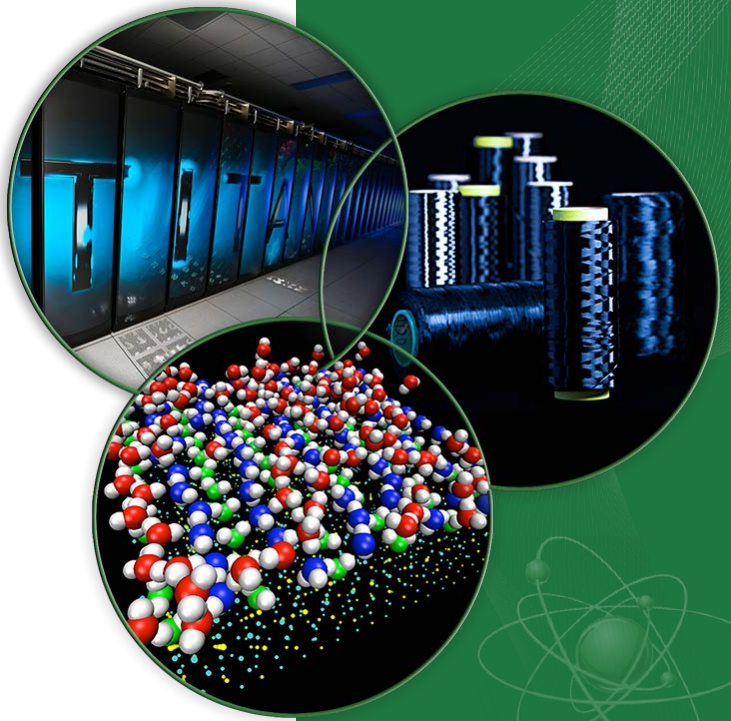
Oak Ridge National Laboratory
Computing and Computational Sciences Directorate

Evaluating Progressive File Layouts for Lustre



Rick Mohr

University of Tennessee

Lustre User Group
April 5-7, 2016

ORNL is managed by UT-Battelle
for the US Department of Energy

Collaborators

ORNL

- Michael Brim
- Jesse Hanley
- Neena Imam
- Sarp Oral

Intel

- Andreas Dilger
- Richard Henwood
- Zhenyu (Bobijam) Xu
- Niu Yawei
- Fan Yong

Outline

- Current state of Lustre layouts
- Progressive File Layout (PFL) overview
- PFL prototype implementation
- PFL evaluation tests
  - Streaming IOR tests
  - Comparison to synthetic dynamic striping
  - Object placement testing
- Summary and future work

Lustre File Layouts Today

- Several Lustre parameters control file layout
  - Stripe size/count/index
  - OST pool
- In practice, stripe size and count are primarily used
- Layout constraints
  - One set of parameters for entire file
  - Parameters chosen at time of file creation
  - Only one layout type (RAID-0) supported

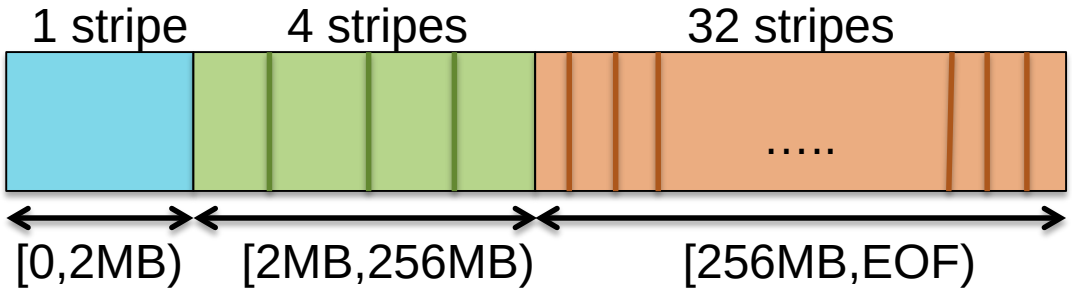
Layout Enhancement

- Under OpenSFS contract SFS-DEV-003, Intel's High Performance Data Division produced a new design for file layouts
- High Level Design document describes several new layouts
  - Composite layouts
  - RAID layouts
  - Compact layouts
  - Large layouts

http://wiki.lustre.org/Layout_Enhancement_High_Level_Design

Progressive File Layout (PFL)

- Progressive File Layout feature is built using Composite Layouts
  - File layout is described by a series of components
  - Each component covers a non-overlapping extent of the file
  - Each component has its own striping parameters



Progressive File Layout Goals

- Single layout definition for multiple file sizes
  - Reasonable performance for a variety of I/O patterns
  - Change stripe layout as file grows
  - Simplify Lustre usage for novice users
- More striping options for advanced users
  - Customization for non-uniform files
  - Different regions of file on different storage
- Stepping stone to more features in the future
  - HSM for file components, new uses for Composite Layouts, etc.

Other PFL Use Cases

- Layout with increasing stripe count not the only use case
  - Striping parameters for each component are independent
  - Choose striping parameters to match the problem
- Examples
  - File striping based on number of processes
    - First component contains shared header (stripe_count=1)
    - Second component contains data for each process (stripe_count=N)
  - Decreasing stripe count
    - First component contains large amounts of raw data
    - Second component contains small post-processed data

PFL Prototype Implementation

- Intel, under contract from ORNL, is developing PFL feature
- A prototype implementation was delivered in the first half of 2015 for evaluation and testing
  - Some limits on functionality
    - No dynamic allocation of new components
    - No support for setting PFL on directories
- Continuing development
  - PFL inheritance from parent directory
  - Integration with existing Lustre code base

PFL Evaluation Tests

- Several different tests were run to evaluate the functionality and performance of PFL prototype
- Streaming IOR and mdtest (Intel)
  - Comparison with traditional Lustre striping
  - Single client and multiple clients
  - Shared file and file per process
- Comparison to Synthetic Dynamic Striping (ORNL)
- Object placement (ORNL)
  - Compare OST object placement for PFL and traditional striping

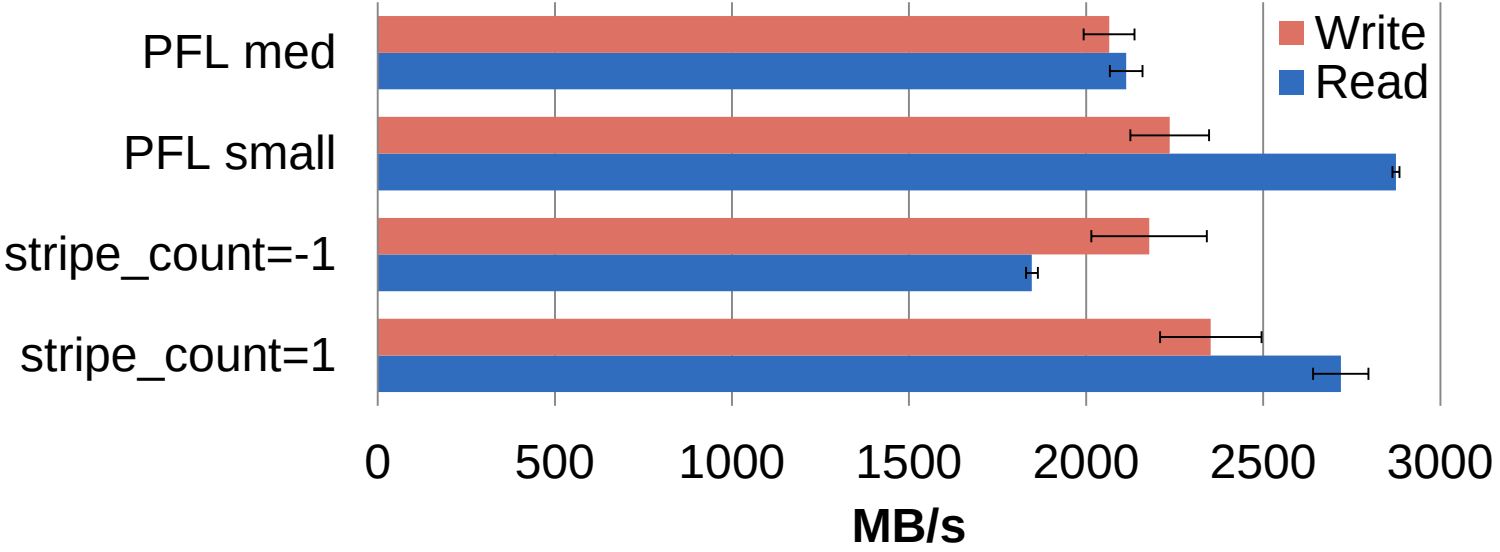
Intel Testing

- Intel's tests were run on Hyperion at LLNL
  - 32 Lustre clients
  - 16 Lustre servers with 52 OSTs (ldiskfs)
  - Mellanox DDR Infiniband
- IOR
  - Single client (16 threads) file per process
  - 32 clients (512 threads) file per process
  - 32 clients (512 threads) shared file
  - Each thread reads/writes 4 GB of data

Test File Layouts

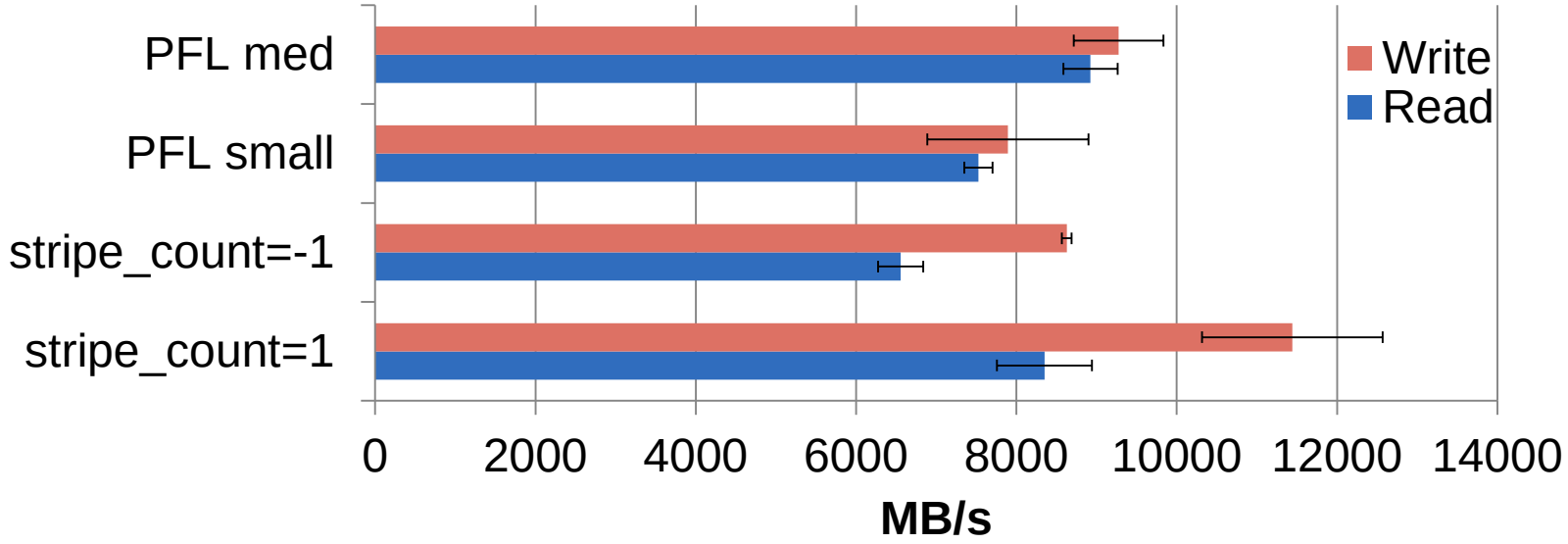
- Traditional
  - stripe_count = 1
  - stripe_count = -1
- PFL small
  - [0,EOF) stripe_count=1
- PFL medium
  - [0,16M) stripe_count=1
  - [16M, EOF) stripe_count=4
- PFL large
  - [0,16M) stripe_count=1
  - [16M, 128M) stripe_count=4
  - [128M, EOF) stripe_count=47

Single Client IOR (FPP) – 16 Threads



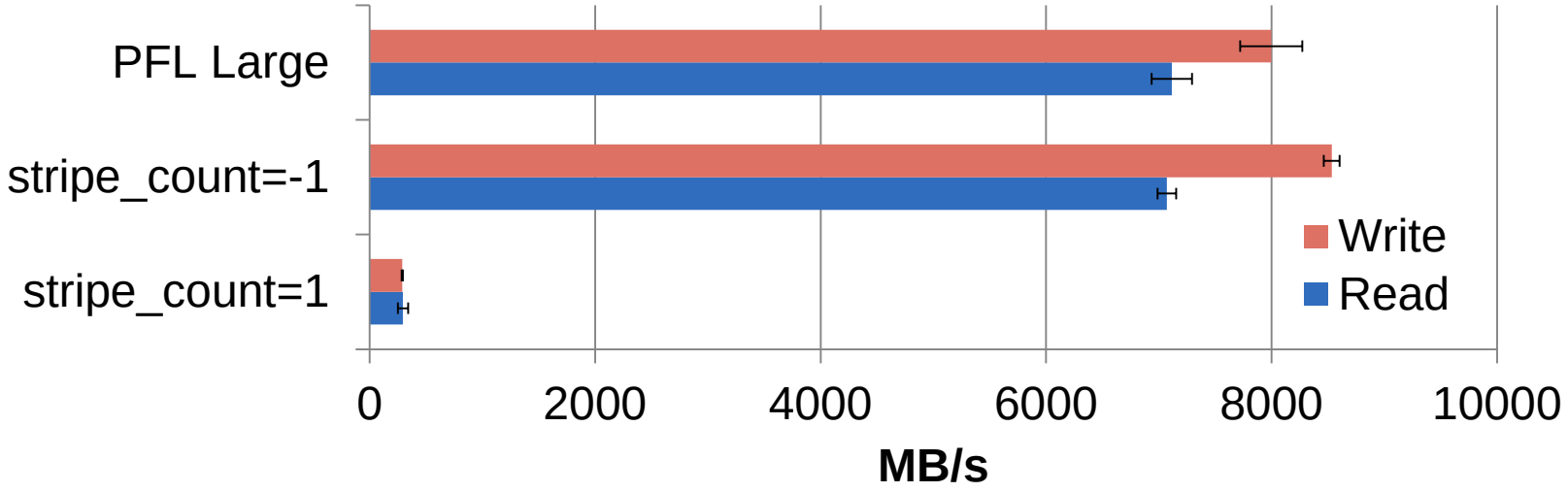
- PFL small and stripe_count=1 are comparable for reads and writes
- PFL med reads are faster than stripe_count=-1 due to less contention
- PFL med write speed is less than stripe_count=-1 (somewhat surprising)
- Higher variance in write speeds possibly due to contention on system

Multi-Client IOR (FPP) – 512 Threads



- Read performance of PFL small is comparable to stripe_count=1
- Write performance of PFL small is much less than stripe_count=1 (large variances could be due to contention on shared test system)
- PFL med performance is better than stripe_count=-1 (less contention)

Multi-Client IOR (Shared) – 512 Threads



- Performance of PFL large and stripe_count=-1 are comparable
- Performance for stripe_count=1 is much lower due to contention (as expected)

ORNL Lustre Testbed

- 8x OSS Servers (Dell R720)
  - 2x Intel Xeon 2630 v2
  - 64 GB RAM
  - Mellanox ConnectX-3 FDR HCA
- 2x MDS Servers (Dell R720)
  - Same as OSS servers except with 128 GB RAM
- 8 OSTs per OSS server
  - OSTs use ZFS backend

Synthetic Dynamic Striping

- Prior to the PFL prototype, a simulated form of dynamic striping was tested
  - Files were split into smaller components
  - Each component was created in a different directory with different stripe counts
  - Applications were modified to perform I/O to file components
    - IOR
    - BLAST

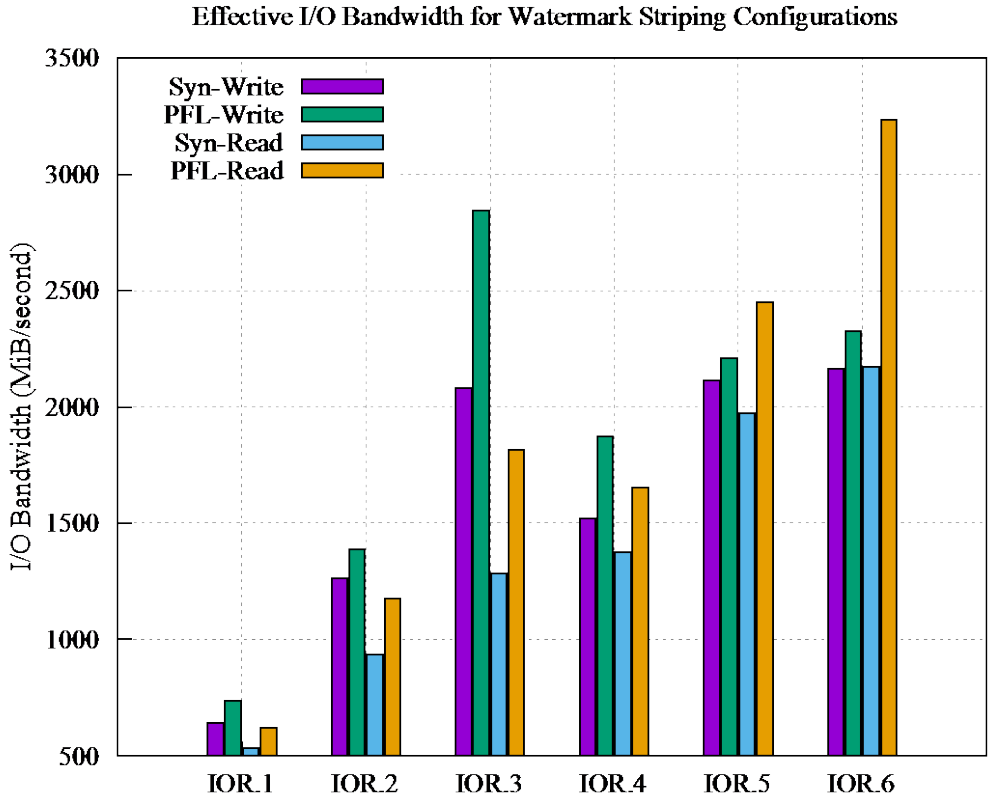
http://arxiv.org/pdf/1504.06833v1.pdf

Dynamic Striping and PFL Test Cases

- IOR POSIX shared file
  - 16 compute nodes with 4 processes per node
  - 4 TB total file size

Standard	Dynamic / PFL
[IOR.1] Entire file: stripe_count=4	[IOR.4] 0-1TB: stripe_count=4 Remainder: stripe_count=8
[IOR.2] Entire file: stripe_count=8	[IOR.5] 0-1 TB: stripe_count=4 Remainder: stripe_count=16
[IOR.3] Entire file: stripe_count=16	[IOR.6] 0-1 TB: stripe_count=4 1-2 TB: stripe_count=8 Remainder: stripe_count=16

PFL vs. Synthetic Dynamic Striping



- Direct comparison of bandwidth between PFL and synthetic striping not really “fair” since different Lustre versions used
- Important result is that PFL behavior follows same trend as synthetic striping

Object Placement Testing

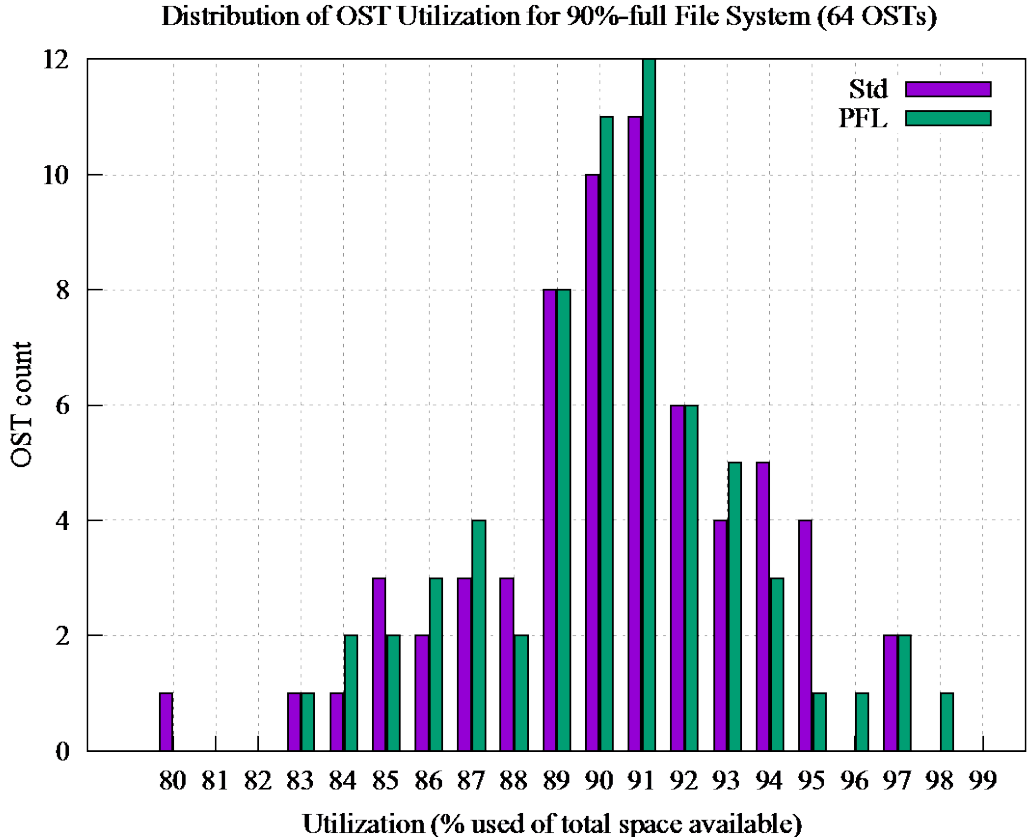
- Users often choose poor striping patterns
  - Large file, small stripe count → Imbalanced OST usage
  - Small file, large stripe count → Sub-optimal performance
- PFL can use increasing stripe count to accommodate multiple file sizes
  - How does using a single PFL layout for all files compare to “ideal” traditional striping?
  - Test scenario
    - Create files with traditional striping and single PFL layout
    - Compare distribution of OST usage

Object Placement Testing (cont.)

- Choose file size distribution (based on OLCF stats)
- Choose PFL Layout
  - [0, 1MB) stripe_count=1
  - [1MB, 64MB) stripe_count=4
  - [64MB, 128GB) stripe_count=16
  - [128GB, EOF) stripe_count=48
- Fill file system to 90% capacity

File Size	Percent	Stripe Count
1 MB	70%	1
64 MB	20%	4
128 GB	9%	16
4 TB	1%	48

Object Placement Results



- Distribution of OST utilization for PFL files is very similar to the distribution seen using “ideal” striping

Summary

- Progressive File Layouts provide additional flexibility when defining the striping configuration for a file
- PFL performance appears to be on par with traditional Lustre (and in some cases better)
- Single PFL layout can be used for files of varying sizes
  - Can simplify Lustre usage for users
  - Will save some headaches for sys admins
- For more test results, see

http://lustre.ornl.gov/ecosystem/documents/papers/LustreEco2016-Mohr-PFL.pdf

Future Work

- PFL Implementation
  - Layout inheritance from parent directory
  - Define PFL layout as default for file system
  - Improved OST allocator
  - Dynamic component instantiation
- PFL Testing
  - More data intensive workloads
  - Increase scaling

Acknowledgements

This work was supported by the United States Department of Defense (DoD) and used resources of the Computational Research and Development Programs at Oak Ridge National Laboratory.

Questions?