Simulation-Aided Performance Evaluation of Server-Side Input/Output Optimizations

Michael Kuhn, Julian Kunkel, Thomas Ludwig

Scientific Computing
Department of Informatics
University of Hamburg

2012-02-16

Parallel Distributed File Systems

- Most operations are expensive to perform
  - Especially true for large amounts of small requests
  - Large number of clients performing many small operations can easily saturate the I/O system
- Many algorithms and optimizations for efficient I/O exist
- Basically two categories:
  - Client-side: trying to minimize the work the servers have to do
  - Server-side: let the servers handle all the work themselves

State of the Art

- Traditionally, data is accessed in contiguous regions
- *Non-contiguous* I/O enables applications to access several regions in one request
- *Collective I/O* explicitly relates I/O performed by multiple clients with each other
- *Two-Phase* is an optimization for collective I/O
  - Clients collaborate during I/O
- Goal: analyze whether comparable performance results can be achieved with server-side optimizations

Simulation Framework

- Presented optimizations are implemented in a simulator as a first step
- *HDTrace* can simulate, trace and visualize applications
- *PIOsimHD* allows simulating arbitrary network topologies, servers and client applications
  - Goal: allow easy and fast prototyping of new algorithms
- Advantages:
  - Not dependent on any specific project environment
  - Can serve as a starting point for adoption into real-life projects

Cache Layers

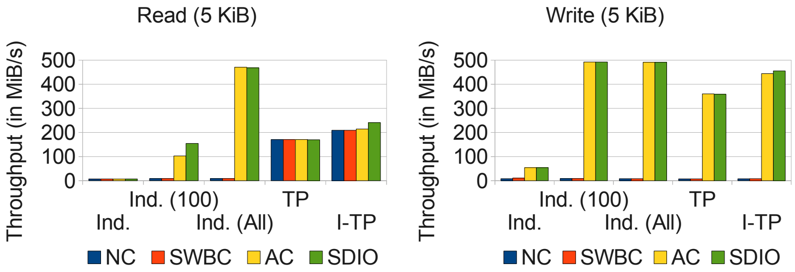
- `NoCache` does no caching at all
  - All I/O operations are forwarded directly to the I/O subsystem
- `SimpleWriteBehindCache` does rudimentary caching
  - Operations are written out in a background thread
  - Write operations do not block the calling client
- `AggregationCache` performs simple read/write optimizations
  - Tries to combine I/O operations with queued ones
- `ServerDirectedIO` additionally reorders I/O operations
  - Merge multiple client requests into larger contiguous operations
    - Unnecessary write requests are discarded early
  - Access to all pending requests

Comparison With Existing File Systems

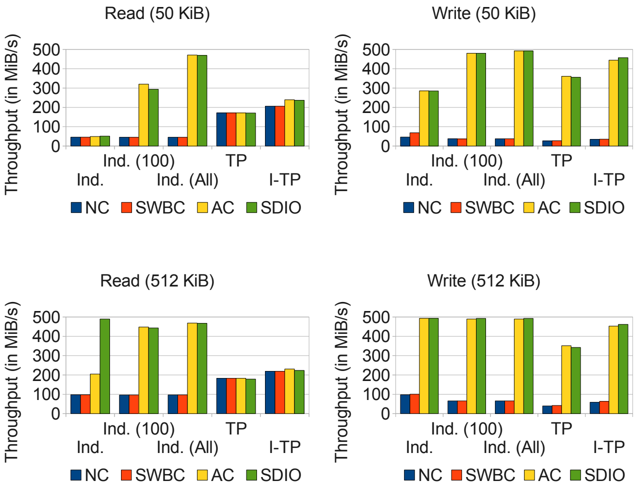
- Comparison with PVFS:
  - Normal buffer size per I/O operation is 256 KiB
  - Only a subset of reads is announced to the I/O subsystem
  - Large reads are fragmented
    - Might cause the access pattern to look like random accesses
    - Can cause a serious performance degradation
  - Read performance can be compared to `NoCache`
- Comparison with Linux:
  - Performs write-behind
  - Some sort of aggregation
  - Observable write performance comparable to `AggregationCache`

Cluster Setup

- Simulated cluster comprised of twenty nodes
  - Ten clients, ten (file) servers:
    - 1 GBits/s NIC
    - 50 MiB/s HDD
  - Maximum I/O throughput of 500 MiB/s
  - Data is striped across all servers with a round-robin scheme
- Comparison uses individual and collective I/O operations
  - 1.000 MiB file divided into data blocks of equal size
  - Individual: one data block (*Ind.*), 100 data blocks (*Ind. (100)*) or all data blocks (*Ind. (All)*) accessed in each iteration
  - Collective: one collective operation to access all data blocks
  - Resembles I/O patterns often found in HPC applications
    - Iterative algorithms perform I/O every *n* iterations



- `NoCache` (*NC*), `SimpleWriteBehindCache` (*SWBC*), `AggregationCache` (*AC*) and `ServerDirectedIO` (*SDIO*)
- Batching operations results in performance gains
- For write operations, less batching is required
  - Can be processed in the background



- Better performance with non-optimizing cache layers
- Due to the larger data block size

- Complex client-side optimizations are not necessarily better than relatively simple server-side optimizations
  - `AggregationCache` and `ServerDirectedIO` deliver better performance
- Necessary to batch operations or use large operations
- Simple server-side optimizations are often sufficient for our use cases
  - Could alleviate the need for sophisticated client-side optimizations
- Some room for improvement:
  - `ServerDirectedIO` does not influence the order in which the clients send their data
  - Benchmarks using SSDs would be interesting