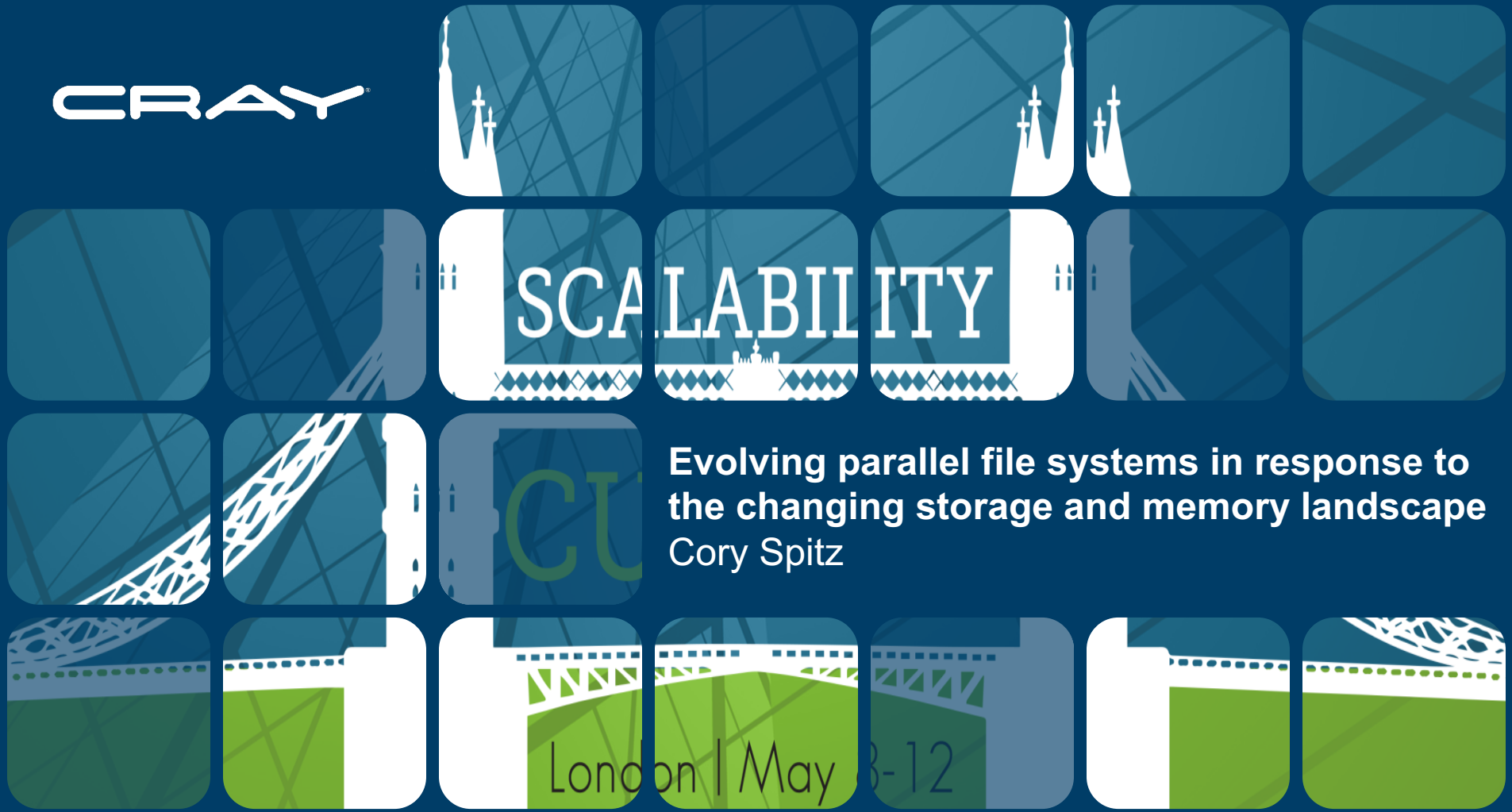


Evolving parallel file systems in response to the changing storage and memory landscape

Cory Spitz

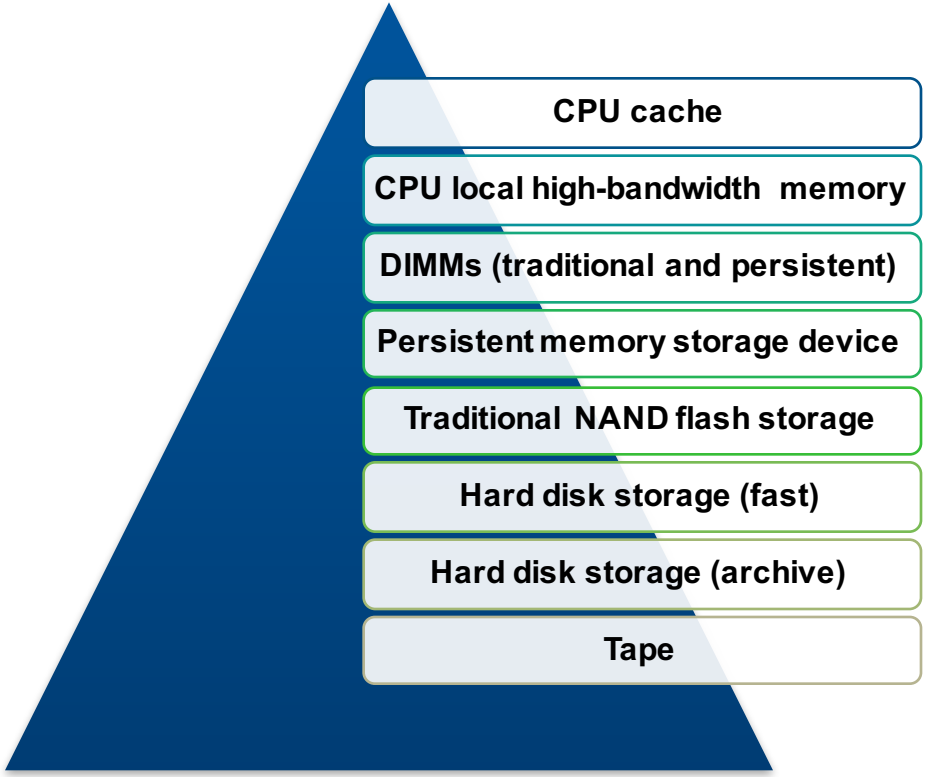
Abstract

Burst buffers and storage-memory hierarchies are disruptive technologies to parallel file systems (PFS), but there isn't consensus among members of the HPC community on how PFSs should adapt to include their use, if at all. There also isn't consensus on how HPC users should ultimately use and manage their data with these emerging technologies. In this BoF we will discuss how HPC and technical computing users want to interact with burst buffers or other storage-memory hierarchies and how their PFS should adapt. What do they expect? Will they want to continue to use POSIX-like semantics for access like with MPI-I/O or HDF5 containers? What do users expect for legacy codes? Generally, what do users, application developers, and systems engineers require? Will they accept exotic solutions or must a de facto industry standard emerge?

Premise

- **Assertion: there is no one canonical definition of an exascale parallel file system**
  - We all seem to agree that there will be lots of devices, components, and threads
  - Can't agree on a single solution for organization, workflows, access methods, or usage/semantics
  - Likely there will be tiered storage architectures as with DataWarp
  - Multiple solutions should emerge, even hybrids
  - For sure, things will be complex
  - How will these complex systems be productive?

Foreseeable memory-storage hierarchy



Survey question discussion

- **Think about how you might use these devices**

- **Possible use cases**
  - Streaming data
  - Small files and heavy metadata
  - Defensive I/O
  - Job steps
  - Out-of-core execution

Data placement

Today's PFS users are accustomed to dealing with data placement issues such as file/directory striping including stripe width and count.

- What control do we expect users to display with a storage hierarchy?
- With a memory hierarchy?
- In what way should the control be expressed? (examples: size, physical location, or unexpressed)

Management

How do you expect to manage data in persistent memory vs. data in the parallel file system?

- **Do you want a transparent file system cache?**
- **Do you want system control (perhaps via job control)?**
- **Do you want user control (application control)?**
- **Do you want all of the above? That is do you want to mix control methods?**
- **How do you envision marshalling data between the two?**
- **Should there exist an API to the PFS?**

Applications

What is the appetite for application changes?

- Is there a requirement to run legacy codes?
- Will users rewrite codes to extract additional functionality?
- Will users rewrite codes to extract additional performance?
- Do you want a middle layer to control application interaction?

Data formats

- Would you like to use data containers?
- Will you store shards of containers in persistent memory?
- What formats would you use?
- Do you want data awareness in HDF5 or MPI-I/O?

Data access

- Will users expect POSIX access to tiered storage?
- Will users need to share data stored in persistent memories?

System level and operator considerations

- Do you want data management control of both memory tiers and storage tiers?
- Does the PFS have a place in a memory hierarchy?
- Would you like to control persistent memory reservations via WLM only?
- Will you like to control stage-in/stage-out via WLM or other system level control?
- How will you control usage of tiers? Via quota or assignment?
- Should data on tiers expire?
- What data movement tools would you like?
- Do you want to drive data movement via policies?

Other

- What do you expect to happen when the tiered storage fills during an application run?
- Will users want fine-grained control over data marshaling?
- What de facto data management tools and methods should Cray users adopt?

Discussion and Q&A

Cory Spitz
spitzcor@cray.com

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Name: Email:

2016 CUG BoF on Parallel Filesystem Access

Abstract: Evolving parallel file systems in response to the changing storage and memory landscape

Burst buffers and storage-memory hierarchies are disruptive technologies to parallel file systems (PFS), but there isn't consensus among members of the HPC community on how PFSs should adapt to include their use, if at all. There also isn't consensus on how HPC users should ultimately use and manage their data with these emerging technologies. In this BoF we will discuss how HPC and technical computing users want to interact with burst buffers or other storage-memory hierarchies and how their PFS should adapt. What do they expect? Will they want to continue to use POSIX-like semantics for access like with MPI-I/O or HDF5 containers? What do users expect for legacy codes? Generally, what do users, application developers, and systems engineers require? Will they accept exotic solutions or must a de facto industry standard emerge?

Premise/inquiry

- **Assertion: there is no one canonical definition of an exascale parallel file system**
  - We all seem to agree that there will be lots of devices, components, and threads
  - Can't agree on a single solution for organization, workflows, access methods, or usage/semantics
  - Likely there will be tiered storage architectures as with DataWarp
  - Multiple solutions should emerge, even hybrids
  - For sure, things will be complex
  - How will these complex systems be productive?

Survey Questions

To explore possible solutions to the inquiry, let's discuss the following questions.

Today's PFS users are accustomed to dealing with data placement issues such as file/directory striping including stripe width and count.

1. What control do you expect users to display with a storage hierarchy?
   Total control, some control, no control (circle one)
2. With a memory hierarchy?
   Total control, some control, no control?
3. In what way should the control be expressed? (examples: size, physical location, or unexpressed)

Please, provide additional feedback:

Name: Email:

How do you expect to manage data in persistent memory vs. data in the parallel file system?

4. Do you want a transparent file system cache?
   Yes or no? (circle one)
5. Do you want system control (perhaps via job control)?
   Yes or no?
6. Do you want user control (application control)?
   Yes or no?
7. Do you want all of the above? That is do you want to mix control methods?
   Yes or no?
8. How do you envision marshalling data between the two?
   Explicit, implicit, or both (circle one)
9. Should there exist an API to the PFS?
   Yes or no?

   Please, provide additional feedback:

What is the appetite for application changes?

10. Is there a requirement to run legacy codes?
    Yes or no?
11. Will users rewrite codes to extract additional functionality?
    Yes or no?
12. Will users rewrite codes to extract additional performance?
    Yes or no?
13. Do you want a middle layer to control application interaction?
    Yes or no?

    Please, provide additional feedback:

Data formats

14. Would you like to use data containers?
    Yes or no?
15. Will you store shards of containers in persistent memory?
    Yes or no?
16. What formats would you use? Explain:

17. Do you want data awareness in HDF5 or MPI-I/O? Explain:

Please, provide additional feedback:

Name: Email:

Data access

18. Will users expect POSIX access to tiered storage? Explain:

19. Will users need to share data stored in persistent memories? Explain:

Please, provide additional feedback:

System level and operator considerations

20. Do you want data management control of both memory tiers and storage tiers?
    Yes or no?
21. Does the PFS have a place in a memory hierarchy?
    Yes or no?
22. Would you like to control persistent memory reservations via WLM only?
    Yes or no?
23. Will you like to control stage-in/stage-out via WLM or other system level control? Explain:

24. How will you control usage of tiers? Via quota or assignment?

25. Should data on tiers expire?
    Yes or no?
26. What data movement tools would you like? rsync, psync, fcp, lustre-data-mover, or other?

27. Do you want to drive data movement via policies?
    Yes or no? If yes, please explain:

    Please, provide additional feedback:

Other

28. What do you expect to happen when the tiered storage fills during an application run?

29. Will users want fine-grained control over data marshaling?
    Yes or no?
30. What de facto data management tools and methods should Cray users adopt?