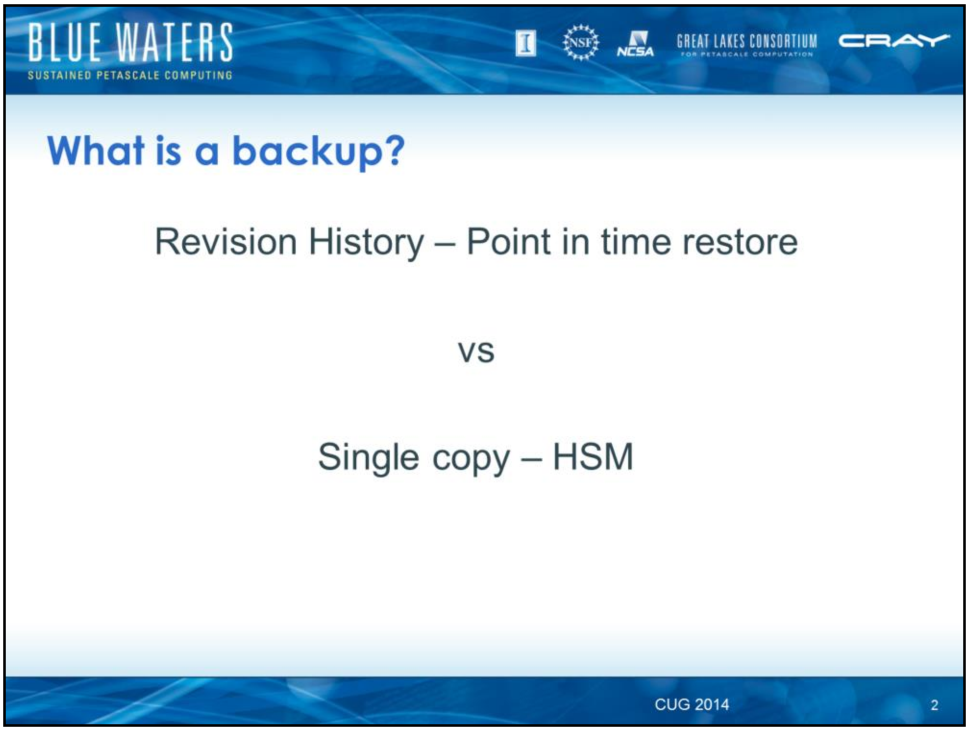

BLUE WATERS
SUSTAINED PETASCALE COMPUTING
Help! My filesystem is too big and I can't back it up!
Automating backups on a 4PiB filesystem.
Andy Loftus (aloftus@Illinois.edu)
GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION
NCSA
NSF
CRAY

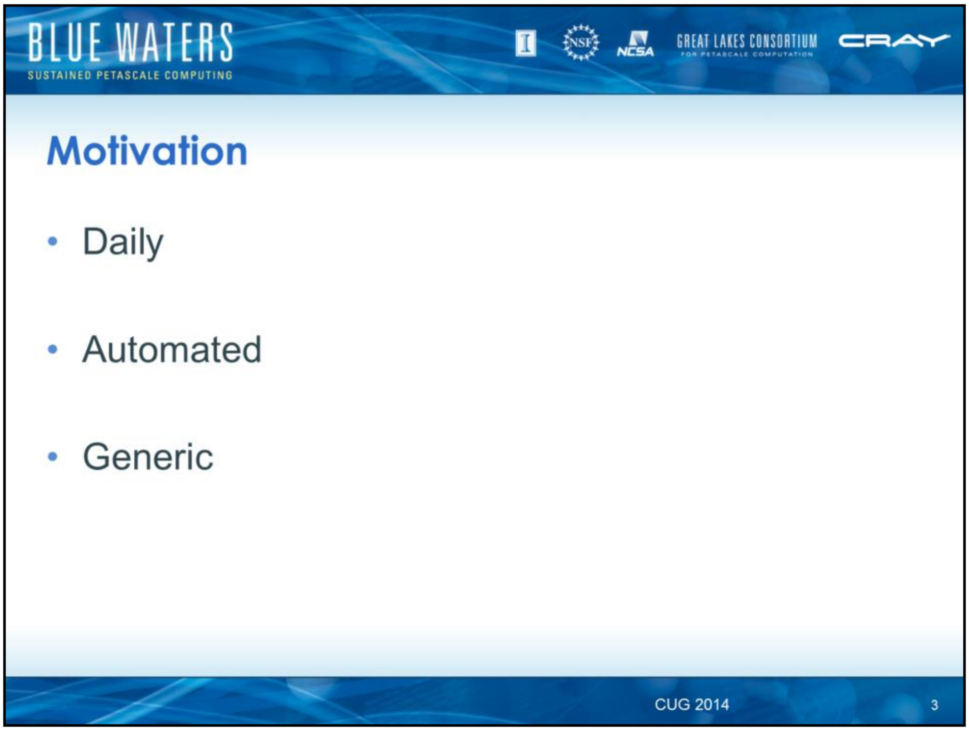


HSM provides only the capability of Single Copy.

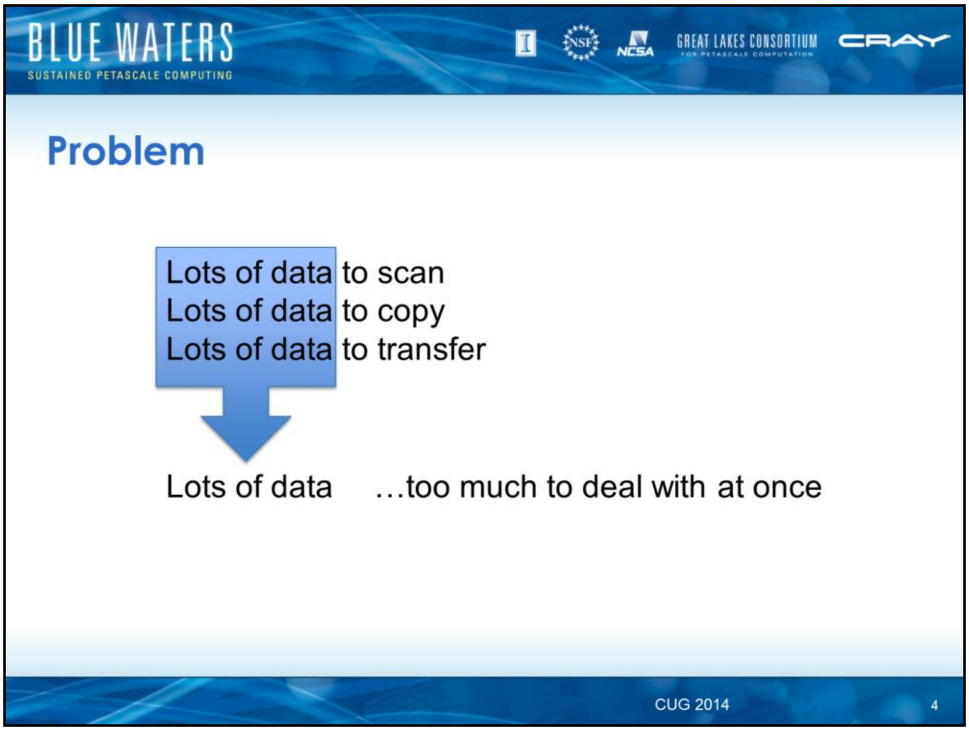
This talk will focus on Backups as Revision History.

This is a custom backup solution.

Why reinvent the wheel?



1. Daily: Want a solution that runs, and completes, every day. Key: It's got to be fast.
2. Automated – daily execution and dynamically adjust to changing environment
    1. Ie: new and deleted users and groups
    2. Counter example - Amanda pdf on Lustre wiki requires a manual configuration of Disk List Entries.
3. Generic - Not tied to any specific filesystem or implementation - portable
    1. Example: Cray Sonexion is a black box. Direct access to low level lustre storage is not allowed.
    2. Don't want to focus on the filesystem structure, just the data.

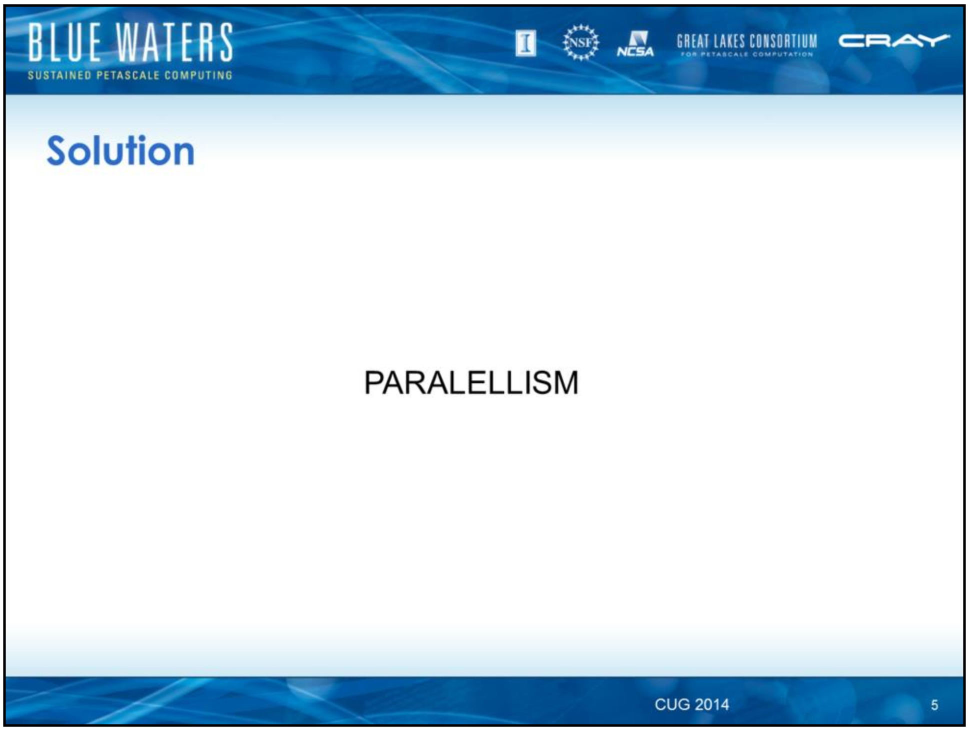


What makes this a hard problem?

- /home – 2 PiB capacity
    - 537 users
    - 16 misc (VM's, boot image servers, etc)
- /projects – 2 PiB capacity
    - 130 projects

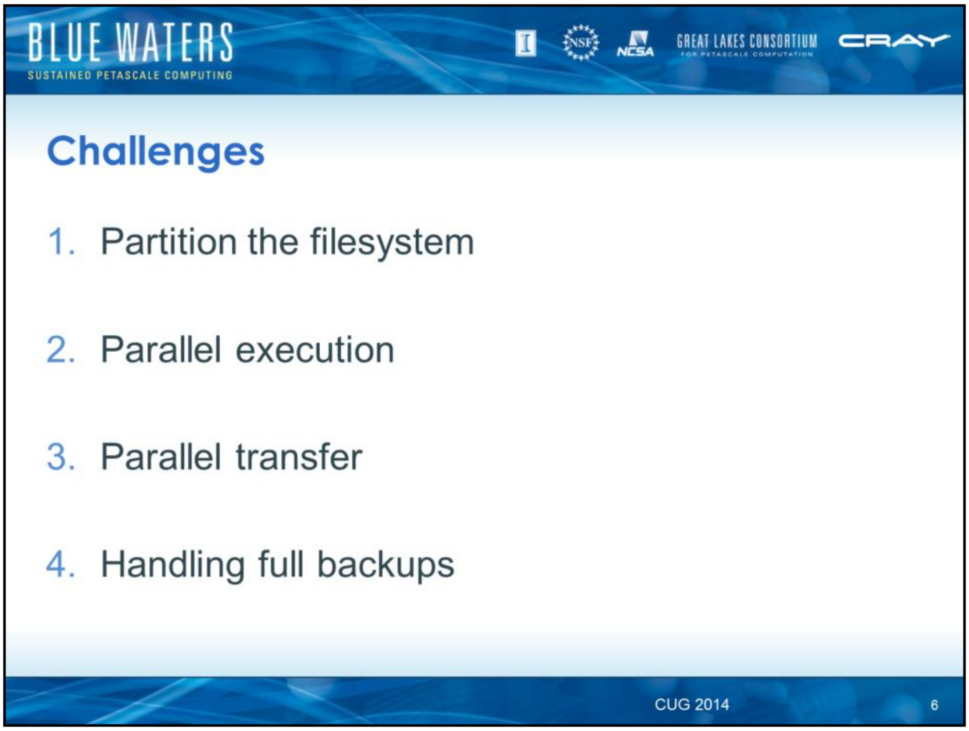
Lots of data

- SCAN to find changes
- making a COPY of the files is the actual backup
- TRANSFER the files to a safe, preferably remote, location



Any solution must take advantage of the parallelism that already exists in the system.

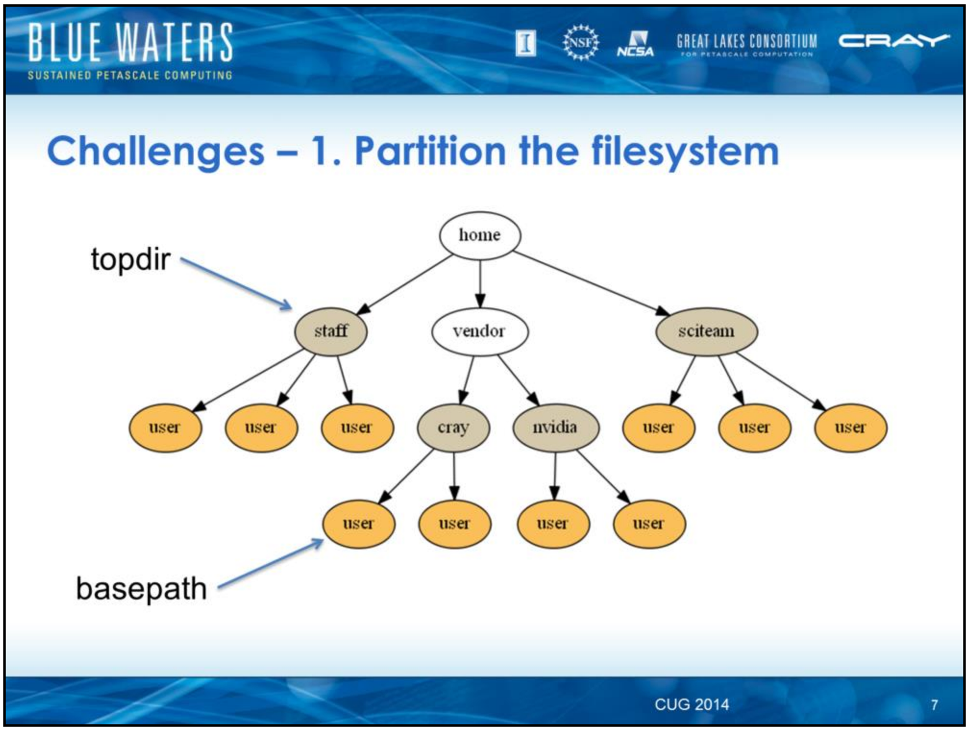
Challenge: Implementing a fully parallel backup solution.



1. Must define non-overlapping targets that can be backed up in parallel.
2. Distribute the backup tasks to run in parallel.
3. Efficiently transfer the backup archives to long term storage.
4. Full backups essentially comprise a copy of the entire filesystem. Using a policy of full backup every 30 days, this becomes a recurring problem.

Solution must be:

- Automated (performed by software)
- Dynamic (resumable/restartable)



Task: Define non-overlapping chunks that can be backed up in parallel.

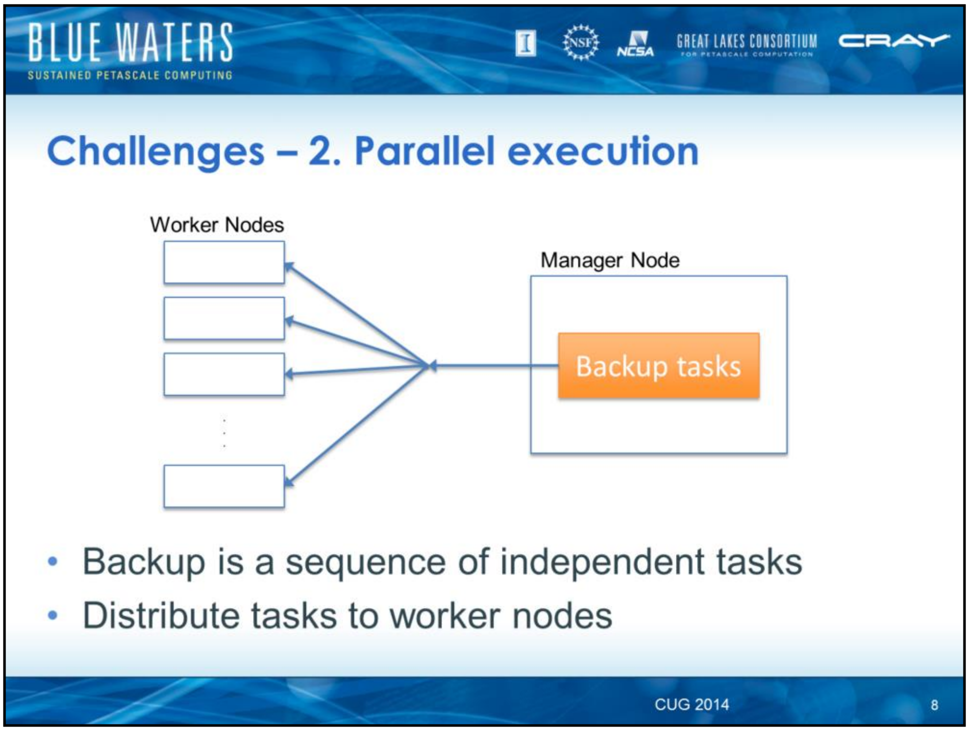
The filesystem structure natually provides good splitting points at user home directories and project specific directories.

Manually define topdirs. Then the software scans exactly 1 layer below for basepaths. The basepaths are the small, non-overlapping chunks that get backed up in parallel.

As new users or projects are added, or deleted, the changes are discovered automatically.

All the goals have been met:

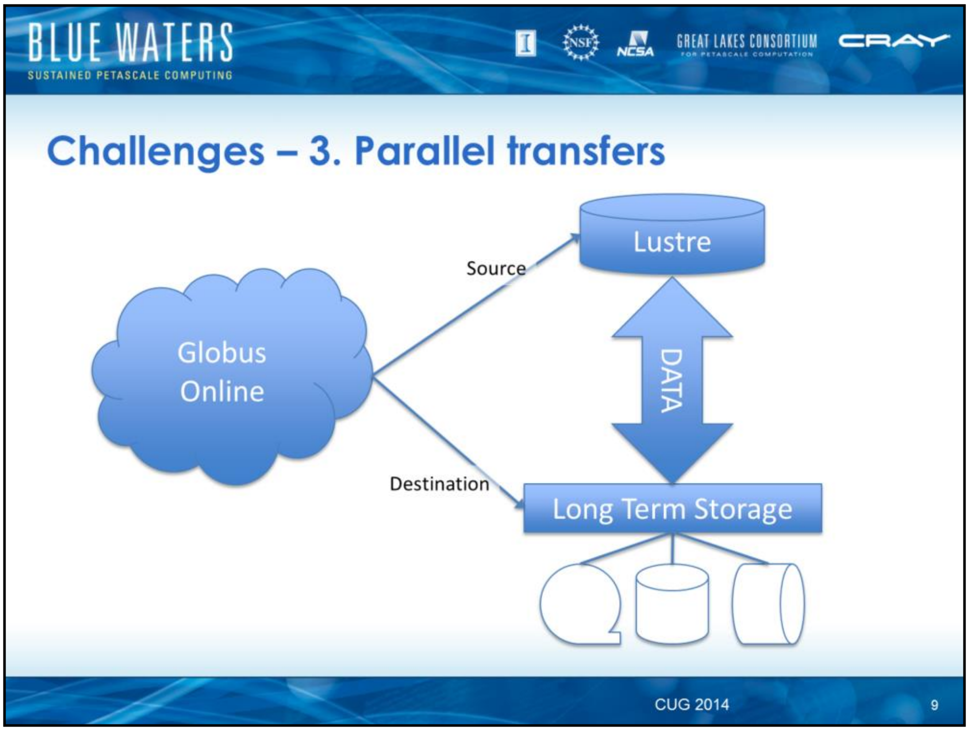
1. DAILY - Small chunks backup faster. Small can mean either few inodes or low capacity use, or both.
2. AUTOMATED - Automated discovery of changes (ie: new users, new projects, etc…)
3. GENERAL – Haven't relied on any filesystem-specific or implementation specific details.



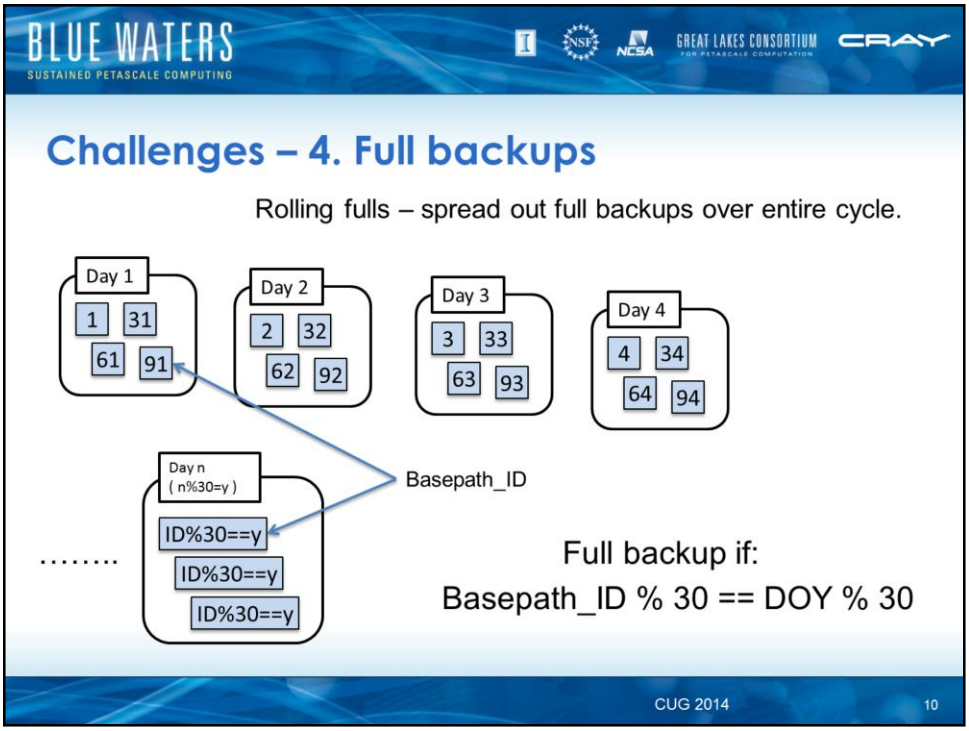
Running backups in parallel requires accounting for two key points:

1. Ensure each backup can run independently (for resiliency, pick up where last part left off)
    1. Works by saving state in a file
2. Running backup tasks in parallel (backups require root)

Cannot schedule jobs using Moab because policy prevents root jobs. Use RPyC to build in a custom solution.



- Some efficient transfer solution usually already configured on a cluster
- 3rd party means the backup software is not tied to a specific implementation
- Backup software only needs to interact with the third party transfer mechanism



Two approaches:

1. One full and incrementals forever.
    1. GOOD: minimizes data to be transferred
    2. BAD: complex mgmt, saves all data forever
2. Periodic fulls (in one form or another).
    1. GOOD: Simple
    2. BAD: Recurring fulls

Idea: Only a few full backups run each day.

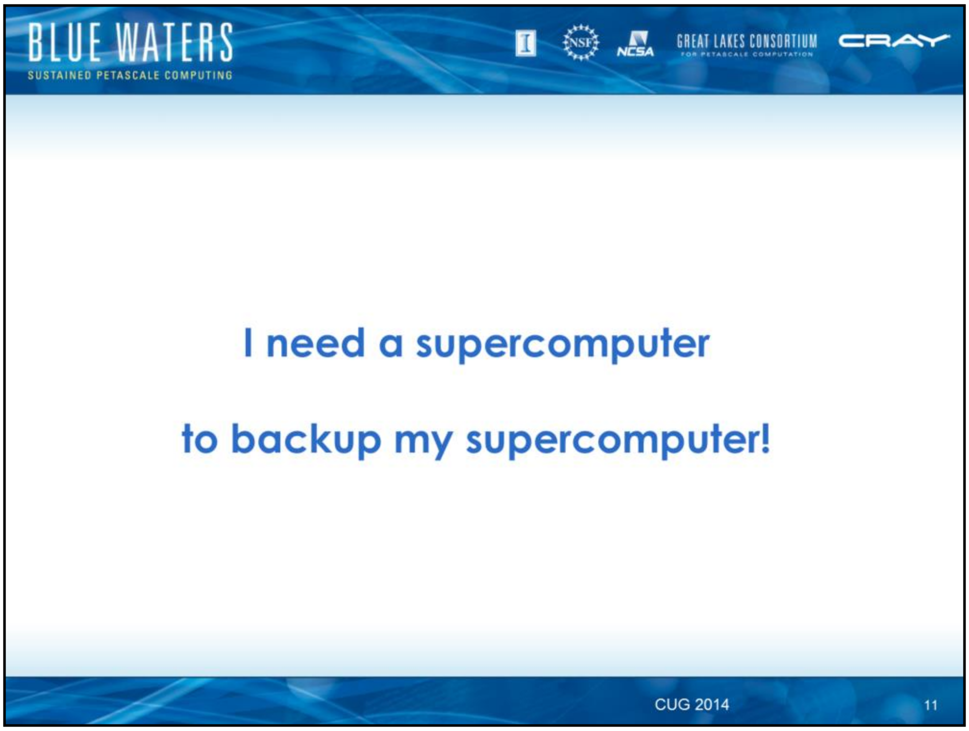
Cycle = how many days between full backups. 30 day retention = 30 day cycle

683 basepaths / 30 day-cycle = 22.8 full backups per day

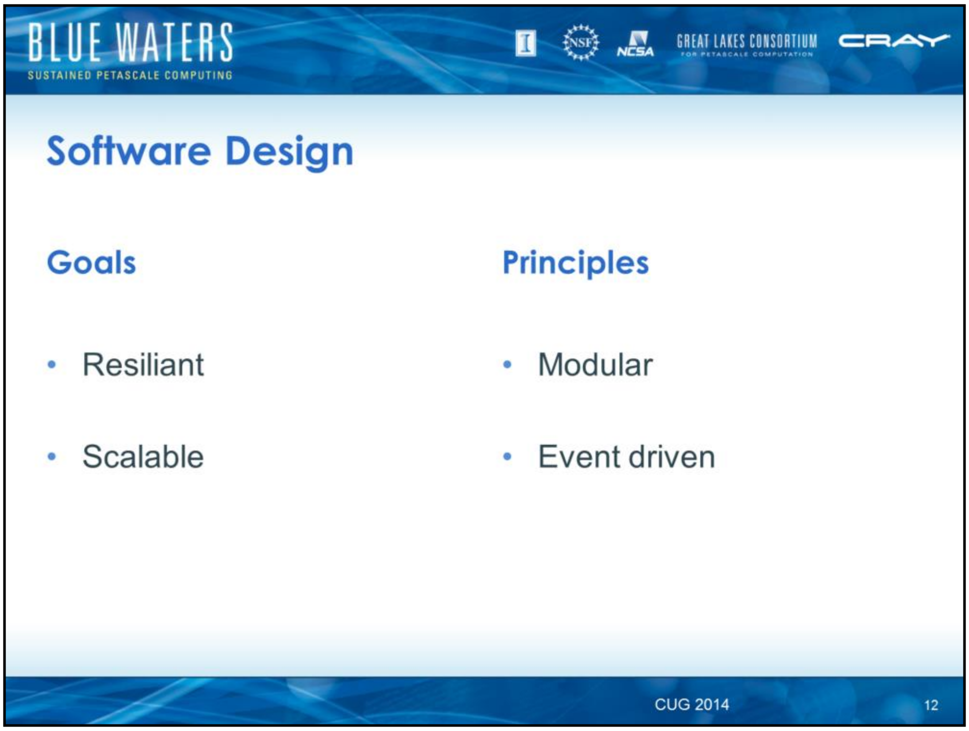
Algorithm: Choose which basepaths get a full backup on a given day.

Assign each basepath a positive integer id.

Basepath % 30 = initial DOY



Backups run on 28 IE nodes, outside of the cluster but mount the filesystem.



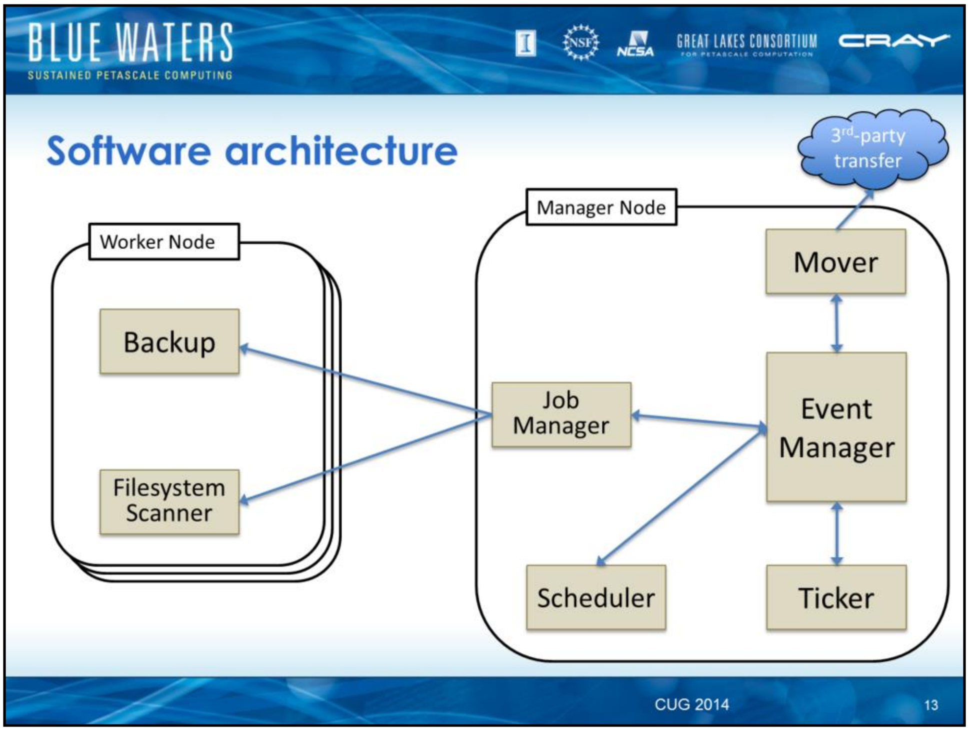
Recall motivations: daily , automated, generic

Design the software to support these driving factors.

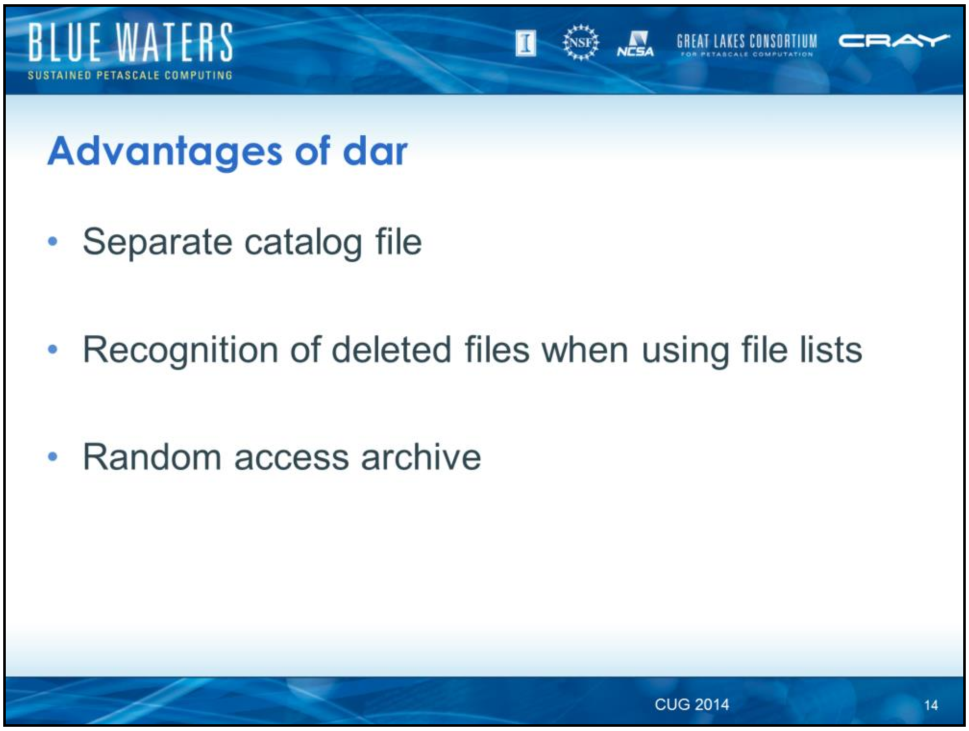
- Resiliancy
  - Ability to tolerate failures (node crashes, network failures, and filesystem failures)
- Scalable
  - Scale for increased performance and/or larger filesystems by simply adding new nodes.

Goals promote modular, event driven architecture.

- Modular
  - Simplicity in both design and ease of maintenance.
- Event driven
  - Events (not time) drive all actions. Activity proceeds based on messages passed between modules.



1. Modules
2. Events (messages)
3. Event Manager as post office
4. Modules register for notification of event types
5. Steps of a backup, move forward based on events



Dar used to do the actual backup

CATALOG

- archive metadata in a separate file
- Reference for incremental backups
- Small, stays on disk (full archive file sent to remote, low cost, long term storage)

FILELIST

- During Incremental, properly handles deleted files

RANDOM ACCESS – doesn't have to scan archive file in order

- Recall – backup as sequence of small steps & software design goal of resiliency (ability to pick up where it left off)
    - Example: create archive & extract catalog are independent steps

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

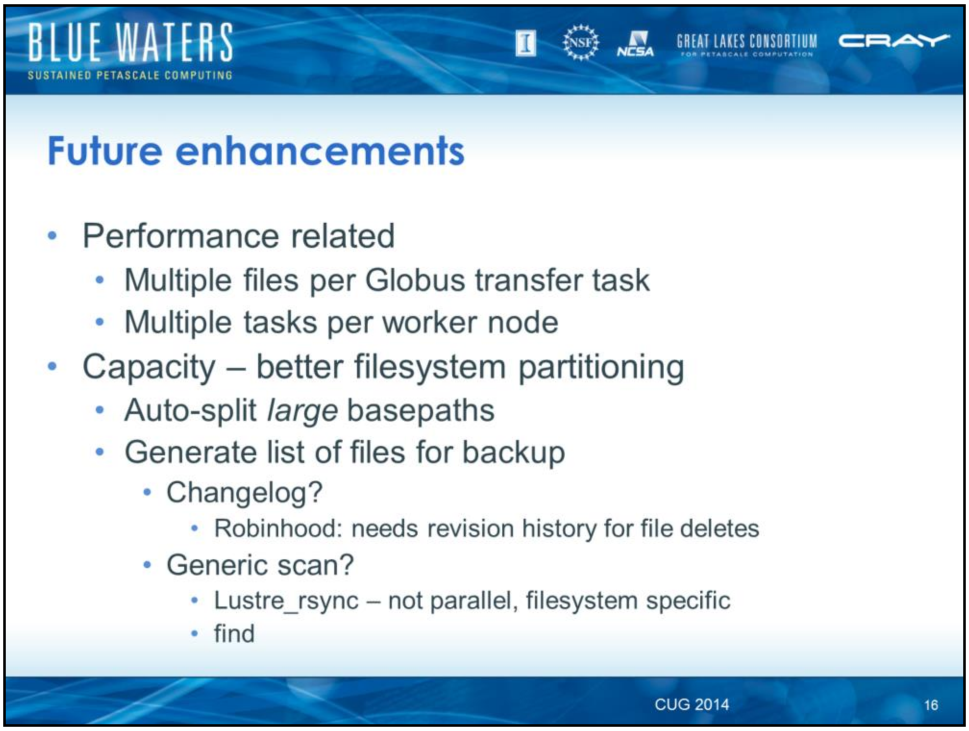
GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

Results

9.6 hours	Average daily backup runtime
2.4 hours	Average full backup time
9 minutes	Average incremental backup time
404.5 million	Average number of inodes scanned daily
51.3 million	Average number of inodes backed up daily
1.61 PiB	Average bytes scanned daily
16.8 TiB	Average bytes backed up daily
10	Number of topdirs
683	Number of basepaths

CUG 2014 15

Averages from Feb 2014 over a 5 day period.



PERFORMANCE

- Bottleneck – inefficient use of Globus
- Currently only a single task per remote host
    - Different nodes do not have to have the same capabilities
        - Some may be able to run more tasks in parallel

CAPACITY

- Initial attempt – automated scan to breakup a project directory
    - Complicated, not scalable
- Robinhood – No revision history, deleted files not detected
    - Thought - data-mining operation beside robinhood, effectively creating snapshots of the filesystem
- Find – Could be pheasable, stat is fast and light, scalability is a concern

Thank you!

- Questions?

Andy Loftus (aloftus@Illinois.edu)