Optimizations for Two-Phase Collective I/O

Introducing Interleaved and Pipelined Two-Phase

Michael Kuhn, Julian Kunkel, Yuichi Tsujita,
Hidetaka Muguruma, Thomas Ludwig

Research Group Scientific Computing
Department of Informatics
University of Hamburg

2011-09-02

1 Introduction
- Motivation
- Two-Phase

2 Interleaved Two-Phase

3 Pipelined Two-Phase

4 Conclusion and Future Work

- There are many algorithms for efficient I/O on parallel distributed file systems
  - The Two-Phase protocol is one of them
- Optimizations can be classified into two categories
  - Client-side optimizations: trying to minimize the work for the servers
    - For example, caching and batching in clients' memory
    - Two-Phase is a client-side optimization
  - Server-side optimizations: let the servers employ their own optimizations
- Two separate improvements for Two-Phase
  - Interleaved Two-Phase from the University of Hamburg
  - Pipelined Two-Phase from Kinki University

- Collective I/O relates I/O operations performed by multiple clients with each other
  - Using individual I/O, they may appear in a random order
  - Clients can collaborate, allowing optimizations that would otherwise be impossible
- Two-Phase is an optimization for collective I/O
  - Implemented in ROMIO
  - Clients share information about their I/O requests
  - Separate communication and I/O phases are used to perform the actual I/O
  - Introduces additional communication overhead

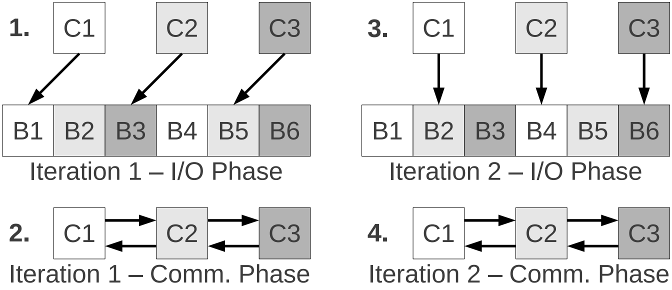

Figure: Two-Phase for a collective read operation with two iterations

0. Initialization: communicate and negotiate file regions
   - a. Decide whether Two-Phase protocol should be used
   - b. Form a file region containing all of the clients' file regions
   - c. Split up file region equally into so-called file domains
     - Each client is then responsible for one file domain

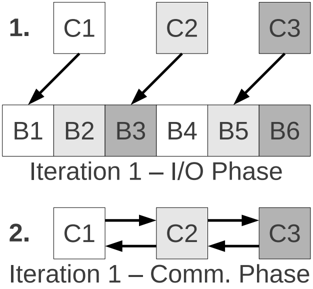

Figure: Two-Phase for a collective read operation with two iterations

1. I/O phase: clients read data
   - Only data within own file domains is accessed
   - Size is limited by an internal buffer used for collective I/O
     - The buffer size currently defaults to 16 MiB

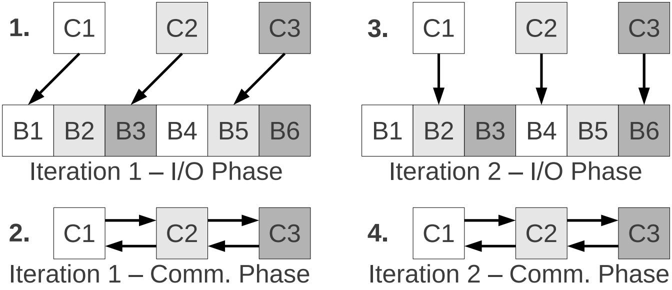

Figure: Two-Phase for a collective read operation with two iterations

2. Communication phase: data is forwarded to the appropriate clients
   - Any client may need to communicate with all other clients

Two-Phase

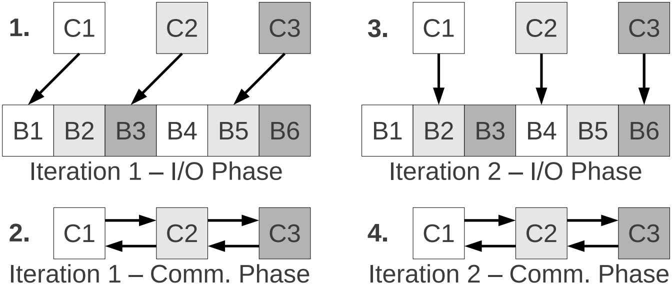

Figure: Two-Phase for a collective read operation with two iterations

- The two phases are repeated until the I/O is completed
- For writing, the order of the phases is reversed
  - Additionally, read-modify-write may be necessary

1 Introduction

2 Interleaved Two-Phase
- Design
- Evaluation

3 Pipelined Two-Phase

4 Conclusion and Future Work

- File domains lead to non-contiguous accesses
- May have a negative effect on performance
- Due to striping schemes, the mapping to the servers may be suboptimal

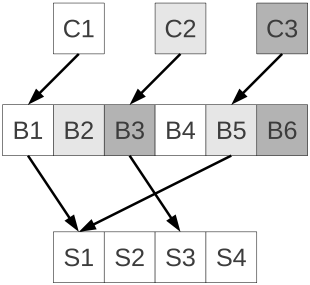

Figure: Suboptimal Two-Phase mapping

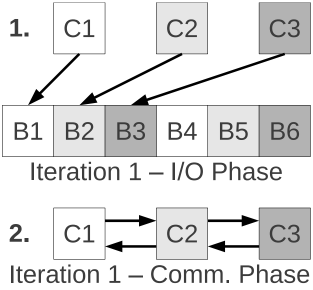

Figure: Interleaved Two-Phase access pattern with two I/O iterations

- Accesses within each I/O iteration are contiguous
- File region is divided into chunks of the size of the Two-Phase buffer
- Chunks are accessed by the clients in a round-robin fashion

- Used PIOsimHD simulator
  - Fast prototyping and evaluation of new ideas
- Ten clients and ten servers
- Test uses collective I/O operations
- File is 1.000 MiB in size
  - Divided into data blocks of equal size
  - Clients access data blocks using a round-robin scheme
- One collective read/write per client
  - Accesses are interleaved

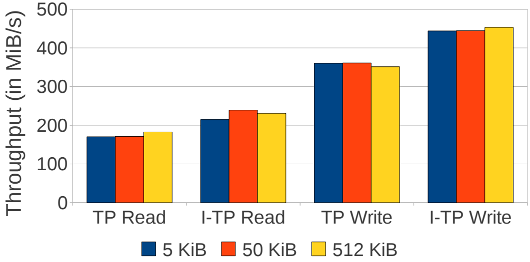

Figure: Interleaved Two-Phase comparison

- Maximum throughput is 500 MiB/s
- Write faster than read, because of write-behind
  - Allows overlapping Two-Phase communication with actual I/O on the servers

1 Introduction

2 Interleaved Two-Phase

3 Pipelined Two-Phase
- Design
- Evaluation

4 Conclusion and Future Work

- In Two-Phase, no I/O is performed during the communication phase and vice versa
- Overlap I/O and communication phases with POSIX threads
- Almost all of the communication time can hidden behind the I/O operations and vice versa
  - Except for startup and finalization
  - Can increase the performance by a factor of two

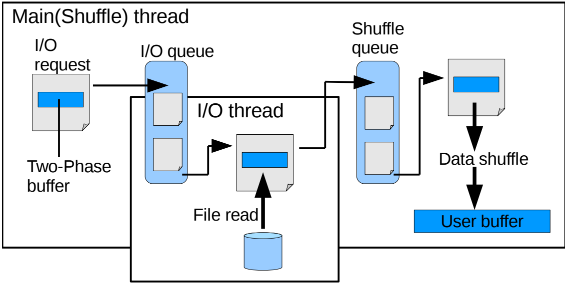

Figure: Pipelined Two-Phase execution

- I/O thread and shuffle thread
  - I/O thread performs data access
  - Shuffle thread exchanges data among peers
- Each thread has a queue of operations
  - Up to four I/O requests can be queued

- Intel Pentium D cluster
- Four clients and five servers
- HPIO benchmark (version 1.55)
- 1 GiB of data is read from a file
  - Divided into four 256 MiB file domains for each MPI process
- Two-Phase buffer size from 1 MiB to 256 MiB
  - Doubling the buffer size halves the number of iterations

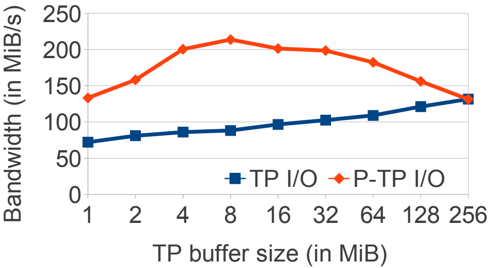

Figure: Throughput of original and Pipelined Two-Phase protocols

- Larger Two-Phase buffer sizes increase performance
- Pipelined Two-Phase better for every buffer size except the maximum
  - Maximum performance with 8 MiB buffer

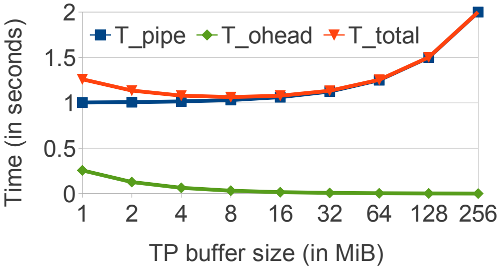

Figure: Estimated operation times in the Pipelined Two-Phase

- More overlap with more iterations

$$T_{pipe} = \left(\frac{N_{itr} - 1}{N_{itr}}\right) \cdot \max(t_{IO}, t_{com}) + \left(\frac{1}{N_{itr}}\right) \cdot (t_{IO} + t_{com})$$

- More overhead with more iterations

1 Introduction

2 Interleaved Two-Phase

3 Pipelined Two-Phase

4 Conclusion and Future Work
- Conclusion
- Future Work

- Two-Phase is a client-side optimization
- The presented modifications promise better performance
  - Up to 30–40% for Interleaved Two-Phase
  - Up to 100% for Pipelined Two-Phase
- Interleaved Two-Phase changes I/O pattern
  - Better suited for the striping in parallel distributed file systems
- Pipelined Two-Phase uses POSIX threads to overlap communication and I/O

- Combine Interleaved and Pipelined Two-Phase
  - Due to being developed in different working groups, this has not happened yet
- Pipelined Two-Phase only implements collective reads at the moment
  - Design of the same protocol for write operation has been started
- Improve the logic used to decide whether to use the Two-Phase protocol
  - For example, use information about the underlying I/O subsystem
- Query underlying file system for information about optimal access patterns