


Hybrid Flash/Disk Storage Systems

Lustre Features, Tier Sizing, & Data Movement

 nrutman@cray.com



CRAY

LUG 2019-05-14

About This Presentation

- You really wanna flash
- But you ain't got no money
- Look at the use cases and tradeoffs
- How much do I need?
- Features that help, inside and outside of Lustre

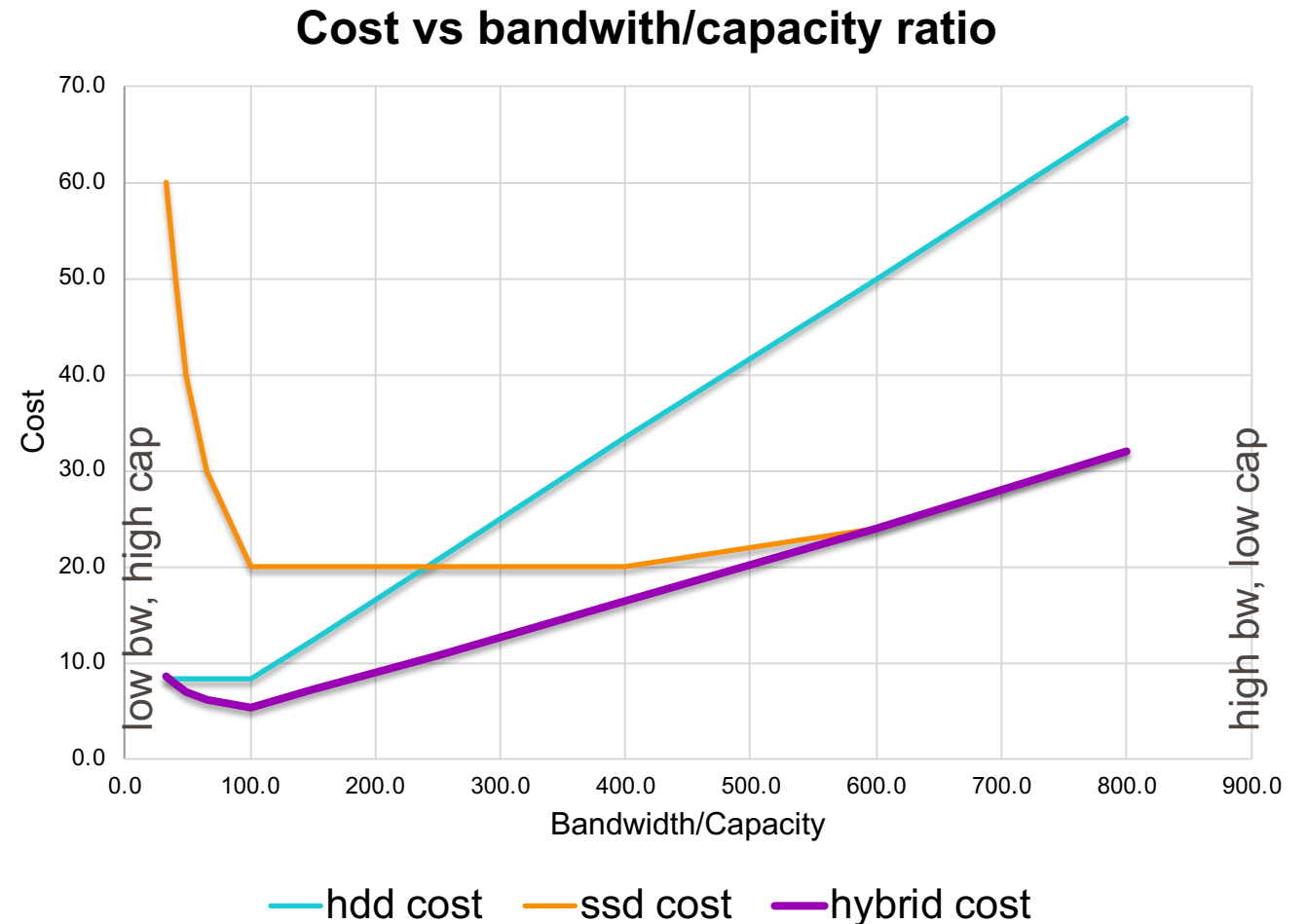


Economic Optimization



- With two media types, can optimize \$ for two constraints (e.g. BW + Capacity)
 - Flash for peak bandwidth
 - Disk for max capacity
- Add the speeds and sizes
- And sign the PO!

Not so fast...



The Bandwidth Fallacy

- If my flash tier goes at 500 MB/s, and my disk tier at 300 MB/s, can I get 800 MB/s for my app?
- File-Per-Process job with 5 nodes writing to SSD for every 3 nodes writing to HDD
 - Non-trivial to set this up in App and/or Lustre – 5/8ths of your files in flash
 - Bifurcated performance in post-processing
- Even more complicated with a Single-Shared-File
 - May be possible with overstriping with an explicit OST list [LU-9846](#)
- So: adding performance of tiers – not so much

$$5 + 3 = 5$$

Initial Data Placement

Should I Stay or Should I Go?



Static

- Place (and leave) your data in the nominally “right” place
 - Stream to HDD, random to SSD

needs

- Permanent capacity in tier
- Placement policies / features / foresight

Dynamic

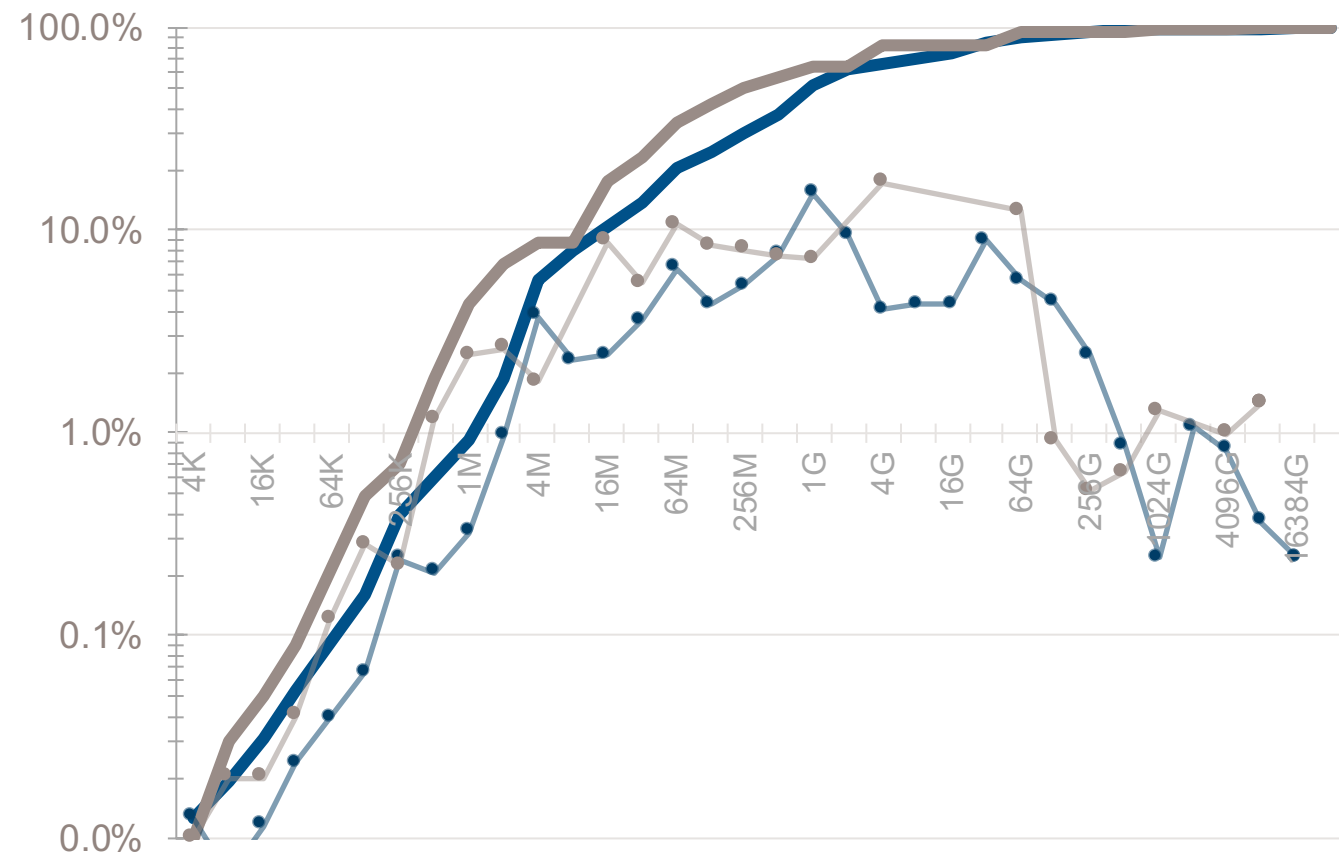
- Temporarily put your working set in flash
 - And move it when done

needs

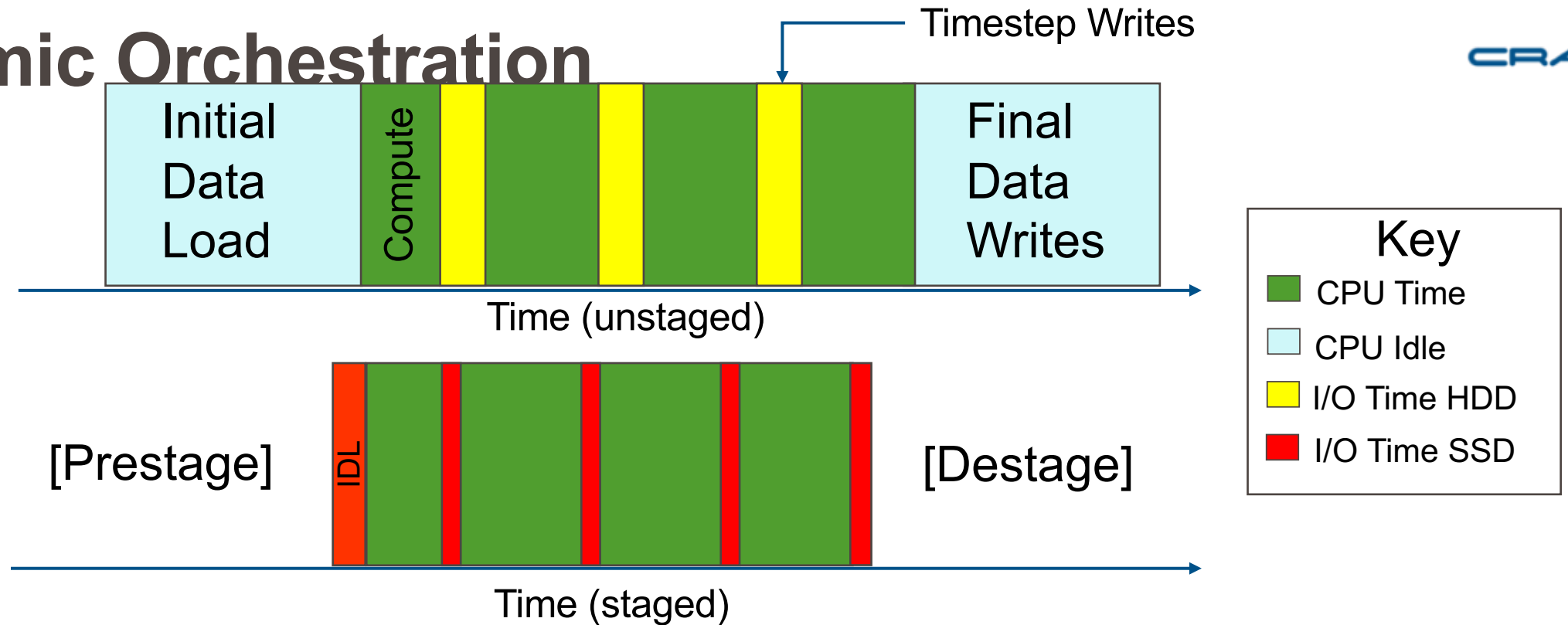
- Efficient, scalable data movement infrastructure
- Orchestration and automation

Static Sizing

- We initially sized our flash for peak bandwidth
- But if we're going to leave files there, we really care about capacity
 - SSD capacity for IOPS files
 - HDD capacity for streaming files
- How big?
 - Small files as a proxy for random
 - Use file size distributions



Dynamic Orchestration



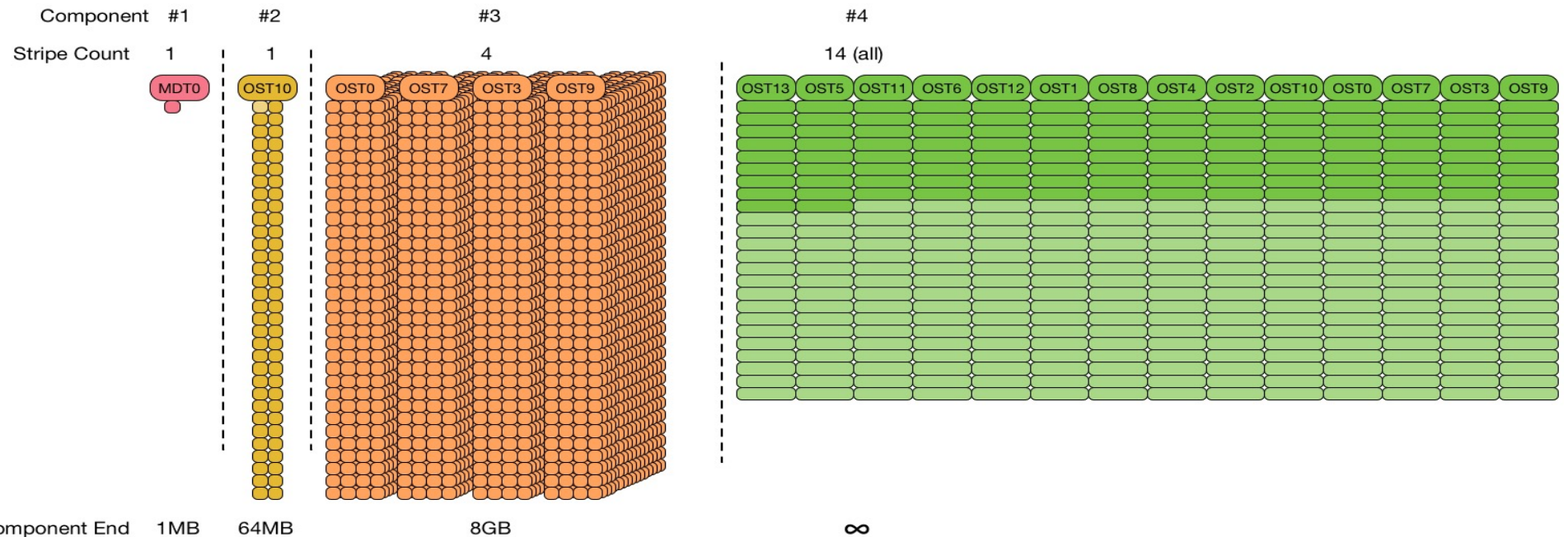
- Compress IDL & timestep writes to flash during “job”
- Reduce job wall time
- Keep CPUs busier
- Flash as “Burst Buffer”
- Pipelining issue - requires intelligent scheduler
- No permanent flash files (need space)
- Data movement requires bandwidth in HDD + SSD = 2x

Initial Placement Controls

- Set up disk and flash pools
- Directory defaults for known apps
- Progressive File Layout (PFL) for unknowns
 - Want “as much as possible” in flash, but no more
 - Thresholds based on file size distributions
- Enforcement
 - Default FS pool = HDD (or PFL)
 - Pool quotas (\neq project quotas) [LU-11023](#)

Three notes on PFL

- Assume we want PFL to fill all tiers at the same % rate (e.g. full in 5 years)
 - But this means flash is empty most of the FS life ☹️
- It becomes less effective to move files “off” of flash – hard to reclaim space ☹️
- Individual PFL files aren’t really mixed media – behave as one or the other, but consume space on both. Flash is “wasted” for large files ☹️



Pool Quotas

LU-11023

CRAY

- OSTs track inode/space usage per user/group/project
- We can sum the usage for any/all groupings of OSTs (i.e. pools)
- The MDS grants quota space to OSTs, can limit grants however it likes; i.e.:
the minimum of all remaining quota space for the pools that OST belongs to.

Limit user Bob to 2G's worth of space on flash OSTs:

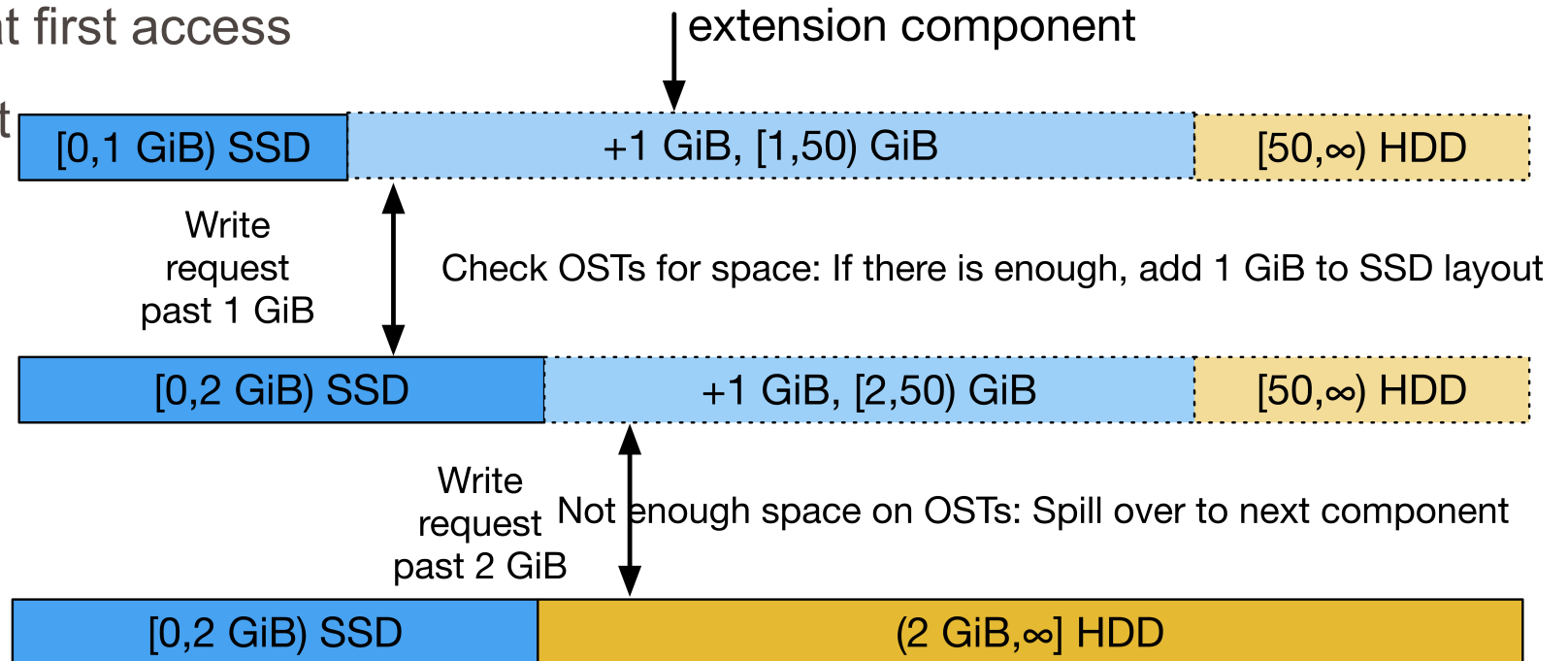
```
ifs setquota -u bob -P flash --block-hardlimit 2G /mnt/lustre
```

- These are not *file* limits, but rather OST space limits. Only data on the flash OST counts toward this limit.

Spillover Space

- A small flash pool will fill quickly, but want to avoid the dreaded *ENOSPC*
- Need *adaptive* layouts for tiering that adjust to remaining space
- So SEPFL: [LU-10070 in Lustre 2.13](#)

- Uninitialized PFL components don't have stripes allocated
- Allocate new components at first access
- Check remaining space first



When We Get Initial Striping Wrong

- Can ENOSPC on small flash OSTs
 - Spillover space just-in-case
 - Pool quotas prevent abuse
- PFL leaves flash empty
- Bursting leaves flash full
- So we must move files at some point
 - Requires efficient copytools
 - Requires polices to select and act

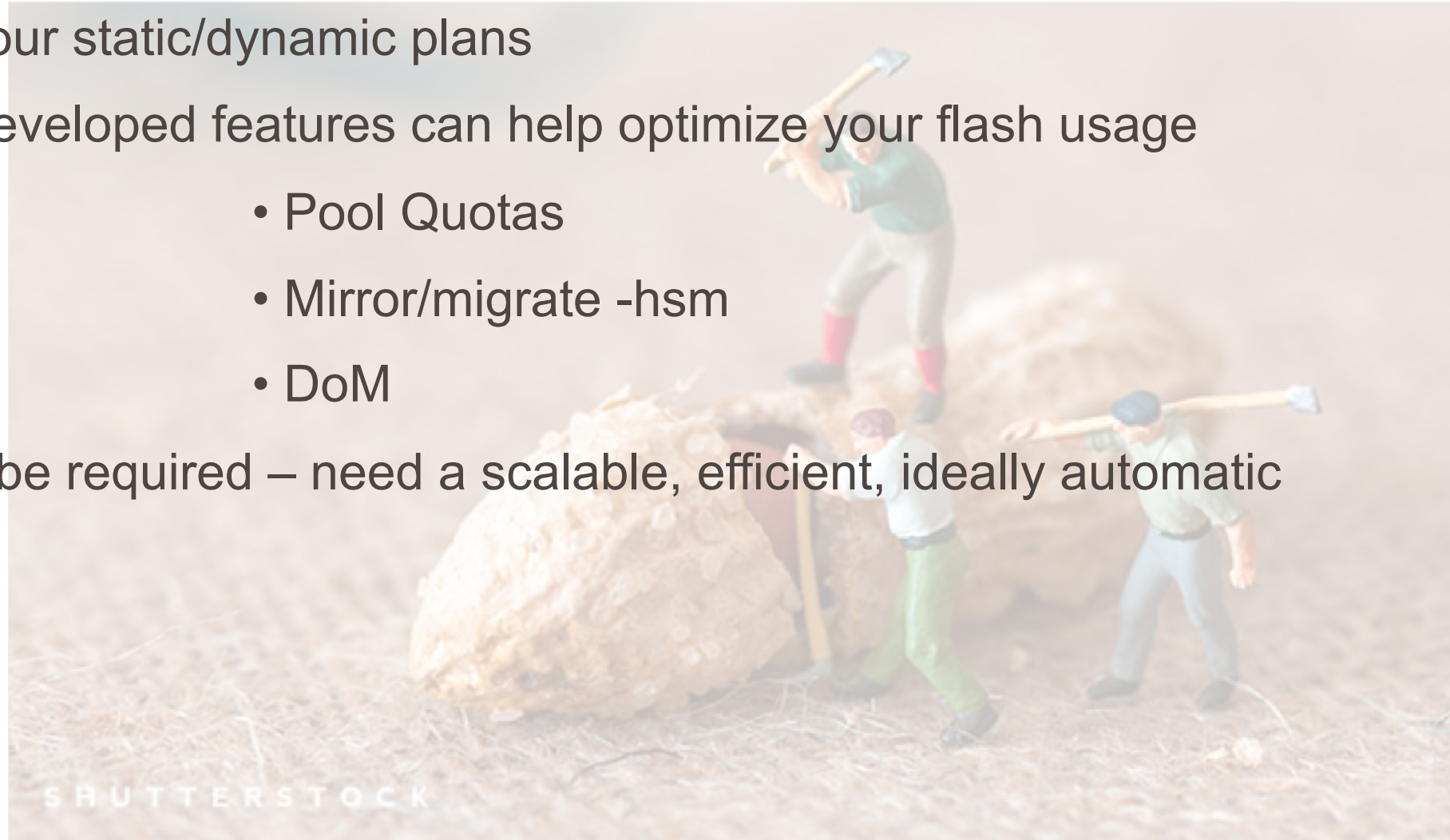


Externalize the Coordinator [LU-10968](#)

- Send HSM requests out of kernel to a more scalable solution
 - Prioritization, expiration, scheduling
 - Kernel memory/pipeline size
 - Directly take data movement requests from Lustre, Job Schedulers, CLI, HSM, etc.
- Expand coverage for intra-Lustre operations
 - hsm migrate [LU-6081](#)
 - hsm mirror sync
- Move data between tiers, multiple FS's, archives


All Together Now

- Your hybrid system will likely be using both static and dynamic data placement
- Sizing depends on your static/dynamic plans
- A number of newly-developed features can help optimize your flash usage
 - Spillover Space
 - Pool Quotas
 - Overstriping
 - Mirror/migrate -hsm
 - PFL
 - DoM
- Data movement **will** be required – need a scalable, efficient, ideally automatic solution



THANK YOU

QUESTIONS?

 nrutman@cray.com



cray.com



[@cray_inc](https://twitter.com/cray_inc)



[linkedin.com/company/cray-inc/](https://www.linkedin.com/company/cray-inc/)



SAFE HARBOR STATEMENT

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts.

These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

