National Aeronautics and Space Administration

Automatically Encapsulating HPC Best Practices Into Data Transfers

Paul Z. Kolano
NASA Advanced Supercomputing Division
paul.kolano@nasa.gov

Outline of Presentation

- Introduction
- Transport tuning and selection
- Global resource management
- File system optimization
- Conclusions

Introduction

- Data transfers are part of life in HPC environments
  - Finite storage capacity
    - Transfer to cheaper tape storage
      - Back up existing data
      - Make room for new data
    - Transfer from tape storage to reprocess old data
    - Transfer between file systems to fix imbalances
  - Finite computational capacity
    - Transfer from off-site systems with cheaper pre-processing
    - Transfer to off-site systems for cheaper post-processing

Introduction (cont.)

- User transfer concerns
  - Ease to use, integrity, turnaround time
- Administrator and owner transfer concerns
  - Environment stability, cost effectiveness
- These items can conflict with each other
  - Easy to use tools or those ensuring integrity may not be fast
  - Easiest file structure may degrade tape performance
  - Fastest turnaround time may lead to resource exhaustion
- Takes HPC expert to reconcile conflicts
  - Understands and applies accepted best practices to achieve fast and efficient verified transfers without impact on stability

Goal

- Let scientists focus on science without wading through documentation on transfer best practices
  - Specify transfers in simplest, naive fashion
    - Source and destination
- Provide tool to perform transfer as if scientist were HPC expert
  - Choose appropriate tools and optimize for best performance
  - Fully utilize available resources without starving other users
  - Manage files appropriately by file system type to ensure efficient access by later and/or behind the scenes processes

Shift: Self-Healing Independent File Transfer

- Satisfies user requirements
  - Simple cp/scp syntax for local/remote transfers
  - End-to-end integrity via checksums and sanity checks
  - High speed via transport selection/tuning and automatic parallelization
- Satisfies administrator and owner requirements
  - Helps prevent resource exhaustion that leads to environment instability
    - Global throttling to allocate resources fairly
    - Load balancing to avoid highly loaded hosts
    - Automatic striping to avoid imbalanced disk utilization
  - Helps prevent wasted resources that impact cost effectiveness
    - Allows easy utilization of idle resources
    - Reduces wasted CPU cycles during jobs due to inefficient disk I/O
    - Prevents issues leading to inefficient tape I/O

Shift: Self-Healing Independent File Transfer (cont.)

- Used in production at NASA's Advanced Supercomputing Facility for over 3.5 years
  - User transfers across local/LAN/WAN
  - Disaster recovery backups to/from remote organizations
  - Rebalancing entire multi-PB Lustre file systems
  - etc.
- Facilitated transfers of over 14 PB in past year
  - 8 PB local transfers
  - 4 PB LAN transfers
  - 2 PB WAN transfers
- "I used to hate archiving data - now I almost look for a reason to archive something" –Shift user

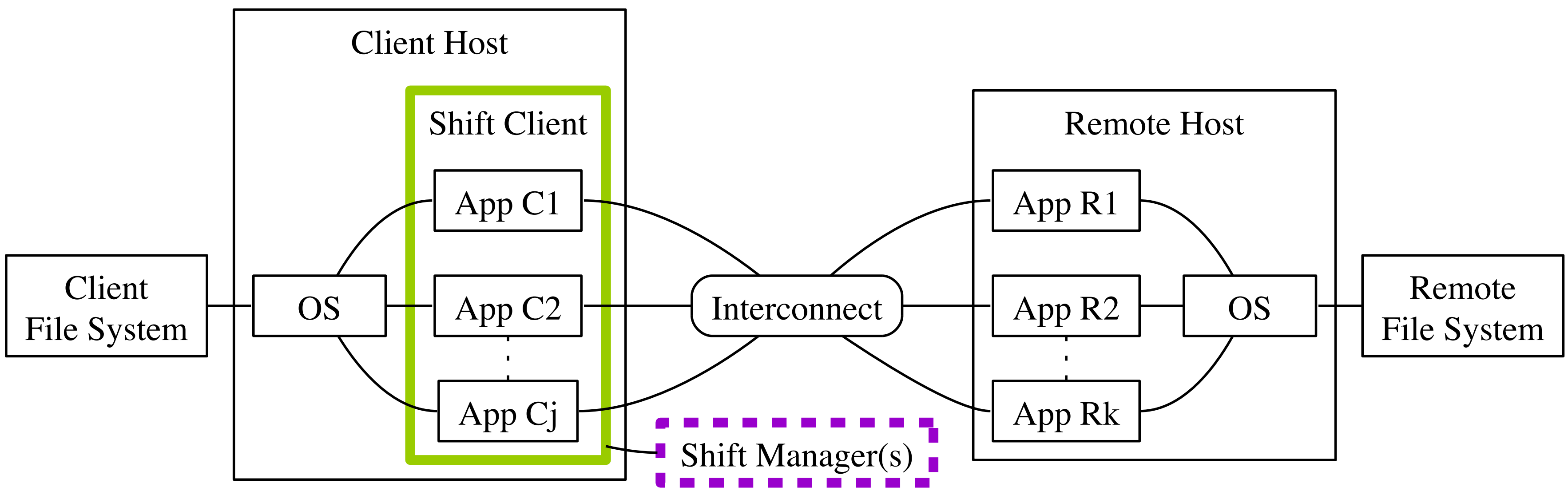
Shift Interface

```
> shiftc --create-tar /nobackup/user1/dataset1 archive1:dataset1.tar
Shift id is 36
Detaching process (use --status option to monitor progress)

> shiftc --status
id | state |         dirs |        files |      file size |  date |       run |     rate
   |       |         sums |        attrs |       sum size |  time |      left |
---+-------+--------------+--------------+----------------+-------+-----------+---------
34 | error |          0/0 | 23121/23121 | 39.5TB/39.5TB | 10/02 | 2d14h32m5s |  175MB/s
   |       |  46222/46242 | 23111/23121 |      79TB/79TB | 10:26 |           |
35 | done  |          1/1 |   5131/5131 |  303GB/303GB | 10/05 |     1m35s | 3.19GB/s
   |       |  10262/10262 |   5132/5132 |  605GB/605GB | 12:28 |           |
36 | run   |        24/24 | 26656/26656 | 1.78TB/1.78TB | 10/06 |  2h48m37s |  176MB/s
   |       |  15463/53312 |    10/26684 | 1.02TB/3.56TB | 12:11 |  1h47m55s |
```

Shift Components

- Command-line client
  - Performs file operations and reports results to manager
- Command-line manager
  - Invoked by clients to track operations and parcel out work



Outline of Presentation

- Introduction
- Transport tuning and selection
- Global resource management
- File system optimization
- Conclusions

Transport Tuning and Selection

- Shift includes built-in local/remote transports and checksum capabilities
  - Fully functional out of the box
  - Perl-based equivalents of cp, sftp, fish, m(d5)sum
- Shift calls higher performance tools when available
  - bbcp, bbftp, gridftp, mcp, rsync, msum
  - Knows how to construct command-lines and parse output
- Tune transports for optimal performance
- Select transports based on transfer characteristics

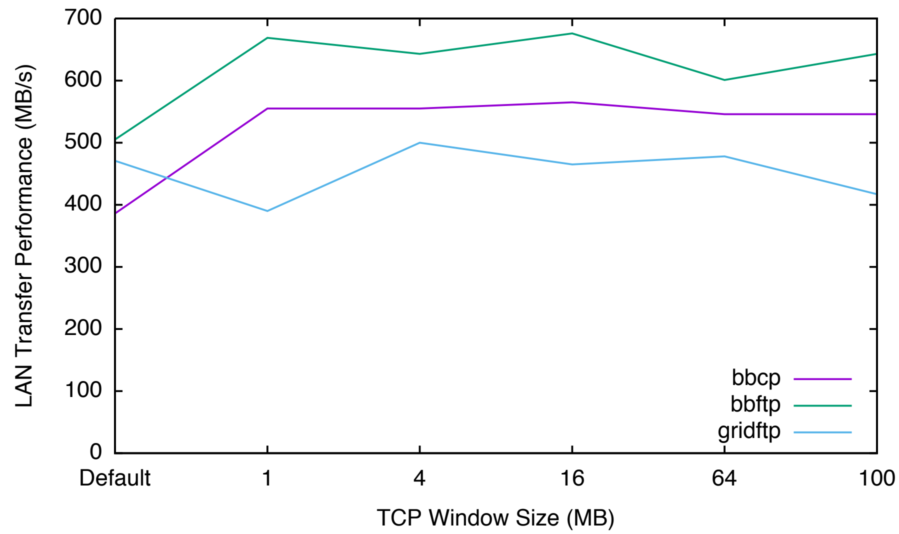
Transport Tuning

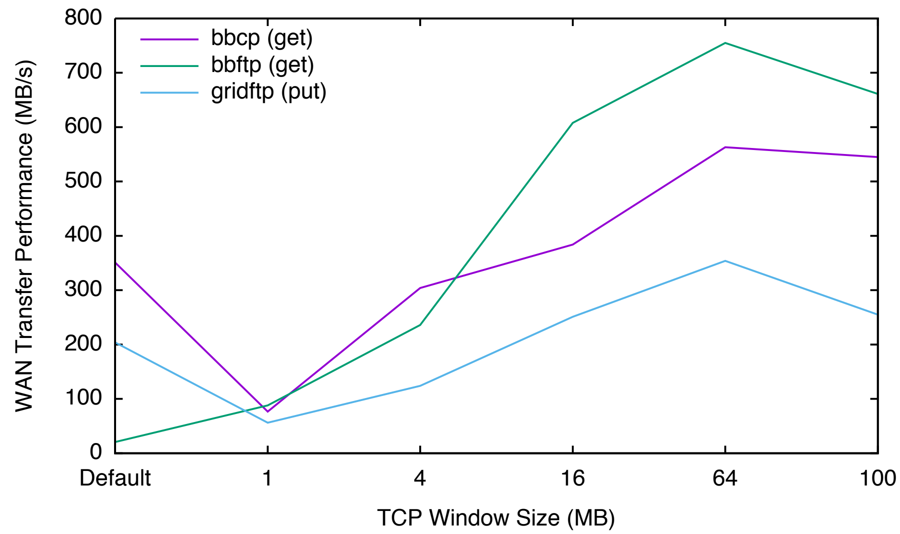
- TCP-based transports
  - bbcp, bbftp, gridftp
  - Choose TCP window size
- Transports with internal parallelism
  - TCP streams (bbcp, bbftp, gridftp) or threads (mcp, msum)
  - Choose appropriate level of parallelism
- SSH-based transports
  - fish, rsync, sftp-perl
  - Choose fastest SSH cipher and MAC algorithm

TCP Window Size Tuning

- TCP window is amount of data sender or receiver willing to buffer while waiting for acknowledgment
- Optimal value is bandwidth delay product (BDP)
  - bandwidth * round-trip time
- Constrained by configured operating system limits
  - e.g. Linux net.core.[wr]mem_max
  - Single stream only achieves bandwidth if limit at least BDP

TCP Window Size Tuning (cont.)



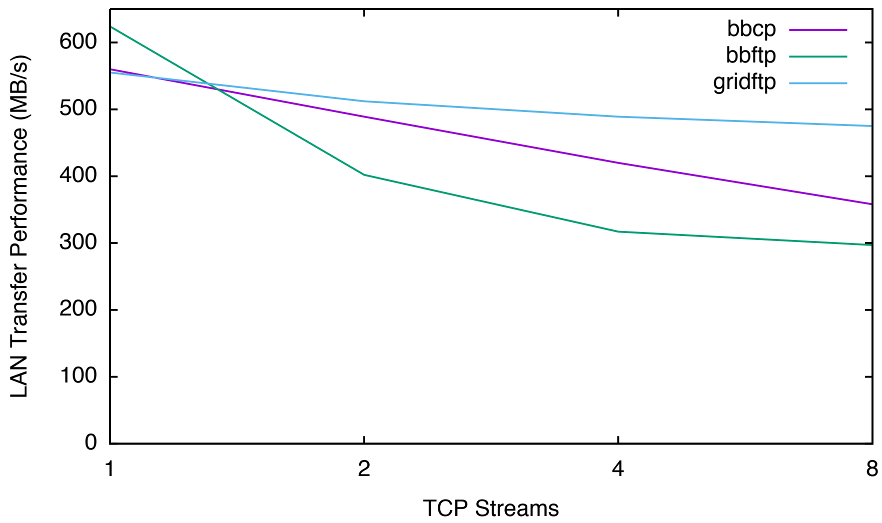


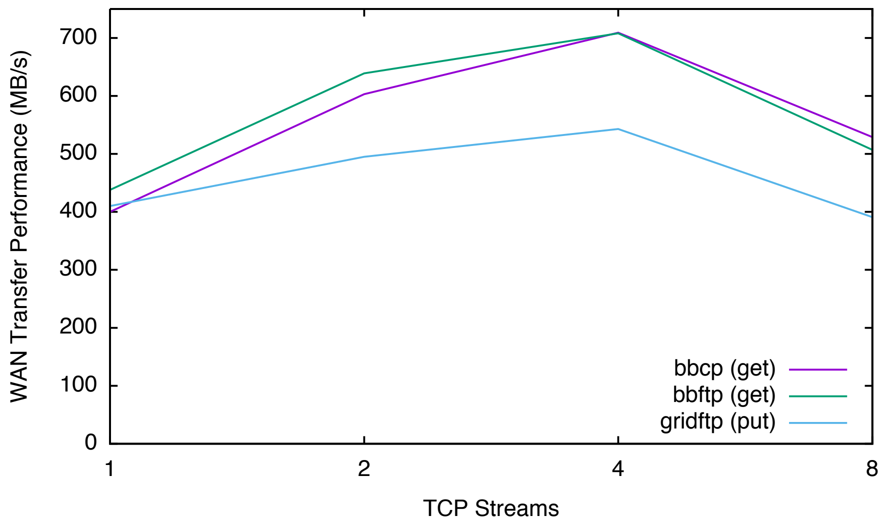
- Shift determines latency using icmp/echo/syn ping
- Shift guesses bandwidth based on network type and client hardware if not given via --bandwidth
  - Bandwidth difficult to compute a priori
- Chooses window size up to operating system limit

Transport Parallelism Tuning

- Number of streams in TCP-based transports
  - Overcome improperly configured TCP window maximums
  - Overcome improperly specified TCP window
  - Overcome interference by cross traffic
- Number of threads in mcp and msum
  - Take advantage of excess resource capacity on one host

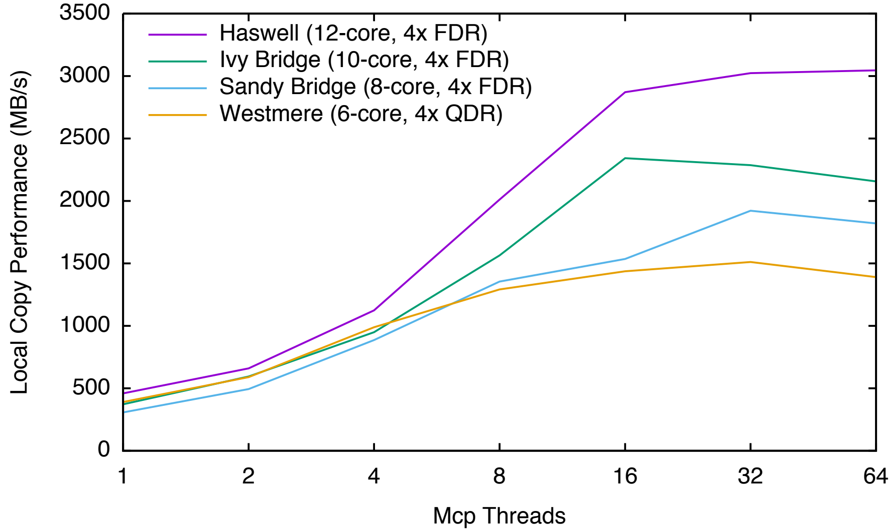
Transport Parallelism Tuning (cont.)





- Shift chooses streams based on bandwidth available beyond operating system window limit
- A minimum value can be configured for LAN/WAN to help overcome cross traffic

Transport Parallelism Tuning (cont.)

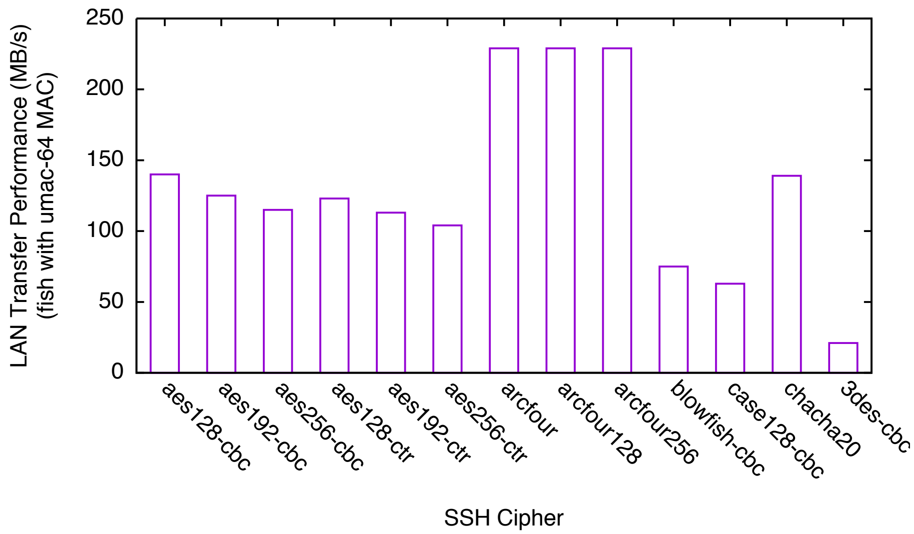


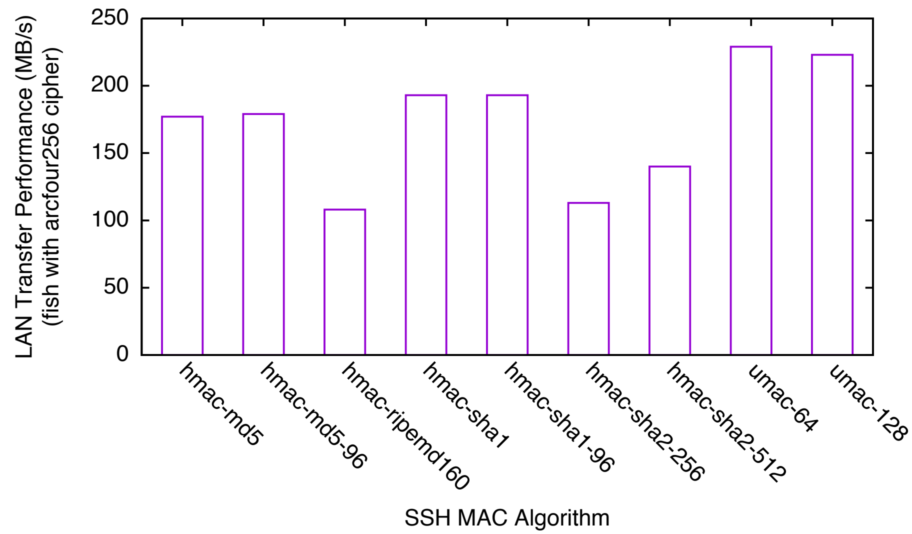
- Threads can be centrally configured on the manager
- High thread counts can induce high load on shared resources
  - Intentionally set lower than optimal at NAS due to high load on archive front-ends

SSH Cipher and MAC Algorithm Tuning

- SSH-based transports use SSH pipe to communicate
  - Performance directly correlated to SSH performance
- SSH does not expose TCP window settings
  - HPN SSH patches can be used for better window handling
- Main SSH tuning parameters available
  - Encryption algorithm
  - Message authentication code (MAC) algorithm

SSH Cipher and MAC Algorithm Tuning (cont.)



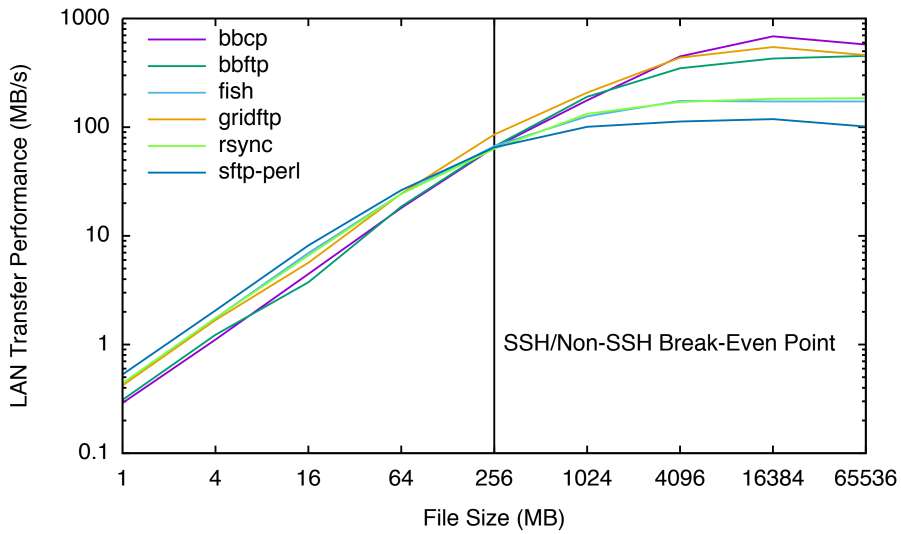


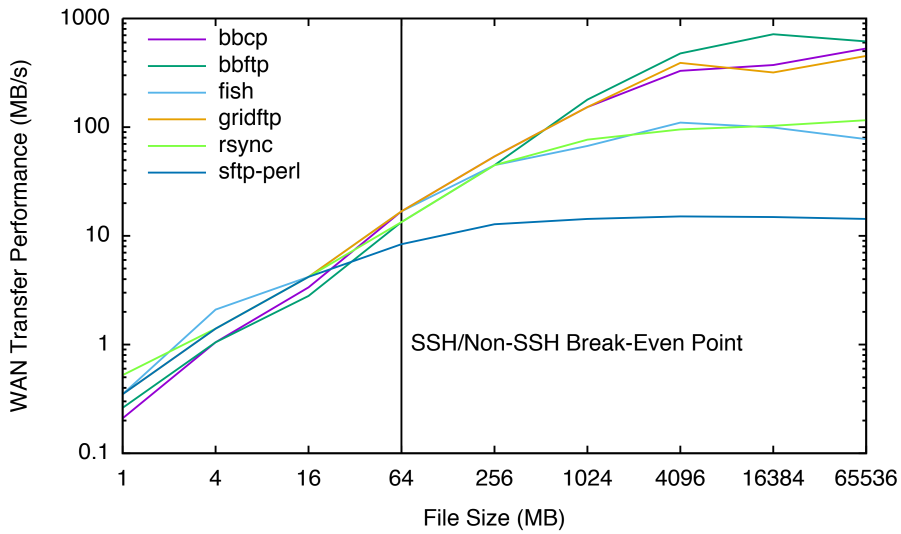
- Shift allows preferred cipher/mac order to be centrally configured
- Availability checked on client host before transfer

Transport Selection

- Different transports have different strengths and weaknesses
- Supporting multiple transports provides opportunity to vary transport across each batch of files within transfer

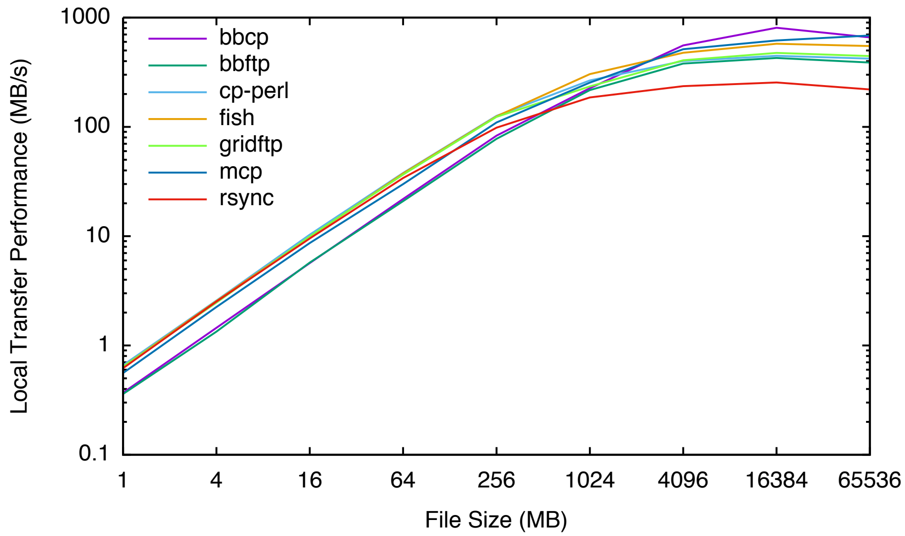
Transport Selection (cont.)





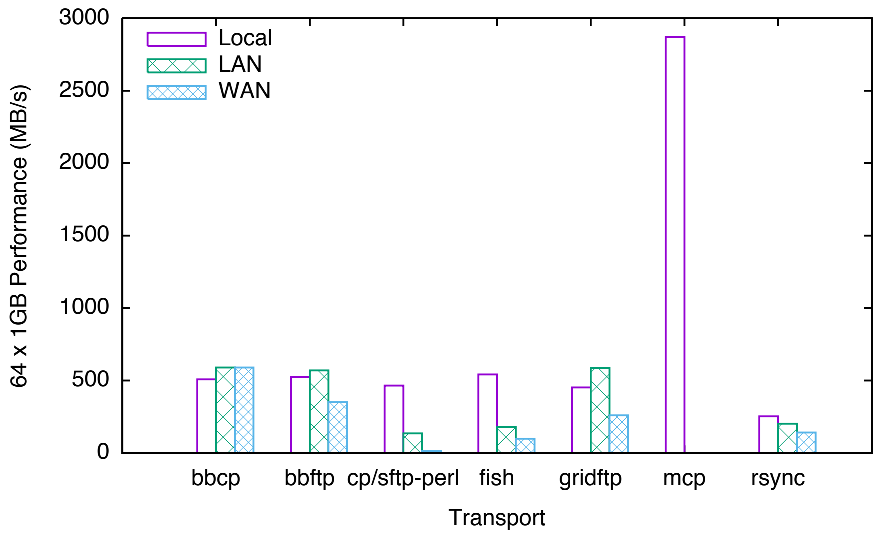
- Using a single transport does not achieve maximum performance
- Shift's support of multiple transports allows it to use the optimum transport for each batch of files

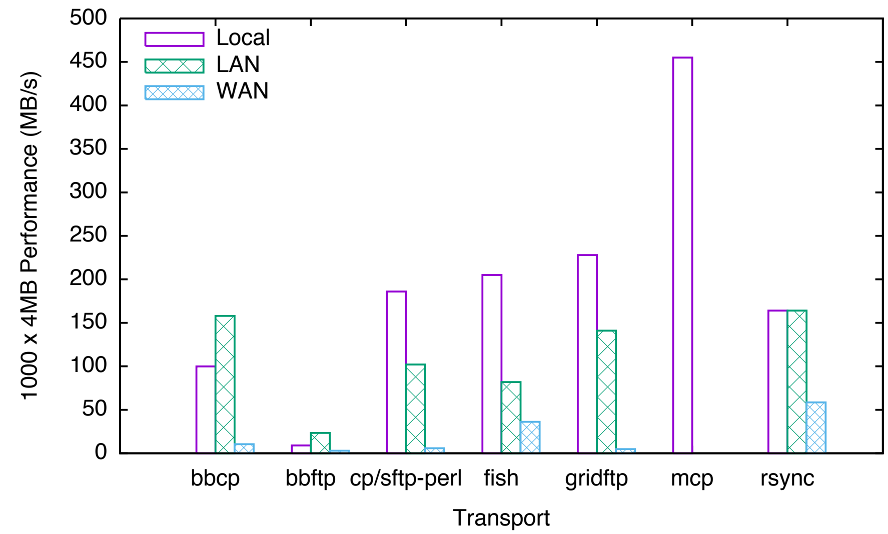
Transport Selection (cont.)



- Traditionally remote transports also perform well for local transfers

Transport Selection (cont.)





- Shift has configurable small file thresholds
  - Preferred local/LAN/WAN transports above/below
- Transport chosen using avg. size of each batch

Outline of Presentation

- Introduction
- Transport tuning and selection
- Global resource management
- File system optimization
- Conclusions

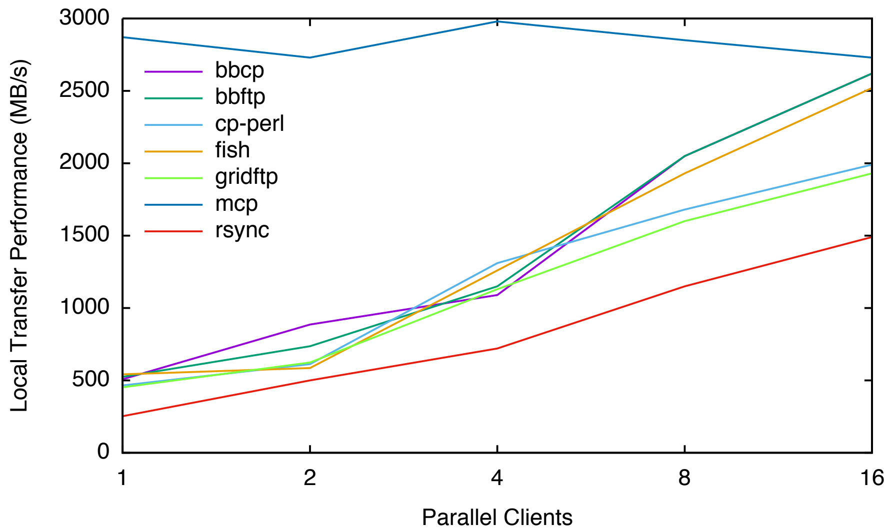
Global Resource Management

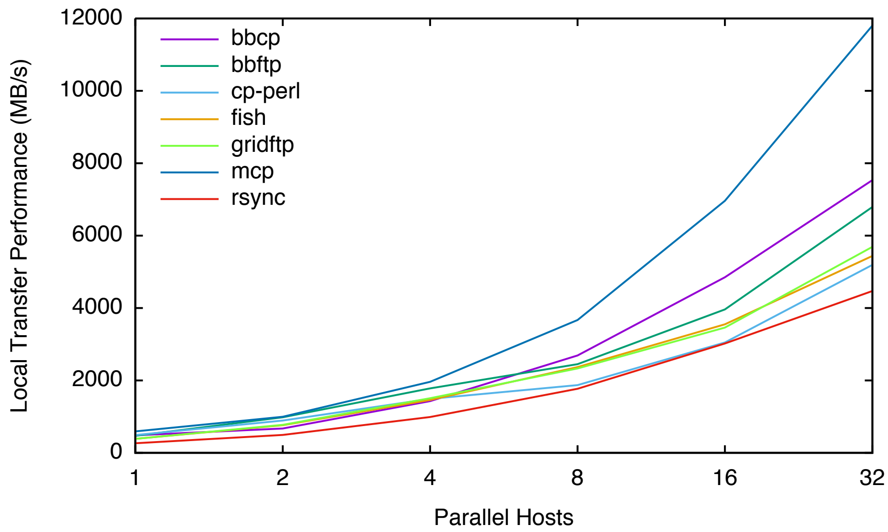
- Transfer parallelization
  - The only way to maximize performance in HPC environments is to use multiple resources at once
- Global throttling
  - If everyone tries to run at the maximum rate possible, everyone loses
- Load balancing
  - Avoid resource exhaustion on individual hosts

Transfer Parallelization

- Single host may have excess capacity
  - Transport does not have its own parallelism
  - Single client cannot fully utilize host's resources
- Full HPC environment may have excess capacity
  - Single system bottlenecks
  - Aggregate resources of many hosts
- Two complementary forms of parallelization
  - Multiple clients running on a single host
  - One or more clients running on multiple hosts

Transfer Parallelization (cont.)

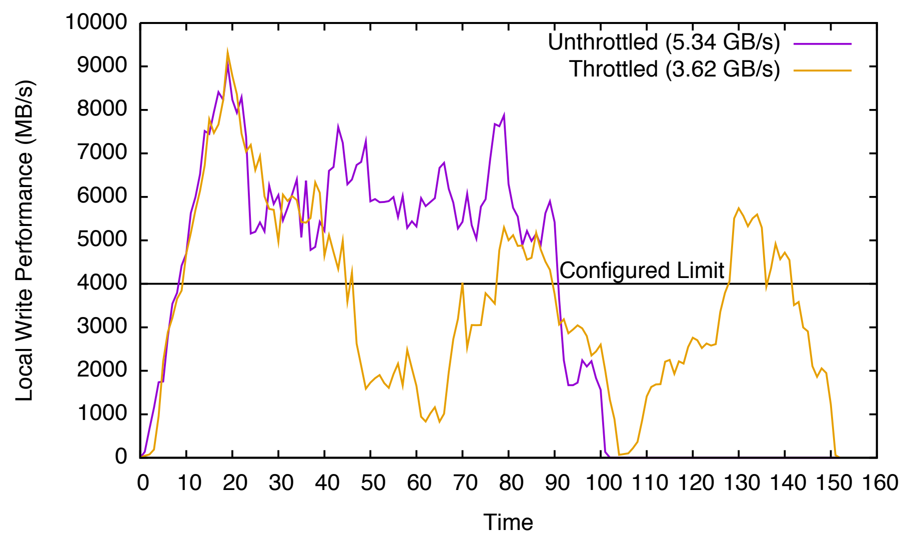




- Shift derives file system equivalency from mount information to determine parallelization candidates
- Shift can automatically parallelize all transfers using centrally configured clients/hosts values
- Can trivially run across allocated compute nodes
  - e.g. --host-file=$PBS_NODEFILE

Global Throttling

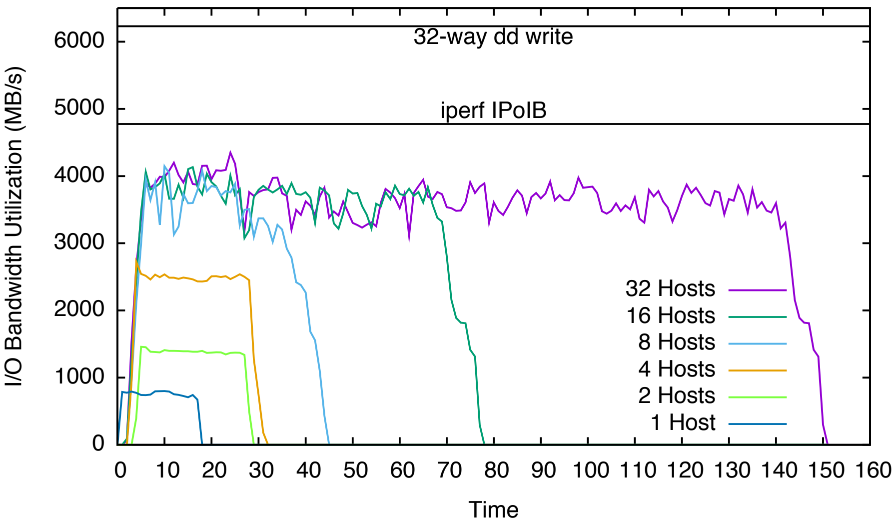
- Support transfers at full speed when excess capacity exists
- Prevent resource exhaustion when systems are busy



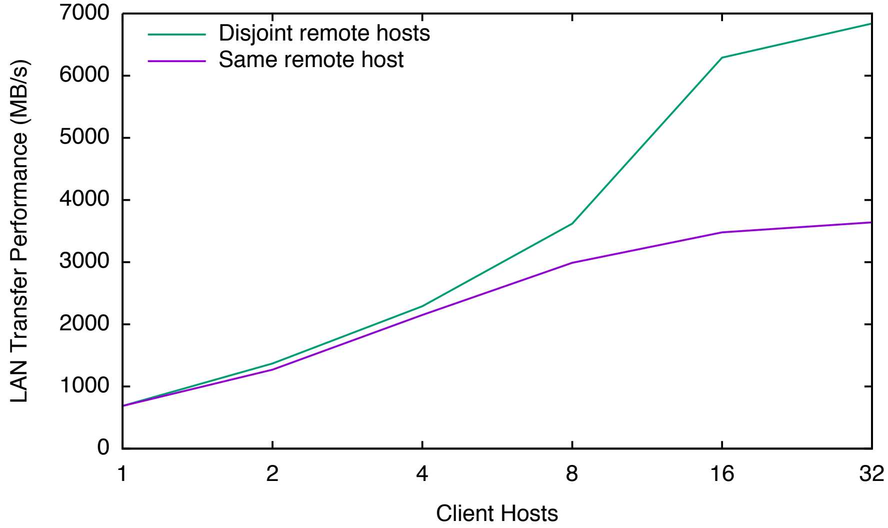
- Shift supports throttling limits on CPU %, disk usage, I/O rate, and network rate
  - Limits can be global, per user, per host, per file system
- Transfers directed to sleep until average reaches (transfer's fair share of) user's fair share of limit

Load Balancing

- Single host resource limitations impact maximum transfer speed of all processes on that host
- Less loaded hosts have more resources available for new transfers



Load Balancing (cont.)



- Shift load balances during host parallelization
  - Picks least loaded hosts when spawning clients
- Shift load balances during remote transfers
  - Remote host switched to least loaded
  - Overlap prevented between parallel clients

Outline of Presentation

- Introduction
- Transport tuning and selection
- Global resource management
- File system optimization
- Conclusions

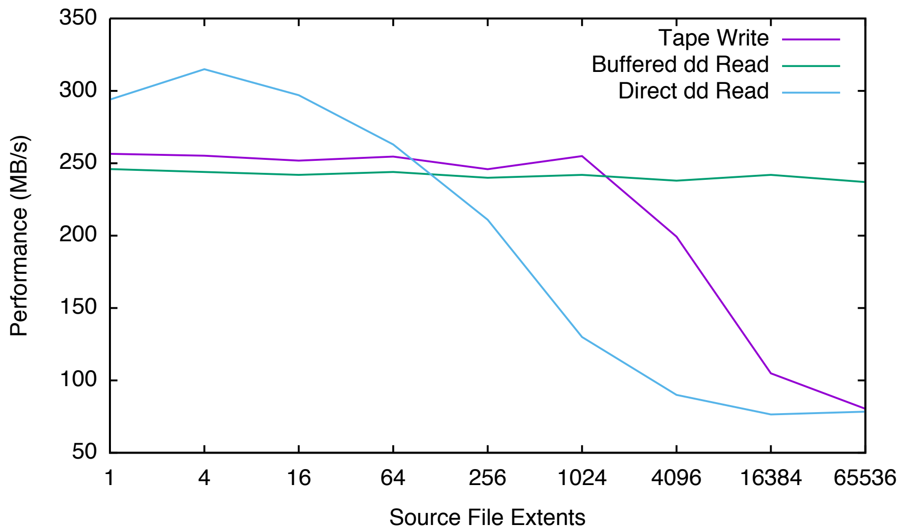
File System Optimization

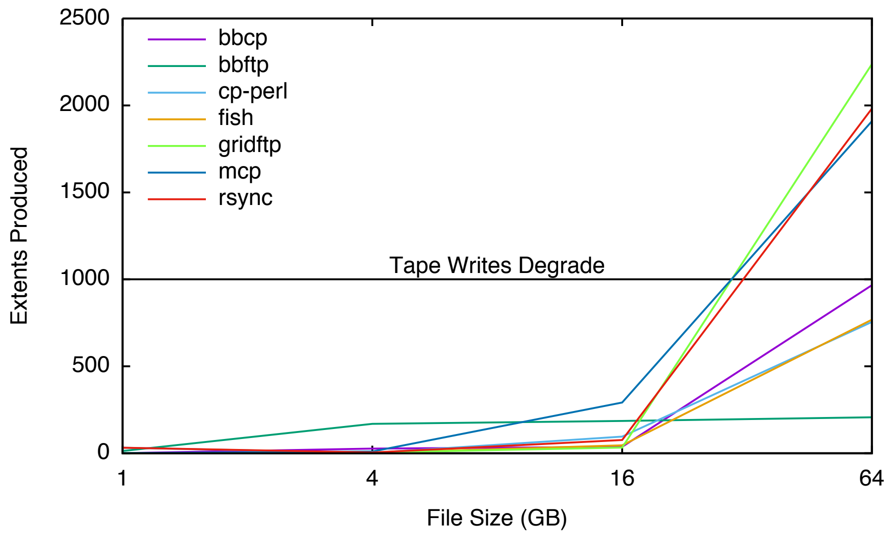
- Different file systems have different idiosyncrasies
  - Can impact performance, stability, and operating cost
  - Not always directly visible to users
- Tape optimization
  - File extents impact tape write speed
  - Sequential retrieval results in inefficient tape movement
- Tar creation/extraction
  - Tape I/O is inefficient with small files
- Lustre striping
  - Too few stripes creates later I/O inefficiencies

Tape Optimization

- Tape-backed file systems have limited parallelism due to large slow-moving physical components
- Inefficient access by one user can have major impact on all others
  - Large number of file extents degrades write speed
  - Sequential file retrieval inhibits minimization of internal tape movement

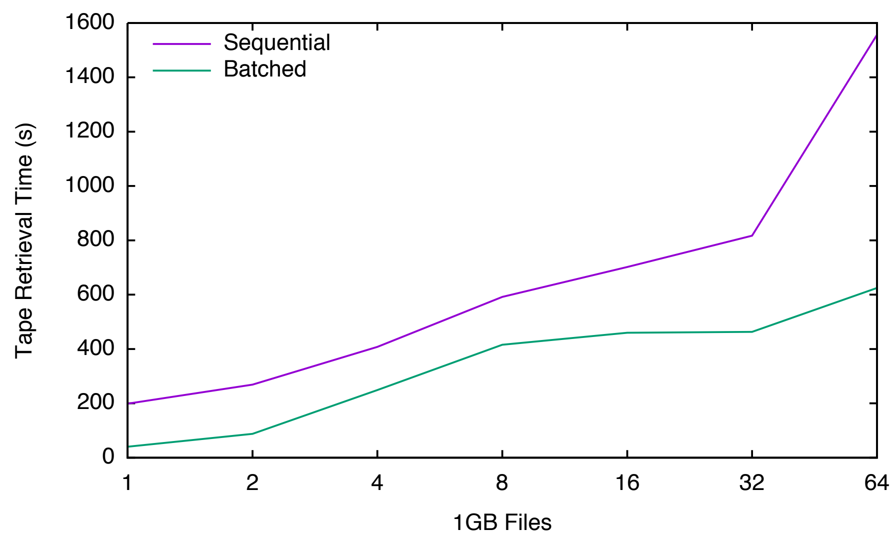
Tape Optimization (cont.)





- Shift can be configured to preallocate files below a given sparsity
  - Minimize extents for most common regular files
  - Minimize disk usage for large sparse files

Tape Optimization (cont.)

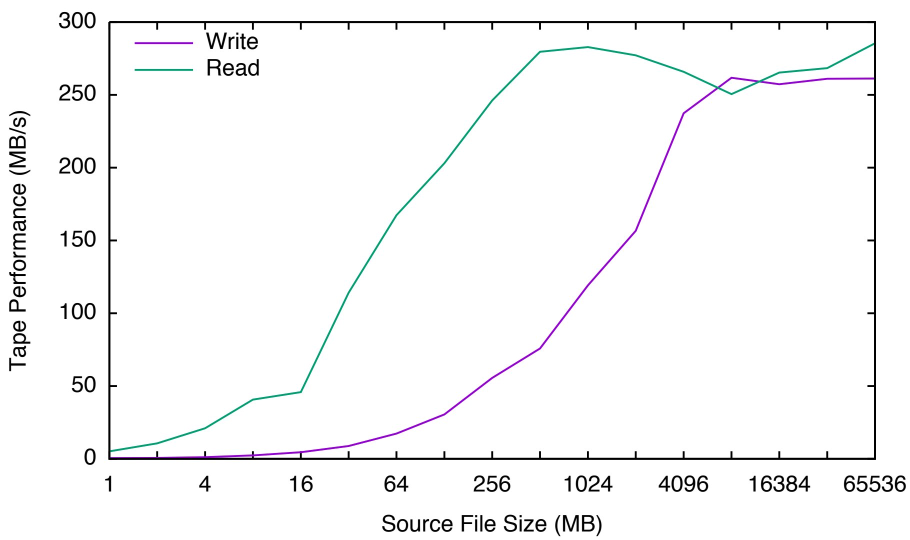


- Files automatically retrieved from tape if offline when accessed
- May be accessed in different order than stored on tape

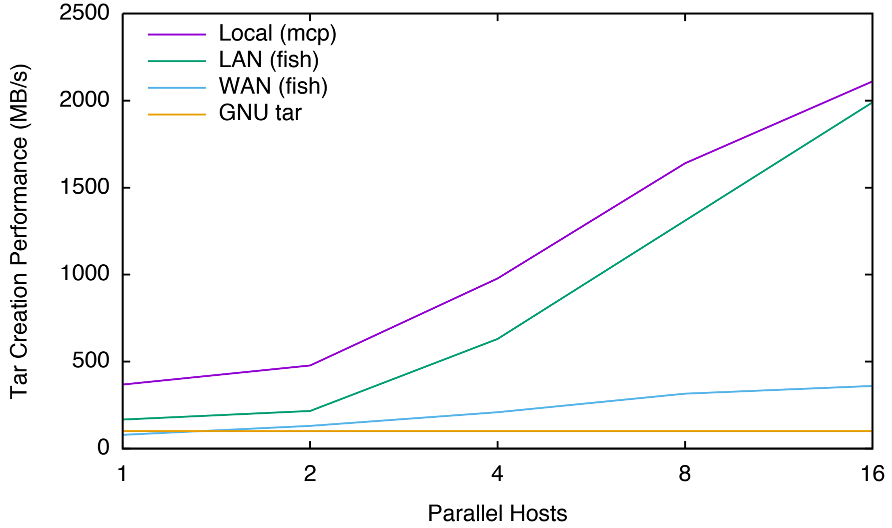
- Shift automatically initiates a retrieval of all source files on SGI DMF file systems
- Retrieval initiated again for files in each batch in case files pushed offline before being transferred

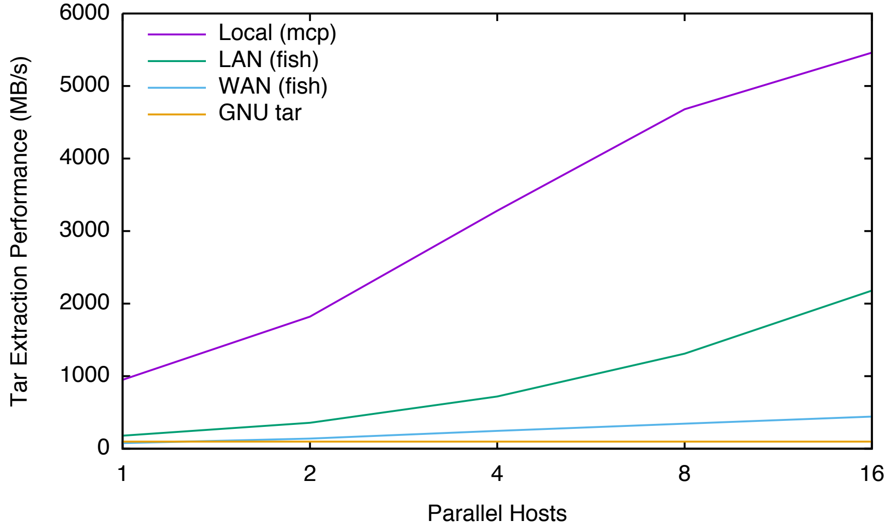
Tar Creation/Extraction

- Users prefer archiving data in original hierarchy
  - Each file accessed as needed
- Small files impact stability
  - Not migrated below certain size so consume limited disk
  - Tape I/O is inefficient
- Normally use tar files
  - Tar is <u>very</u> slow (100 MB/s)
  - Must retrieve to view contents
  - No assurance of integrity
  - Retrieve entire tar for one file
  - May not have quota to create



Tar Creation/Extraction (cont.)



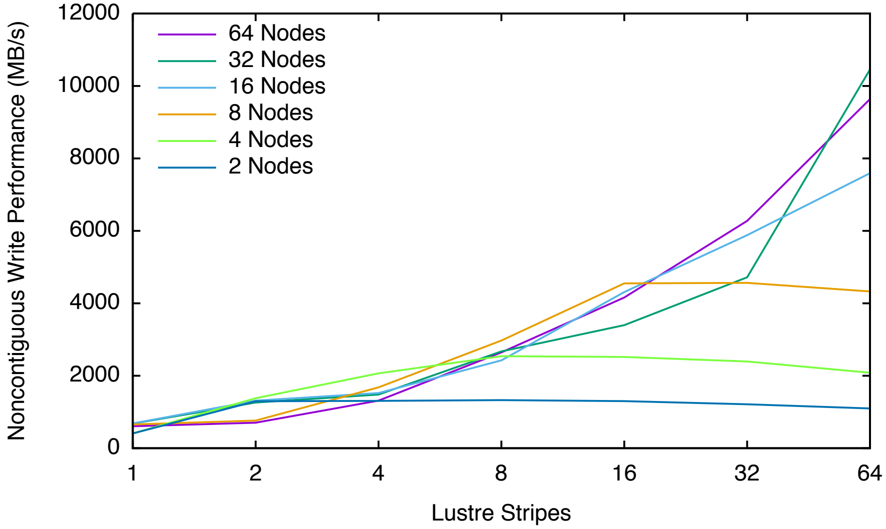


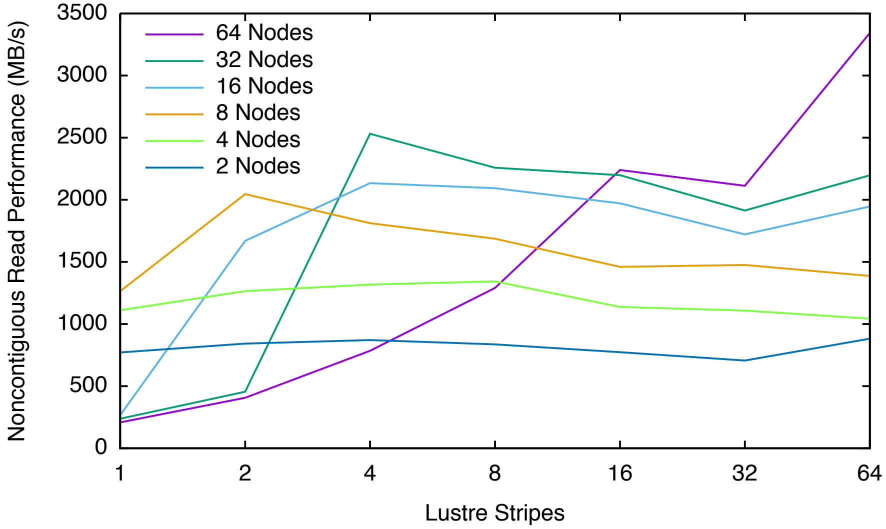
- Shift has built-in tar creation/extraction
  - Uses high performance transports and parallelism
  - Automatic creation of index files to see contents
  - Integrity verification of every file added/extracted
  - Automatic split of tar files at configurable size
  - Direct creation/extraction over network without use of quota

Lustre Striping

- Striping must be set before a file is first written
- Higher stripe count
  - More I/O bandwidth available for large files
  - More balanced distribution of large files over OSTs
- Lower stripe count
  - Less contention during metadata operations
  - Less wasted space for small files
- Striping can only be changed by copying file

Lustre Striping (cont.)





- Shift automatically stripes files according to size
  - Increases write performance during parallel transfers
  - Increases read performance during later job access
    - Reduces wasted CPU cycles due to I/O
- Preserves non-default striping when applicable

Conclusion

- Shift is an automated transfer tool that encapsulates HPC best practices to achieve better performance while preserving stability of HPC environments
  - Centralized configuration in manager component allows policies to be changed globally across all clients
  - New best practices can be incorporated transparently without user in the loop
- Shift is open source and available for download
  - http://shiftc.sourceforge.net
- Contact info
  - paul.kolano@nasa.gov
- Questions?