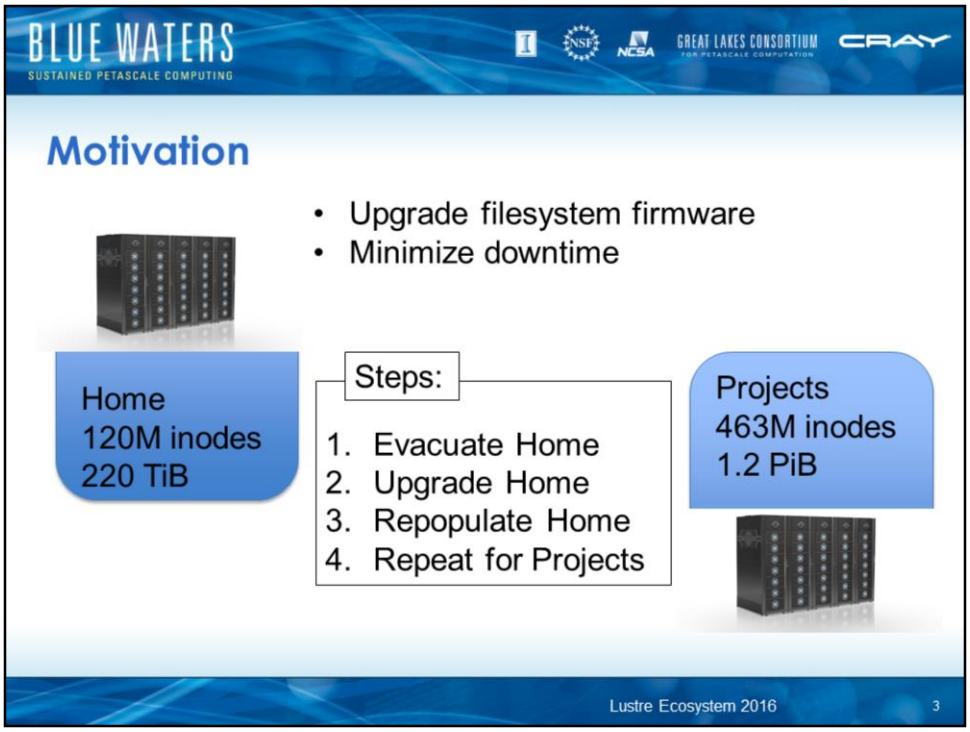

BLUE WATERS
SUSTAINED PETASCALE COMPUTING
Parallel Synchronization
of Multi-Pebibyte File Systems
Andy Loftus (aloftus@Illinois.edu)
NCSA
NSF
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
CRAY

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

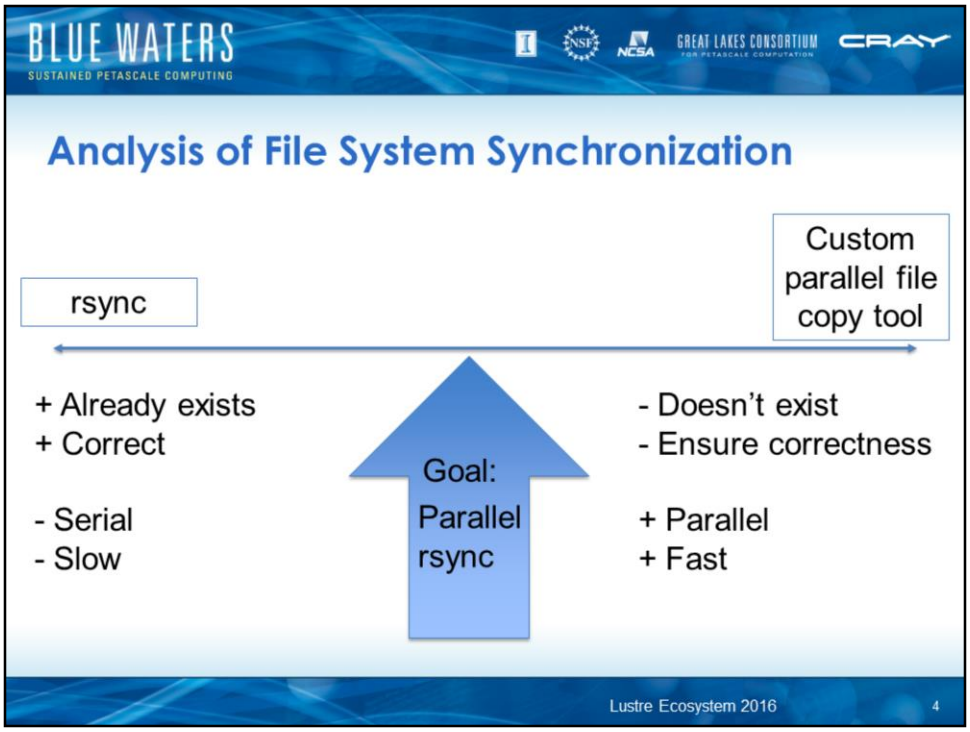


Key points:

+ Firmware upgrade takes 5 days.

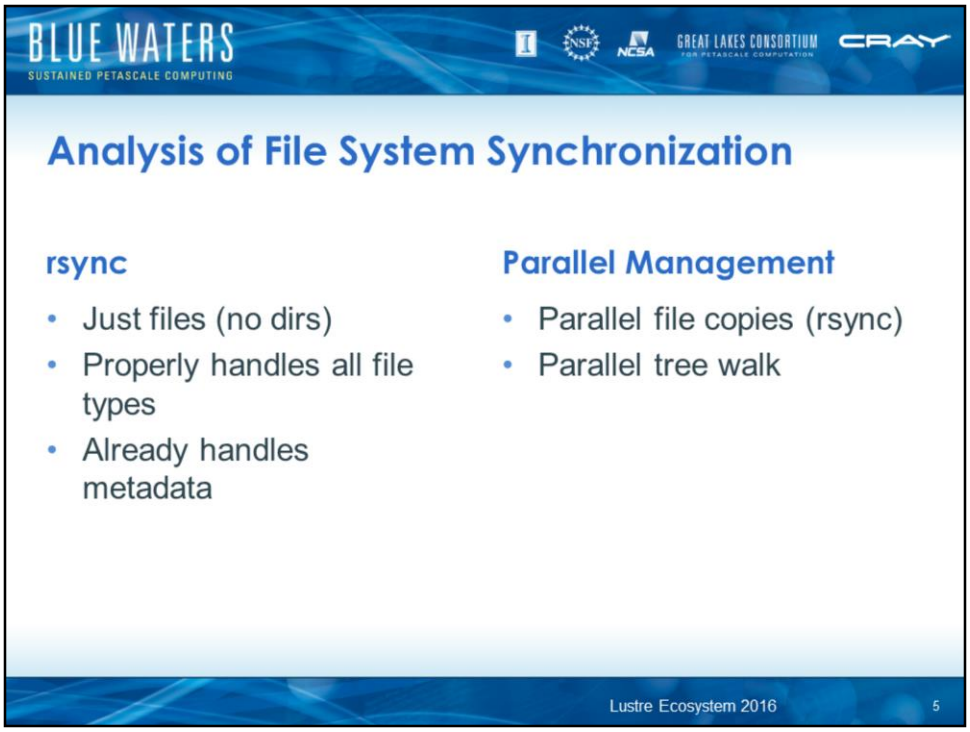
+ Also allowing several weeks testing time

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



Key point:

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



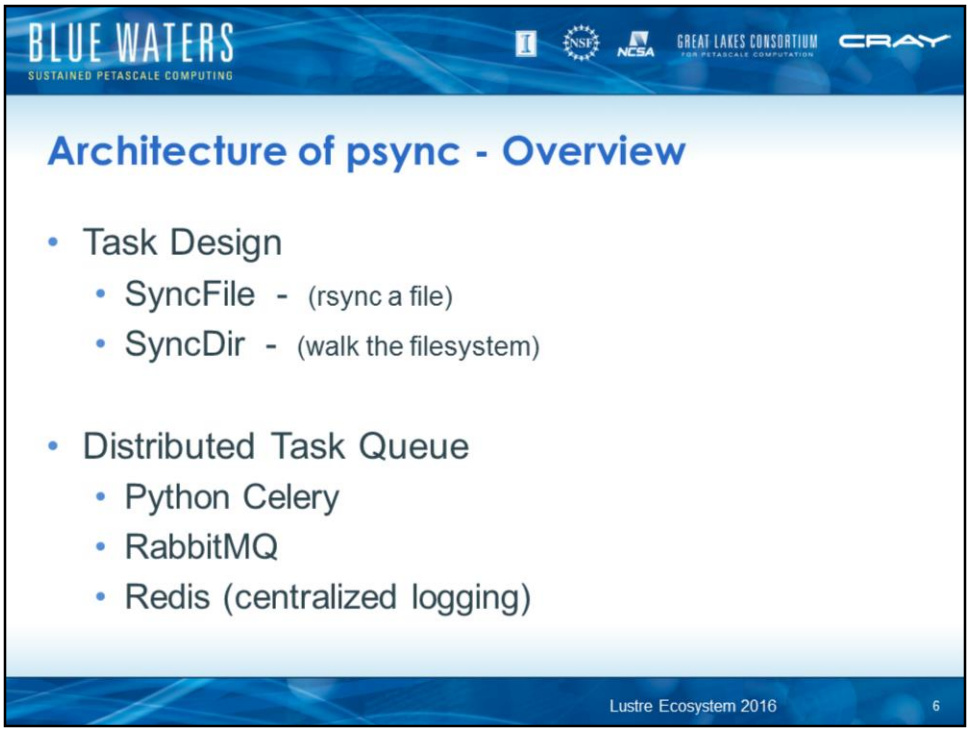
Keypoint:

+ reduce rsync to operate on the smallest, atomic unit of work: a file

+ Custom code will manage:

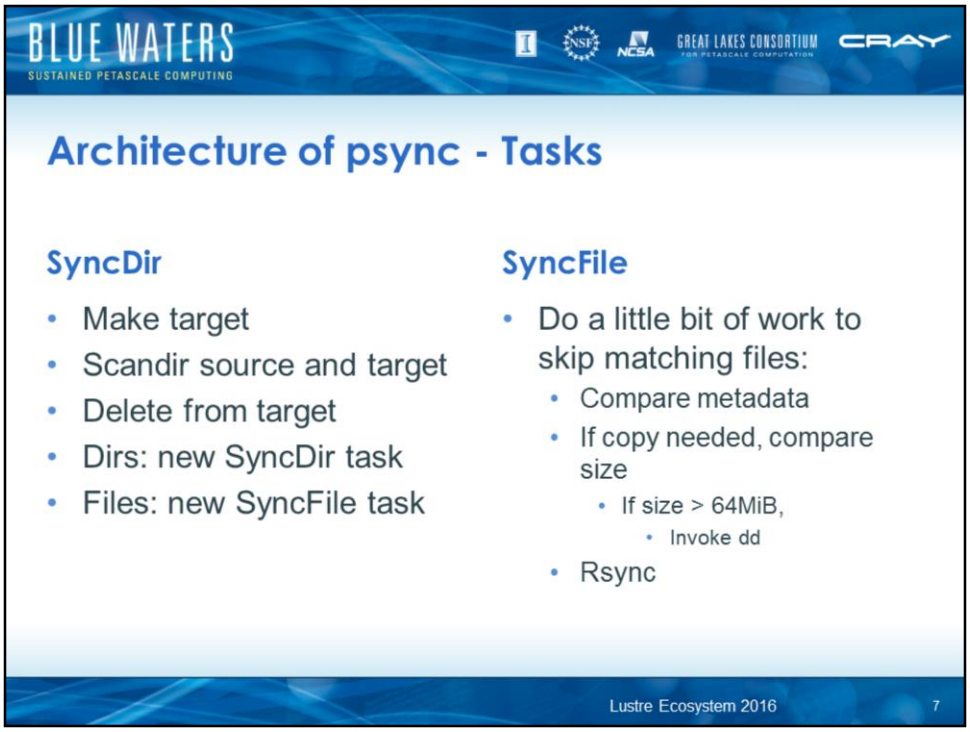
  - rsync's in parallel

  - file system traversal in parallel



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



Keypoints:

+ syncdir handles target deletes inline

+ all subdirectories can be processed in parallel

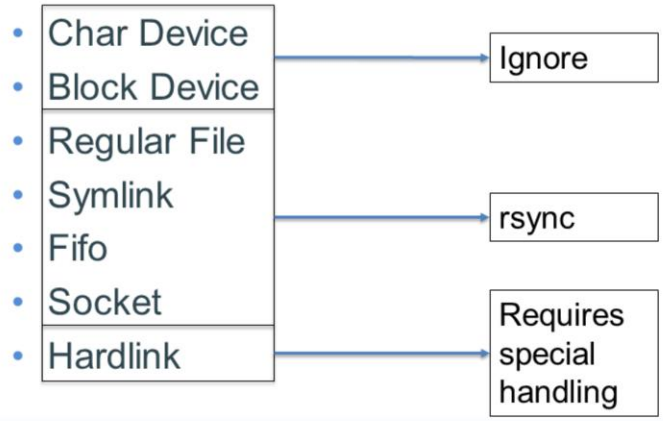
+ syncfile checks metadata so only invoke rsync if necessary

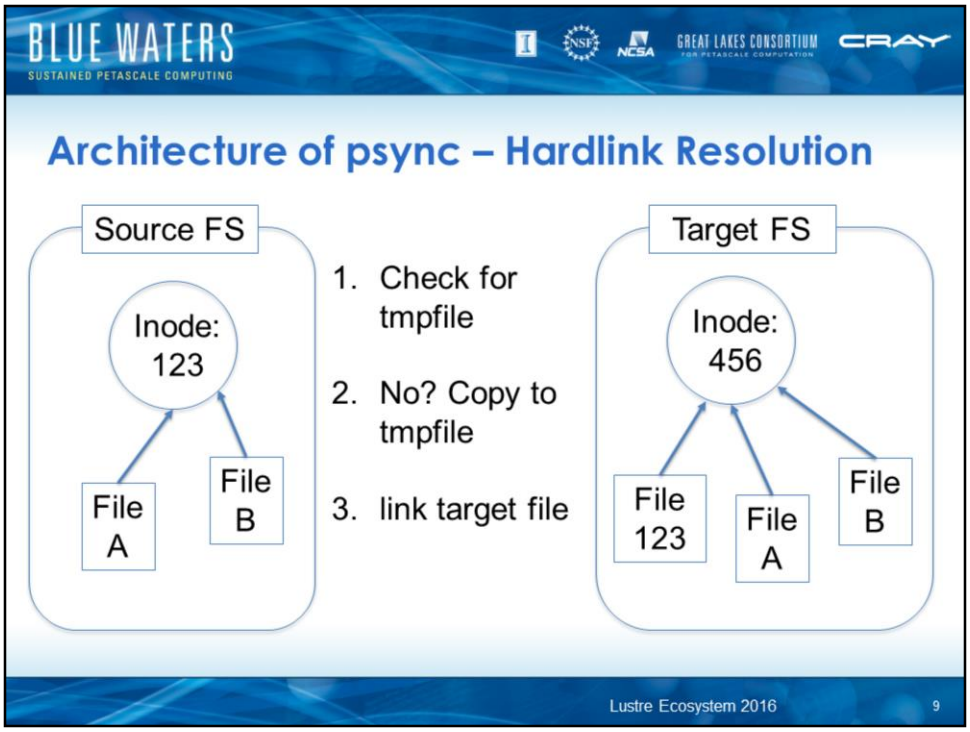
+ dd is more efficient to copy “larger” files because blocksize can be increased

  - 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:

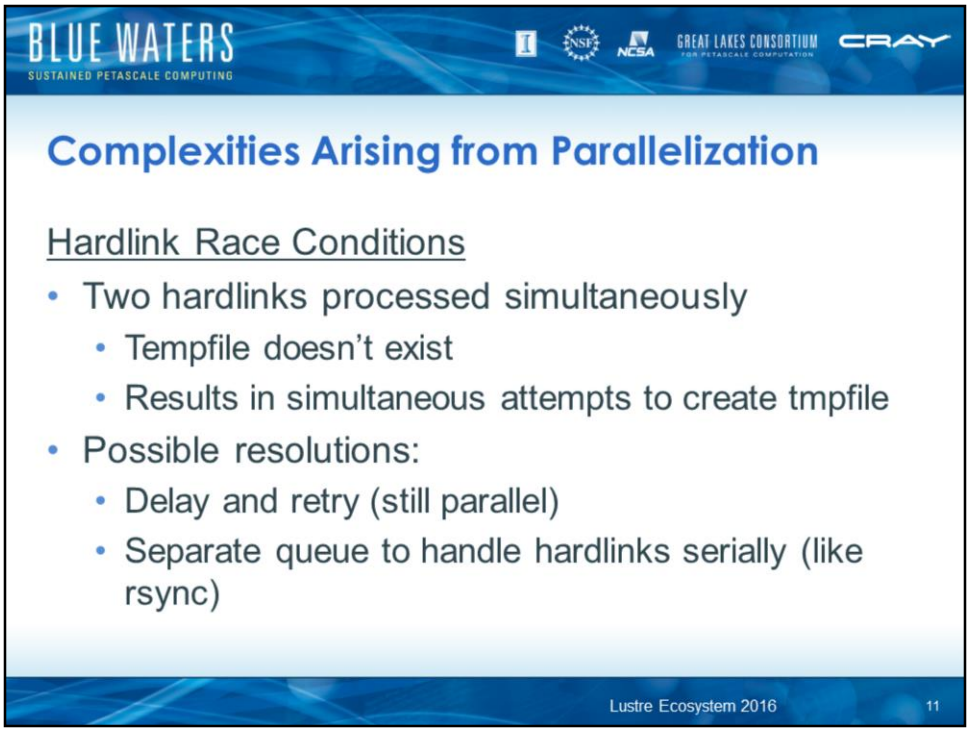
+ tempfile name == source file inode number (FID)

+ 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)

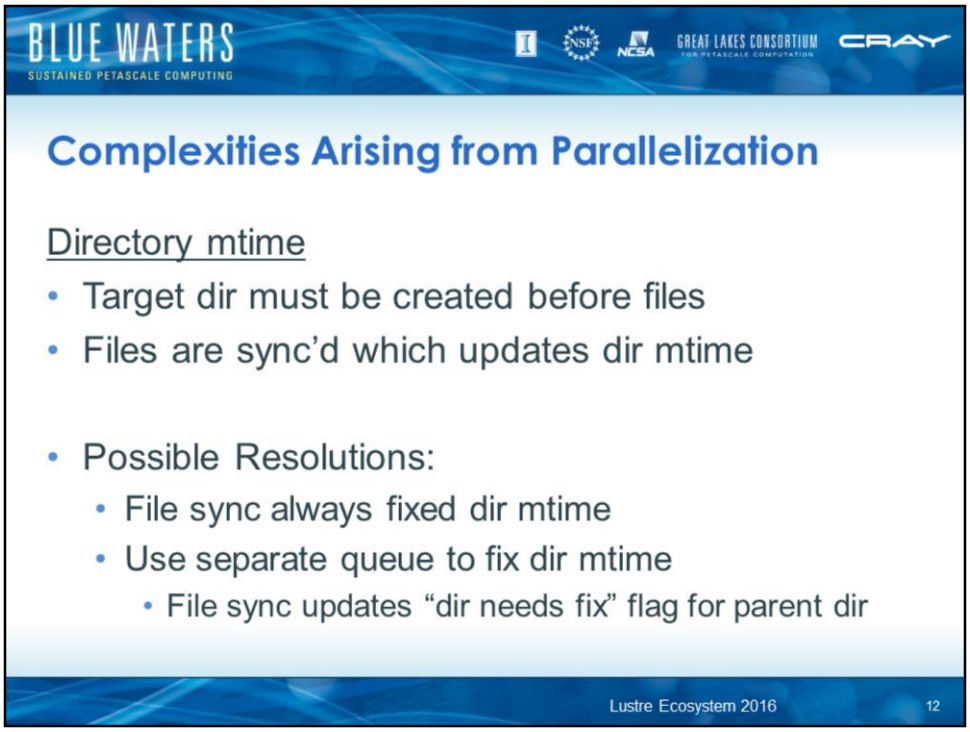
- Programming is special in that the tasks we undertake are simple in principle, but difficult in practice.

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



Key point:

+ Workaround is to run another psync



Key point:

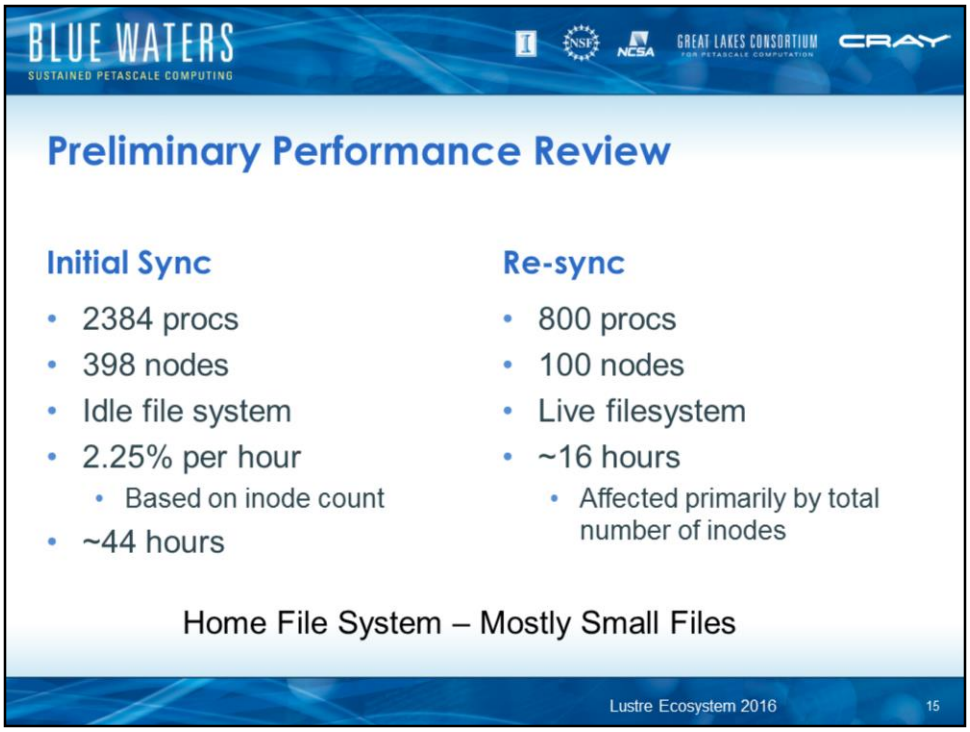
+ Workaround: run another psync

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

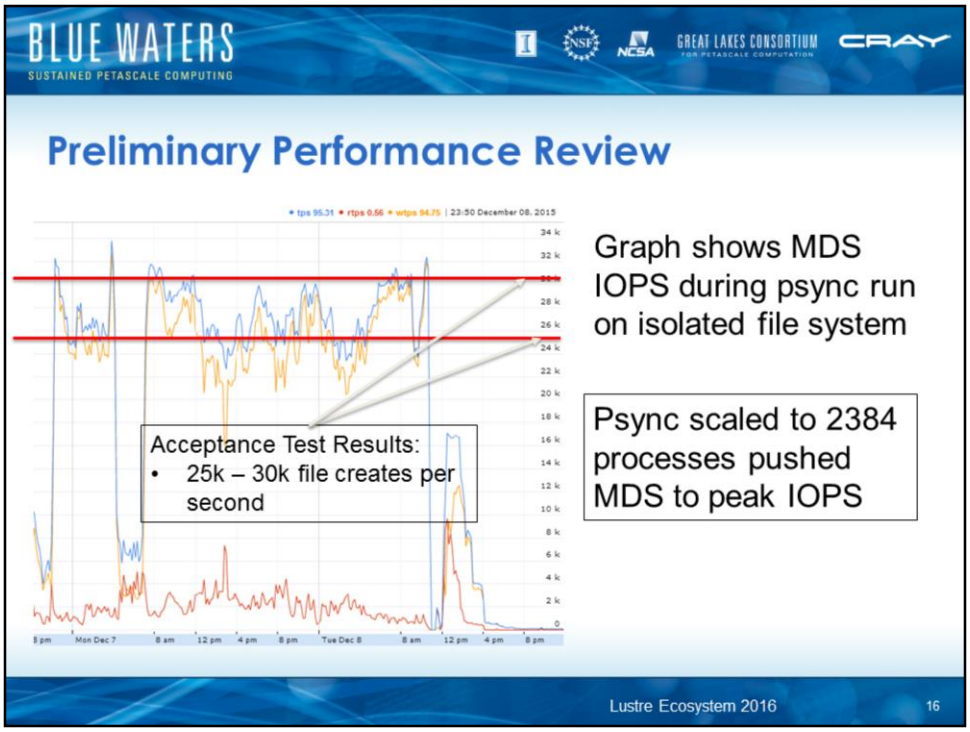
- TPS and MiB/s depend entirely on filesystem layout
- Initial sync time affected primarily by usage
- Re-sync time affected primarily by total inodes



Key Points:

+ Initial sync – data and metadata

+ Resync – mostly metadata

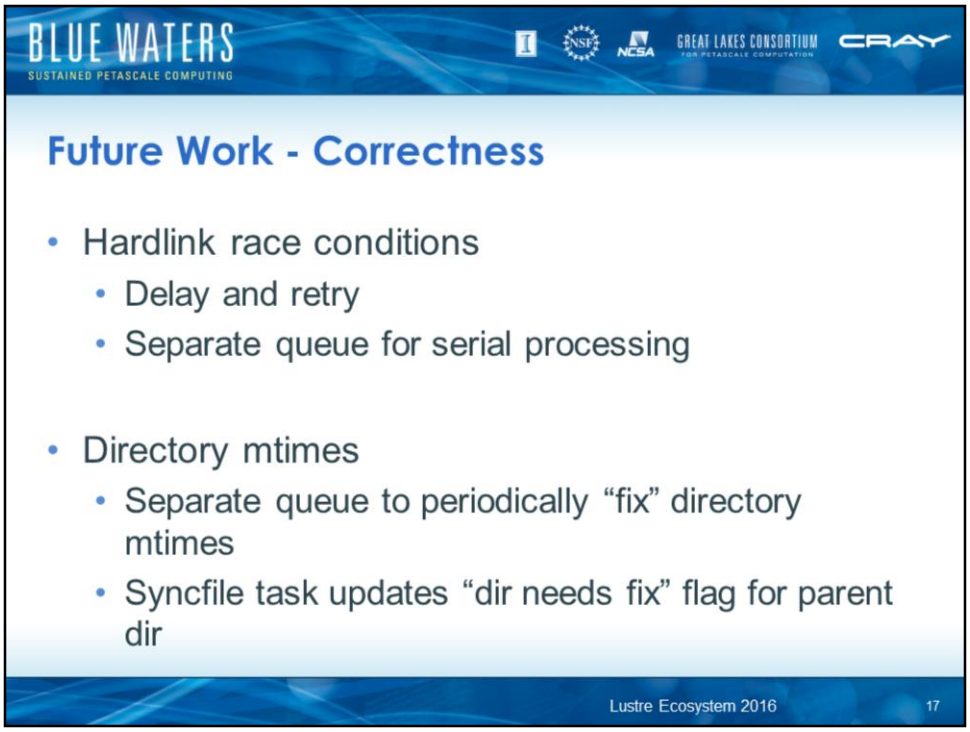


Keypoints:

+ synchronization of “Home” file system (ie: mainly small files)

+ syncdir (mainly dir scan) and syncfile run simultaneously

+ acceptance test results generated from mdtest

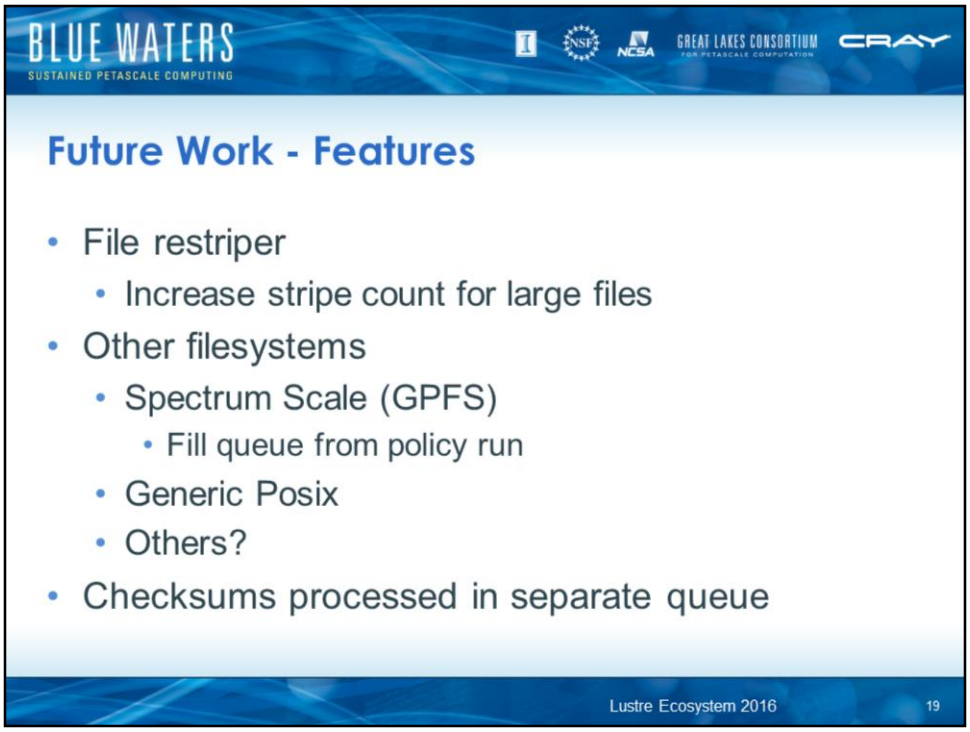


Keypoints:

+ 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



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