POSIX IO extensions for HPC

Brent Welch, Panasas, [welch@panasas.com]

There is a group of HPC afficiandos that are defining some new POSIX file system APIS that are cluster and parallel processing friendly. I presented slides on this at SC05, and can easily replay the information in, e.g., 5 minutes or less.

Here is an outline of the information:

POSIX IO APIs (open, close, read, write, stat) have semantics that can make it hard to achieve high performance when large clusters of machines access shared storage.

A working group (see next slide) of HPC users is drafting some proposed API additions for POSIX that will provide standard ways to achieve higher performance.

Primary approach is either to relax semantics that can be expensive, or to provide more information to inform the storage system about access patterns.

POSIX was created when a single computer owned its own file system.

Network file systems like NFS chose not to implement strict POSIX semantics in all cases (e.g., lazy access time propagation)

Heavily shared files (e.g., from clusters) can be very expensive for file systems that provide POSIX semantics, or have undefined contents for file systems that bend the rules

The goal is to create a standard way to provide high performance and good semantics

Ordering (stream of bytes idea needs to move towards distributed vectors of units)
   readx(), writex()

Coherence (last writer wins and other such things can be optional)
   lazyio_propogate(), lazyio_synchronize()

Metadata (lazy attributes issues)
   statlite()

Locking schemes for cooperating processes
   lockg()

Shared file descriptors (group file opens)
   openg(), sutoc()

Portability of hinting for layouts and other information (file system provides optimal access strategy in standard call)

The full slide deck has excerpts from the POSIX man pages that the group is creating so the APIs have meat behind them.