Research Directions in Parallel I/O for Clusters

Walt Ligon
Parallel Architecture Research Lab
Clemson University

Cluster 2002

Basic Tenets

- Parallel I/O: critical problem for cluster computing
  - Important applications need high performance parallel I/O
  - Enough hardware to deliver the required performance
- Software remains in research and development
  - Have achieved remarkable goals in one or more key areas
- Great reluctance to commit to any file system
  - File systems do not address enough issues at once
  - Package is not robust enough for widespread use

Critical Goals

- **High performance with scalability**
- **Flexible, efficient integration with parallel codes**
- **Reliability/fault tolerance**
- **Portability, manageability**

Research Issues

- Interfaces and semantics
- Distributed locking, caching, and redundancy
- Implementation methods
- Benchmarking and other evaluation methods

example: PVFS v2

Talk Outline

- **Interfaces and semantics**
- Locking and atomicity
- Redundancy and reliability/fault tolerance
- Implementation and portability
- Benchmarking

Issues with Interfaces

- Compatibility and portability
  - With old utilities (like Posix)
  - With existing programming models (like MPI)
  - With various internal interfaces (like VFS)
- Extra information
  - Non- contiguous requests
  - Data distribution
  - Semantic issues
- Partial completion status
  - Fault detection / recovery

PVFS v2 Interfaces

- Guiding principles
  - Expandability
  - Feature availability
- Server/client request protocol
  - Architecture independent
- System interface
  - VFS- like, exposes all internal features
- User interfaces
  - Posix- like
  - MPI- IO

Issues with Semantics

- Caching
  - Data (and forced write- back)
  - Directory entries
  - Metadata
- Locking
- Concurrent access
- Redundancy and recovery
- Security

PVFS v2 Semantics

- Guiding principles
  - Semantics often conflict with performance goals
  - No single set of semantics is right for every situation
- High- performance choices
- Implementations of alternative choices supported
  - caching
  - redundancy
  - locking
- Expect more choices in the future

Talk Outline

- Interfaces and semantics
- **Locking and atomicity**
- Redundancy and reliability/fault tolerance
- Implementation and portability
- Benchmarking

Distributed Locking

- Region- based locks are still used in file systems
  - Work well in hardware but
  - Not scalable in software
  - Mostly used to achieve atomicity
- Atomicity in metadata and some data operations
  - Can be implemented without locks
  - May be provided by client (service is not needed)
- Implemented with locks
  - Lots of state on clients
  - Lots of I/O, poor scalability

Conditional Operations

- Taken from modern SMP hardware designs
  - Load Locked
  - Store conditional
- Allows local operations to proceed
- Conditional store operations check for atomicity violation
- Could this be applied to a parallel file system?

PVFS v2 Approach

- Clients obtain version tags (vtags) during read.
- Vtag identifies a region and a state.
- Conditional write only succeeds if vtag is current
- Can build locks from this primitive
- But ...
  - This does not solve all locking problems
  - Poor performance in pathological cases if not implemented well

Talk Outline

- Interfaces and semantics
- Locking and atomicity
- **Redundancy and reliability/fault tolerance**
- Implementation and portability
- Benchmarking

Redundancy in Parallel File Systems

- Typical approach is to use RAID redundancy
- Significant performance/scalability issues
  - Locking issues
  - Bottleneck issues
  - Extra I/O
- Parity is slow, mirroring faster

Don’t Need Redundancy All The Time

- **Redundancy on demand**
  - **Scratch files**
  - **Checkpoint/commit**
  - **Long- term storage**
- **Need selectable redundancy policy**
  - **Multiple redundancy mechanisms**
  - **Mirroring vs. Parity**
  - **On update vs. on commit/close**

PVFS v2 Redundancy

- Redundancy support in distribution subsystem
- Fault- tolerant interface design
- Redundancy levels
  - Mirroring
  - Lazy Redundancy
    - on close
    - on commit
    - partial redundancy
- Depends heavily on atomic operation capability

Talk Outline

- Interfaces and semantics
- Locking and atomicity
- Redundancy and reliability/fault tolerance
- **Implementation and portability**
- Benchmarking

Implementation Issues

- PVFS modules
  - network transports (BMI)
  - storage (Trove)
  - flow protocols
  - distributions (and redundancy)
  - requests
- Request "wire" protocol
- Independent of OS structures and types

Talk Outline

- Interfaces and semantics
- Locking and atomicity
- Redundancy and reliability/fault tolerance
- Implementation and portability
- **Benchmarking**

Benchmarking

- Need standardized benchmarks for parallel I/O
  - measurement procedure
  - reporting format
  - terminology
- Test a range of workloads
  - small/large transactions
  - contiguous/non- contiguous
  - metadata operations
- Both synthetic and application benchmarks

I/O Benchmark Consortium

- **Open group working to establish an effective set of benchmarks for parallel I/O**
- **Have national lab and university involvement**
- **Need industry involvement**
- **Need input from applications groups**

`http://www.mcs.anl.gov/~rross/pio-benchmark/index.html`

Conclusions

- **Important research issues**
  - **locking, redundancy, scalability**
  - **interfaces, semantics**
- **We need a joint effort to reach goals**
  - **open, flexible, common platform**
  - **good benchmarks**

Conclusions

- Important research issues
  - locking, redundancy, scalability
  - interfaces, semantics
- We need a joint effort to reach goals
  - open, flexible, common platform
  - good benchmarks
- The conference is over - I need a beer!!!!