IMCa: A High-Performance Caching Front-end for GlusterFS on InfiniBand

Ranjit Noronha and Dhabaleswar K. Panda
Network Based Computing Lab
The Ohio State University
<noronha, panda>@cse.ohio-state.edu

Outline of the Talk

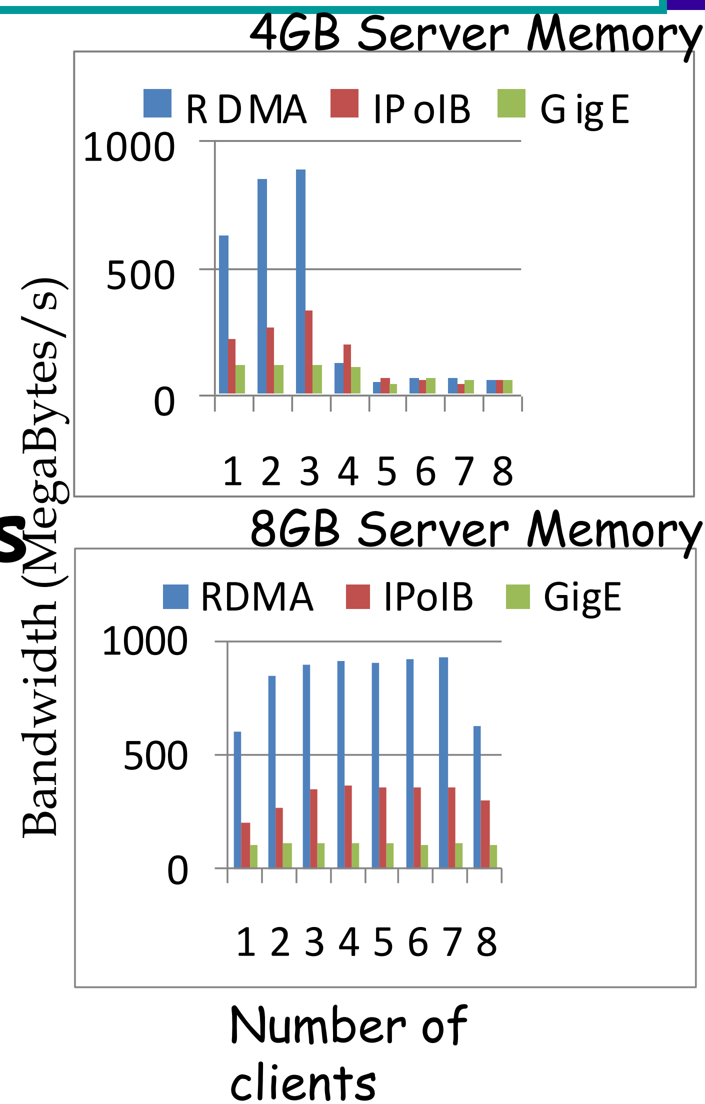
- Background and Motivation
- Architecture and Design of IMCa
- Experimental Evaluation of IMCa
- Conclusions and Future Work

Background

- Large Scale Scientific and Commercial Workloads
- Petascale Computers have arrived
- High-Performance access to the I/O data is crucial
  - Parallel applications is often limited by I/O
- Clustered/Parallel File Systems have evolved to meet this challenge

Motivation – Limitations of disks

- File System performance still dependent on disk performance
- Single Server Bandwidth Drop With Multiple Clients
- Parallel I/O Bandwidth From Multiple Servers



Motivation – Challenges With File Systems

- **Performance for Small Files**
  - Generally difficult to achieve
  - Many environments with a large number of small files
  - Storing on the same disk block provides limited benefit
  - Striping does not provide benefit
  - Store on different servers
- **Cache Coherency Problems**
  - Client side cache provides good performance
  - Non-coherent client cache limited when there is sharing
  - Limited Scalability of coherent caches
- **Server Load Problems**
  - RDMA reduces overhead from TCP/IP
  - RDMA based transport protocols cannot reduce copying costs within the file system

Problem Statement

- Which file-system operations are potential targets for caching?
- What are the alternatives to the traditional client cache/server cache architecture?
- What are the advantages and disadvantages of alternate cache architectures?
- How do we provide the performance of the non-coherent client cache without the scalability problems of the coherent client cache?

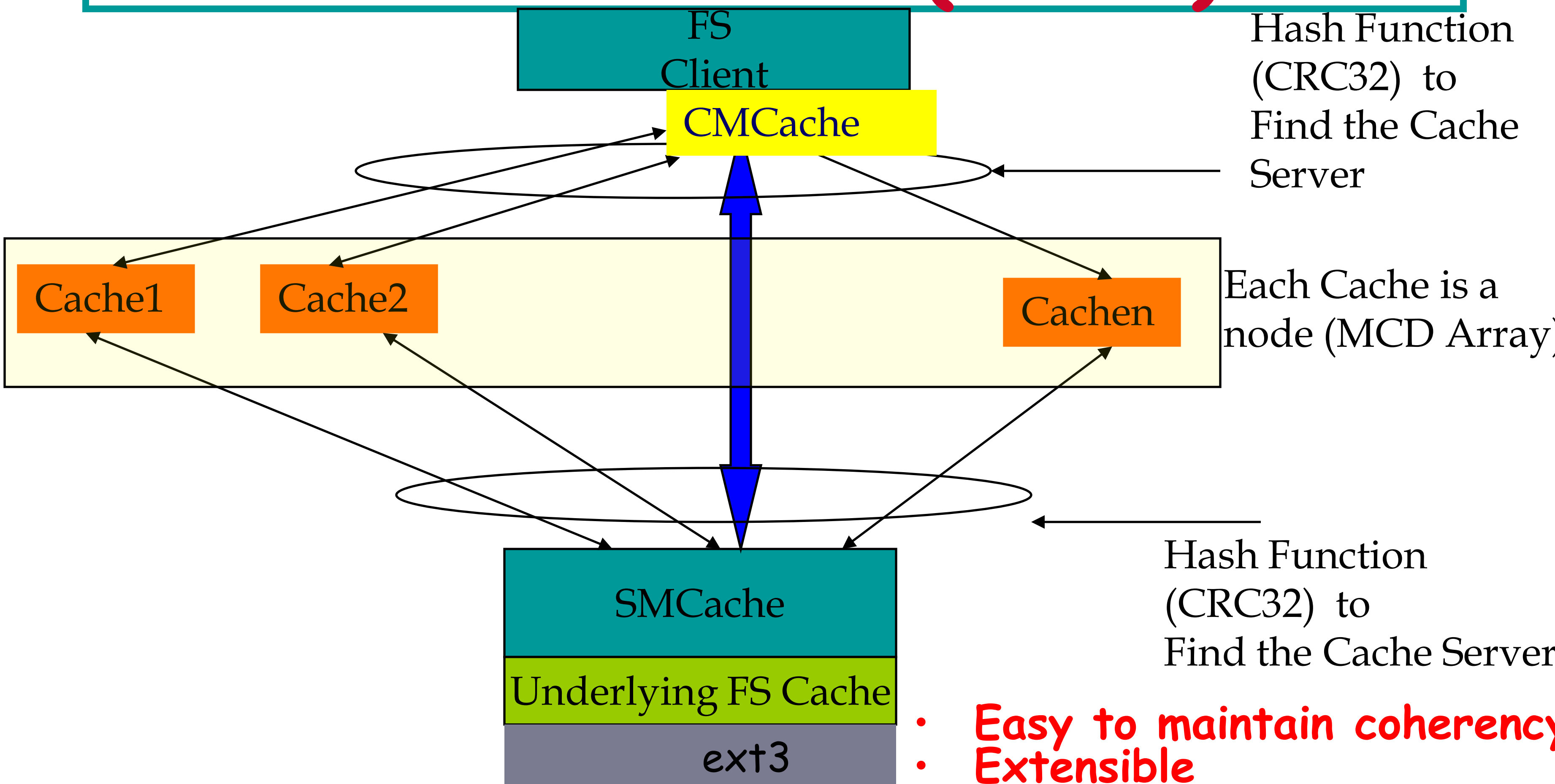
Outline of the Talk

- Background and Motivation
- Architecture and Design of IMCa
- Experimental Evaluation of IMCa
- Conclusions and Future Work

Potential File System Operations That May Be Cached

- Potential Targets For Caching
  - Should be something the client reads
  - Should be possible to uniquely identify cache target
  - Should be possible to chunk the data element
- Small Operations → Stat, Create, Delete, Open
- Stat
  - Read by the client
  - Used as a form of update by many applications
  - Should be used
  - Should be updated on read/write operations on the server
- Create/delete
  - Not read by the client
  - Delete should invalidate previous cache entries
- File Open
  - Not a target for caching, but may be used for prefetching
- Data Transfer Operations
  - Read and Writes
  - Blocks Needed

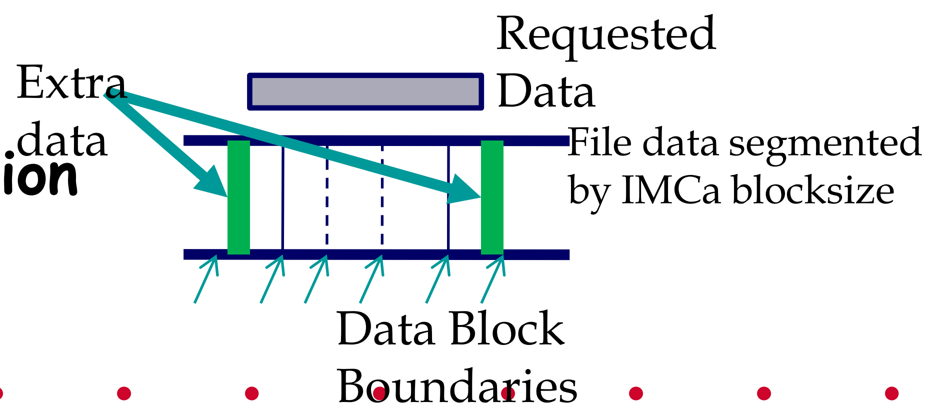
Intermediate Cache Architecture (IMCa)



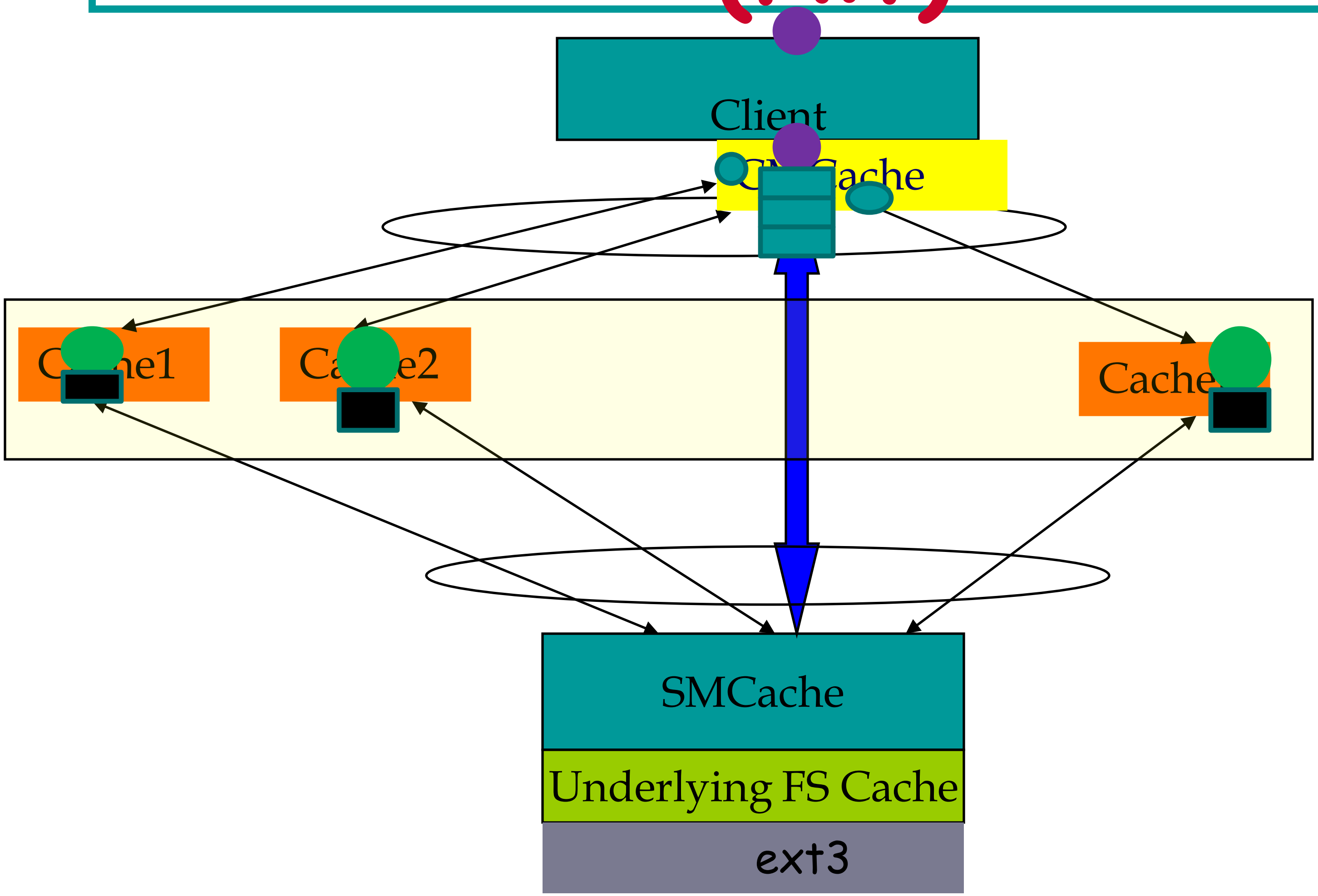
- Easy to maintain coherency
- Extensible
- Can multiple Cache nodes provide benefit?

Need for Blocks In IMCa

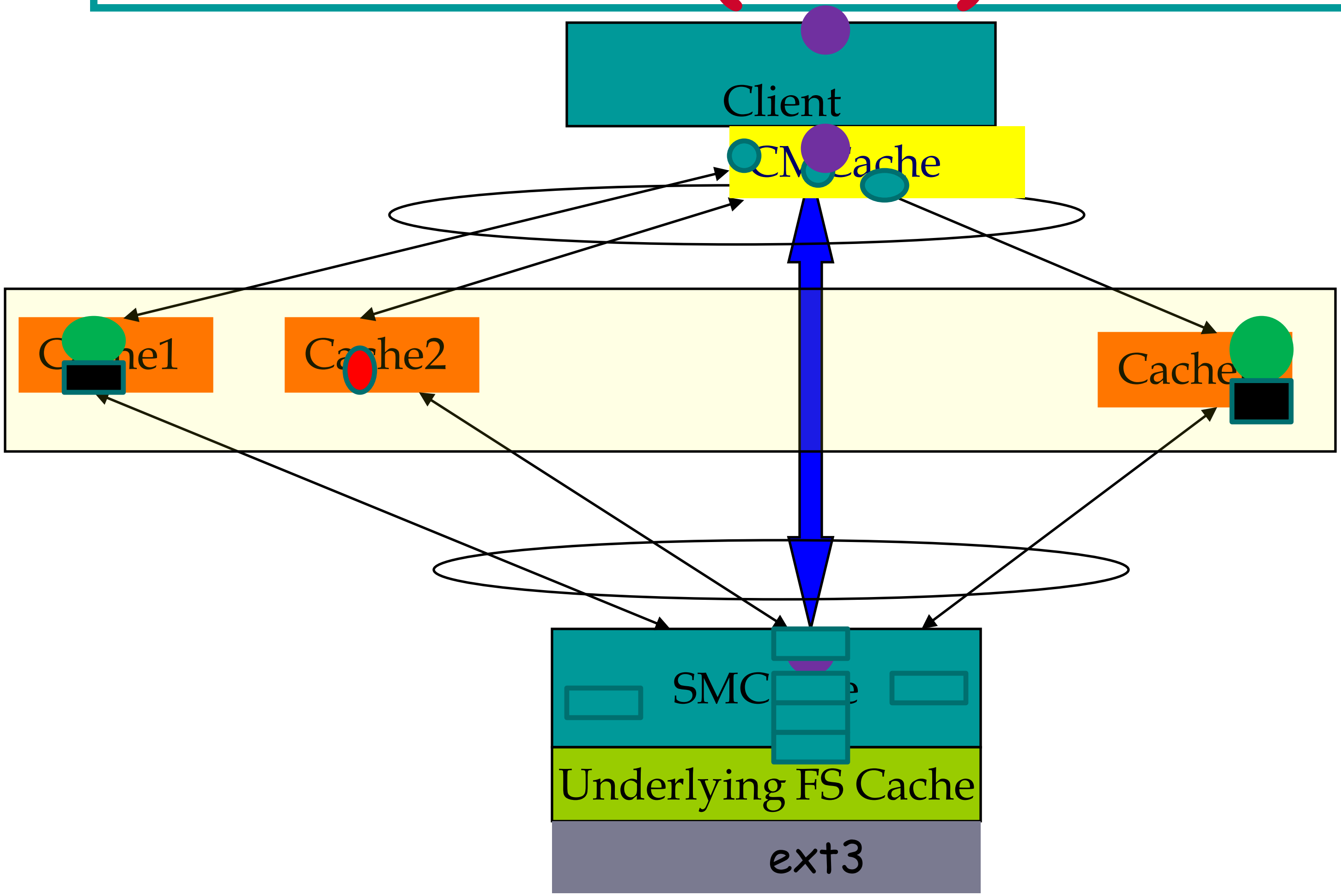
- Most file system store data on the disk as blocks
- Parallel file-systems stripe data across multiple servers
- IMCa uses a fixed block size to store data across the cache servers
  - Block size should provide good performance for most small files
  - Should avoid
    - excessive fragmentation



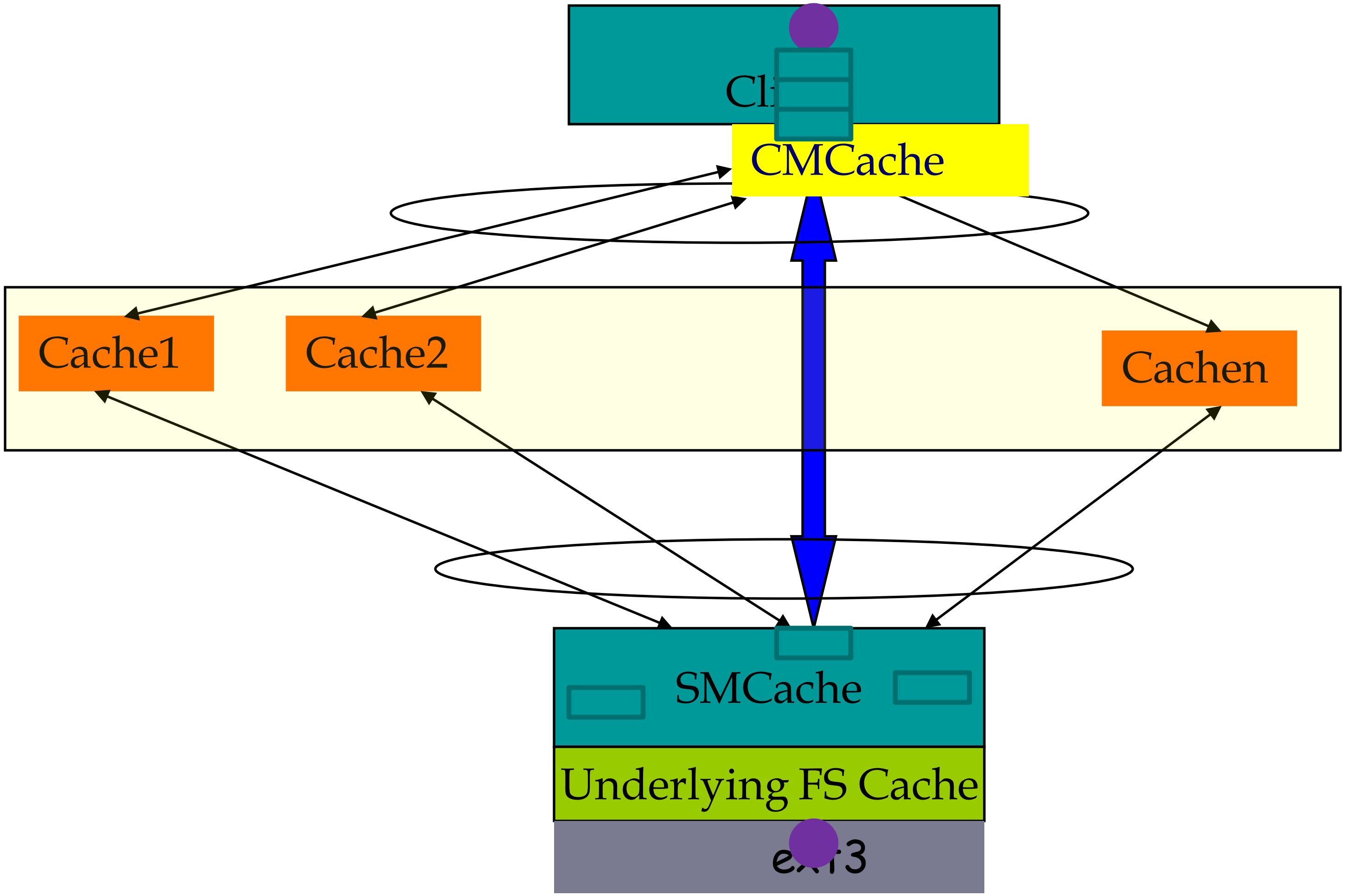
Design-Read Operations (Hit)



Design-Read Operations (Miss)



Design-Write Operations



Advantages/Disadvantages of IMCa

- Fewer Requests Hit the Server
- Latency for requests read from the cache is lower
- MCDs are self-managing
- Failures in MCDs do not impact correctness
- Additional node elements needed especially for caching
- Cold Misses are expensive
- Additional Blocks/Data Transfers Needed
- Overhead and delayed updates

Outline of the Talk

- Background and Motivation
- Architecture and Design of IMCa
- Experimental Evaluation of IMCa
- Conclusions and Future Work

GlusterFS File System

- Clustered File System
- Client and Server in userspace
- Use FUSE interface to translate FS calls from the kernel to the user daemons
- No Stripping → data distributed across servers
- Possible to apply translators at the server and client to perform different functions
- www.glusterfs.org

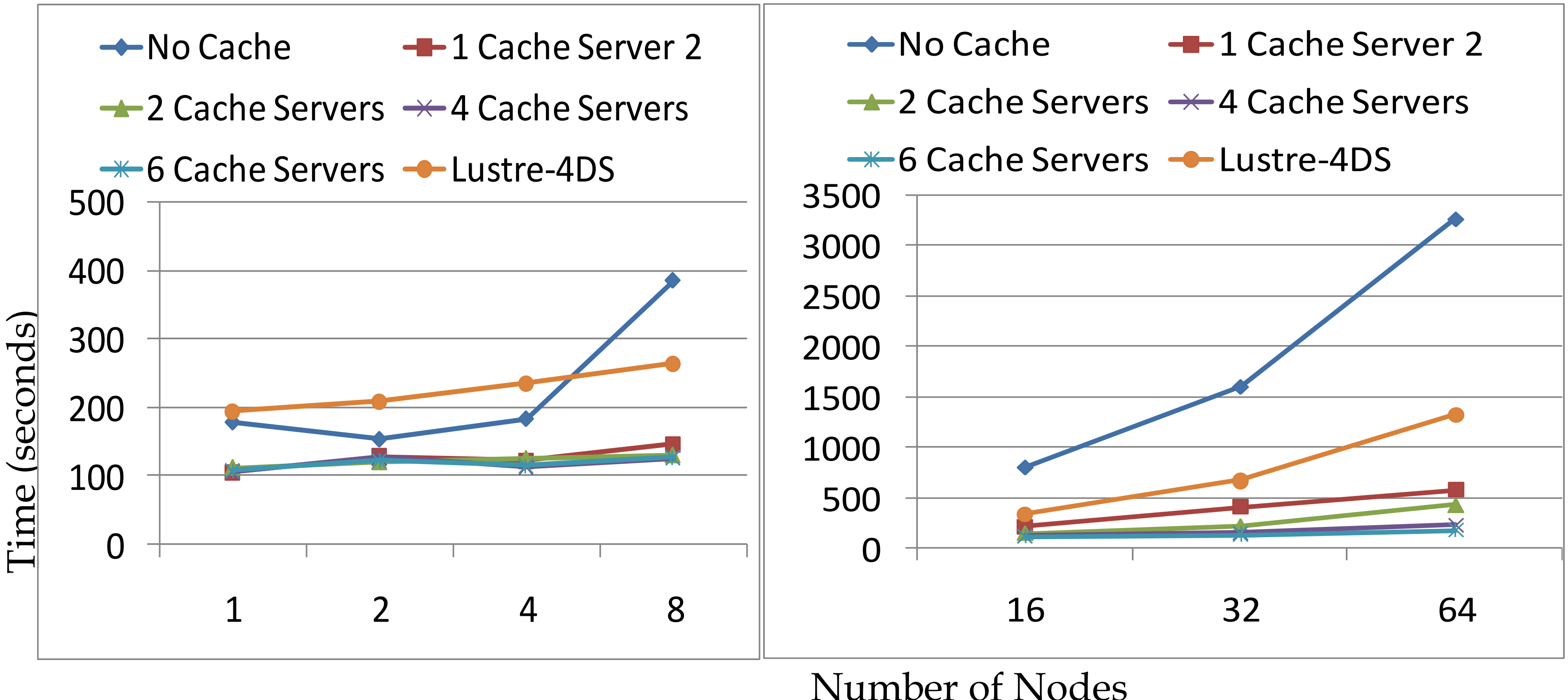
Experimental Setup

- 64-node cluster
  - 8-core Intel Clovertown
  - 8 GB memory
- InfiniBand DDR is the interconnect
- GlusterFS file-system
- The data servers each have 8 RAID highpoint disks
- Communication protocol is IPoIB in Reliable Connected (RC) mode
- MCDs run on independent nodes and use up to 6GB of memory
- CMCache and SMCache use a CRC32 hash function for locating data on the MCDs
- Lustre 1.6.4.3 is used with a socklnd for comparison

Experiment-stat

- Consists of two stages
- First stage (untimed)
  - 262144 files created by a single node
- Second stage (timed)
  - each node tries to perform a stat on each of the 262144 files sequentially

Stat performance

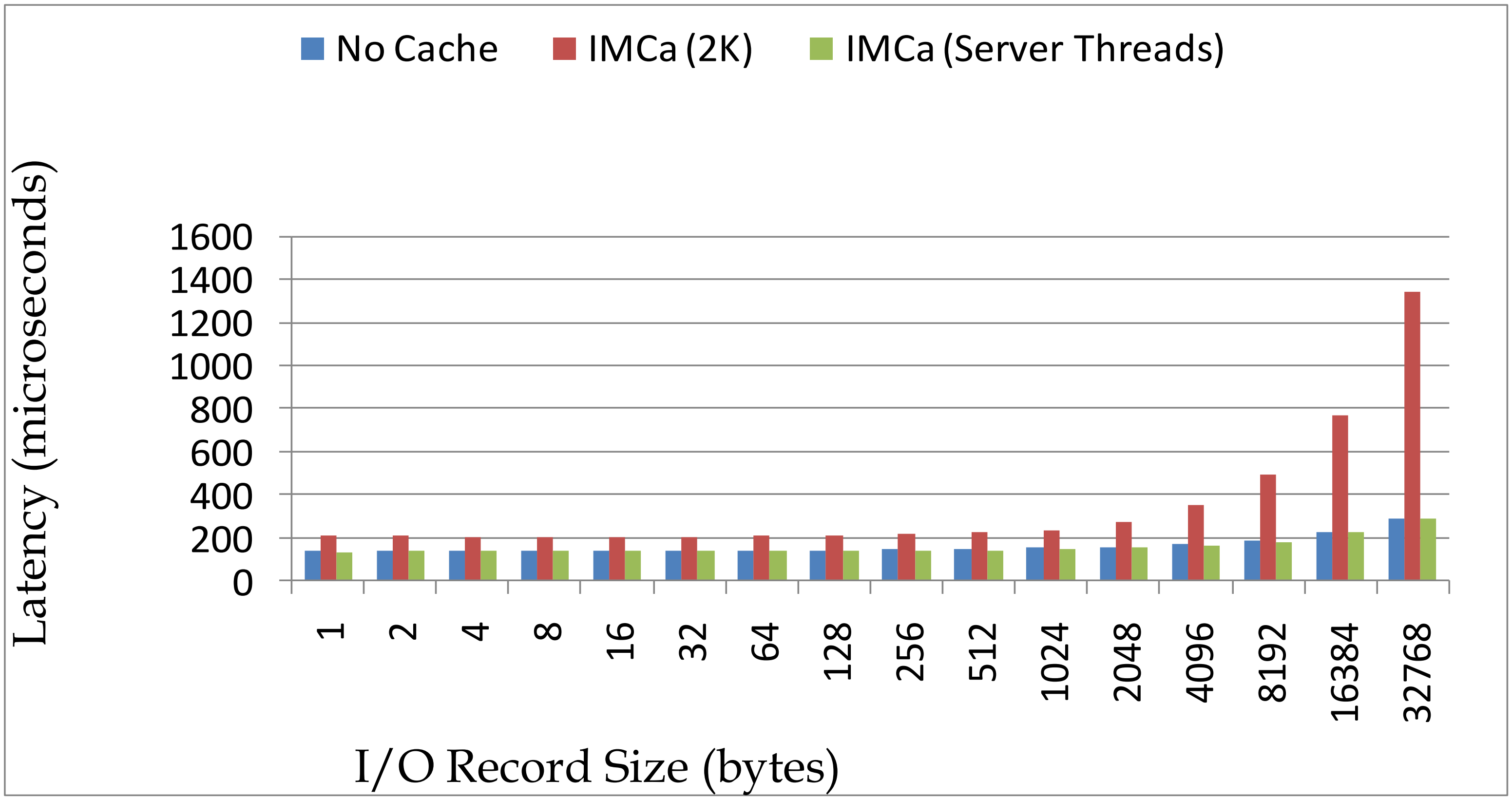


- *Time to stat 262144 different files*
- *Benchmark has two phase create (untimed), followed by stat (timed)*
- *82% improvement at 64 nodes*

Experiment-Write Single Client

- One Client
- Writes 1,024 records of size r sequentially to the file
- Measure time for this to complete

Write - Single Client

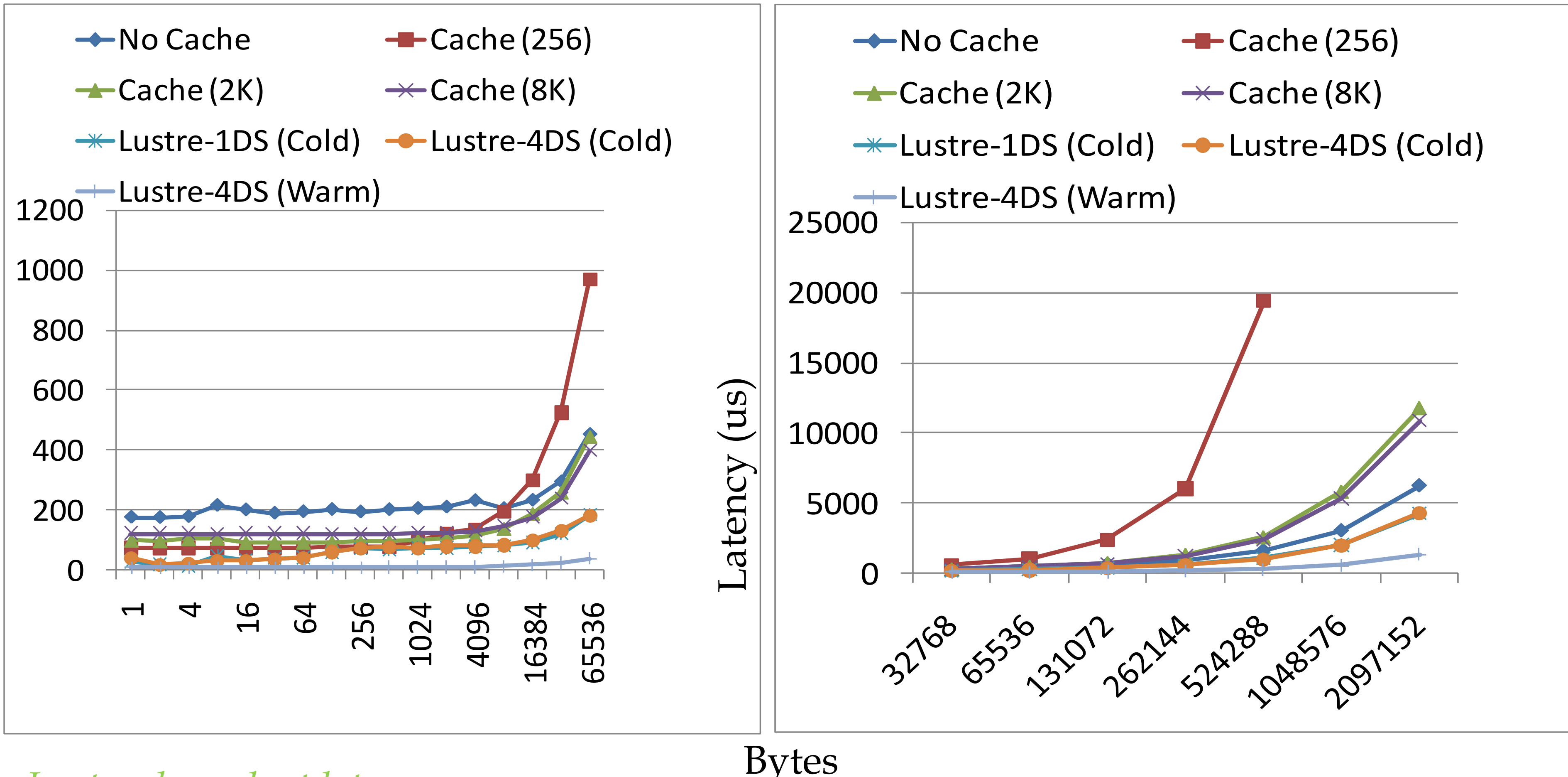


- 2KB block size
- Server thread helps performance

Experiments-Read

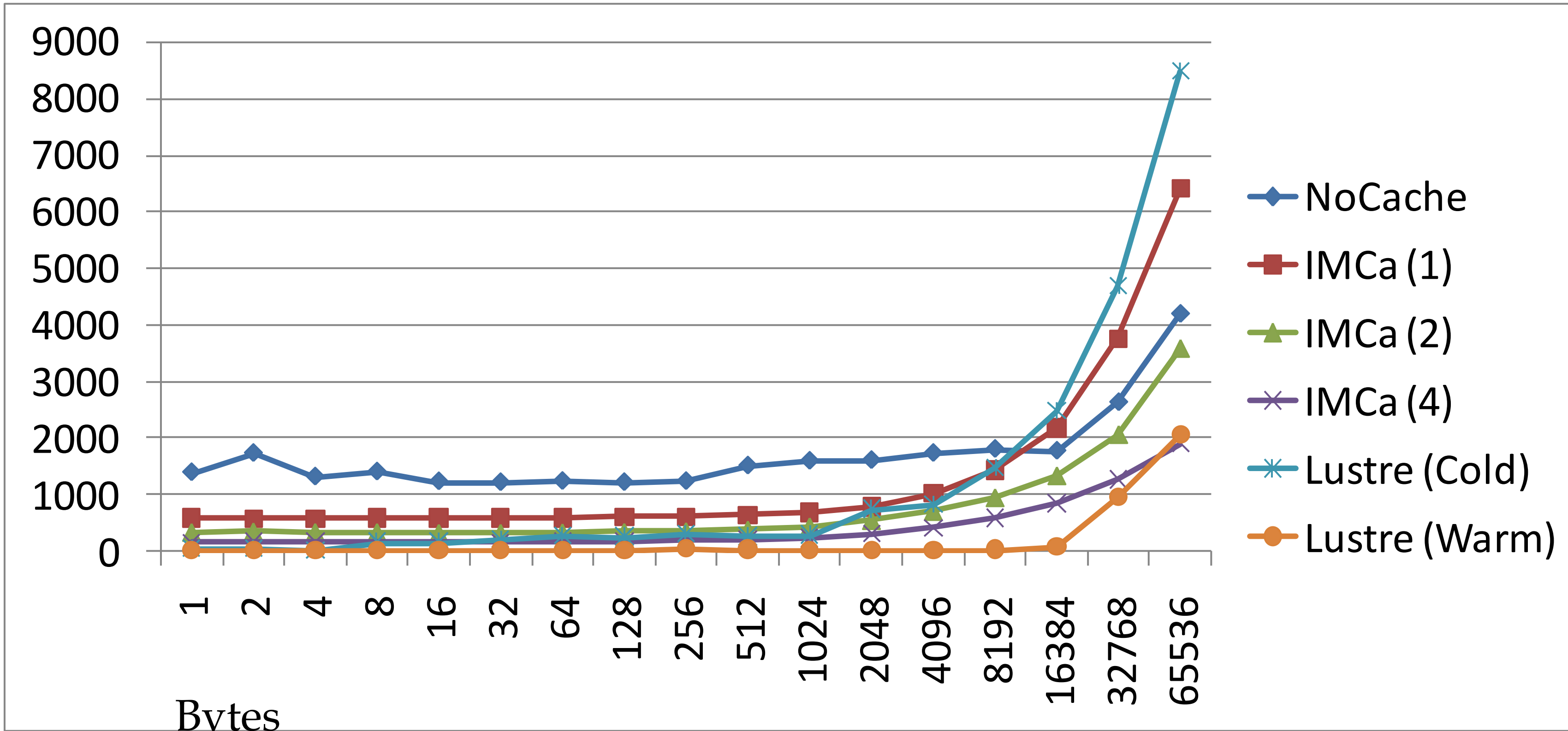
- Single Client Read
  - Follows Write component of the benchmark
  - Move file pointer to the beginning of the file
  - Read 1,024 records of size r sequentially to the file
  - Measure time for this to complete
- Multiple Client Read
  - Each client uses a separate file
- Multiple Client Read Shared
  - Same file used by every client
- Lustre configurations
  - Cold Client Cache → Unmount between Write and Read
  - Warm Client Cache → No unmount between Write and Read

Read Latency (Single Client)



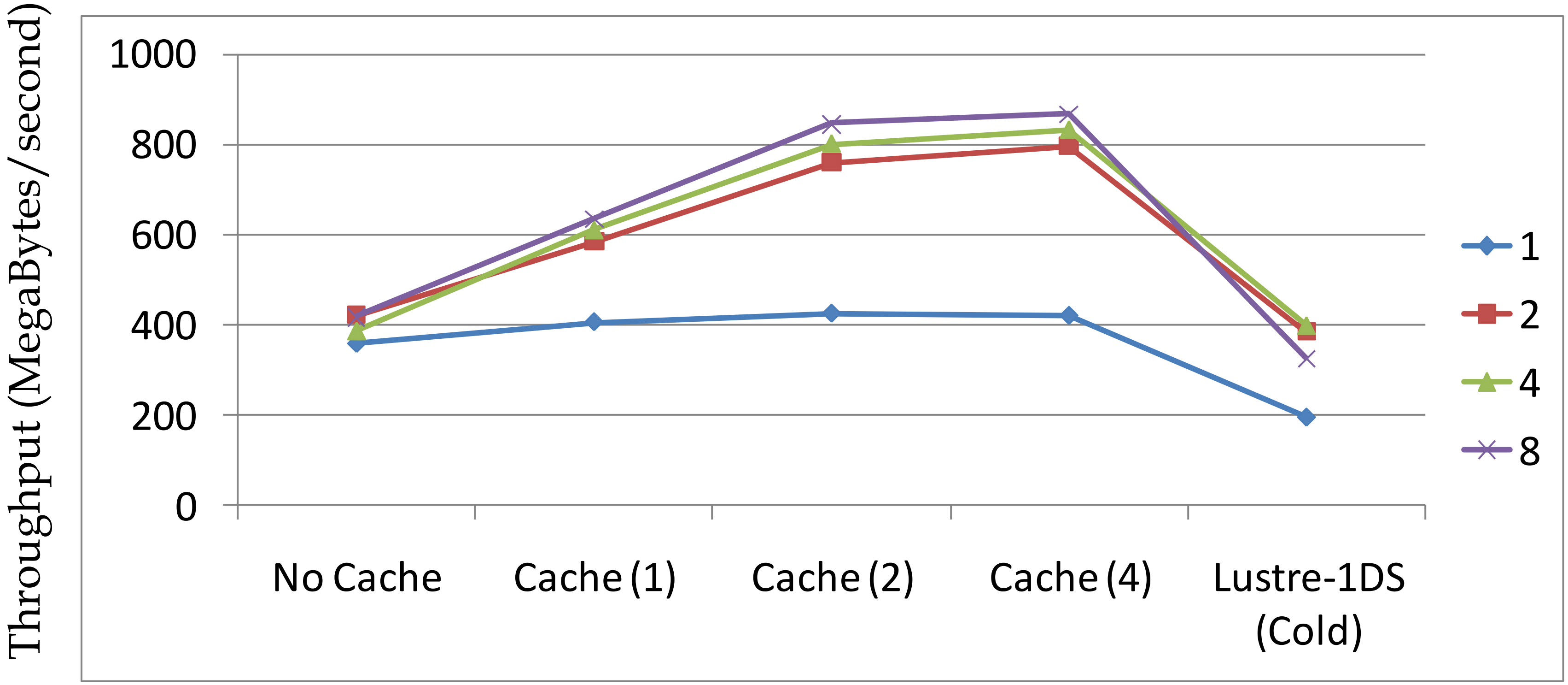
- *Lustre shows best latency*
- *Cache provides benefit for small message sizes*

Read Multiple Client (32 clients)



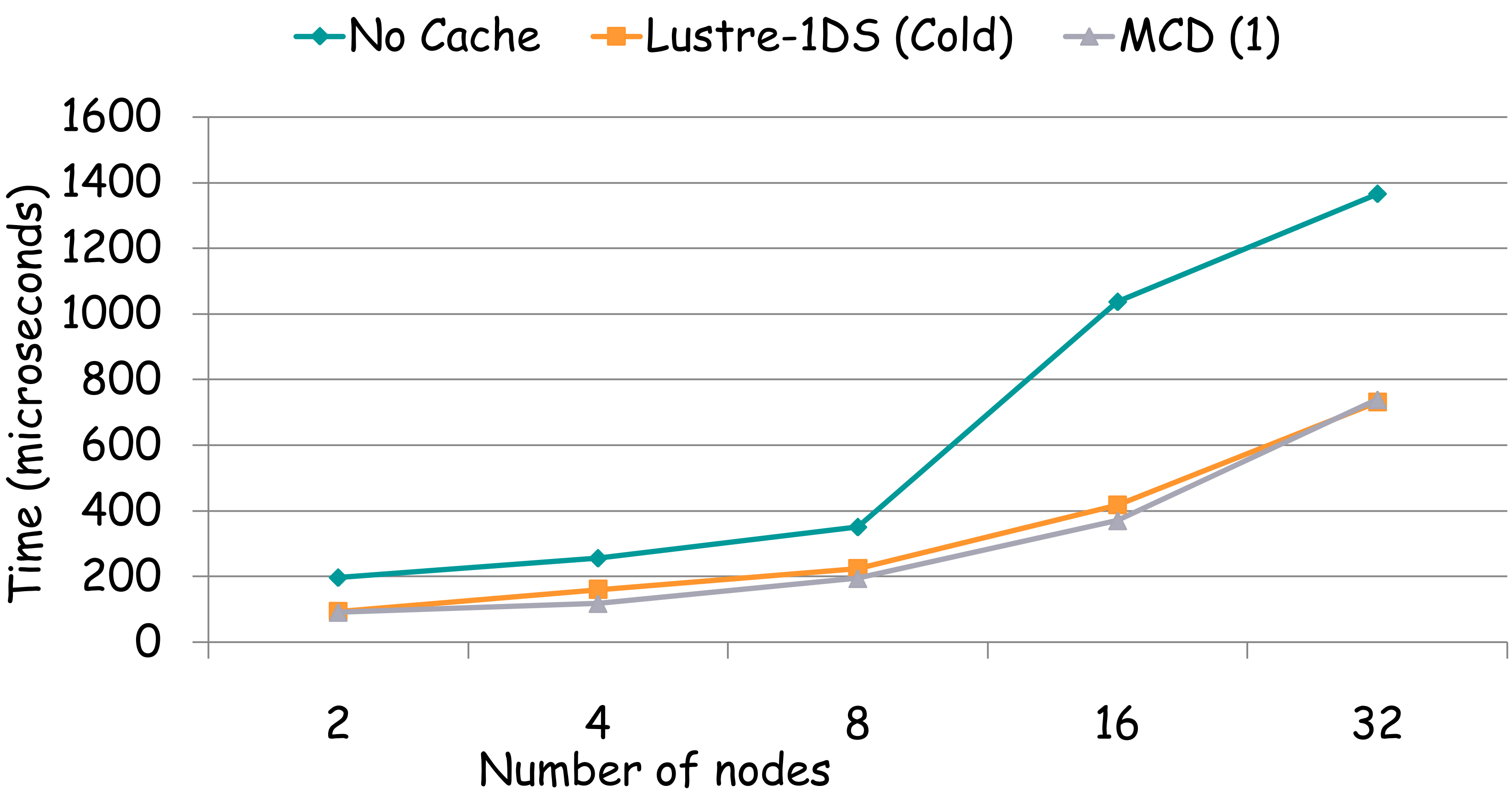
- 51% improvement in latency at 16K
- Multiple MCDs help reduce capacity misses

Iozone throughput



- 1, 2, 4, 8 IOzone threads, 1GB files, 2KB block size
- 325 MB/s (NoCache) -> 868 MB/s (4 MCDs)

Read-Shared Latency



- IMCa helps improve performance over NoCache case

Outline of the Talk

- Background and Motivation
- Architecture and Design of IMCa
- Experimental Evaluation of IMCa
- Conclusions and Future Work

Conclusions and Future Work

- Proposed, Designed and Evaluated an Intermediate Cache for GlusterFS
- Good improvement in stat performance
- Improvement in latency/throughput of read operations
  - Depends on block size
- Would like to evaluate the performance with RDMA
- Would like to evaluate distribution algorithms

Acknowledgements

Our research is supported by the following organizations

Thank you

{noronha, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/