LA-UR-13-26028
Approved for public release; distribution is unlimited.

Title: Improving Small File Creation Performance in a Parallel Log-Structured File System

Author(s): Bonnie, David J

Intended for: LANL Mini-Showcase

Issued: 2013-07-31

Los Alamos
NATIONAL LABORATORY

Improving Small File Creation Performance in a Parallel Log-Structured File System

August 1, 2013

David Bonnie

LA-UR-13-

NNSA
National Nuclear Security Administration

Many currently contributing to PLFS

- LANL: David Bonnie, Hugh Greenberg, Gary Grider, Brett Kettering, David Shrader, Aaron Torres, Alfred Torrez
  - Past Contributors: Ben McClelland, Meghan McClelland, James Nuñez
- EMC: John Bent & The EMC Engineering Team
- Carnegie Mellon University: Chuck Cranor, Garth Gibson, Annika Peterson
- Other Academics: Jun He (UW-Madison), Kshitij Mehta (U. Houston)

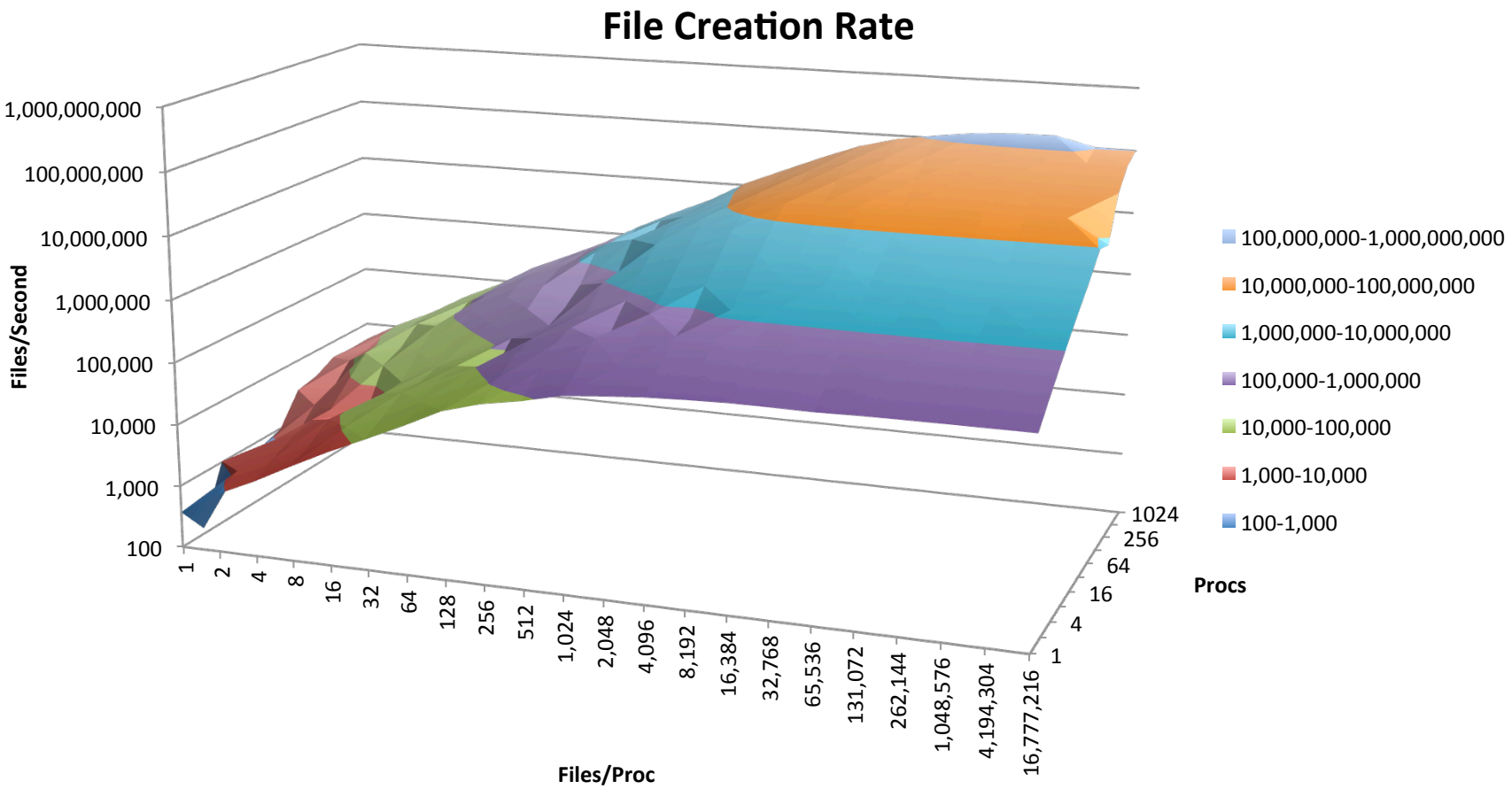
The problem

- Creating small files on a parallel file system is slow
- PLFS has a new “1-N” small file mode to help mitigate this
  - Initial results were promising; ~300x speedup over PLFS N-N
  - 20,000 / second -> ~7,000,000 /second w/ 1024 pes
- Essentially transforms a metadata problem into a data problem through file packing
  - 1 logical create per process instead of 1 create per file
  - Should be able to saturate the network…but it wasn’t.
  - Why?

Low hanging fruit

- 7M files / second is roughly 270 MB/s in raw metadata throughput
- Test platform can do ~6 GB/s…
- Why can't we push it faster?
  - First, improve process->file efficiency with glibc buffering
  - Second, profile running code to find obvious inefficiencies
    - New 1-N mode changed code path / loops
    - Restructure tightly looped and/or slow code (C++ strings!)
  - Third, reduce memory usage where possible

Peak of 146,391,418 creates/sec at 1024 pes and 1M files per pe



Other misc

- The rollover from these optimizations into N-1 and N-N modes was not insignificant
- Small block size writes drastically improved
  - Due to both code efficiency improvements and glibc buffering
- Found various bugs and corrected them through the process
  - Thank you Valgrind!
- Integrated new features including buffering into the new configuration architecture (YAML config files)