Application Controlled Parallel Asynchronous Input/Output

Shujia Zhou[1], Amidu Oloso[2], Megan Damon[1], Tom Clune (NASA-GSFC SIVO)

Abstract:

File systems continue to be a major performance bottleneck for many applications across a variety of hardware architectures. Most existing attempts to address this issue (e.g., PVFS), rely on system resources that are not typically tuned for any specific user application. Others rely on special hardware capabilities such as shared-memory.

We have developed an MPI-based Parallel Asynchronous I/O (PAIO) software package that enables applications to balance compute and I/O resources directly. PAIO uses a queuing mechanism to stage the data, sent over in parallel from compute nodes, on the reserved I/O nodes. Because the bandwidth of the inter-processor network greatly exceeds that of the file system, significant performance improvements can be achieved under a bursty I/O load provided sufficient memory is available for the I/O nodes. The results of PAIO for typical weather applications on an SGI Altix and other architectures are presented.

Goal:

To develop an application-controlled, portable, high-performance I/O library to maximize computing efficiency.

Background:

Many high-performance computing applications need to write simulation data to disks for analysis in a short turnaround time. Typically such an operation is constrained by the underlying file system. The desire for high-resolution simulations and the trend of rapidly increasing computational nodes further burden the file system. The low performance of typical file systems forces compute nodes to idle for a significant time while writing the data to disks. For example, the NASA Goddard Space Flight Center (GSFC) Cloud Model (Figure 1) requires 1,913 wall-clock seconds, ~46% of the total simulation time, to write twenty-three 3D single-floating-point arrays of 1024 x 1024 x 41 to one file on a disk of an HP AlphaServer SC45 in running a one-model-hour simulation using 256 CPUs during which six such writes occurred.

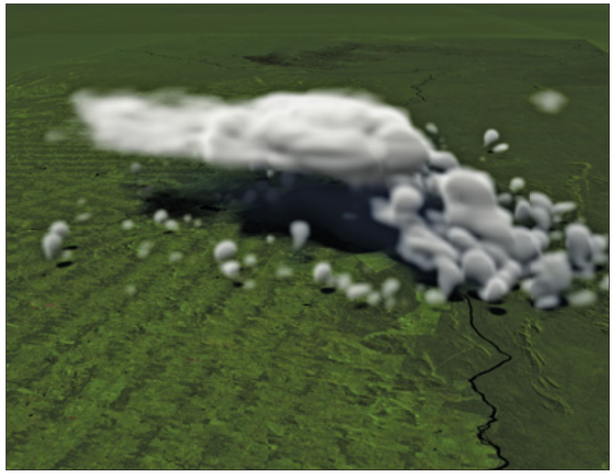

Figure 1: NASA GSFC Cloud Model is capable of simulating the cloud formation process and providing physical input for global circulation models. The image is one snapshot at Lat 10.14 S x 61.91 W (Brazil) created in a simulation with 256 x 256 x 41 resolution. The area covered is 64km x 64km with a maximum height of 22.437 km. The background is imposed with the Landsat satellite data. Image by James Williams (GST).

[1] - Northrop Grumman Corporation
[2] - Advanced Management Technology, Inc.

PVFS-1 Preliminary Examination:

We have examined the performance of PVFS-1 using one storage node on the NASA Thunderhead cluster (Figure 2). The purpose of our experiment was to test the buffering capability of the I/O nodes, not the disk performance. The test application written in Fortran was executed on one compute node. To bypass the limitation of local memory (i.e., 1 GB) on the compute node, the application writes 20 small 1D Fortran arrays to the I/O node.

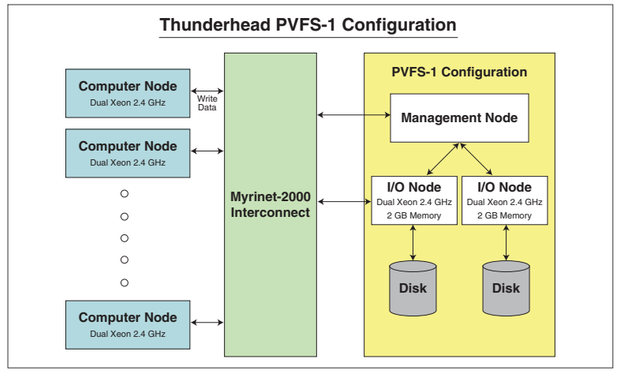

Figure 2: The schematic of PVFS-1 configuration on the NASA Thunderhead cluster.

As indicated in Figure 3, the buffering capability does alleviate the I/O bottleneck on the Thunderhead cluster. We observed that once the total data size reaches ~1.35 GB, which is 67.5 % of the memory on the I/O node (i.e., 2 GB), a sharp decease in write performance occurs: from ~72 to ~55 MB/s. We conjecture that the performance drops precipitously at the point where the data volume exceeds the available buffer cache.

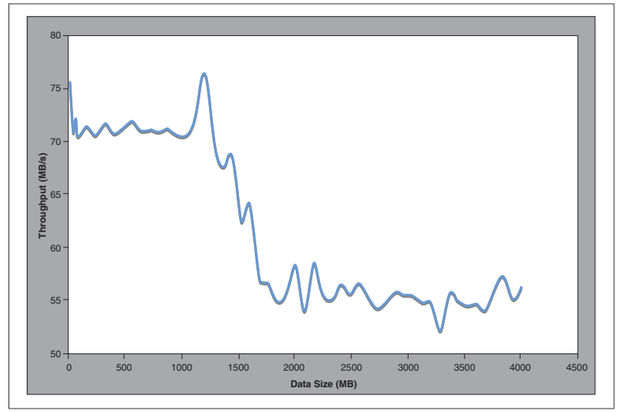

Figure 3: The throughput of PVFS-1 on the NASA Thunderhead cluster as a function of data size.

Approach:

- Move data out of compute nodes to I/O nodes via inter-processor network
  - Harness the bandwidth of the inter-processor network, which greatly exceeds that of the file system
- Allow a user to determine when to send the data which to I/O nodes and when to flush the data into the disks
  - Balance the bandwidth of the inter-processor network, the number of I/O nodes, the memory size of the I/O node, and the disk speed
- Use queuing mechanism to optimize the amount of data in I/O nodes
  - Cache data according to available memory in I/O nodes

PAIO Technical Description:

As illustrated in Figure 4, the process of writing data into a disk is decomposed as:

- Send data from a compute node to a corresponding I/O node
- Store the data in a queue
- Pull the data out of the queue
- Write the data into disks

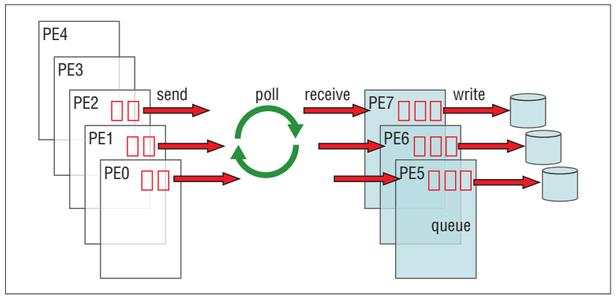

Figure 4: The system architecture of the application-controlled parallel asynchronous I/O.

In addition, a polling mechanism is used between Send– and Receive–operations to support a more flexible mode of operation. Moreover, the code is written in MPI to ensure portability and Fortran 95 to provide a user-friendly interface. Finally sending data in parallel from multiple compute nodes to the corresponding I/O nodes is implemented to aggregate the bandwidth of the inter-processor network.

Results:

We have performed a series of performance tests using our application-controlled PAIO for an output pattern typical of weather and climate applications. Namely, 10 single-precision arrays of 1440x720x70 (~290MB per array) are written to a disk consecutively. A test run on the GSFC SGI Altix, which has unidirectional internal network bandwidth of 3.2 GB/s and 2 GB of memory per processor, shows that the speed of writing these arrays to a disk with a single processor is ~298 MB/s. By using PAIO, data is sent from a compute node (Process 0) to the I/O node (Process 5) at 590 MB/s. By eliminating unnecessary data copies, we further improved the performance to ~735 MB/s, which is a 2.5X improvement over the direct disk performance. Since the total size of the 10 arrays exceeds the available memory on a single node of the HP AlphaServer SC45, we performed a smaller test with just eight arrays of slightly smaller size (~1.37 GB total) on that platform. With PAIO, we achieved ~121 MB/s, which is a 3X improvement over writing data directly to a disk. To further exploit inter-processor bandwidth, PAIO has the capability of using multiple I/O nodes to cache the data. In the above configuration with eight arrays, we observe speedups of 1.7X and 2.3X by using two and three I/O processors, respectively.

Conclusion:

The results clearly indicate that our application-controlled PAIO library is capable of considerably increasing I/O performance.

Next Steps:

- Investigate variations of communication strategies for sending data to the I/O nodes
- Enable use of netCDF to support common data format
- Enable use of MPI I/O to further optimize management of distributed data
- Use PAIO in production codes to measure real benefit to the end user

Acknowledgments:

We would like to thank the NASA Center for Computational Sciences (NCCS) for access to the Explore SGI Altix supercomputer, and John Dorband (NASA GSFC) for his suggestion on the system caching mechanism of PVSF-1 and for his assistance in using the NASA Thunderhead cluster. We would also like to thank Wei-Kuo Tao (NASA GSFC) for providing the NASA GCE code and Xiping Zeng (UMBC) for assistance in using the code.