Abstract Storage
Moving file format-specific abstractions into petabyte-scale storage systems

Joe Buck, Noah Watkins, Carlos Maltzahn & Scott Brandt

SRL
UCSC Systems Research Lab
Introduction

• Current HPC environment separates computation from storage
  – Traditional focus on computation, not I/O
  – Applications require I/O architecture independence

• Many scientific applications are data intensive
• Performance increasingly limited by data-movement
HPC Architecture

Gateway nodes
run parallel file system
client software and
forward I/O operations
from HPC clients.

Commodity network
primarily carries storage traffic.

Storage nodes
run parallel file system
software and manage
incoming FS traffic
from gateway nodes.

Enterprise storage
controllers and large racks
of disks are connected via
InfiniBand or Fibre Channel.

640 Quad core PowerPC
450 nodes with 2 Gbytes
of RAM each

900+ port 10 Gigabit
Ethernet Myricom
switch complex

136 two dual core
Opteron servers with
8 Gbytes of RAM each

17 DataDirect S2A9900
controller pairs with 480
1 Tbyte drives and 8
InfiniBand ports per pair

HW bottleneck is here. Controllers
can manage only
4.6 Gbyte/sec.
Peak I/O system
bandwidth is
78.2 Gbyte/sec.

Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

Diagram courtesy of Rob Ross, Argonne National Laboratory
HPC Architecture

Current bottleneck in the controllers

Gateway nodes
run parallel file system client software and forward I/O operations from HPC clients.

Commodity network
primarily carries storage traffic.

Storage nodes
run parallel file system software and manage incoming FS traffic from gateway nodes.

Enterprise storage
controllers and large racks of disks are connected via InfiniBand or Fibre Channel.

640 Quad core PowerPC
450 nodes with 2 Gbytes of RAM each

900+ port 10 Gigabit Ethernet Myricom switch complex

136 two dual core Opteron servers with 8 Gbytes of RAM each

17 DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

Diagram courtesy of Rob Ross, Argonne National Laboratory
HPC Architecture

Future bottleneck:
I/O nodes / storage nodes
network

Gateway nodes
run parallel file system
client software and
forward I/O operations
from HPC clients.

Commodity
network primarily
carries storage traffic.

Storage nodes
run parallel file system
software and manage
incoming FS traffic
from gateway nodes.

Enterprise storage
controllers and large racks
of disks are connected via
InfiniBand or Fibre
Channel.

- 640 Quad core PowerPC
- 450 nodes with 2 Gbytes
  of RAM each
- 900+ port 10 Gigabit
  Ethernet Myricom
  switch complex
- 136 two dual core
  Opteron servers with
  8 Gbytes of RAM each
- 17 DataDirect S2A9900
  controller pairs with 480
  1 Tbyte drives and 8
  InfiniBand ports per pair

Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

Diagram courtesy of Rob Ross, Argonne National Laboratory
Approach:
Move functions closer to data

- Use spare CPU cycles at intelligent storage nodes
  - Replace communication with CPU cycles
- Provide storage interfaces with higher abstractions
- Enable file system optimizations due to knowledge of data structure
- Do this for small selection of data structures
  - This is not another object-oriented database!
Why Now?

- Parallel file systems move more intelligence into storage nodes anyways
- Advances in performance management and virtualization
- Moving bytes slated to be a dominant cost in exa-scale systems
- Scientific file formats and operators increasingly standard
  - NetCDF, HDF
- Structured abstractions have seen recent success
  - BigTable, MapReduce
  - CouchDB
Abstract Storage
Storage as an Abstract Data Type

• ADT decouples interface from implementation
• Only few ADTs necessary, e.g.:
  – Dictionary (Key/value pairs)
  – Hypercube (Coordinate Systems)
  – Queue
• Optimize each one for each parallel architecture
  – Data placement
  – Performance management
  – Buffer cache management (incl. pre-fetching)
  – Coherence
ADTs and Scientific Data

• Scientific data is normally multi-dimensional, lending itself well to this approach
  – Multi-dimensional and hierarchical structures are readily mapped onto data types

• Multiple structures mapped onto (portions) of the same data for more efficient access
  – Operate on the appropriate structure (matrix, row, element, etc)
Implementation Challenges

• **Programming model** for implementing ADTs
• Everything based on **byte streams**
  – Current storage APIs (e.g. POSIX)
  – Current file system subsystems
    • Buffer cache
    • Striping strategies
    • Storage node interfaces
• Need awareness of **structured data**
  – New interfaces at various storage layers
Prototype: Ceph Doodle

• Focus: Programming model for implementing ADTs

• Construction and test framework for:
  – Storage abstractions
  – ADT implementations
  – Programming models (flexibility, ease-of-use)

• Based on object-based parallel file system architecture (e.g. Ceph).
Ceph Doodle Features

• Rapid prototyping:
  – Uses RPC mechanism
  – Written in Python

• Support for plugins for different ADTs
  – Byte stream (implemented as storage objects)
  – Dictionary (implemented as **skip lists**)

Ceph Doodle Overview

Clients use application-specific interfaces

Data types are cross-cutting system modules

Striping and caching are optimized per data type

Mappings route ADT RPCs to storage nodes

Client Application

ADT-Operation(…)

Data Type

ADT-Operation(…)

RPC_X(Op, ObjID, Context)
RPC_Y(Op, ObjID, Context)
RPC_Z(Op, ObjID, Context)
...

Striping & Caching Strategy

RPC to OSD With Object

RPC ADT Operation(Object, Context)

Client OSD

Data Type

Clients use application-specific interfaces

Data types are cross-cutting system modules

Striping and caching are optimized per data type

Mappings route ADT RPCs to storage nodes
Dictionary Implementation: Skip lists
Splitting skip lists across nodes
Future Work

• Building on top of Ceph
  – New dynamically loadable object libraries
• Redesigning caching
  – Data structure boundary aware v.s. pages
  – Pre-fetching = access patterns = ADT parameters
• Rethinking striping strategies
• Unified views supported by virtual ADT layer
• Embedding versioning and provenance capturing into file system
Thank you

buck@cs.ucsc.edu