ABSTRACT

DINAKAR, DIVYA. Online Output Data Reduction via BG/P I/O Nodes. (Under the direction of Xiaosong Ma.)

Petascale scientific applications face severe data bottlenecks, especially with the increasing computation hardware parallelism and heavy I/O resource contention. In many situations, scientists have to save intermediate results at a much lower frequency than desired, sacrificing analysis/visualization resolution for better overall execution efficiency.

In this research work, we develop a new approach to result data output, which performs online data reduction to summarize data into histograms, a step routinely performed by scientists when examining large amounts of data, but usually carried out as a post-processing task. Other post processing operations could vary from simple exercises like calculating the min, max and average; to more complicated algorithms like multidimensional binning. Our efforts focus on the BlueGene/P supercomputer where we stage the data through IO nodes. We leverage the BlueGene architecture to avoid node overhead for online data processing and reduction. Such online processing/reduction is taken out of the way from the main computation, by leveraging existing monitoring software launching and multi-level data reduction frameworks. Our solution also does not affect the applications’ execution model, allowing compute processes to communicate normally and enable such online data processing through parallel standard I/O interfaces. We evaluated our work using the GTC particle application, on thousands of cores. Our results indicate that the online data reduction framework is able to shrink the output size by orders of magnitude without bringing noticeable overhead or requesting additional nodes.
Online Output Data Reduction via BG/P I/O Nodes

by
Divya Dinakar

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2010

APPROVED BY:

__________________________  ____________________________
Frank Mueller                  Yan Solihin

__________________________
Xiaosong Ma
Chair of Advisory Committee
DEDICATION

To Mom and Dad ...
BIOGRAPHY

Divya Dinakar was born on April 18, 1985 in Bangalore, India. She completed her Bachelors degree in Computer Science Engineering in June 2007 from B.M.S College of Engineering, Bangalore, India. After graduating in 2007, she worked at Spikesource Software Pvt. Ltd, a startup in Bangalore. She was also an intern at IBM Corporation during her undergraduate studies. In Fall 2008, she came to North Carolina State University, Raleigh, NC to pursue graduate studies in Computer Science. She interned at Intel Corporation as a performance modeling intern during the summer of 2009. Upon the completion of her thesis research, she plans to graduate in August 2010.
ACKNOWLEDGEMENTS

I would like to begin by thanking my advisor, Dr. Xiaosong Ma for her support, guidance and encouragement throughout my graduate studies. Being her first female student makes me proud and grateful for giving me a sense of direction in research. She has been very patient and inspiring and always made me think of problem solving rather than implementing to be the goal of good research. I would also like to thank my committee members Dr. Frank Mueller and Dr. Yan Solihin for their guidance.

I am grateful to the members of the systems research group – Amey Deshpande, Kartik Vijayakumar, Jagan Srinivasan, Manav Vasavada, Feng Ji, Fei Meng, Xing Wu, Abhik Sarkar, Zhe Zhang, Heshan Lin and others for their cooperation and valuable inputs along with all the fun we had while working together. I am specially thankful to Abhishek Sreenivasa who has been a great teammate - cooperative, hardworking and lots of fun. I am obliged to my friends at NCSU – Aliasgar Pedhiwala, Rashmi Bhat, Sayli Karmarkar and Sindhura Ramegowda who made my graduate life an enjoyable experience and stood by me through good and bad in these two years. I am truly grateful to Nirmal Thacker, for being my strength on several occasions. I also thank Mr. Wolfram Sauer for guiding me well during my internship at Intel Corporation, which helped in shaping my research and career interests. I am most appreciative of all the support provided by my research co-ordinators Scott Klasky, Martin Schulz, Dong Ahn, Gregory Lee, Jay Lofstead, Hassan Abbasi, Emily Jacobson and Bronis Supinski.

I have been fortunate to have Nithyambika Gurukumar, Shruti Suresh and Supreetha Sudhakar as friends, and Swaroop Dinakar as brother who have stood by me through my frustrations and successes. I prefer to remain indebted to my family and friends – Rashmi Bhat and Sanjeev Mehta along with my parents, who have always been the sources of inspiration in my life.
# TABLE OF CONTENTS

List of Figures ................................................................. vi

Chapter 1 Introduction ......................................................... 1
  1.1 Motivation and Problem Definition .................................. 1
  1.2 Research Overview .................................................. 3
    1.2.1 Summary of Contributions ................................... 3
  1.3 Thesis Outline ....................................................... 4

Chapter 2 Background ....................................................... 5
  2.0.1 Blue Gene/P Overview ............................................ 5
  2.0.2 MRNet ............................................................. 6
  2.0.3 LaunchMon ....................................................... 7
  2.0.4 ADIOS ........................................................... 8
  2.1 Related Work ......................................................... 9

Chapter 3 End to End Reduction Framework Architecture ............. 12
  3.0.1 BlueGene Reduction Architecture Overview .................... 12
  3.0.2 Application Case Study: GTC-P ................................ 15

Chapter 4 Implementation of End to End Reduction Framework ........ 18
  4.0.3 Filter Implementation .......................................... 18
  4.0.4 Histogram Compression ......................................... 19
  4.0.5 Communication Tree ............................................ 19
  4.0.6 ADIOS Extention ............................................... 20

Chapter 5 Evaluation .......................................................... 21
  5.1 Overview ............................................................ 21
    5.1.1 Experimental Environment ................................... 21
    5.1.2 Evaluative study and implementation of alternate approaches for histogram reduction using MPI ................................. 22
  5.2 Results ............................................................... 23
    5.2.1 Impact of compression ....................................... 23
    5.2.2 Visible cost of PDF generation ................................ 24
    5.2.3 Impact on total application time ............................ 26
    5.2.4 BG-REDUCT setup time overhead ............................ 28

Chapter 6 Conclusions ....................................................... 29

References ................................................................. 31
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>IBM Blue Gene/P Architecture [41]</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>End to End Reduction Framework: High Level Architecture</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>BG-REDUCT levels</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>GTC-P I/O and Memory Analysis</td>
<td>16</td>
</tr>
<tr>
<td>5.1</td>
<td>Impact of compression</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>Size of output on disk</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>Visible cost of PDF generation</td>
<td>26</td>
</tr>
<tr>
<td>5.4</td>
<td>Dissection of BG-REDUCT histogram generation</td>
<td>26</td>
</tr>
<tr>
<td>5.5</td>
<td>Split up of alternate approaches</td>
<td>27</td>
</tr>
<tr>
<td>5.6</td>
<td>Impact on total execution time</td>
<td>27</td>
</tr>
<tr>
<td>5.7</td>
<td>BG-REDUCT Setup time overhead</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation and Problem Definition

Most ultra-scale parallel applications are simulations created to generate data that help scientists understand their phenomena or processes of interest. However, such applications running on state-of-the-art supercomputers face severe challenges regarding I/O and data processing today. It is well known that the I/O subsystem is further lagging behind the processors and the memory hierarchy, in both access speed and the degree of hardware parallelism. As the parallel execution scale increases, the overall scalability of these applications becomes more and more limited by the data pathway. In particular, with the common supercomputer charging mode, the relatively slow and unscalable I/O operations turn out to be more costly – the application-visible I/O time will be multiplied by the total number of nodes allocated, in counting the total resource units consumed by the job.

Scientists facing this dilemma thus often have to make uneasy choices regarding their output volume, through configuring the periodic I/O frequency (such as “one dump every $n$ computation timesteps”). Typically, to ensure that their massively parallel simulations use supercomputer resources effectively toward solving large problems and achieving desired level of spatial details, scientists sacrifices the temporal resolution in saving output data. This is done by
allocating a total execution time budget to parallel I/O operations, say 5% or 10% of the total job wall time. The periodic I/O frequency is then adjusted to fit in the given budget.

Meanwhile, even when scientists have been frugal in generating data, 100,000-core runs on today’s petascale machines can easily produce large amounts of output (hundreds of GBs or several TBs per timestep). Such enormous datasets are expensive to manage, move, and process, especially when tasks such as visualization are mostly carried out on users’ local clusters with limited processing and storage capabilities. As a result, data reduction, as a common post-processing step, is frequently used to extract important information from the raw datasets. The reduction results often replace the original output in subsequent analysis and visualization.

The above situation has recently motivated active research in data staging and online data analysis, where a separate staging area is created to serve dual purposes: accelerating periodic I/O and performing data analysis/reduction on the fly. For example, the new I/O middleware PreDatA [48] has demonstrated that with a rather small extra sets of staging nodes (1.5% of the original allocation), important yet relatively unscalable common data processing tasks such as sorting can be carried out on these nodes while the normal compute nodes proceed with their computation. With asynchronous data processing and I/O, the overall execution time enhancement still outweighs the increase in node allocation.

This promising approach, however, may not be easily ported to all ultra-scale platforms. One example is the IBM Blue Gene system, where I/O nodes are distributed and reside on the node boards, each working with a fixed number of compute nodes.[7]. The compute nodes are allocated in “partitions” or “blocks”, a power-of-two unit that includes at least one I/O node. This causes the compute nodes to be allocated in blocks of 32, 64, or more. Unlike other platforms, where typically an arbitrary number of nodes can be requested for a job, this rigid node allocation makes it difficult to create a small staging area. Although with large-scale production runs that use thousands of nodes, one can request 32 or 64 additional staging nodes and stay within the 1-2% budget, the placement of such staging nodes is challenging. With the compute node interconnection and application inter-process communication both tuned for
power-of-two setups, forcing one out of 32/64 nodes on a board to be a staging node may disturb the communication pattern and create load imbalance. On the other hand, it is unwise to place all staging nodes together on a separate node board, which easily create communication and I/O bottlenecks.

1.2 Research Overview

To address this problem, in this research work we present a new approach: to exploit the I/O nodes on BG/P machines for online simulation result data reduction. This approach brings multi-fold advantages. First, it allows BG/P applications to run without explicitly allocating staging nodes, solving the block-allocation problem and waiving the resource cost associated with using more nodes. Second, it utilizes the computation and memory resources available on the I/O nodes, otherwise under-utilized. Third, by performing on-the-fly data reduction at the I/O nodes, co-located with blocks of compute nodes, we exploit fast intra-node-board interconnection and data locality, while achieving load balance. Fourth, by completing the global data reduction among the I/O nodes, the output data processing can be overlapped effectively with computation without much disturbance to the inter-compute-node communication, by taking advantage of the separate interconnection network between the I/O nodes available on Blue-Gene systems. Finally, this allows the client application to simply use parallel I/O interfaces (augmented with standard or custom data reduction filters) for output, hiding all I/O node related computation and communication details from users.

1.2.1 Summary of Contributions

We consider the major contributions of this work as follows.

- We designed and built BG-REDUCT, a framework that performs end-to-end online simulation output data reduction, leveraging several existing building blocks, including the MRNet multicast/reduction tool [37], the LaunchMon infrastructure for co-locating tool daemons with parallel applications’ computation processes [5], and the ADIOS parallel
I/O middleware [16]. BG-REDUCT is transparent to application codes, and can be enabled by adding less than 5 lines to the application code and minimal job script changes.

• As a proof-of-concept prototype, we designed and implemented a multi-dimensional histogram building filter, representing the PDF (Probability Distribution Function) for particle data attributes. We adopted a multi-phase reduction scheme that effectively use resources at both the compute and I/O node side, and explored histogram compression techniques to reduce memory consumption and network traffic.

• We evaluated BG-REDUCT with GTC-P, a leadership-class SciDAC plasma particle simulation code. Our results demonstrate that BG-REDUCT is far more efficient as compared to existing models that we evaluated against and provides multifold benefits to parallel applications running on the Blue Gene/P platform.

This work was performed in collaboration with Abhishek Sreenivasa from North Carolina State University under the guidance of Dr.Scott Klasky from Oak Ridge National Laboratory, and Martin Schulz, Dong Ahn and other members at Lawrance Livermore National Laboratory.

1.3 Thesis Outline

The rest of the thesis is organized as follows: In Chapter 2, we provide the background to our research followed by related work. In Chapter 3, we describe in detail, the architecture of BG-REDUCT and provide an overview of the GTC-P, the application under study. In Chapter 4, we describe the implementation of BG-REDUCT and discuss our staging mechanism, histogram building and compression algorithms. In Chapter 5, we present the evaluation of BG-REDUCT against MPI based reduction techniques. We conclude our work in Chapter 6.
Chapter 2

Background

2.0.1 Blue Gene/P Overview

The IBM Blue Gene/P [41] is an architecture that drives performance and scaling while driving down cost. The system has been designed to increase the FLOPS/Watt and showcases a tiered architecture that boosts performance at low power. The system consists of 5 different networks, namely, a three dimensional torus for point to point message passing, a tree based Global Collective network for broadcast and reduction, a Global Interrupt network, 10 Gigabit ethernet for file I/O, and a Control network, to match different requirements. Every rack consists of 32 node cards and each node card houses 32 quadcore nodes. Top500.org lists 3 Blue Gene Solutions among the top 10 fastest supercomputers as of today.

The system has 2 kinds of nodes- the compute nodes and the IO nodes. The compute nodes are where users submit batch jobs to carry out all the computation tasks. These compute nodes run a lightweight operating system called the Compute Node Kernel. The compute nodes are diskless. Hence they need to route the I/O operations elsewhere. The data from a compute node is routed to its corresponding IO node, which is connected to the disks via the 10 Gigabit network. The IO nodes are responsible for performing all the I/O operations on behalf of the compute nodes and run normal Linux. Each compute node has 4GB of shared RAM and every
node card contains 1 or 2 IO nodes. The IO nodes obtain data and system calls from the compute nodes via system call forwarding using the Control and IO Daemon (CIOD). It is important to note that the IO node is transparent to the user. Figure 1 shows the Blue Gene/P environment [41].

2.0.2 MRNet

MRNet [37] is a software-based multicast/reduction network for building scalable performance and system administration tools. MRNet incorporates a master-slave architecture to build a tree based overlay network (TBON) of processes. The master serves as the frontend while the leaves of the tree serve as backends. Data flow in the MRNet network takes place in the form of streams. Using filters that manipulate the aggregate data that flows through the network, MRNet efficiently performs data reduction on the backend data. MRNet allows its users to develop custom data aggregation filters for complex data analytics and manipulation. It allows flexible organization of the tree work as well as supports scalable data aggregation. With a
packed binary representation of the data, MRNet presents high bandwidth communication. Scalable multicast ensures avoiding serialization of data as the number of backends increase.

2.0.3 LaunchMon

LaunchMON [5] is a software infrastructure for co-locating tool daemons with the distributed processes of an target HPC application. It seeks to help tool builders create highly portable and scalable tools through a standard framework that leverages underlying native Resource Management (RM) system services. LaunchMON is composed of four components: the LaunchMON Engine; the front-end API (FE API); the back-end API (BE API); and the middleware API (MW API). The LaunchMON FE and BE APIs provide information about application processes and scalably launch tool daemons on remote nodes. Similarly, the LaunchMON FE and MW APIs enable scalable launching and connection of middleware daemons. The Engine interacts with the RM to determine when, where and how to perform the services of the other components. A compact application layer network protocol, LMONP, enables interactions between all of LaunchMON’s components.

The main motivation of developing LaunchMON is to overcome challenges of ad hoc mechanisms for tool daemon launching, most frequently combining remote access commands like ssh or rsh with manual protocols to co-locate daemons with an application. While this scheme allows rapid software development, such an RM-agnostic scheme poses significant problems on high-end systems such as IBM Blue Gene or Cray XT family that do not support direct remote access services. The development and deployment of LaunchMON have been very successful. It has been ported on many high-end computing systems including Linux clusters, BlueGene/L and P, and Cray XT4 and 5. It has also been adopted to underpin key scalable development tools including DDT [8] as well as STAT [10, 20, 6] that has run successfully with 212,992 MPI tasks [20]. Using LaunchMon, we are able to design a staging area daemons that are launched on the BGP IO Node. Traditionally, developing a framework that involves BGP IO nodes would be almost impossible to achieve.
2.0.4 ADIOS

To meet the needs of scalable bandwidth, numerous studies have exploited different techniques to meet the bandwidth demands of scientific applications. However, the I/O performance on different systems strongly depends on the underlying machine characteristics and configuration. Thus, it is desirable that the high end user is able to choose the most efficient I/O method for different systems with the minimal effort in terms of user code alternation. ADIOS, the Adaptable I/O System, is an I/O componentization developed to address this issue. ADIOS allows the end user to select different I/O methods, including methods which use I/O staging [2]. The ADaptable I/O System (ADIOS [26], [25]) is a user layer I/O library developed at Oak Ridge National Laboratory to address the above issues.

The motivation of ADIOS is to offer an easy-to-use, flexible I/O APIs for scientists to manage their data in the code that may need to be written, read, or processed during simulation runs. ADIOS also provides a novel binary packed (BP) file format that has been demonstrated to achieve excellent read/write performances, scalability and I/O resiliency on the Cray XT4/XT5 with Lustre [35]. The ADIOS-BP file format of ADIOS, together with associated methods for data characterization, enables efficient data analysis. This file format is a metadata rich log file format designed to take advantage of concurrency on the file systems and has been shown to be efficient for both writes and reads on high end computing systems. In addition, ADIOS uses an external XML file to describe user data and therefore the routines in the user code can transparently change the way they process data. Different XML setups cause the code to behave differently without having to either change or recompile the source code. ADIOS affords the end user selecting different I/O methods for different groups of data in an application, such as restarts and diagnostics, so that the best performance can be achieved by employing the most efficient methods based on I/O patterns and run-time platform characteristics.
2.1 Related Work

Scientific Data Processing  Scientific data processing, including tasks such as reduction, analysis, and visualization, has been an active area of research. ADIOS [25] is an input/output and data organization API library for high performance computing using delayed consistency methods for the process of data output from the compute nodes of petascale machines. The authors demonstrate advantages derived from resilient data organization coupled with lightweight methods for data indexing and simple data characterization techniques that were introduced in the ADIOS API and BP format. LIVE [3] is a data workspace for creating data processing overlays called IOgraphs. LIVE exhibits a mechanism for extracting generated data based on a self-describing binary format and IOgraphs, which provides filters for distributed data transformation and output data size reduction. Ma et al. developed Godiva [31], a lightweight, database-like data management interface for batch-processing scientific data visualization tools to manage in-memory data, with transparent I/O optimizations such as caching and prefetching.

Recently online data-analytics and visualization has particularly attracted research efforts. Ma et al. discuss several techniques [28] for data reduction and challenges for in-situ visualization. Schroeder developed web based multi-point in-situ data tools for scientific data analysis [38]. Hercules [44] applies an end-to-end approach to tightly couple together all simulation components, including meshing, partitioning, solver, and visualization, and runs all components on the same supercomputer, eliminating intermediate I/O and data movement between simulation components. It requires scaling data analysis and visualization operations to the level where simulation runs and all simulation components need to be changed for addressing efficiency issues.

Our proposed BG-REDUCT system is complementary to the above offline and online data processing tools, by providing a novel framework for online data processing to be offloaded and isolated from compute nodes running a parallel simulation. This enables many existing data reduction/analysis/visualization techniques to be deployed in a transparent and scalable manner on the IBM BlueGene systems, an important class of petascale supercomputers.
**Data Staging**  It has been long recognized that the supercomputer I/O subsystem does not scale well enough to catch up with the computation side. One approach to address this problem is to quickly move output data away from the compute nodes. Recently, PreDatA [48] was developed as a data analytics framework for petascale applications, also integrated with the ADIOS I/O library. Designed for the Cray XT architecture, PreDatA uses a separate set of *staging nodes* to perform online analytics tasks, such as data sorting and binning, in addition to fast, asynchronous I/O. DataTap [1] is another system closely related to our BG-REDUCT. It introduces an abstraction known as "data services" to manage data from source to sink. Datatap servers serve as staging nodes carrying out online data processing, such as online file format conversion (from raw data to an intermediate binary format to scientific formats like HDF5). In some sense these systems are similar to the SDIO (Server Directed I/O) model [39], plus existing studies in aggressive write data buffering [17, 29]. The difference is that the staging nodes are much more versatile than the I/O servers in SDIO, capable of many data processing tasks beyond draining output data to the underlying parallel file system. Our work enables the data staging plus online processing realized by PreDatA to become available on BlueGene systems, where using additional staging nodes is difficult and inefficient. Also, by eliminating the explicit staging nodes requirement, online output data processing becomes more transparent and cost-effective.

There are several studies on data staging for better I/O performance. For example, DataStager [2] leverages server directed IO over RDMA for fast asynchronous parallel I/O and reduced performance perturbation to the computation performance, through intelligent I/O and network scheduling. DataStager copies data into a small fixed amount of space on the compute nodes and uses scheduling algorithms to move the data into the staging area. This approach is similar to BG-REDUCT, where local histograms are generated on the compute nodes and then moved to the staging area. PDIO [42] showcase portals-enabled compute nodes for high performance remote file I/O. DART (Decoupled Asynchronous Remote Transfers) [14] builds a thin communication layer on top of Portals library [11] to allow fast low-overhead access to data at
the compute elements, supporting high-throughput low latency asynchronous IO. The high-throughput data streaming substrate uses metadata rich outputs to support in-transit data processing and data redistribution for coupled simulations. BG-REDUCT uses the communication layer provided by MRNet to move data from compute nodes to I/O nodes.

**I/O Scalability and Efficiency** Another group of work aims at improving the HPC I/O stack for more scalable and efficient I/O. One example is LWFS (Lightweight File System) [34], designed to support a core set of critical I/O functionality required parallel scientific applications. In particular, a highly relevant project is Fast Collective-Network Protocol (FCNP) [36], a low-overhead, high-bandwidth network protocol developed for fast communication between the B/P compute nodes and I/O nodes.

**Data Compression Algorithms** Data compression has been explored for both HPC I/O and scientific data archiving. Lee et al. combined data compression with server-directed I/O to speed up online wide-area job output data migration [21]. Peter Buneman et al., [12] exploit hierarchical scientific data format properties to develop an archiving technique that is both efficient in its use of space and preserves the continuity of elements through versions of the database. Adaptive Coarsening [45] is an adaptive sub-sampling compression strategy that enables the compressed data product to be directly manipulated in memory without requiring costly decompression to produce a non-progressive multi-resolution representation, subdividing the dataset into fixed sized regions and compressing each region independently. In a recent work [40], difficulties encountered in deploying bitmap indexes with scientific data and queries from real-world domains are solved through the use of multi-resolution, parallelizable bitmap indexes, which support a fine-grained trade-off between storage requirements and query performance. Our BG-REDUCT framework exploited data packing and bitmap indexing techniques for sparse histogram compression, and is capable of incorporating other scientific data compression techniques as data filters.
Chapter 3

End to End Reduction Framework Architecture

3.0.1 BlueGene Reduction Architecture Overview

The BG-REDUCT framework consists of a single master FrontEnd (FE) and multiple BackEnds (BE) and intermediate communication processes in the staging area as shown in Figure 2. The FE is typically a Blue Gene/P login node while the BEs are Blue Gene/P compute nodes. We leverage Blue Gene/P IO nodes to form the staging area. Data may flow upstream or downstream through the reduction tree. Two key technologies, MRNet [37] and LaunchMon [5] are used to establish the end to end framework.

**Location-aware mapping**  It is important to note that the compute nodes on a particular nodecard have the corresponding IO node as their parent. In this manner we establish a parent-child connection between every IO node and 32 compute nodes that are physically located on the same nodecard. Such a location-aware mapping between child and parent avoids network related latencies for parent-child communication in the reduction tree.
FrontEnd Node  The FrontEnd Node (FE) is the master node of the BG-REDUCT end to end reduction framework. The FE typically resides on the Blue Gene/P login node and runs the MRNet and LaunchMon frontends. The FE is the first to launch and the last to exit in the reduction framework. The FE has several responsibilities such as:

1. Setting up the MRNet FrontEnd
2. Setting up the LaunchMon FrontEnd
3. Launching the application code
4. Launching MRNet communication processes on the IO nodes
5. Sending and synchronizing data reduction requests with the BEs
6. Writing reduced data to disk
7. Thoroughly tearing down the network upon task completion
**BackEnd Nodes**  A BackEnd Node (BE) in our end to end reduction framework resides on the Blue Gene/P compute node. The BE runs the MRNet Backend as well as the application process. They form the leaves of the end to end framework. Every BE has a parent node that forms the upper level of the communication tree. The corresponding IO node on the same nodecard as the BE is the parent of a BE. Thus, every 32 BEs on a particular nodecard have a common parent. The application process that the BE houses is a typical MPI petascale parallel scientific code. In our framework, the BE is launched from the FE instead of independently through the conventional resource manager job script.

**Intermediate Nodes**  A novel contribution of our research is the ability to leverage IO nodes for data reduction. By creating a staging area with IO nodes, we provide multifold benefits to users. With our approach, users are given access to one extra resource for every 32 Nodes at no extra cost. This number translates to approximately 3% extra nodes free of usage charges. In the absence of such a staging area, users would need to request these additional resources in the form of compute nodes, which not only would add to the user’s supercomputing account, but would also introduce network latency issues due to location-unaware mapping of the parent and child on compute nodes. IO nodes on Blue Gene/P are not accessible to users. We are able to leverage them by launching LaunchMon backends on them. The most significant contribution of LaunchMon is making available, a mechanism to launch processes on IO nodes.

The IO nodes form the middle level of the BG-REDUCT end to end framework. These nodes are parents to BEs and provide a communication link to the FrontEnd Node. Staging Nodes thus belong to the intermediate layer of the MRNet tree and run MRNet communication processes. In addition to providing a communication fabric between the FE and BEs, the staging nodes also perform data reduction on the intermediate data they receive from the BEs.
3.0.2 Application Case Study: GTC-P

Our data reduction framework serves massively parallel scientific applications. As we step into peta and exascale computing, the data needs of these applications have proliferated. Scientific applications normally compute data in the areas of fusion, astronomy, fluid dynamics and similar fields. Such calculations involve presence of large number of particles such as electrons and neutrons that operate in magnetic and electric fields. The application we study for our research is a PIC fusion code known as GTC-P [23]. As a fusion code GTC-P deals with particles and electrons that are scattered over an electromagnetic grid. These particles are characterized by voltage, velocity, weight and similar attributes and travel through the three dimensional space of the radial grid through the course of the application’s lifetime.

Output data files for GTC-P range from megabytes to terabytes depending on the problem size. For meaningful runs, the problem size is quite large resulting in output file sizes at least in gigabytes. Large data output sizes translate to I/O overhead causing scientists to perform I/O tasks less often than desired. We have learnt that for GTC-P, scientists currently carry out I/O once in every 400 iterations of the computation loop. Of course, we would like this output frequency to be much higher. Also, higher frequency I/O results in massive disk space
usage. The I/O timing analysis for GTC-P is shown in Figure 5. As we scale from 64 to 2048 processes, the percentage of total time for I/O activities increases rapidly. Similarly Figure 6 shows disk space requirements if the application saved frequent outputs. In our case study, we reduce a particles array named zion. Figure 7 gives us a brief overview of space requirements of zion in the final output. This number translates to about 89% of the total output file size.

The reduced output for the GTC-P case study is a 5 dimensional particle distribution histogram that characterizes zion in multiple dimensions such as rho, theta, phi, velocity and voltage. This 5D histogram is generated hierarchically at various levels of the reduction framework. A 5D histogram whose input is a very large data set could easily use up a tremendous amount of memory. However, we integrate state of the art compression techniques to shrink
the multidimensional binned data to eradicate issues that may arise due to the popular limited Blue Gene/P memory problems or network latency due to large packet sizes. Hierarchical multi-dimensional histogram binning is discussed in the following section.
Chapter 4

Implementation of End to End Reduction Framework

4.0.3 Filter Implementation

Data reduction is carried out by the aid of custom MRNet filters that we develop according to our problem requirements. Two filters are developed to compute the multi dimensional histogram representing zion in the GTC-P application. Minimum and maximum values of all the particles’ attributes are required to set the parameters for histogram computation at the BackEnds. Since each BackEnd is aware of only local particle data, a min-mix filter is used to determine the minimum and maximum values of all the particles’ attributes in the system. At Staging and FrontEnd nodes, this filter finds the minimum and maximum value at that level by finding the lowest among all the mimima and highest among all the maxima from the values received from its children.

The second filter is the histogram filter which computes the histogram of the particles in the system. At each stage, the filter performs the following steps:

- Receives compressed histogram from its child. At the leaves, local histograms are computed.
• Decompresses the intermediate histogram and performs an element-wise sum of the all the histograms to generate another intermediate histogram which represents a small percentage of the total particles in the system.

• Compresses the resulting histogram and sends the same to its parent. At the FrontEnd, the final histogram is written out to the disk.

4.0.4 Histogram Compression

Since we are employing a 5 dimensional histogram, we are aware that only a few bins would have non-zero values, essentially resulting in a highly sparse result dataset. We use efficient compression techniques to pack the sparse data for transportation through the reduction tree. In order reduce the network bandwidth and memory requirements in Staging and FrontEnd Nodes, compression is a highly recommended approach. Each compressed histogram consists of metadata, which is essentially bit array indicating whether a bin contains non-zero value. Only bins that have non-zero values are included in the compressed histogram. The compressed histogram is decompressed by reading the metadata and copying the non-zero bins into corresponding position in full sized histogram. For a 5 dimensional histogram of dimensions 64 x 64 x 64 x 16 x 16, there are a total of 67,108,864 bins. In a full sized decompressed histogram where each bin occupies 2 bytes, the size of the histogram is 134.21MB. In a compressed histogram, the size is dependent on metadata and the number of non-zero valued bins. Our metadata has a fixed overhead of 8.38 MB. For example, with 90% of the bins in the histogram empty, the size of the compressed histogram is 21.80 MB (8.38 + 13.42).

4.0.5 Communication Tree

The FrontEnd(FE) of BG-REDUCT is the most crucial piece of the system as it is responsible for launching all BackEnds as well as spawning the IO node daemons. With the aid of the LaunchAndSpawnDaemon API provided by LaunchMON, the FE launches the MPI job on all compute nodes. Normally process is carried out by the resource manager of the system. With
our implementation, we bypass LoadLeveler, Blue Gene/P’s resource manager and job scheduler to launch MPI processes internally. The BEs form the leaves of the MRNet tree. We use the *backend-attach mode* of MRNet to facilitate the compute nodes to connect to the MRNet tree. The middle level of the MRNet tree resides on IO nodes. These processes are spawned by placing LaunchMON backends on IO nodes and then launching MRNet communication processes from within these LaunchMON Backend processes. Once the IO nodes attach to the MRNet tree, they write out config files that are read by compute nodes to obtain connection information. In this manner, the communication tree is established.

### 4.0.6 ADIOS Extention

We integrate the End to End Reduction Framework with ADIOS, to provide a staging platform for Blue Gene/P systems. The Reduction method is introduced as a ADIOS Transport Method that users can use in the input xml file. Our framework is designed in such a manner that application scientists need minimal changes to the application code. Currently we provide particle distribution function generation and particle smoothing as extensions. However, custom data filters can be easily implemented and plugged in to suit users’ requirements. The integration of BG-REDUCT with ADIOS may be accomplished by adding BG-REDUCT into the Transport Method API provided by ADIOS. We plan to provide some basic scientific filters similar to our current histogram filter.
Chapter 5

Evaluation

5.1 Overview

BG-REDUCT is a flexible data reduction infrastructure with pluggable data reduction filters. These filters can be easily developed to suit application needs. In our research, we analyze the effects of online data reduction for applications such as GTC-P. We analyze results for hierarchical multidimensional data binning of particle data in GTC-P. Experimentation involves comparison of performance of staging based versus MPI based histogram generation as well as histogram generation using MPI-Reduce native function. We also analyze performance as well as overheads at each stage of the binning process. We study the visible cost of histogram generation, impact on total execution time and various overheads.

5.1.1 Experimental Environment

Experiments are run on the Oak Ridge National Laboratories Blue Gene/P Eugene platform. Eugene is composed of two racks (2048 compute nodes) of the standard BlueGene/P configuration. Each node has a single quad core processor, for a total of 8192 compute cores. Eugene uses the standard IBM software stack. Each rack has 32 IO nodes; each IO node serves the I/O requests from 32 compute nodes. The IO nodes are connected to each other and to disk.
over a 10 Gigabit Ethernet network, provided by a 256 port Myricom switch. The system uses two GPFS filesystems, one for scratch space (70 TB) and a second for longer term code storage (18 TB). The GPFS system includes 8 file servers and 2 metadata servers. Data is stored in 24 LUNs, each of which is approximately 3.6 TB in size. Individual LUNs are an 8+2 array of DDN disks, which communicate through dual DDN SA29500s using Infiniband.

5.1.2 Evaluative study and implementation of alternate approaches for histogram reduction using MPI

We develop 2 alternate implementations of histogram generation for comparison with BG-REDUCT. Both versions are developed in order to highlight the importance of the staging area. Our comparison implementations are MPI-based and designed to process data exclusively in compute nodes in the absence of staging. In this section, we provide a brief overview of these implementations.

MPI Send/Receive based reduction

Histogram generation and data reduction in the MPI Send/Receive based reduction is carried out completely at the compute nodes. We do not request additional resources in this approach and assign reduction tasks to a few nodes running the application. Rank 0 is assigned as the Master and plays the role of the BG-REDUCT FrontEnd. A reduction tree is constructed, similar to BG-REDUCT and the same compression algorithms used for BG-REDUCT are leveraged for data compression. It is important to note that in this setup the Blue Gene/P 3D Torus network is responsible for all communication tasks since they involve MPI based communication between compute nodes.

MPI_Reduce based reduction

the MPI_Reduce based histogram processing implementation relies completely on the native MPI_Reduce call provided by MPI. This approach is a simpler implementation and does not
deploy any data compression techniques. Local histograms are generated at all compute nodes and reduced at Rank 0. The Blue Gene/P architecture is designed to perform all reduction operations through a special Global Collective Network, which is used for communication in the MPI Reduce based reduction technique.

5.2 Results

5.2.1 Impact of compression

BG-REDUCT uses sophisticated compression techniques to reduce the size of data flow through the network. Compression is applied at all levels of the reduction tree. In this section, we evaluate and discuss the effect of compression. Figure 5.1 (a) represents the compression achieved by BG-REDUCT at the Backends. At the BackEnds, we are able to achieve about 93% compression of the histogram reducing the size of the histogram from 130MB to just over 8MB in all cases. We achieve such high levels of compression as local histograms are generally very sparse in nature. Figure 5.1 (b) depicts compression at the FrontEnd. As the number of processors increase, the total number of particles in the system increases. As a result, the final histogram represent an enormous number of particles for larger runs. We use 128,000 particles per node for all our runs. We scale the problem in such a way that the number of particles per node remains the same but as we increase the number of nodes, we increase the area of the grid where the particles travel. This form of scaling is known as weak scaling. From the figure we see that we are able to achieve close to 90% compression for 64 processors but this value falls to roughly over 45% for 1024 processors.

Figure 5.2 shows the impact of binning and compression on the size of output data written to disk. As the binning parameters remain constant even though the total number of particles in the system increases, the histogram output is always 130MB. This compared to raw output data is not only much lesser in volume but also shows scalability. Each iteration of raw output written to disk would only increase with problem size. For example, we see that each iteration
of output for 1024 nodes is 15GB of data.

### 5.2.2 Visible cost of PDF generation

In this section we analyze the visible cost of PDF (Particle Distribution Function) or histogram generation as seen by the application. We highlight the advantages of using a staging area in order to offload processing to a separate set of nodes. Our BG-REDUCT framework performs remarkably well and is noticeably faster than MPI Send/Receive based and MPI Reduce. All error bars in figure 5.3 are plotted using 95% CI. Figure 5.3 outlines the visible cost at the compute nodes in all three approaches. BG-REDUCT computes and generates a histogram in under 20 seconds for up to 512 processors and takes 42.8 seconds for 1024 processors. MPI Send/Receive based reduction performs observably slower and generates histograms with the same algorithms as BG-REDUCT but takes well over 100 seconds. This result clearly shows that the same operations performed without a staging area may be exceptionally slower. MPI Reduce based technique performs the worst as shown in Figure 5.3. MPI Reduce is a stable and reliable alternative but its performance is a serious drawback. This result points out the advantages of data compression when large amount of data needs to be communicated between
processors. On the whole, we clearly show that the performance of BG-REDUCT is a cut above both MPI Send/Receive and MPI Reduce based data reduction techniques.

![Figure 5.2: Size of output on disk](image)

We now analyze each histogram generation technique and provide a split up the visible cost. Figure 5.4 shows that BG-REDUCT spends less than 5% of its total time in calculating the Min and Max values. Roughly 10-20% of the time is spent in histogram computation. Majority of the time is spent in histogram compression. This value drops as we move from 64 to 1024 processors.

![Figure 5.3: Visible cost of PDF generation](image)
processors as histograms become denser and less time is spent in compression. The rest of the time can be accounted for as time for sending the histogram through the tree and other network related waiting.

Figure 5.4: Dissection of BG-REDUCT histogram generation

![Histogram Generation Chart](image)

Figure 5.5: Split up of alternate approaches

(a) Dissection of MPI Send/Receive

(b) Dissection of MPI Reduce

Figure 5.5 (a) and (b) show the split up of histogram generation for MPI Send/Receive and
MPI\texttt{Reduce} based implementations. MPI Send/Receive based technique spends about 95% of the time in network related calls like MPI\texttt{Send} and MPI\texttt{Receive}. Here the Torus network is used for communication. The MPI\texttt{Reduce} function accounts for more 99% of the time spent in histogram generation in the MPI\texttt{Reduce} based implementation. Even though the Global Collective Network is used by MPI\texttt{Reduce}, it still shows slow performance.

5.2.3 Impact on total application time

![Figure 5.6: Impact on total execution time](image)

Figure 5.6 shows the impact of the time taken for data reduction on total application execution time. BG-REDUCT takes less than 7% of the total execution time. MPI Send/Receive based data reduction being slower has a larger impact on the application as shown in the figure. MPI\texttt{Reduce} based data reduction accounts for greater than 50% total execution time. This clearly is not acceptable for applications. However, we do observe that ADIOS write calls in original I/O still perform better than BG-REDUCT. We can account for this performance improvement as ADIOS only spends time writing the raw output to disk without post processing. In, BG-REDUCT all BEs are involved in computing local histograms before the data is sent out to the staging nodes and this adds to the total application time. Also, BG-REDUCT has
certain amount of a one time setup overhead for building the communication tree.

5.2.4 BG-REDUCT setup time overhead

BG-REDUCT requires a certain amount of time for initial setup. This number however is not very large. BG-REDUCT requires a few seconds for setup as shown in Figure 5.7, which is plotted with 95% CI. We see that BG-REDUCT does not have significant setup overheads even though it establishes a sophisticated communication tree with upto thousands of nodes.
Chapter 6

Conclusions

We develop BG-REDUCT, a scalable and efficient framework for data reduction and post processing. With BG-REDUCT, we introduce a staging area that could be integrated with ADIOS to provide efficient staging for Blue Gene/P. Data reduction has endless benefits as we enter the exa-scale age of parallel computing. With terabytes of data being generated every few iterations, large amounts of raw data can be reduced to few megabytes of useful, processed, intelligent data. BG-REDUCT provides support to such data reduction and we demonstrate the advantages of data binning and histogram generation.

By leveraging several existing building blocks, including the MRNet multicast/reduction tool [37], the LaunchMon infrastructure for co-locating tool daemons with parallel applications’ computation processes [5], and the ADIOS parallel I/O middleware [16], BG-REDUCT provides a scalable and efficient framework as demonstrated in the results. BG-REDUCT is transparent to application codes, and can be enabled by adding less than 5 lines to the application code and minimal job script changes. We adopted a multi-phase reduction scheme that effectively use resources at both the compute and I/O node side, and explored histogram compression techniques to reduce memory consumption and network traffic. We evaluated BG-REDUCT with GTC-P, a leadership-class SciDAC plasma particle simulation code and compared performance with MPI Send/Receive based and MPI_Reduce based reduction techniques. BG-REDUCT
out-performs both techniques and proves to be more efficient.

BG-REDUCT shows multiple benefits as listed below.

- Drastic reduction in the size of output files from terabytes and gigabytes to a few megabytes
- Time savings as a result of avoiding large data writes
- Ready to use meaningful data without offline processing
- Higher frequency of meaningful output for every application run
- No additional compute resource overhead due to staging area
- More work done at the same cost

We plan to extend our work to providing many more data reduction filters such as data sorting and smoothing and tightly integrating BG-REDUCT with ADIOS to provide a solid and scalable reduction framework for ADIOS on Blue Gene/P.
REFERENCES


