Spillover Space: Dynamic Layouts for Tiering

Self-Extending Layouts (SEL)

Tiering

- Different types of storage hardware with different tradeoffs of latency, capacity, and cost:
  Flash, HDD, SMR[1] HDD
- Modern file systems often include multiple types, separated in tiers
- Lustre supports tiering: Different OST HW, OST pools, file layouts
- Problem: Tiering splits the file system, and it becomes easier to run out of space
- Space rebalancing becomes more important

[1]Shingled Magnetic Recording

Traditional Lustre Layout

- RAID0 striping across OSTs
- OST space is checked at file creation time
- Static: Cannot be changed once file is created
- If an OST runs out of space, cannot write to files striped to that OST
- Same striping for the entire file

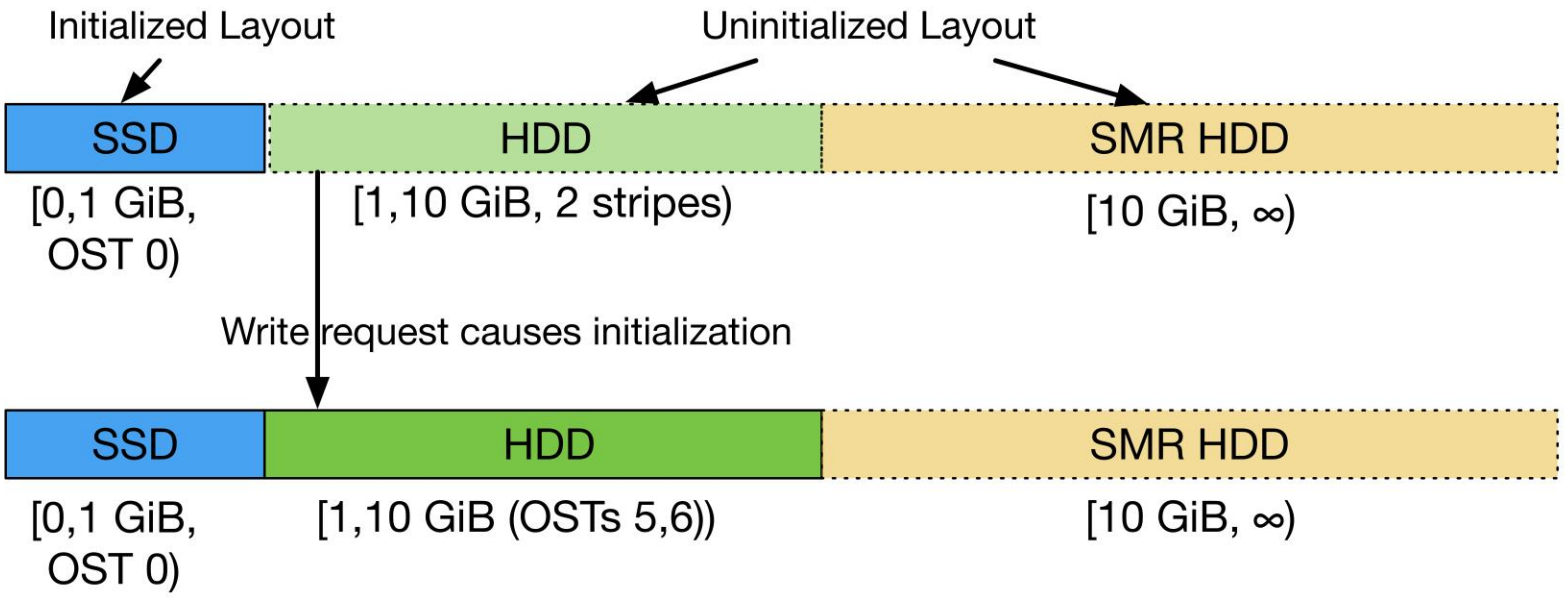
Progressive File Layouts

- Different layouts for different regions of a file
- Fits nicely with tiering – Small files on flash, large files split between flash and disk

Example PFL Layout

PFL: Layout Initialization

- When a client writes to a component for the first time, it asks the server to initialize it:



PFL: Limitations

- PFL improves space balancing as components are initialized when needed
- But components are large and we check OST space only at initialization time
- If an OST runs out of space, we cannot write to components already using that OST
- Initialized components are static and cannot be changed

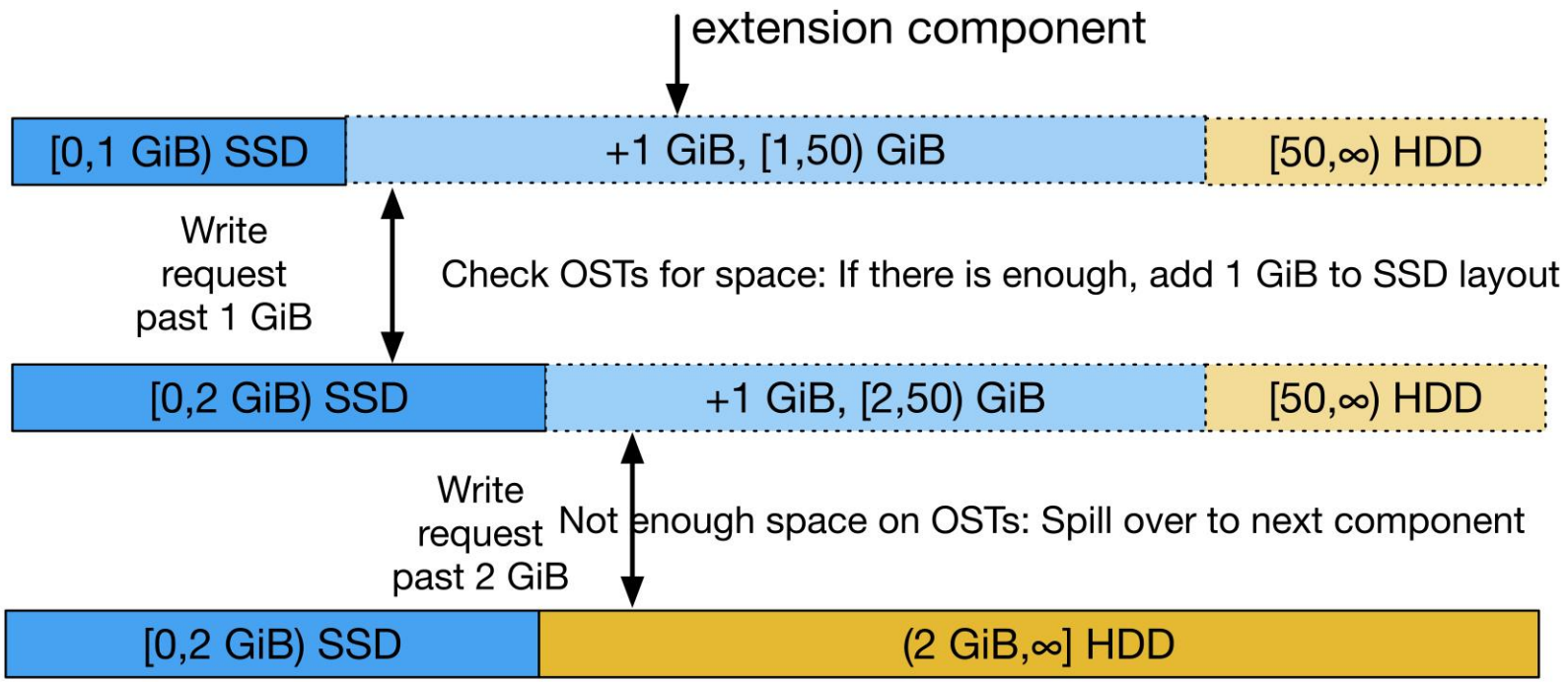
Dynamic Layouts

- We would like layouts to handle conditions such as running low on space
- Changing layout during i/o is an old idea, but very hard to do well
- If we can manage layout in smaller granularity, we can be more dynamic
- Many tiny components would use too much MDT space, but there is another way

New Feature: Self-Extending Layouts

- Gives layout in smaller chunks, rather than whole components
- Component is split in two parts: Initialized and extension space
- When more layout is needed, check if there is enough free space on the current OSTs
- If there is enough space, initialized part is extended
- Otherwise, switch to a new component on different OSTs

Self-Extending Layouts



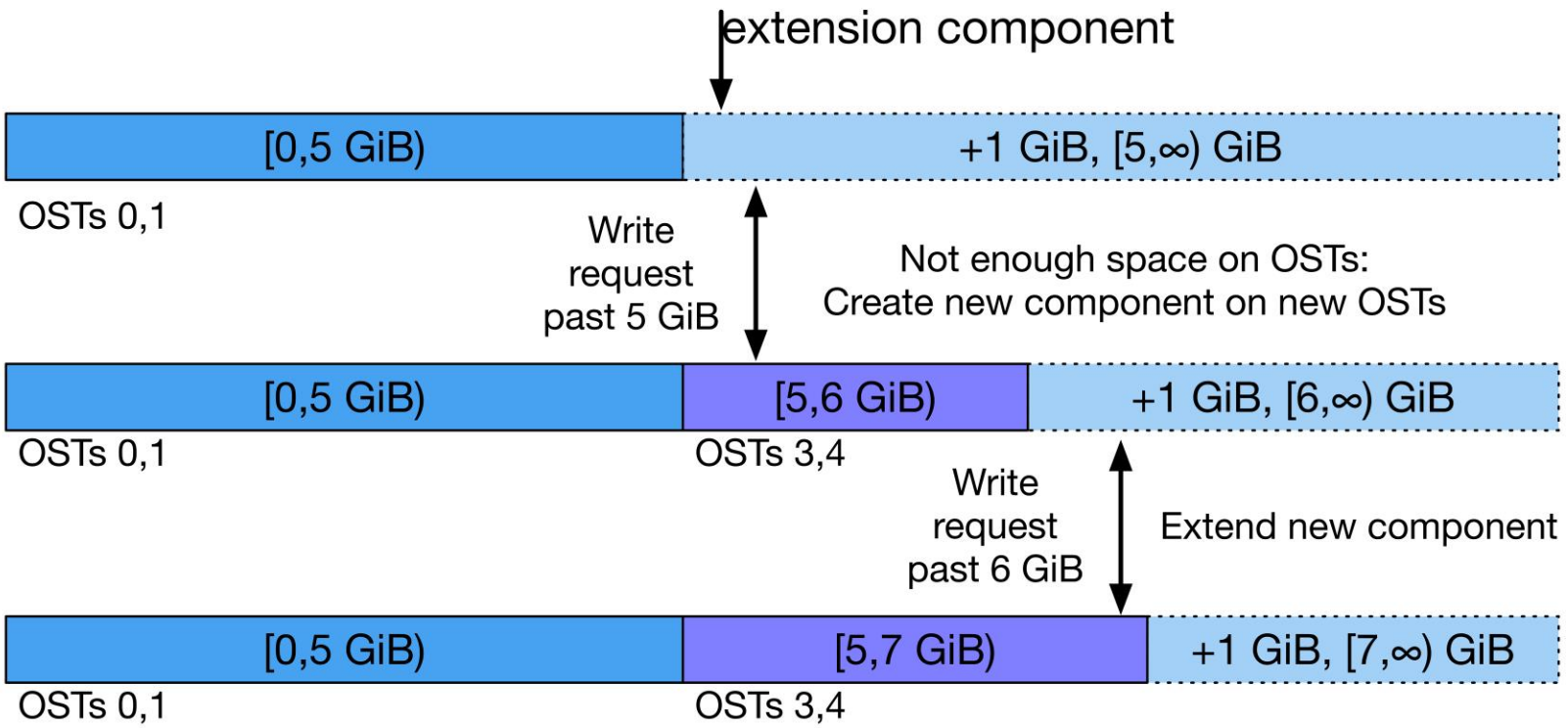
Self-Extending Layouts

- Because layout is given in chunks, it is still possible to get ENOSPC if an OST fills up before the chunk runs out
- Chunk size is a tradeoff between performance and the chance of getting ENOSPC
- Low space check is tunable: OSTs report "low space" at a threshold
- "low space" check & handling will eventually be policies with various choices

Self-Extending Layouts: Self Spillover

- Single tier layouts cannot spill over to other tiers when an OST is low on space
- Solution: Spill over within the tier
- Create a new component, using the previous component as a template
- Start writing to new OSTs

Self-Spillover



Self-Extending Layouts: Self-Spillover

- Broadly applicable: Doesn't need multiple tiers
- All files can benefit:
  When creating a non-PFL file, make it PFL with a self-extending component at the end
  When creating a PFL file, make the last component self-extending
- Automatic space rebalancing for any system(!)

Development Status

- LU-10070 at Whamcloud
- Targeting Lustre 2.13 upstream (sooner from Cray)
- We encourage you to join upstream review if interested

Finally

- **Any questions?**
- Happy to answer questions later or by email
- Developer: Patrick Farrell, (paf@cray.com)