Using Simulation to Validate Performance of MPI(-IO) Implementations

Julian M. Kunkel

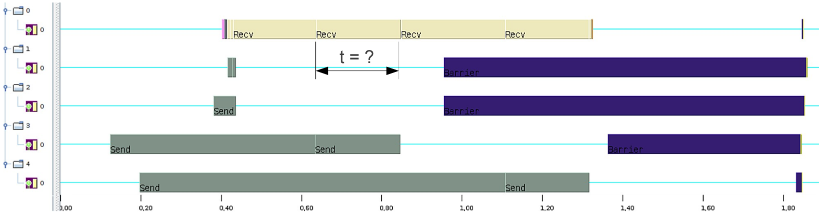
University of Hamburg

5. Juni 2013

Outline

Motivation

- Communication and I/O should utilize the system efficiently
  - Performance lost in MPI-IO degrades many applications
- Is observed communication or I/O performance as expected?
  - Is MPI using the best communication algorithm?
  - Is MPI loosing performance due to system issues?
- Users have a hard time to assess observed performance



Endphase of an application

Assessing observed performance is difficult

- Supercomputers are complex
  - Hardware components and software stack
  - Deployed optimizations
  - Interaction between optimizations
  - Network topology
- Communication and I/O algorithms of MPI are non-trivial

Validating MPI-IO Performance

Goals

- Reveal performance issues prior to production
- Prevent unexpected performance degradation

Proposed solution

Include automatic performance oriented tests in MPI which

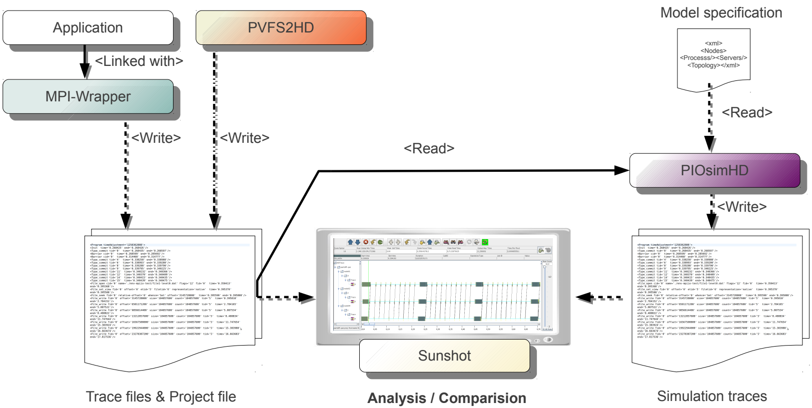
1. Run elementary MPI-IO benchmarks to build a system model
2. Run sophisticated collective and I/O benchmarks
3. Estimate theoretical performance and compare results
4. Assess performance differences with an expert system
5. Report system model and found inefficencies

Manual Approach Using Simulation

1. Runs elementary MPI-IO benchmarks to build a system model
2. Runs sophisticated collective and I/O benchmarks
   - *Record P2P communication pattern of collectives and I/O*
3. *Use simulation to estimate behavior and runtime*[1]
   - *Replay recorded P2P and I/O activities*
4. *Compare runtime and traces to identify issues*

[1] Performance of collectives could be approximated knowing the MPI-internal algorithm

Analysis Workflow



PIOsimHD – Virtual Laboratory

Goals

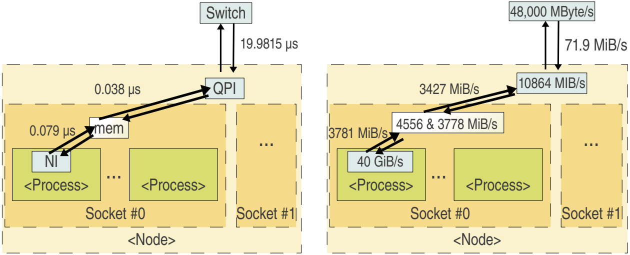
- Foster understanding of performance factors in clusters
- Assist MPI-IO research

Model

- Discrete event simulation
- Supports fundamental hardware characteristics
  - Throughput, Latency
  - HDD: Sequential transfer rate, average access time, track-to-track seek time, RPM
  - Network: Store&Forward, Data-flow oriented
- Modular, alternative device- and I/O cache models
- Abstract parallel file system (no MD)
- Server-side I/O-Caching algorithm & Two-Phase I/O

Model Characteristics

- Parameterization by using MPI point-to-point operations
- HDD access time and latency from data sheet
- HDD sequential transfer rate measured with IOZone



(a) Latency of edges (b) Throughput

Network model for the working groups Westmere cluster

Slow Communication on our GigE Cluster

Unexpected network behavior

- Performance of 67 MiB/s – behind expectation of 117 MiB/s
- High P2P variance – sometimes very slow operations (by 0.2 s)

Automatic analysis would have detected these inefficencies!

Tedious analysis of the software issue

- Behavior happens with MPICH2 and Open MPI
- Invisible in TCP benchmarks
- Newer kernel fixes throughput issue
- Variance disappears using CentOS

⇒ Reason is still unknown

Analysis of MPI-IO behavior using simulation

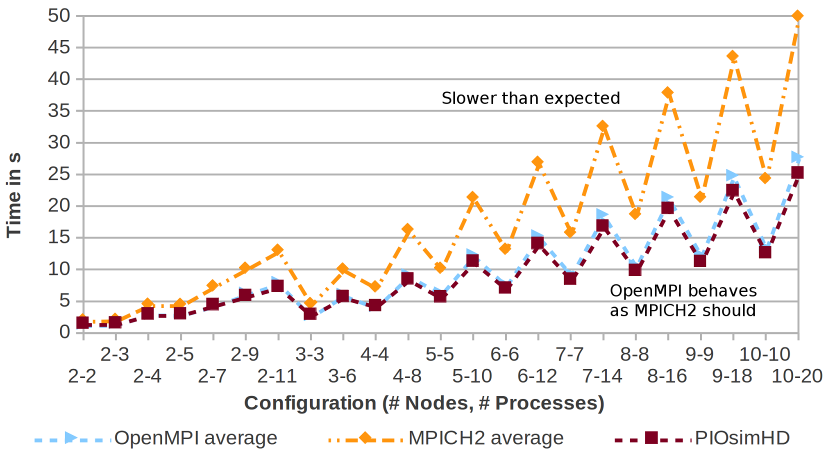
Experimental setup

- 10 node Ubuntu cluster with 120 cores
- Open MPI 1.5.3, MPICH2 1.3.1, Orangefs-2.83

Conducted experiments

- P2P communication schemes: Root, PingPong, SendRecv, Ring
- Collectives: Bcast, Gather, Scatter, Reduce, ...
- Parallel I/O: 4-levels-of-access, tmpfs vs. HDD
- Application: Jacobi-PDE solver

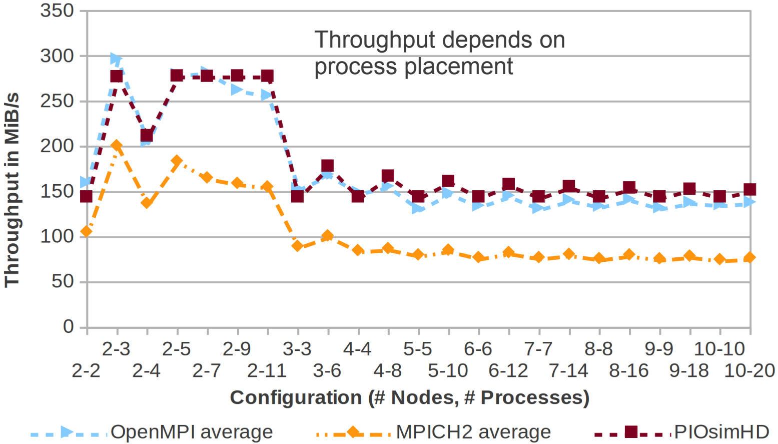
P2P: Processes Communicate with Rank 0 (Sendrecv)



(a) Time

100 MiB messages

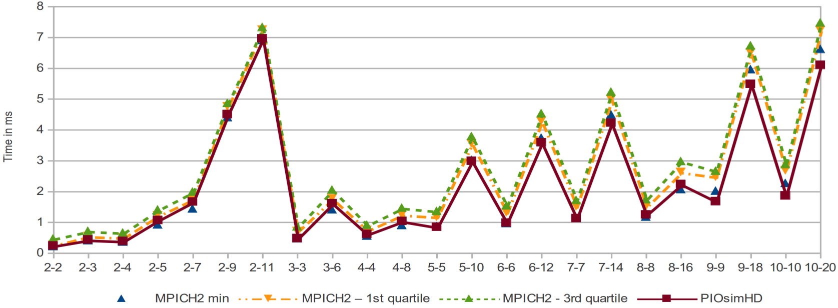
Estimating Throughput Looks Trivial...



(b) Aggregated throughput

100 MiB messages

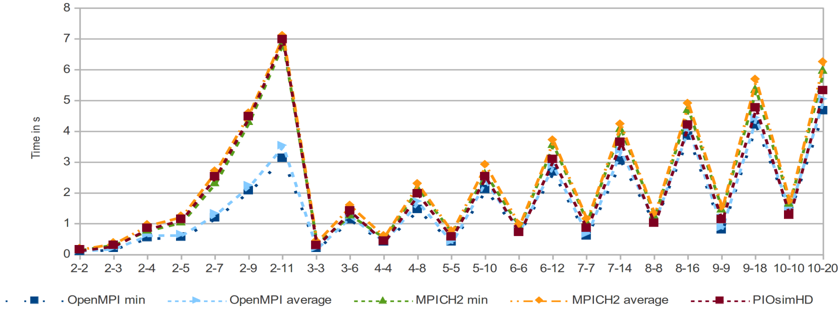
Collective Communication



`MPI_Allgather()`, 10 KiB of data (10,000 repeats)

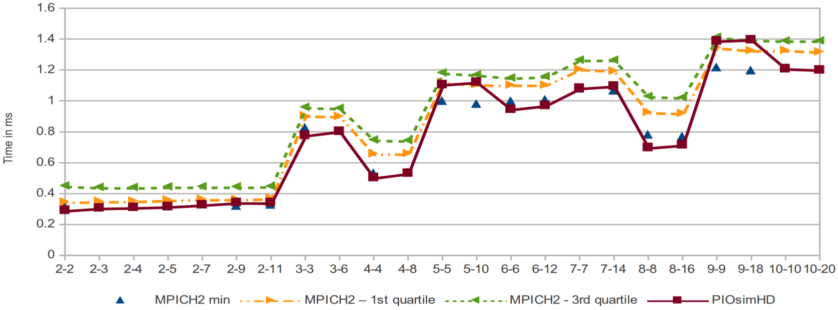
No unexpected performance degradation, but SMP-unaware algorithm*

Collective Communication



`MPI_Allgather()`, 10 MiB of data (12 repeats)

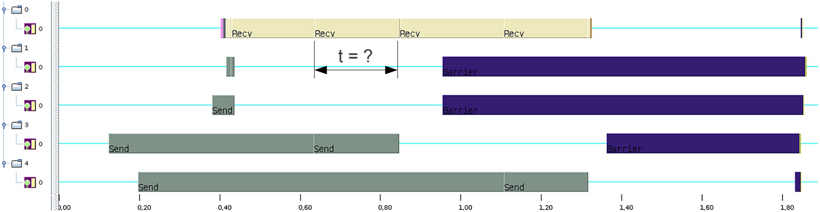
Collective Communication



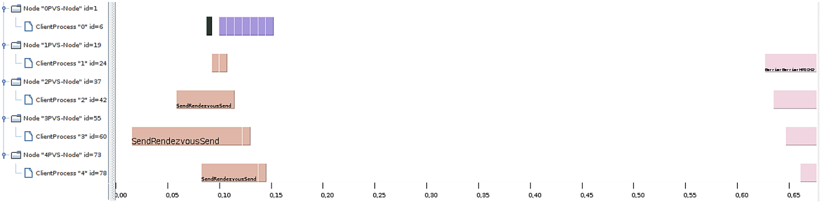
`MPI_Allreduce()`, 10 KiB of data

Is the discrepancy worth to investigate?

Jacobi-PDE: Comparing Invididual Operations



(a) Observed



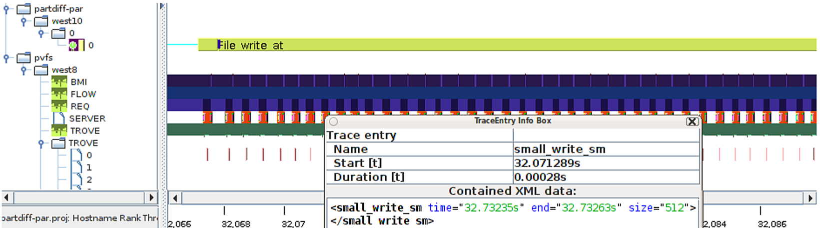
(b) Simulated

Final phase of the solver – sometimes communication is 0.2 s slower as expected

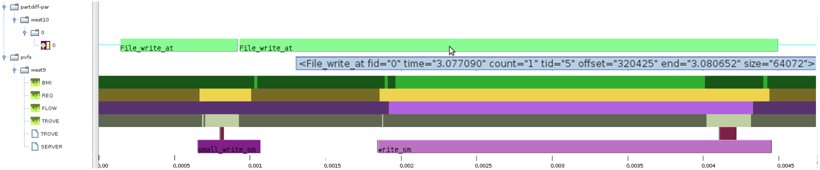
Parallel I/O Example

- Our PDE testprogram outputs data for analyzing convergence behavior
  - Appending 64 KiB of data to a file
- Time for the operation
  - Measured 70 ms
  - Simulated 2 ms
- Reason: I/O is split into 512 Byte requests
  - Identified by introspecting client and server activity
- Applying the undocumented hint `romio_pvfs2_listio_write` ⇒ 3.4 ms

Investigating Behavior of PVFS-Servers



(a) Default operation; details of one request are shown



(b) With supplied hint

Screenshot of the PDE's data exchange for one client and one server

Summary & Conclusions

- MPI is not always operating optimally
  - Algorithms may not be able to extract performance
  - System might degrade performance unexpectedly
- Assessing of performance is difficult
- PIOsimHD is a virtual laboratory to research MPI-IO
  - With the help of simulation issues could be identified
- Analysis of system / library issues is non-trivial
  - Best to identify reasons in an integrated/upon installation
  - Automatic evaluation seems possible
- Future work: integration of an automatic tool in Open MPI