CSE 726
Data Intensive Distributed Computing
Spring 2011

A Review of PVFS: A Parallel File System for Linux Clusters, *P. Carns et al., Linux Conf. 2000.*

Based on Presentation by: Sathish
Written by: Rishi Baldawa

In PVFS, Carns et al try to develop a Virtual Parallel File system on top Local File System, hence the name PVFS, for Linux Systems providing dynamic distribution of IO and meta data. The main goals of the authors were high bandwidth concurrent read/writes from multiple nodes to a single file, support for multiple APIs, use of common UNIX commands for DFS, Application of APIs without constant recompilation, robustness, scalability and easy installations and use. The Paper provided a relatively cheap Distributed File System that could be easily applied to any Linux Based Cluster independent of significant hardware requirements and could be use for the applications such as scientific research, media streaming, complex computations etc.

PVFS allows user(s) to store and retrieve data using common UNIX commands (such as ls,cp and rm) where data stripped in round robin fashion and stored on multiple independent machines with different network connections. Data is stored in a distributed fashion to reduce single file bottlenecks and increase the aggregate bandwidth of the system. The systems are treated as nodes (or daemons) which can perform one or more of the following operations:

- Run applications,
- Handle metadata
- Store information for I/O operations

The major components of PVFS are File Manger (which manages all the meta data for PVFS Files), IO servers (which handle the storing and retrieval operations performed on the distributed data stored on local disks), PVFS native APIs (libpvfs is used to provide user-level access to PVFS servers to perform operations) and PVFS Kernel Support provides the functionality to mount PVFS on Linux-OS nodes. There can be only one File Manager for the whole PVFS system. Applications can access data in 3 ways: Linux Kernel interface, PVFS native API and ROMIO MPI-IO interface.

Advantages:

- PVFS is easy to install and configure.
- It provides high performance for I/O intensive parallel or distributed applications.

- It is also compatible with existing applications so that you don't have to modify them to run with PVFS.
- PVFS is well supported by the developers due its openness.

Disadvantages:

- PVFS is stateless hence cannot provide data redundancy or fault tolerance.
- As it has a single manager, this can lead to many bottle necks at management level especially when the size of cluster increases.
- PVFS uses TCP/IP and hence incurs all the issues of TCP/IP protocol.
- PVFS has a poor security model.
- PVFS is OS dependent.

In the presentation and the discussion that took thereafter, certain points were made. the whole process of making calls are user level is not completely efficient but the way to make system calls in a simplified manner to support other features such as ease of use, scalability, robustness, etc. Also, when the Kernel is used during trapping calls with PVFS library loaded, the application doesn’t really use PVFS syscall wrapper hence the Figure 4. B is ambiguous. PVFS is stateless and hence doesn’t support redundancy and fault tolerance. The strip size is calculated dynamically. The size of fragments is fixed (unless it is the last stripe in which case it could be less than the fixed one) and such that there are no read bottlenecks when the data is retrieved. The discussion was brief and could not be extended due to time constraints.