

BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Parallel Synchronization of Multi-Pebibyte File Systems

Andy Loftus (aloftus@Illinois.edu)



GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION



Outline

1. Motivation
2. Analysis of file system synchronization
3. Architecture of psync
4. Complexities arising from parallelization
5. Preliminary performance review
6. Future work

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

UNIVERSITY OF MICHIGAN | NSF | NECA | GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION | CRAY

Motivation

- Upgrade filesystem firmware
- Minimize downtime



Home
120M inodes
220 TiB

Steps:

1. Evacuate Home
2. Upgrade Home
3. Repopulate Home
4. Repeat for Projects

Projects
463M inodes
1.2 PiB



Lustre Ecosystem 2016 3

Key points:

+ Firmware upgrade takes 5 days.

+ Also allowing several weeks testing time

+ Already have the storage capacity to store Home elsewhere and keep using the filesystem in the meantime.

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF
NECA
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

Analysis of File System Synchronization

rsync	Custom parallel file copy tool
<ul style="list-style-type: none">+ Already exists+ Correct - Serial- Slow	<ul style="list-style-type: none">- Doesn't exist- Ensure correctness + Parallel+ Fast

Goal:
Parallel
rsync

Lustre Ecosystem 2016 4

Key point:

+ rsync already exists and does the right thing. Just need to parallelize it!

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

UNIVERSITY OF MICHIGAN | NSF | NASA | GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION | CRAY

Analysis of File System Synchronization

<h3>rsync</h3> <ul style="list-style-type: none">• Just files (no dirs)• Properly handles all file types• Already handles metadata	<h3>Parallel Management</h3> <ul style="list-style-type: none">• Parallel file copies (rsync)• Parallel tree walk
--	--

Lustre Ecosystem 2016 5

Keypoint:

- + reduce rsync to operate on the smallest, atomic unit of work: a file
- + Custom code will manage:
 - rsync's in parallel
 - file system traversal in parallel



Architecture of psync - Overview

- Task Design
 - SyncFile - (rsync a file)
 - SyncDir - (walk the filesystem)
- Distributed Task Queue
 - Python Celery
 - RabbitMQ
 - Redis (centralized logging)

Lustre Ecosystem 2016 6

Key points:

+ Two task types:

- SyncDir (walk file system)
- SyncFile (rsync an individual file)

+ Use a distributed task queue to schedule and run tasks in parallel

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF
NECA
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

Architecture of psync - Tasks

SyncDir

- Make target
- Scandir source and target
- Delete from target
- Dirs: new SyncDir task
- Files: new SyncFile task

SyncFile

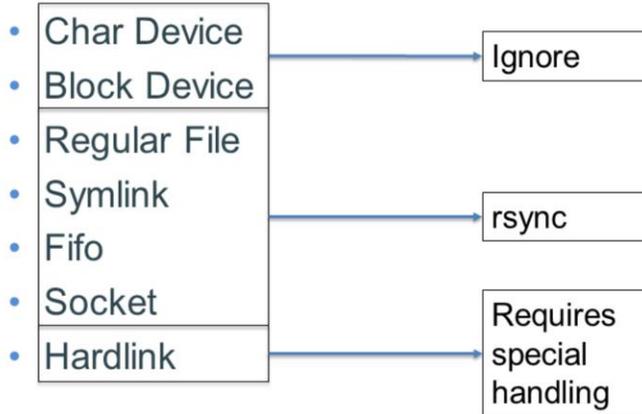
- Do a little bit of work to skip matching files:
 - Compare metadata
 - If copy needed, compare size
 - If size > 64MiB,
 - Invoke dd
 - Rsync

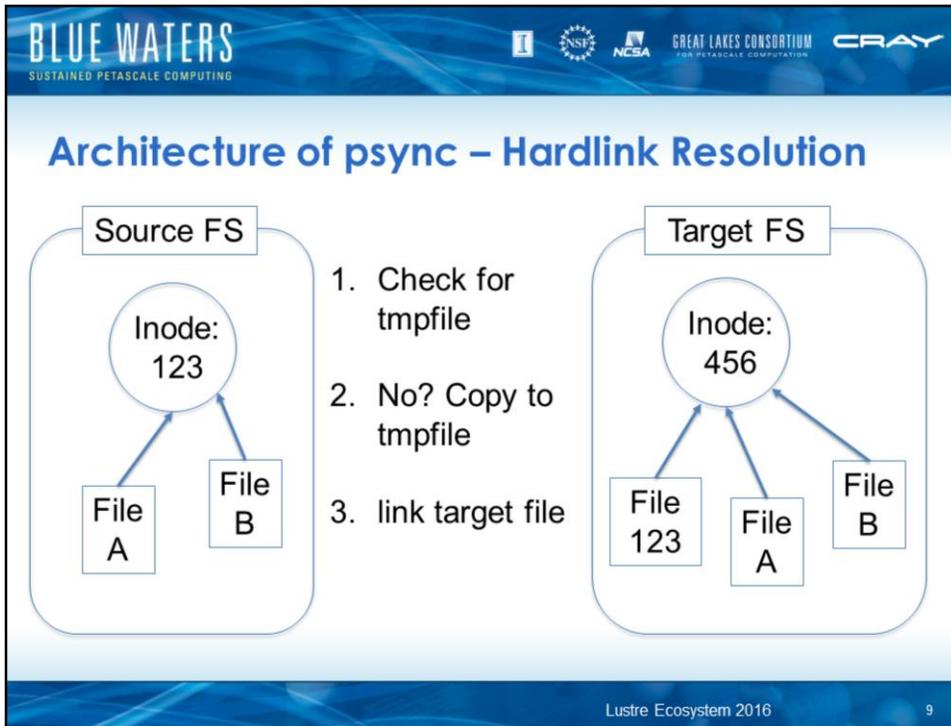
Lustre Ecosystem 2016 7

Keypoints:

- + syncdir handles target deletes inline
- + all subdirectories can be processed in parallel
- + syncfile checks metadata so only invoke rsync if necessary
- + dd is more efficient to copy "larger" files because blocksize can be increased
 - cannot affect rsync blocksize when copying between local file systems
- + rsync is always invoked if data is copied or if metadata needs update

Architecture of psync – Special Files





Keypoints:

+ tempfile name == source file inode number (FID)

+ tempfile remains until end of psync

+ Necessary for all files when sync'ing a live filesystem (hardlinks can be created at any point in time)

- Programming is special in that the tasks we undertake are simple in principle, but difficult in practice.

- <http://c2.com/cgi/wiki?RubberDucking>

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF
NSA
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

Complexities Arising from Parallelization

Hardlink Race Conditions

- Two hardlinks processed simultaneously
 - Tempfile doesn't exist
 - Results in simultaneous attempts to create tmpfile
- Possible resolutions:
 - Delay and retry (still parallel)
 - Separate queue to handle hardlinks serially (like rsync)

Lustre Ecosystem 2016 11

Key point:

+ Workaround is to run another psync

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF
NCSA
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

Complexities Arising from Parallelization

Directory mtime

- Target dir must be created before files
- Files are sync'd which updates dir mtime
- Possible Resolutions:
 - File sync always fixed dir mtime
 - Use separate queue to fix dir mtime
 - File sync updates "dir needs fix" flag for parent dir

Lustre Ecosystem 2016 12

Key point:

+ Workaround: run another psync

Preliminary Performance Review

- Blue Waters – Cray XE6 compute nodes
 - Dual AMD Interlagos 6276 CPUs @ 2.3GHz
 - 64 GiB RAM
- Compute to LNET routers
 - Cray Gemini high speed network (~6 GiB/s)
- LNET routers to Lustre
 - QDR Infiniband
- Cray Sonexion 1600 filesystem appliance

Preliminary Performance Review

- TPS and MiB/s depend entirely on filesystem layout
- Initial sync time affected primarily by usage
- Re-sync time affected primarily by total inodes

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF NCSA GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION CRAY

Preliminary Performance Review

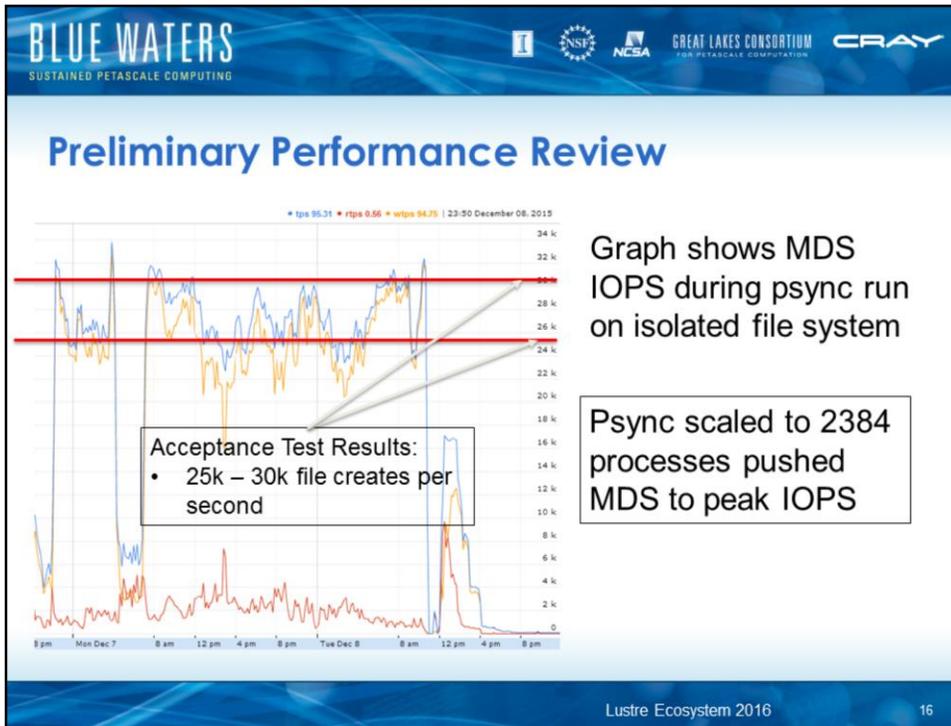
Initial Sync	Re-sync
<ul style="list-style-type: none">• 2384 procs• 398 nodes• Idle file system• 2.25% per hour<ul style="list-style-type: none">• Based on inode count• ~44 hours	<ul style="list-style-type: none">• 800 procs• 100 nodes• Live filesystem• ~16 hours<ul style="list-style-type: none">• Affected primarily by total number of inodes

Home File System – Mostly Small Files

Lustre Ecosystem 2016 15

Key Points:

- + Initial sync – data and metadata
- + Resync – mostly metadata



Keypoints:

- + synchronization of “Home” file system (ie: mainly small files)
- + syncdir (mainly dir scan) and syncfile run simultaneously
- + acceptance test results generated from mdtest

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

INSF
NECA
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

Future Work - Correctness

- Hardlink race conditions
 - Delay and retry
 - Separate queue for serial processing
- Directory mtimes
 - Separate queue to periodically “fix” directory mtimes
 - Syncfile task updates “dir needs fix” flag for parent dir

Lustre Ecosystem 2016 17

Keypoints:

+ Currently, failed hardlinks and directory mtimes will get fixed on a successive sync

Future Work - Performance

- Short circuit metadata queries
 - Check ctime (only) first
 - Only get stripe information if necessary
- Use lustrelib for lustre-specific metadata access
 - “lfs getstripe” causes two (or more) file opens



Future Work - Features

- File restriper
 - Increase stripe count for large files
- Other filesystems
 - Spectrum Scale (GPFS)
 - Fill queue from policy run
 - Generic Posix
 - Others?
- Checksums processed in separate queue

Lustre Ecosystem 2016 19

Keypoints:

- + Filesystem interaction is contained in a Python module. Create a new module for a different filesystem.
- + Checksum processed by same node that performed rsync could result in read file from cache instead of from disk
- + Checksum tasks could be isolated to nodes optimized for CPU intensive tasks

Summary

- Psync
 - Works properly
 - Known issues soon to be fixed
 - Workarounds are okay for now
 - Scales to thousands of processes
 - Limited only by:
 - MDS IOPS max load
 - Acceptable response time on live filesystem
 - Generic solution for any (Lustre) filesystem regardless of file and/or directory sizes

Thank You!

- Questions?
- Psync source
 - <https://github.com/ncsa/pylut>