I/O & API

SANDEEP KORRAPATI

Overview

- I/O Significance
- Standard Semantic
  - POSIX
  - MPI
- Coordinaton
- Atomic Operators
- DISC & HPC

I/O Significance

- Larger parallel computers as a result of the demands in Computational Science.
- I/O bound Computational science applications.
- High-level parallel I/O libraries, I/O middleware, and parallel file systems require increasing specialization.

Need for Parallel I/O

- Sequential I/O becomes a bottleneck
- Tedious pre- and post-processing is necessary to split and merge global files if parallel I/O is not used.
- Standard Unix I/O is not portable.
- True parallel I/O requires multi-process access from a parallel file system.

POSIX

- Portable Operating System Interface – IEEE Standard.
- Assures code portability.
- Implementing POSIX semantics in multi-node file systems is extremely difficult.
  - Ex: Immediate visibility of Writes to a File.
- Many file access protocols chose not to implement POSIX semantics.
- POSIX Compliance & Conformance

POSIX Contd...

- Allows to overlap compute tasks with I/O processing, to increase determinism.
- Supported functionality:
  - Send multiple I/O requests at once from different sources
  - Cancel ongoing I/O requests
  - Wait for request completion
  - Inquire the status of a request: completed, failed, or in progress

POSIX API Issues

- “Stream of Bytes“ – Ordering
  - Developed for a Single Machine with a single memory sapce to a streaming device.
- Extensions
  - A set of mandatory functions and a set of extensions to that set of mandatory functions.
  - Awkward for use in HPC. Ex: listio
- Metadata
  - A standard, portable set of semantics would be useful.

POSIX API Issues, Contd...

- Locking Schemes
  - To support cooperating groups of processes is also necessary.
  - Assumes, only a a single process would want exclusive write access at an instance.

MPI

- MPI → Message Passing Interface
- A specification for developers and users of message passing libraries.
- Goal: To provide a widely used standard for writing message passing programs.
- The interface attempts to be
  - Practical
  - Portable
  - Efficient
  - Flexible
- Interface specifications have been defined for C/C++ and Fortran programs.

Reason for Using MPI

- **Standardization**
  - MPI is the only message passing library which can be considered a standard.
- **Portability**
  - No source code modification when an application is ported to a platform that supports the MPI standard.
- **Performance Opportunities**
  - Vendor implementations should be able to exploit native hardware features to optimize performance.
- **Functionality**
  - Over 115 routines are defined in MPI-1 alone.
- **Availability**
  - Variety in both vendor and public domain.

MPI-2

- **Dynamic Processes**
  - Extensions that remove the static process model of MPI. Provides routines to create new processes.
- **One-Sided Communications**
  - Shared memory operations (put/get) and remote accumulate operations.
- **Extended Collective Operations**
  - Non-blocking collective operations and application of collective operations to inter-communicators
- **External Interfaces**
  - Routines to layer on top of MPI, such as for debuggers and profilers.
- **Additional Language Bindings**
  - Describes C++ bindings and discusses Fortran-90 issues.
- **Parallel I/O**
  - Describes MPI support for parallel I/O.

MPI-IO

- Ememrging Standard mechanism for file I/O within HPC applications.
- MPI-IO relaxes POSIX semantics
- Interface that allows applications to manage cache coherency themselves.
- MPI_FILE_SYNC guarantees freshness of data.
- Similar to NFSv4 semantics in how it flushes and revalidates data at specific points in time.

MPI-IO Contd...1

- SYNC/BARRIER/SYNC
  - To ensure data consistency among nodes.
  - Is a way to enforce an ordering and separation of I/O operations in time.
  - 1. SYNC: Places written data on filing servers.
  - 2. BARRIER: Clients wait for other clients to flush dirty data to the servers, ensuring that no client issues read requests until all clients have the same view of file contents.
  - 3. SYNC: Perform file revalidation by ensuring written data is visible to all nodes.

MPI-IO Contd...2

- Features of MPI-I/O
  - Parallel read/write
  - Non-contiguous data read/write
  - Non-blocking read/write
  - Collective read write
  - Portable data representation across platforms
  - HPF style distributed array syntax
- Advantages of MPI-I/O
  - Performance
  - Portability
  - Convenience

Coordination of Access

- File system for coordiantion of access in Distributed systems.
  - POSIX consistency semantics or Locks.
- Features of File System over message passing and shared memory.
- File System to manage persistent shared data.
- Forms of coordination in the file system include exclusive access to shared data and atomic updates to metadata.

Coordination of Access, Contd..

- Implicit Coordination: Sequential consistency of I/O accesses.
- File system interfaces to explicitly coordinate between processes.
- Coordination provided through locks traditionally.
- Critical accesses to the file when locked.

Active Storage

- Model of computation that moves functions from compute nodes to the file system or storage device.
- Data accesses become local operations at the storage device.
- Thus simplify coordination of File Accesses.

Atomic Operations in Filesystems

- File systems provide tighter coordination of I/O accesses than is often required by the application.
- As file systems are forced to provide the same semantics for all applications.
- Proposal: A set of basic atomic operators to allow the application to construct their own coordination models efficiently.

Properties of Atomic Operators

- State Variables
  - The state that the operation is performed on.
  - Typed state variables and interfaces to create, modify and remove state.
- Persistence
  - Values across application runs, or among many applications.
- Global Namespace
  - To uniquely reference state variables and perform atomic operations on them.
- Per-File Granularity
  - An application coordinates processes around a file. Persistent state variables should exist on a file.

Operators on Attributes

- Atomic operators were implemented using extended attributes.
- Persistent variables that can be placed on a file.
  - Name – String scoped to the file
  - Value – Variable length buffer
- Leveraging extended attributes avoids introducing new file system interfaces or requiring any modifications to the client operating system or file system software.
- Extended attributes as functions, to support Atomic operators.

Queue Interfaces

- Coordinating between processes, as well as communicating with processes to grant lock requests.
- Notification mechanisms built into file systems, add complexity.
- Simple Queue Interface
  - Enqueue Operation
  - Dequeue Operation

Coordination Outside File System

- The MPI-2 one-sided routines provide atomic access to an entire memory region, allowing for a sufficiently clever algorithms built on top of MPI-2 RMA methods.
- Drawback: Independent shared file pointer operations may stall the locking algorithm’s progress

DISC &HPC

- Big Data – IT Industry and Scientific Computing Community.
- Data Intensive Scalabel Computing
  - Manage, process and store massive data-sets in a distributed manner.
- Internet services stack of Google(GoogleFS) & Yahoo(HDFS):
  - Distribute Processing: MapReduce, Hadoop
  - Program Computation: Sawzall, Pig
  - Store Data: Bigtable, HBase

DISC &HPC, Contd...

- High Performance Computing – Cluster File Systems
- Cluster file systems: necessary to meet the scalabilty demands of highly parallel I/O access pattterns
- HPC
  - Distributed-memory numerical simulations
  - MPI, Infiband
- DISC
  - Web search and Data analytics
  - MapReduce, Gigabit Ethernet

DISC &HPC, Contd...

- Main Issues:
  - Parallel programming, fault tolerance, data distribution and resource load balancing.
- Much of DISC functionality can be supported by HPC file systems.
- The other direction, serving HPC applications with DISC file systems, is not clear.
- Scalability is the main reason for abandoning POSIX.
- A little relaxing of POSIX in most HPC Filesystems.

DISC &HPC, Contd...

- Parallel processing of Massive Data Sets(DISC).
- Large Computation divided into many tasks.
- Avoids potential bottlenecks.

DISC data processing extensions to HPC file systems.

- HDFS‘s achitecture resembles a HPC parallel file system.
  - Data and Metadata on different servers.
  - Files are divided into chunks.
- Differences
  - HDFS exposes file layout information to Hadoop
  - Fault tolerance: HDFS uses triplication instead of RAID.

Experiment

- A shim layer to plug PVFS into Hadoop framework stroting data on compute nodes.
- Queries layout information from underlying parallel filesystem.
- Emulates triplication by writing on behalf of client.
- Result: „Hadoop-on-PVFS“ comparable to „Hadoop-on-HDFS“

Conclusion

- Compliance to POSIX IO with some additional features.
- MPI IO : As it is already successful with many computing systems.
- Active Storage.
- Atomic Operators (with some further study).

- Thank You.