Petascale I/O using HDF-5 *

Nicholas Malaya
Institute for Computational Engineering and Sciences
The University of Texas at Austin
nick@ices.utexas.edu

Karl W. Schulz
Texas Advanced Computing Center
The University of Texas at Austin
karl@tacc.utexas.edu

Robert Moser
Institute for Computational Engineering and Sciences
The University of Texas at Austin
rmoser@ices.utexas.edu

ABSTRACT

Our work is focused on performing Petascale Direct Numerical Simulations (DNS) of turbulent flows. An essential performance component of these simulations are the restart files, as a single petascale simulation will write on the order of a petabyte. Petascale I/O requires both performance and dataset maintainability, for archival and post processing of the velocity fields for statistics. This paper presents benchmarks and comparisons between a single shared file written using the HDF-5 library and a POSIX compliant I/O library on several top-10 machines and filesystems. It is shown that a properly tuned HDF-5 routine provides strong I/O performance, which coupled with the metadata handling and portability available to the file format, indicates that the lower performance provides a worthy tradeoff. The benchmarks presented were provided from real turbulence simulations and required extensive tuning for different platforms, and in particular, between different file systems (GPFS and Lustre).

Categories and Subject Descriptors

B.4.3 [**Interconnections (Subsystems)**]: Parallel I/O

General Terms

Design, Measurement, Performance

Keywords

HPC, Petascale I/O, Computational Fluid Dynamics, DNS

INTRODUCTION

The flow of fluids is central to many important natural phenomena and engineering technologies and in many cases, turbulence plays a dominant role. Unfortunately, turbulence is notoriously complex, and its effects are difficult to predict or manipulate. The one reliable tool that we do have for analyzing the effects of turbulence is direct numerical simulation (DNS), where the governing equations of fluid mechanics (the Navier-Stokes equations) are solved with high accuracy in time and three-dimensional space. Although DNS is too computationally expensive for engineering calculations, high quality DNS is an excellent tool for investigation of turbulent flow physics [6, 5].

Effective petascale computation of turbulent flows is a challenging problem on several fronts, requiring large volume communication in all-to-all exchanges between nodes, effective fft implementations, etc. I/O plays a particularly important role because of performance requirements as well as the extensive metadata requirements for post-processing by various researchers.

We are developing an open-source DNS codebase to be tuned for various platforms. In addition to requiring efficient FFT and MPI collective performance, another critical component for petascale DNS applications is the inclusion of an efficient and well tuned I/O subsystem to output and catalog the significant data generated by the simulation.

REQUIREMENTS

Petascale machines should enable effective computation of velocity fields at 8192^3 (or equivalent) sizes. The storage requirement of double precision velocity fields with three components of velocity are $(8 \times 3 \times N_x N_y N_z)$ and as a result, restart files will be on the order of ten terabytes. The frequency with which a checkpoint is written depends on system reliability, but once per wall clock hour is typical. Approximately 100 such files are archived (e.g. one petabyte) for postprocessing. This becomes a significant fraction of the runtime of a simulation ($\approx 10\%$).

Analysis of DNS results is itself a computational challenge. The primary task is to compute statistical diagnostics averaged over the restart fields saved from the simulations. Once the simulations have been performed, the fields they generated can be used like an experimental facility to "measure" most any diagnostic quantity of interest. Scientific advances will come from repeated analysis of the simulation data to

answer new questions and test hypotheses, by a number of researchers over a period of years.

Therefore, a DNS requires more than just effective write speeds. Postprocessing is also a supercomputing problem, requiring millions of CPU hours on many processors. In addition to simulation performance, extensive metadata, single file output and efficient field organization can greatly facilitate effective postprocessing of DNS fields.

Our code achieves massive parallelism in part from a 2D decomposition which divides a $N_x N_y N_z$ domain into (e.g.) M "pencils" (shown in figure 1) each of size $(N_x/M_1)N_y(N_z/M_2)$ where M_1 and M_2 define a 2-D processor grid (with $M = M_1 \times M_2$, in principle up to $N_x N_z$). However, simply duMPIng these pencils to a file complicates the postprocessing and also increases the difficulty in running problems of different processor count. As a result, we desire to write the turbulent field as a cube that assumes no particular decomposition or processor count.

Depending on the underlying file system implementation, the use of many file I/O, where each MPI task writes a single file containing a portion of the decomposed solution, can be fast and easy to implement. However, at large scale, this approach can become extremely cumbersome when it comes to subsequent archiving, file transfers, postprocessing, and general interactivity. In addition, this method of writing may become increasingly infeasible for a filesystem to support given the sheer number of files produced during a run on massive core counts.

The current DNS codebase utilizes many file I/O, which has required significant overhead. In particular, commands to list files (`ls`) can require several minutes to return, and removing millions of files (using `rm`) can take hours. The code requires additional processing to enable reading the fields by SILO[1], where it is essentially combined into a single file output. Finally, no metadata is attached to output files, leaving the problem parameters of the output files badly documented.

As a result, a major impetus for the current work is to explore the feasibility of adopting parallel I/O within a single solution file using modern I/O libraries for portability and metadata inclusion.

Several modern high level I/O libraries were considered (NetCDF, HDF-5, MPI-IO, etc.). The HDF-5 library was chosen because it appeared to possess the most advantages. It is portable and possesses many internal optimizations that enable the user to quickly achieve high-parallel I/O efficiency. Besides portability and readability by a variety of visualization and analysis tools, there is separate storage in HDF-5 file for metadata and datasets, so the data blocks can be of a compatible size and aligned with file system blocks. Tests conducted implied that the efficiency of HDF-5 is nearly as high as the MPI-IO library on which it is based, so there is essentially no speed advantage in writing binary files using MPI-IO. The perceived advantages and disadvantages between many file I/O and single file HDF-5 (unified HDF) output are summarized below.

Many File I/O:

- Pros
  - Parallelism
  - High Performance
  - I/O code is simple
- Cons
  - Huge number of files
  - Domain is decomposed.
  - Difficult to read on different number of processors
  - Requires additional post-processing for visualization
  - Difficult to ls, archive, transfer files

Unified HDF I/O:

- Pros
  - Single file output
  - Decomposition independent
  - Read by host of visualization software
  - Intrinsic metadata support
  - Files are endianness independent (portable)
- Cons
  - Slower than many file I/O
  - I/O strategy requires partitioning information
  - Requires more tuning for performance
  - Massive file size
  - Requires HDF library support

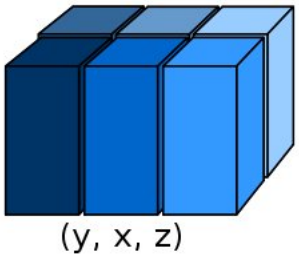

Figure 1: The two dimensional data decomposition.

RESULTS

A library written in HDF-5 was developed with the requirements listed in the previous section. This library was benchmarked on several currently top-10 machines[4] (University of Texas's Ranger, University of Tennessee's Kraken and Blue-Gene/P Intrepid at Argonne National Laboratory), and the results found for the aggregate write speeds are displayed in figs 3-4. On node performance and communication interconnect speeds are the principle timing information used to

determine the number of cores used in a turbulence simulation. Thus, as far as I/O is concerned, we are more interested in weak scaling problems than strong scaling. For each test conducted, the total write size per task was 20 MB. Thus, the largest single file output was 163 GB. Write speeds can vary greatly because of simultaneous write operations from various applications competing for resources. As a result, to improve the reliability of the measures, 10 samples were taken for each processor count. The individual many file write speeds were determined by averaging the time reported by each task using `MPI_Wtime()`.

The current HDF-5 library has a small number of tuning parameters that can be set by the user based on file system characteristics. Several of these parameters, such as H5Pset_hyper_cache and H5Pset_sieve_buf_size lead to substantial performance gains.

On Ranger and Kraken, the Lustre file system[3] is sufficiently exposed that several "dials" can be adjusted. The Lustre file system is made up of I/O servers and disks called Object Storage Servers (OSSs) and Object Storage Targets (OSTs), respectively. A file can be striped and written across multiple OST's concurrently. File striping increases I/O performance by writing to multiple OST's simultaneously and therefore increasing the available I/O bandwidth. Aligning I/O operations to stripe boundaries, with transfer sizes an even multiple of the stripe size greatly improved performance. The maximum stripe count is currently limited to 160 on a Lustre system. This is not a variable parameter on the GPFS[2] filesystem used by Blue-Gene/P Intrepid, where the file system automatically distributes disk blocks across all the disks. Meta data typically exists as very small chunks of data. As a result, metadata writes are best aggregated by increasing the H5Pset_meta_block_size to the I/O request size, such as 1 MB.

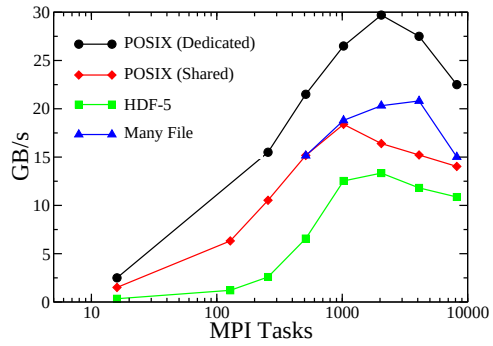

Figure 2: Write speeds as a function of MPI tasks on Ranger (Lustre Filesystem) during normal (shared) and quiet (dedicated) production.

It is typical to compare the write speed of any application to the peak filesystem throughput. However, this metric is misleading, as no application can duplicate the "quiet system" under realistic run conditions because other users I/O needs causing contention. As a result, a POSIX compliant I/O library was written and optimized to provide a more realistic comparison. POSIX compliant is used here to imply `fopen()`, `fseek()`, `fwrite()`, etc. function calls. This POSIX library was carefully tuned on all the machines tested, with stripe count (on Lustre filesystems), stripe sizes and block write sizes set based on the recommendations of the respective supercomputing staff and shown in table 1. This POSIX library provides the most direct point of comparison against the HDF-5 library: it also produces a single file output, and stores the velocity fields in an identical manner (i.e. a cube).

parameter	Ranger	Kraken	Intrepid
stripe count	1	1	N/A
stripe size	20 MB	20 MB	N/A

Table 1: Tuning parameters used for POSIX I/O

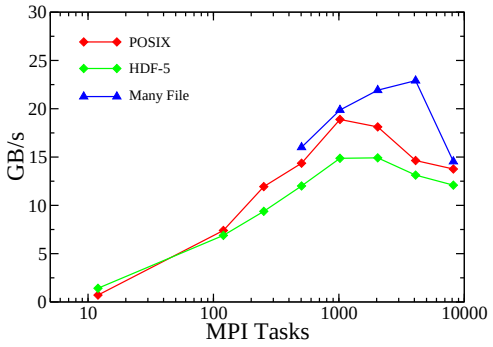

Figure 3: Write speeds as a function of MPI tasks on Kraken (Lustre Filesystem) during normal (shared) production.

A comparison between POSIX I/O on a "quiet" and "noisy" machine is shown in fig 2. One can plainly see that performance is severely reduced in a production (shared) environment. As a result, the quoted peak throughput of 30 GB/s for Ranger is not a realistic point of comparison to the library, with 18 GB/s write speeds presenting a much more realistic target. As a result, the peak write speeds from our tuned HDF-5 library of 13.34 GB/s should not be viewed as a loss of over 50%, but a much more modest 26.6% reduction.

These benchmarks were repeated on Intrepid and Kraken and are shown in figs. 3 and 4, respectively. We are not in possession of the quiet write speeds as with Ranger, but quoted filesystem peak I/O throughput is 30 GB/s and 65 GB/s for Kraken and Intrepid, respectively. As with the previous plot, a comparison between the POSIX writer and the HDF-5 Library reveals competitive performance.

It should be noted that in all the systems plotted, file system contention causes serious degradation of performance at larger core counts. The large number of processes overwhelm the file system resources (Object Storage Targets).

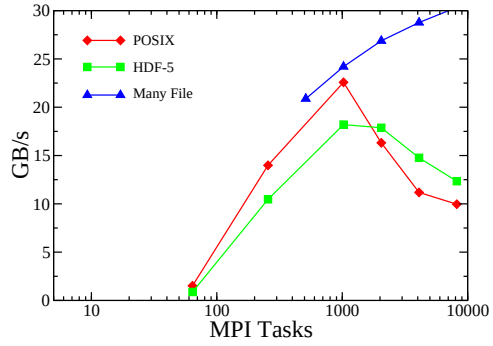

Figure 4: Write speeds as a function of MPI tasks on Intrepid (GPFS Filesystem) during normal (shared) production.

This can be alleviated in part by subsetting the I/O. In particular, using a "relay" strategy, where the total number of MPI tasks are broken into sets of writers, significantly reduce filesystem contention. This was tested on the problems and machines sizes shown above, and worked well, providing sustained writes only slightly reduced from the peak measured speeds. For Ranger and Kraken, the processors were broken up in to batches of 1024 and 1020, respectively, and the simulation was observed to maintain nearly peak write speed for as high as 32k processors.

These write times are sufficiently quick that the HDF-5 set of routines on the target problem size(8192^3), using the relay strategy outlined above, and writing one restart per wall clock hour, will require less than 10% of the runtime of the DNS, which is within the requirements stated above.

CONCLUSIONS AND FUTURE WORK

This paper provided an analysis of write speed timings conducted between many file I/O, a POSIX compliant single-file write and a library written in parallel HDF-5. A program with properly tuned HDF-5 provides lesser, but not unreasonably reduced I/O performance in comparison to the other implementations. Coupled with the intrinsic metadata handling, dataset portability and widespread availability and support of the library, the high-level library HDF-5 is a reasonable selection for Petascale I/O needs.

As a result of this study, the HDF-5 subroutines used for benchmarks in this paper are being incorporated into the Petascale Direct Numerical Simulation Library, replacing the many file I/O scheme previously used. Additionally, this tuned, parallel HDF-5 library is being incorporated into several other projects at the center for Predictive Engineering and Computational Sciences (PECOS), and upon further code-review and testing, will be released open source (LGPL) for public use.

We are particularly interested in further developing this library for extreme core count runs on next generation machines. Aside from further tuning of HDF-5 library parameters, future work will focus on implementing overlap between computation and asynchronous I/O, which using RDMA technology should be possible on a next generation machine such as NCSA's Blue Waters or ALCF's Mira.

FFTW's success as a fourier transform library comes because of the excellent performance available on a variety of platforms. It achieves this portability by supporting a variety of algorithms which are selected based on tests performed by a subroutine call, `fftw_plan`. In much the same way, the authors of this paper conclude that HDF-5 could greatly benefit from a similar auto-tuning project that that would allow the software to determine the best values of the tuning parameters.

ACKNOWLEDGMENTS

The authors would like to thank TACC, ALCF and NICS for computational resources and tuning assistance. This work was supported in part by NSF PetaApps award OCI-0749286, as well as the Center for Predictive Engineering and Computational Sciences (PECOS) at the University of Texas.

REFERENCES

[1] *https://wci.llnl.gov/codes/silo/.*

[2] *http://www.almaden.ibm.com/StorageSystems/projects/gpfs/.*

[3] *Peter J. Braam. Lustre: a scalable high-performance file system, Nov. 2002.*

[4] www.top-500.org.

[5] J. Jiménez and R. D. Moser. What are we learning from simulating wall turbulence? *Phil. Trans. Royal Soc. A*, 365:715–732, 2007.

[6] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *ARFM*, 30:539–578, 1998.