Abstract Storage: Moving file format-specific abstractions into peta-byte filesystems

Joe Buck, Noah Watkins, Carlos Maltzahn, Scott Brandt

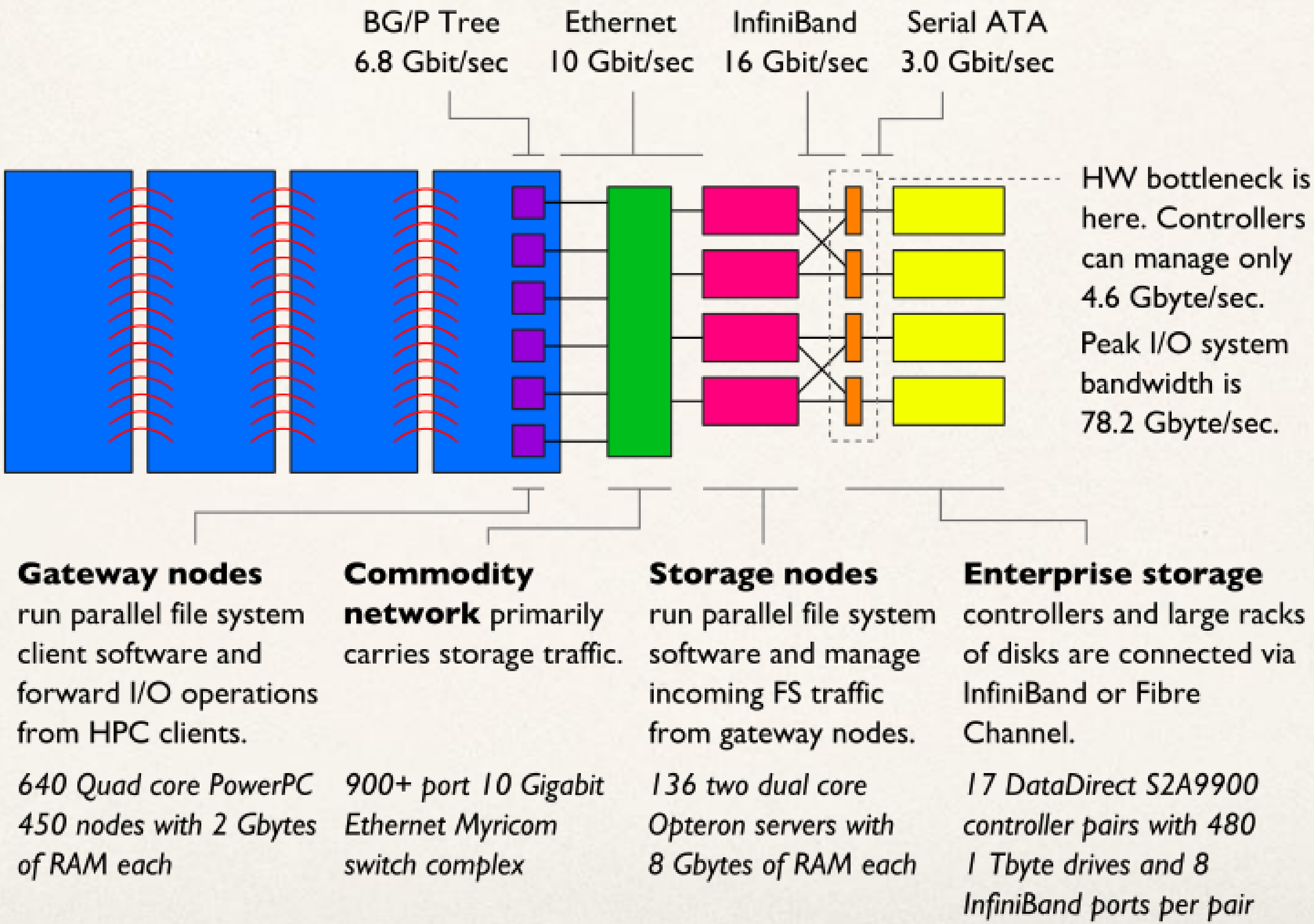
SRL Research Symposium - October 21st, 2009

Introduction

- Current HPC separates computation and storage
  - Focused on computation, not I/O
  - Applications require I/O independence
- Many new scientific applications are data intensive
- Data movement is becoming a bottleneck

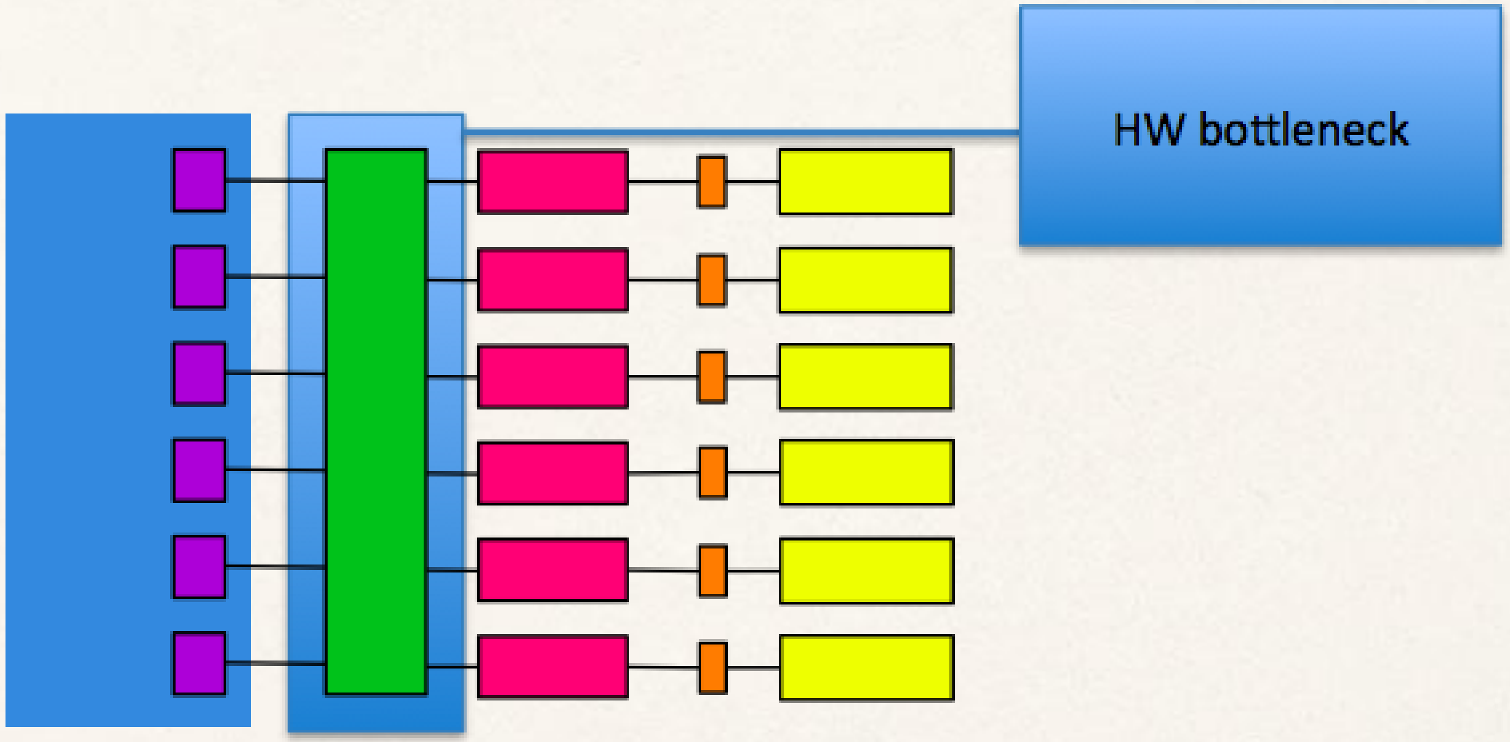
HPC Architecture

(diagram courtesy of Rob Ross, Argonne National Laboratory)



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

Future Bottlenecks



Higher number of smaller storage nodes
Compute/Storage boundary becomes the bottleneck

Our Solution:
Move functions closer to the data

Our Solution

- Use spare cycles on storage nodes
- Provide more abstract storage interfaces
- Maintain data’s structure in the storage system
- Small selection of structures & abstractions

Why now?

- Intelligent nodes in parallel filesystems
- Performance management and virtualization advances
- Data movement dominating cost in exa-scale
- Standardization of scientific format
- Recent successes of structured data

Abstract Storage

- Treat storage like abstract data types in code
- Only a few ADTs are necessary
  - Dictionary, Hypercube, Queue
- Optimize each structure/interface for parallel architecture
  - Data placement
  - Performance
  - Coherence

ADTs and Scientific Data

- Most scientific data is multi-dimensional and well-formatted
- Mapping multiple structures onto a single data-set is a natural fit
- Different write/read patterns

Implementation Challenges

- Programming model for implementing ADTs
- Existing systems built on byte-streams
  - Current storage API (POSIX)
  - Current filesystem subsystems
    - Buffer cache, striping strategies, storage node interfaces
- Need awareness of data structure at all levels
  - New interfaces at each layer

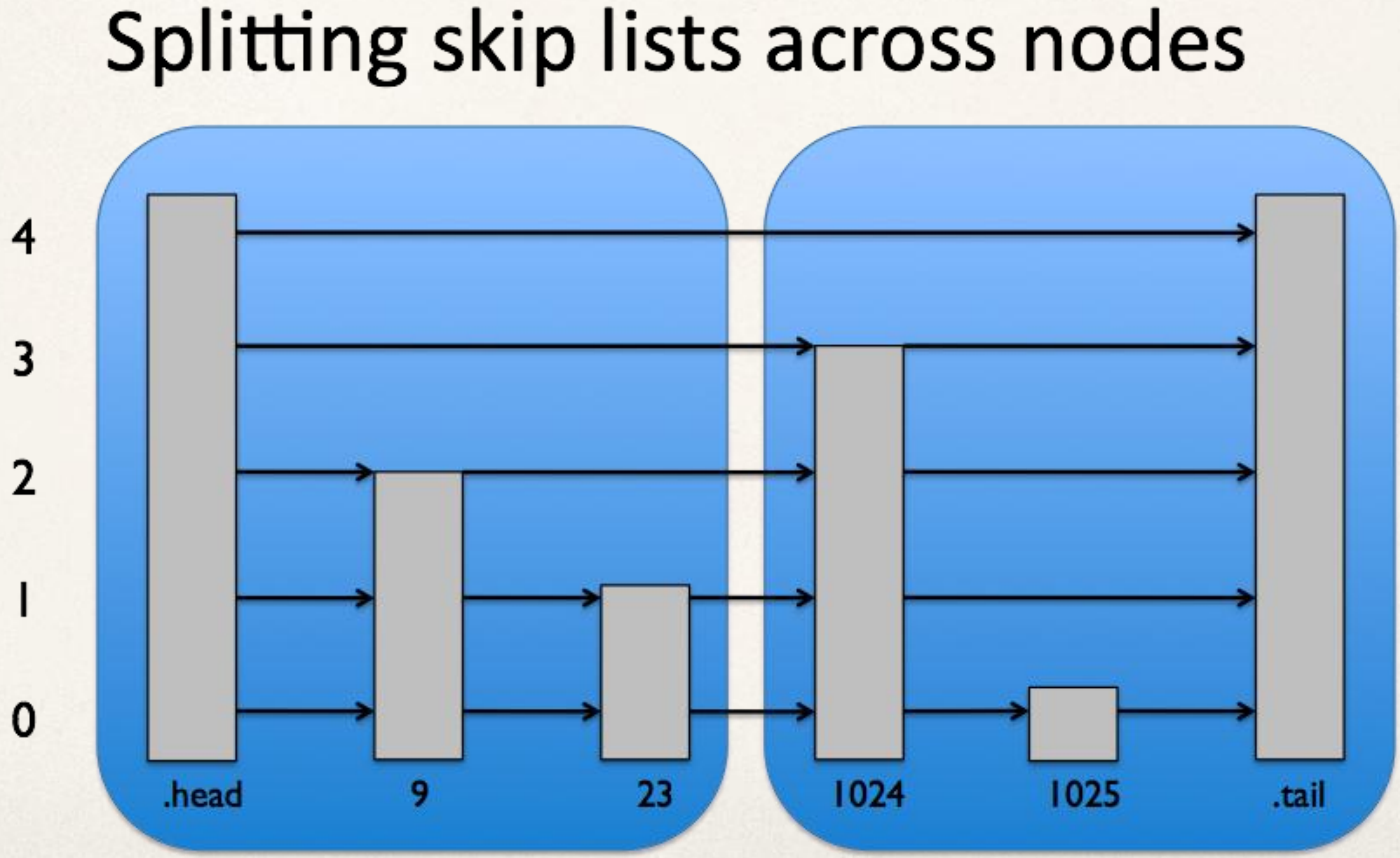
Prototype: Ceph Doodle

- Focus: programming models, ADT interfaces
- Built a framework for implementing and testing:
  - Storage abstractions
  - ADT implementations
  - Programming models
- Loosely modeled after Ceph

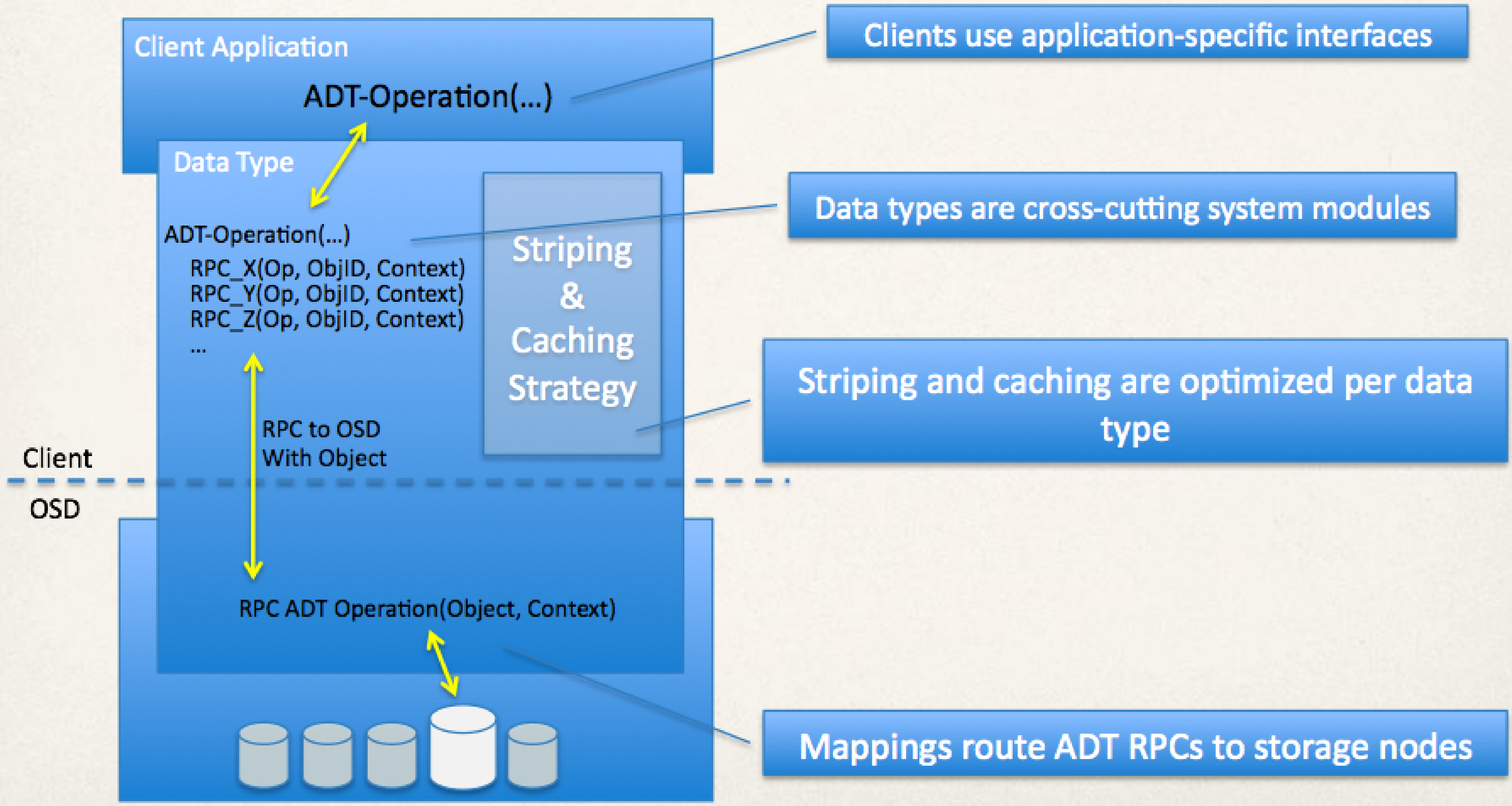
Ceph Doodle Features

- Rapid prototyping
  - Uses RPC mechanisms
  - Python
- Support plugins for different types
  - Bytestream (implemented as storage objects)
  - Dictionary (implemented as a skiplist)

Skiplist Implementation



Ceph Doodle Exposed



Roadmap

- Building on top of Ceph
- Redesigning sub-systems
  - Cache, striping strategies, pre-fetching
- Designing new interfaces to storage
- Performance increases
- Adding views, queries, provenance

Current Status

- Collaborating with database group
- Focusing on consuming structured data and providing
  - query support
  - mapping and indexing
  - provenance
- PDSW 2009

Thank you

Contact: Joe Buck
Email: buck@soe.ucsc.edu

SRL
UCSC Systems Research Lab