch data between clusters
  - Use $HOME for long living data and when backup is required
  - Archive huge and unused data sets

How does striping work?

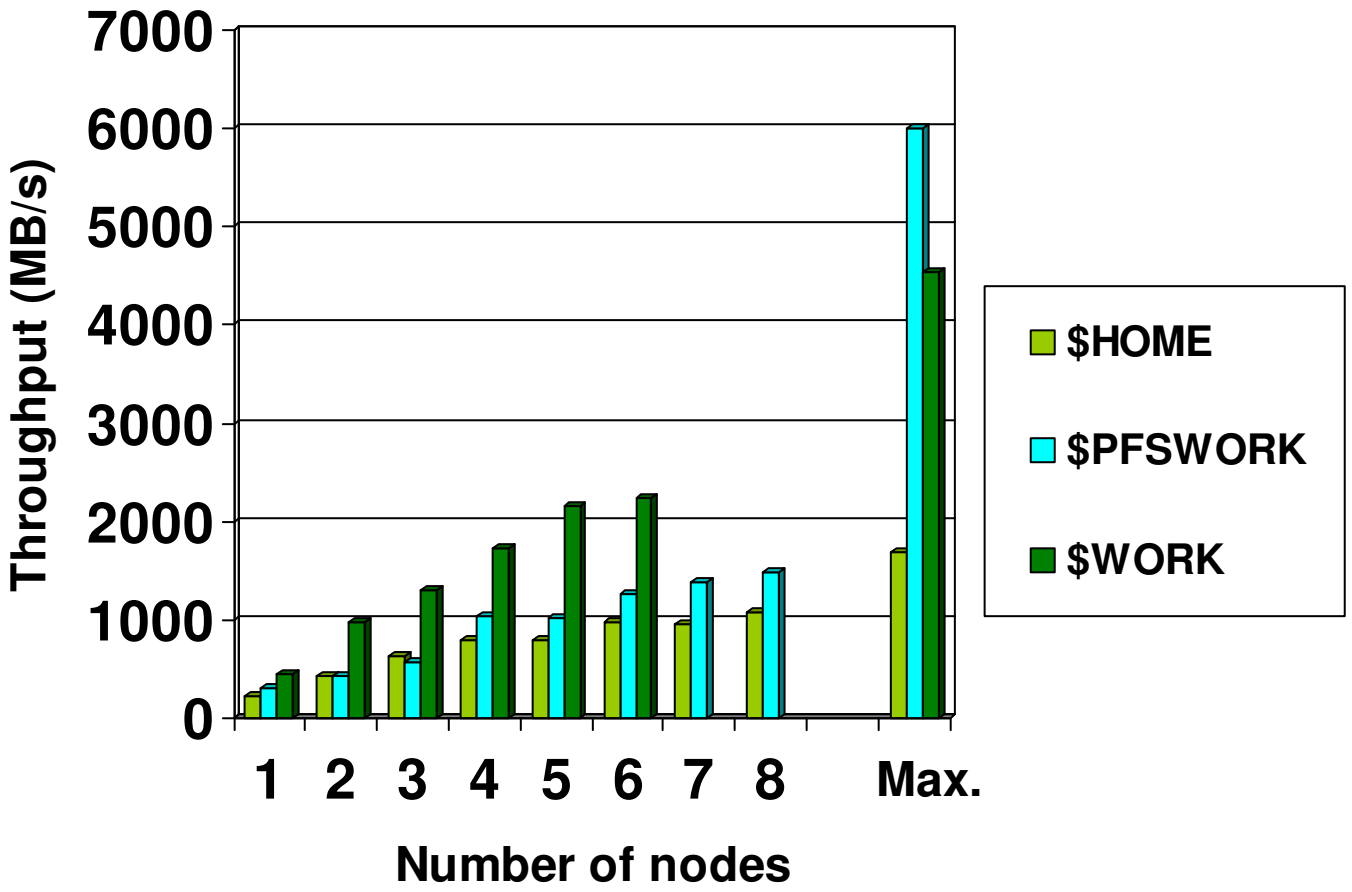


- Parallel data paths from clients to storage

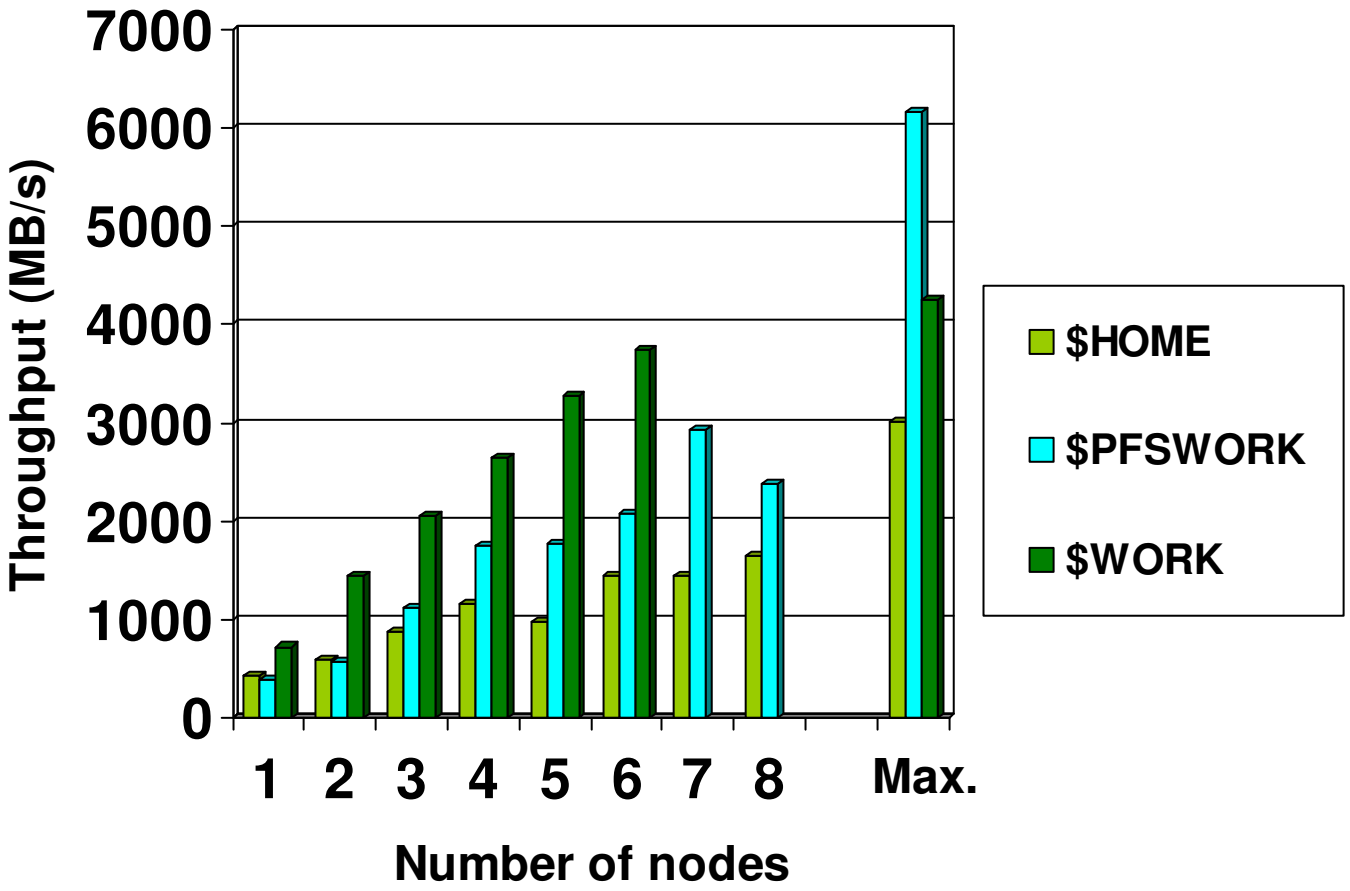
Using striping parameters

- Important hints to use striping
  - Striping parameters inherited from parent directory or file system default
    - To change parameters for existing file, copy it and move it back to the old name
  - Each new file is automatically created on new set of OSTs
    - No need to adapt striping parameters if many files are used in similar way!
- Adapt striping parameters to increase performance
  - OST performance is usually limited by storage subsystem
    - At HC3 pretty similar for all 3 file systems: ~200 MB/s
    - Increasing the stripe count might improve performance with few large files
  - Adapt stripe size to match application I/O pattern
    - Only makes sense in rare cases
- Show and adapt striping parameters
  - Show parameters: *lfs getstripe <file|directory>*
  - Change the stripe count: *lfs setstripe –c <count> <file|directory>*
  - Change the stripe size: *lfs setstripe –s <size> <file|directory>*

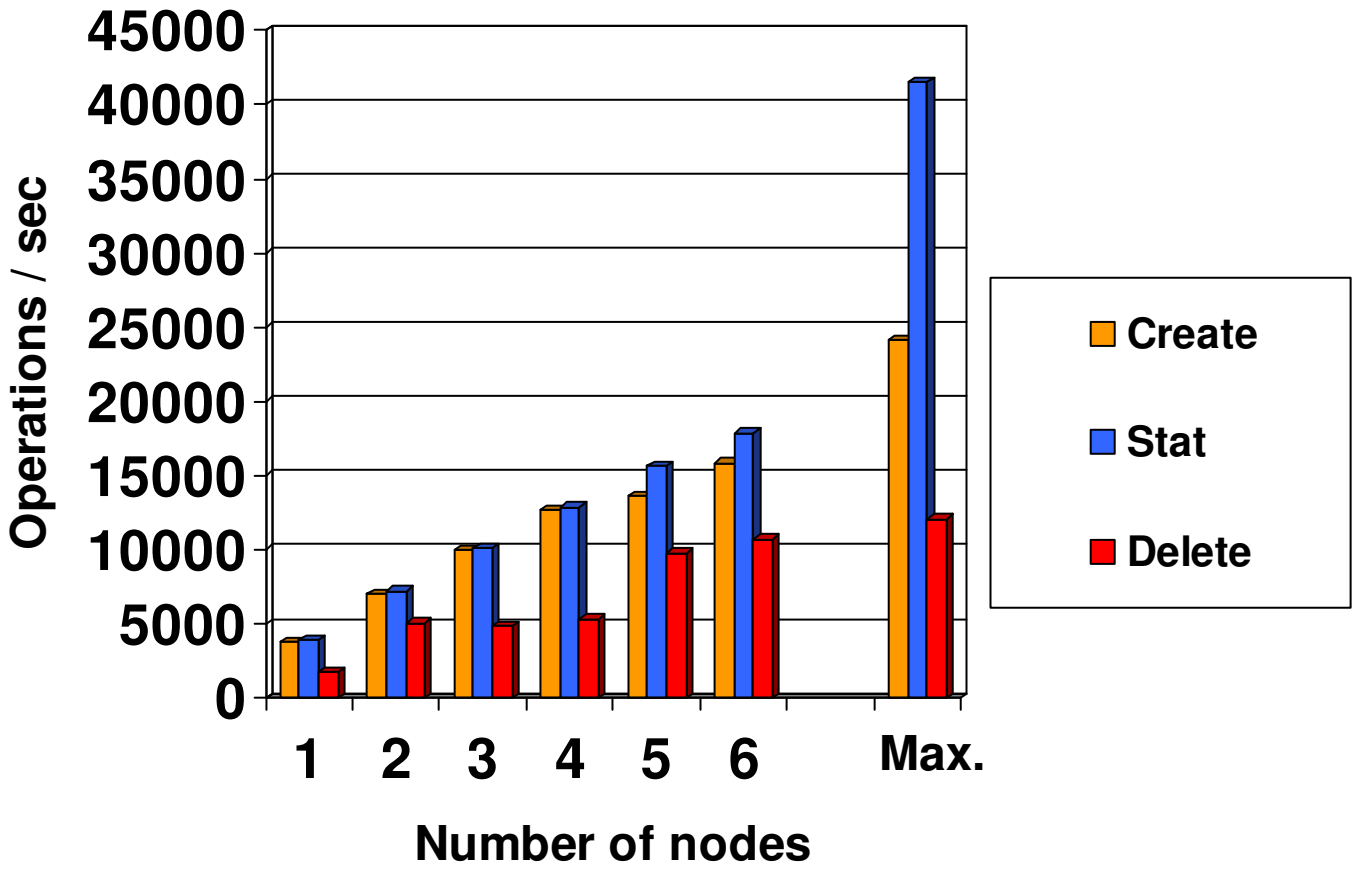
Write performance with default stripe count



Read performance with default stripe count



Metadata performance



Best practises (1)

- Change nothing if you use few (< 100) small (< 10MB) files
- Increasing throughput performance:
  - Use moderate stripe count (4 or 8) if only one task is doing IO
    - Improves single file bandwidth per client
  - To exploit complete file system bandwidth use several clients
    - Different files from different clients are automatically distributed
    - For one shared file use chunks with boundaries at stripe size
    - If many tasks use few huge files set stripe count to -1
  - Use stripe count 1 if lots of files are used in the same way
  - Collect large chunks of data and write them sequentially at once
  - Avoid competitive file access
    - e.g. writing to the same chunks or appending from different clients
    - e.g. competitive read/write access
  - Use $TMP whenever possible
    - If data is used by one client and is small enough for local hard drives

Best practises (2)

- Increasing metadata performance:
  - Avoid creating many small files
    - For parallel file systems metadata performance is limited
  - Avoid searching in huge directory trees
  - Avoid competitive directory access
    - e.g. by creating files for each task in separate subdirectories
  - If lots of files are created use stripe count 1
  - Use $TMP whenever possible
    - If data is used by one client and is small enough for local hard drives
    - This also allows to reduce compilation times
  - Change the default colorization setting of the ls command
    - alias ls='ls -color=tty‘
    - Otherwise ls command needs to contact OSS in addition to MDS

Understanding application I/O behaviour

- Application properties with impact on I/O:
  - Number and size of files
  - Access pattern, i.e. random or sequential
  - Buffer size of file operations
  - Type and number of operations (read, write, create, delete, stat)
  - Number of clients executing the operations
- External factors with impact on application I/O:
  - Overall usage of file system components
    - Storage, OSS, MDS, networks, clients
    - Administrators can tell the current status
- For help to analyse and improve I/O performance
  - contact roland.laifer@kit.edu