Improving I/O Forwarding Throughput with Data Compression

Presented by Benjamin Welton
welton@cs.wisc.edu
Overview

• Overview of the need for I/O enhancements in cluster computing

• Discussion of related work

• A brief introduction to I/O forwarding and IOFSL

• Description of the implementation

• Performance testing of various compressions in the I/O forwarding layer
Why are I/O optimizations needed?

- Computational power and memory have been increasing at a fast pace with every generation of supercomputer.
- This means faster cores, more cores, and more memory.
Why are I/O optimizations needed?

- Interconnects, however, have not been increasing at the same rate as core computation resources.
Why are I/O optimizations needed?

- An example of the divergence between interconnect bandwidth and node computation can be seen when comparing Blue Gene/L and Blue Gene/P Nodes.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Interconnect Bandwidth</th>
<th>Node Computation</th>
<th>Ratio Comp /Band</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Gene/L</td>
<td>2.1 GB/Sec</td>
<td>2.8 GF/Sec</td>
<td>1.3:1</td>
</tr>
<tr>
<td>Blue Gene/P</td>
<td>5.1 GB/Sec</td>
<td>13.7 GF/Sec</td>
<td>2.68:1</td>
</tr>
</tbody>
</table>
Why are I/O optimizations needed?

- This divergence can cause serious performance issues with file I/O operations.
- Our goal was to find methods to reduce the overall transfer size to alleviate the bandwidth pressures on file I/O operations.
Related Work

• Wireless network compression of network traffic [Dong 2009]
• MapReduce cluster energy efficiency using I/O Compression [Chen 2010]
• High-Throughput data compression for cloud storage [Nicolae 2011]
Brief introduction to HPC I/O

- HPC I/O generates large amounts of data
- As computation workload increases, so does I/O data requirements
- High data rates are required to keep pace with high disk I/O request rates

Blue Gene/P I/O transfer rate (per minute) [Carns 2011]
Brief introduction to HPC I/O

- Obtaining high I/O throughput requires a highly optimized I/O framework

- Some optimization techniques already exist (e.g. collective I/O, subfiling, etc)

- Current optimizations may not be enough to keep pace with increasing computation workloads
I/O Compression

- An existing I/O middleware project (I/O Forwarding Scalability Layer, IOFSL) was used to experiment with I/O compression.
IOFSL

• IOFSL is an existing I/O forwarding implementation developed at ANL in collaboration with ORNL, SNL, and LANL

• Compressed transfers are an extension of this framework

• Compression was implemented internally to allow for client applications

[Ali 2009]
Compression

- Only generic compressions were chosen for testing
- Compressions requiring knowledge of the dataset type (e.g. floating point compression) were not implemented.

<table>
<thead>
<tr>
<th>Compression</th>
<th>Throughput</th>
<th>Output Size</th>
<th>CPU Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bzip2</td>
<td>Low</td>
<td>Small</td>
<td>High</td>
</tr>
<tr>
<td>Gzip</td>
<td>Moderate</td>
<td>Medium</td>
<td>Moderate</td>
</tr>
<tr>
<td>LZO</td>
<td>High</td>
<td>Large</td>
<td>Low</td>
</tr>
<tr>
<td>No Compression</td>
<td>Highest</td>
<td>Largest</td>
<td>None</td>
</tr>
</tbody>
</table>
Compression Implementation

- Compression and decompression are done on the fly
- Two different methods were implemented for message compression
  - Block style compression
  - Full message compression
Block Style Compression

- Block style compression uses an internal block encoding scheme for I/O data.
- Used for LZO and can be used for Floating Point Compression (or any compressor without a block compress function)
Full Message Compression

- Treats entire message as one compressible block (visible to IOFSL, External compression has own internal blocking)
- The message does not have to be fully received to start decoding
- Used by Bzip2 and Gzip
Results

- All testing was done on a Nehalem-based cluster. With data written to memory and client counts between 8 and 256 clients per forwarder.

- Testing was done on two different interconnects (1 Gbit Ethernet and 40 Gbit Infiniband)

- Testing was done using a synthetic benchmark with a variety of datasets.
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Format</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>Null Data</td>
<td>Binary</td>
<td>/dev/zero</td>
</tr>
<tr>
<td>Text</td>
<td>Nucleotide Data</td>
<td>Text</td>
<td>European Nucleotide Archive</td>
</tr>
<tr>
<td>Bin</td>
<td>Air / Sea Flux Data</td>
<td>Binary</td>
<td>NCAR</td>
</tr>
<tr>
<td>Compressed</td>
<td>Tropospheric Data</td>
<td>GRIB2</td>
<td>NCAR</td>
</tr>
<tr>
<td>Random</td>
<td>Random Data</td>
<td>Binary</td>
<td>/dev/random</td>
</tr>
</tbody>
</table>
Dataset Compression Ratio

![Diagram showing compression ratios for different datasets and compression methods: bzip2, zlib, LZO. The x-axis represents different types of data (Zero, Text, Bin, Comp, Rand), and the y-axis represents the compression ratio (comp/orig).]
Bzip2 Ethernet

- Worst performing on both Ethernet and IB
- Only when using the most compressible datasets is write performance improved
Gzip on Ethernet

• Decent performance for compressible datasets
• Uncompressible datasets show slight degradation in write performance
LZO on Ethernet

• Fastest rates of compression
• In cases where the file does not compress, performance is about equal to the no-compressed read/write
LZO on Infiniband

- Tested to show a case where congestion is not a factor for transfer
- For writes, compression shows positive effect on throughput
- Reads show a decrease in throughput for data that is not compressible
Result Overview

- LZO is by far the fastest compression tested
- Low complexity compressions (such as LZO) can produce faster transfer rates on bandwidth-limited connections (and faster connections using data with a high compression ratio)
- High complexity compressions (Bzip2) show drastic performance degradation, especially on non saturated high speed connections
Future Work

- Implementation of specialized compressions, such as floating point compression, which could result in drastically increased performance
- Storing data compressed on the file system instead of decoding it on the I/O Forwarder
- Adaptive compression techniques which would enable or disable compression of a particular block depending on whether or not it compressed well
- Testing with hardware compression
Hypothetical Hardware Compression Data Rates

Read Hardware Compression

Write Hardware Compression

Data Rate MB/s

Bin  Comp  Text

24056  17848  8136

Data Rate MB/s

Bin  Comp  Text

5016  3722

Methods:
- bzip2
- zlib
- LZ0
Acknowledgements

- IOFSL Team @ Argonne
  - Dries Kimpe
  - Jason Cope
  - Kamil Iskra
  - Rob Ross
- Other Collaborators
  - Christina Patrick (Penn State University)
- Supported by DOE Office of Science and NNSA
Improving I/O Forwarding Throughput with Data Compression

Questions?

Presented by Benjamin Welton
welton@cs.wisc.edu