Performance of a direct numerical simulation solver for turbulent combustion on the Cray XT3/XT4

Ramanan Sankaran and *Mark R. Fahey*
National Center for Computational Sciences
Oak Ridge National Laboratory

Jacqueline H. Chen
Sandia National Laboratories, Livermore, CA

Abstract
Parallel performance of a flow solver, S3D, for direct numerical simulation of turbulent combustion is evaluated on the latest Cray XT system installed at the National Center for Computational Sciences (NCCS). S3D is a massively parallel code that uses MPI for message passing parallelism. It has been ported to various platforms at the Office of Science computational facilities and scales very well to a large fraction of the system. In the past, S3D showed near perfect parallel scaling in the performance studies conducted on the Cray XT3 system. However, S3D was found to exhibit higher parallel communication overheads on the Cray XT system at NCCS following a recent upgrade. This article presents the results of the scaling study and an analysis of the parallel and dual-core performance.

1 Introduction

Direct numerical simulation (DNS) of turbulent combustion is a scientific tool used to gain fundamental insight into the coupling between fluid dynamics, chemistry and molecular transport in reacting flows. A major challenge faced in solving the governing equations for reacting flows lies in the large range of highly coupled length and time scales that must be resolved by the simulation. The DNS approach fully resolves the length and time scales described by the governing equations and the only models deployed are for the evaluation of the transport properties (viscosity, thermal conductivity, mass diffusivity etc.) and chemical kinetics (reaction rate constants). DNS is a computationally expensive technique and will remain intractable for solving industrial problems for the foreseeable future due to the inability to resolve the entire range of scales that exist in practical combustion devices.

In recent years, the rapid advance of computational capabilities has presented significant opportunities for increasingly realistic DNS of turbulent combustion flows, which are very well suited to assess the validity modeling approaches and their underlying assumptions. The current terascale computational platforms allow DNS of combustion using canonical configurations and using geometries similar to the flames studied experimentally in laboratories. With full access to the spatially and temporally resolved fields, it is possible to extract, under well controlled conditions, detailed information about small scale flame behavior that cannot be measured by experiments. This information can then be used to develop modeling strategies for computational fluid dynamics (CFD) approaches that do not resolve the entire range of scales.

In section 2 we present the turbulent combustion simulation code S3D. In section 3 we describe the model benchmark problem used for the performance and scalability studies. Section 4 focuses on the parallel scalability of S3D particularly on the Cray XT3/XT4 platform. Section 5 briefly discusses multicore performance. Lastly, in section 6 we present our conclusions.

2 S3D Overview

S3D is a massively parallel DNS solver developed at Sandia National Laboratories [1, 2]. S3D solves the full compressible Navier-Stokes, total energy, species and mass continuity equations coupled with detailed chemistry. It is based on a high-order accurate, non-dissipative numerical scheme. The governing equations are solved on a conventional three-dimensional structured Cartesian mesh. Spatial differentiation is achieved through eighth-order finite differences along with tenth-order filters to damp any spurious oscillations in the solution. The differentiation and filtering require nine and eleven point centered stencils, respectively. Time advancement is achieved through a six-stage, fourth-order explicit Runge-Kutta (R-K) method [3].

S3D is parallelized using a three-dimensional domain decomposition and MPI communication. Each MPI process is in charge of a piece of the three-dimensional domain. All MPI processes have the same number of grid points and the same computational load. Inter-processor communication is only between nearest neighbors in a three-dimensional topology. The messages for a typical problem are approximately 80KB. A ghost-zone is constructed at the processor boundaries by non-blocking MPI sends and receives among the nearest neighbors in the three-dimensional processor topology. All-to-all communications are only required for monitoring and synchronization ahead of I/O.

S3D shows good parallel performance on several architectures and can make effective use of a large fraction of the Office of Science leadership computing platforms [4]. The rapid growth of computational capabilities has presented significant opportunities for DNS of turbulent combustion. Increases in computational power allow for increased grid size and/or a larger number of time-steps in the simulation. Both of these are favorable to the scientific goals by helping achieve a higher Reynolds numbers, a larger sample of turbulent structures for better statistical measures, and a longer simulation time to obtain a more complete temporal development of the turbulent flame. Increases in computational power also allow for the solution of a larger number of species equations, thereby allowing the simulation of fuels with higher chemical complexity. Recently, S3D has been used to perform several epic simulations of combustion problems ranging from premixed flames (200 million grid points, 18 variables) [5], non-premixed flames (500 million grid points, 16 variables) [6], to lifted jet flames (1 billion grid points, 14 variables) [7].

The National Center for Computational Sciences (NCCS) at the Oak Ridge National Laboratory (ORNL) is one of the primary provider of computational resources for the DNS of turbulent combustion through the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program [8]. The INCITE program has provided a multi-year allocation on the NCCS systems for performing DNS of turbulent combustion using S3D. The primary system at the NCCS is a Cray XT3/XT4 system named Jaguar. After an upgrade in February of 2007, it has 124 cabinets and 11508 compute nodes, of which 5212 are XT3 nodes and 6296 are XT4 nodes. Each compute node contains a 2.6GHz dual-core AMD Opteron processor. Prior to the upgrade the system consisted of only 5212 XT3 nodes. In this paper, we study the performance and scalability of the S3D code on Jaguar, in this hybrid configuration.

3 Benchmark problem description

Code performance and scaling are evaluated by simulating a pressure wave problem, where the propagation of a small amplitude pressure wave through the domain is computed for a short period of time. The test is conducted with detailed CO-H_2 chemistry consisting of 11 chemical species and mixture-averaged molecular transport model. Due to the detailed chemical model, the code solves for 11 species equations in addition to the five fluid dynamic variables. The 11 species chemical mechanism corresponds to that used in an earlier INCITE simulation [6], and several other benchmark studies [4]. More recently, larger chemical mechanisms are being used in production simulations. However, we are retaining the CO-H_2 chemistry for the benchmark studies to maintain continuity and also because the effect of chemical mechanism size on the simulation cost can be determined easily. The simulation's initial condition consists of a Gaussian temperature profile centered in the domain with periodic boundary conditions. When integrated in time, the initial temperature non-uniformity gives rise to pressure waves and spreading of the temperature profile. The problem size is kept at 50^3 grid points per MPI-thread. This size is representative of the number of grid points per MPI-thread in production simulations. The code performance is measured by the computational cost (in core-hours) per grid point per time step. A lower simulation cost will allow an increased grid size and/or a larger number of time-steps, both of which allow the simulation of higher Reynolds number regimes, more complete temporal development of the solution and larger statistical sample sets for better accuracy.

4 Parallel performance

In this section, the parallel performance of S3D is demonstrated on the Cray XT3/XT4 system at the NCCS. The results are shown in Figure 1. The batch queue system has the option of allocating only XT3 or XT4 nodes for a job. This feature was used in the benchmarking experiments to isolate the performance on XT3 and XT4 nodes, since the XT4 nodes have a higher memory (10.6 GB/s as opposed to 6.4 GB/s) and sustained network bandwidth (6 GB/s as opposed to 4 GB/s.) At large processor counts, the experiments made use of a mixture of both XT3 and XT4 nodes, which is labeled as XT7 in the figure. The performance of S3D on the system when it consisted of only the dual-core XT3 nodes (August 2006) is also shown for reference. It is noted here that this experiment is a weak parallel scaling test in that the number of grid points per MPI-thread is kept constant, as described in the earlier section. A perfect parallel scaling will manifest itself as a flat scaling curve implying that the cost of simulation does not increase when larger number of cores are utilized.

The increase in cost going from one to two cores, seen in all the cases, is not related to inter-node communication and therefore is discussed separately in the next section. Here, we limit ourselves to the scaling behavior seen at large core counts. First, it is noted that S3D achieved perfect scaling on the XT3 system prior to the XT3/XT4 hybrid upgrade, seen from the flat curve. The following observations are made with respect to the parallel performance of S3D shown in Figure 1.

In the experiments performed using the XT3 or XT4 nodes exclusively, the parallel scaling is flat until 2,048 cores, but shows a sudden jump and increase in cost when going from 2,048 to 4,096 and 8,192 cores. This was not observed on the pre-upgrade XT3 system. As a consequence, the cost of execution on 8,192 XT3 cores on the upgraded system is significant higher than it used to be. It is probably not a mere coincidence that the increase in cost happens at the same core count (2,048 to 4,096) on both the XT3 and XT4 portions of the system. The request to allocate either the XT3 or the XT4 nodes exclusively might be the cause for less efficient job placement across the nodes. This could be causing a longer communication time than usual.

4.1 MPI Communication Time

Production jobs usually require $>$10,000 cores and hence, will not and cannot request exclusive XT3 or XT4 nodes. Therefore, this anomaly may not be an important factor in a production run. However, finding the source of the loss of parallel efficiency might also help determine the reason for the higher cost of execution on 8,192 and 12,000 cores of the combined system when compared against the cost of execution on the old XT3 system. There is also a noticeable jump in cost when going from 16,000 cores to 20,000 cores. In effect, the cost of execution increases from 67μs per grid point per time step for a 2 core XT3 node experiment to 83μs per grid point per time step for a 22800 core experiment. This translates to, roughly 25% increase in cost of execution when S3D is scaled on the entire system.

The parallel execution overhead on the current XT system is not high enough to desist the application from utilizing a major fraction of the resource. Such less than perfect scaling has been observed on other systems before. However, the scaling degradation noticed here is still intriguing considering that S3D used to scale perfectly on the XT3 system. The following section presents further analysis of the source of overheads and possible approaches for trouble-shooting.

The parallel performance of S3D is further analyzed by profiling the cost of MPI calls in the above mentioned experiments using the FPMPI library [9]. Figure 2 shows the average cost of the different MPI calls obtained from experiments performed on 2,048 and 4,096 XT3 cores. The total execution time was 878 and 1,048 seconds on 2,048 and 4,096 XT3 cores, respectively. It is seen that the increase in execution time is mainly due to the longer time spent in MPI_Wait on 4,096 cores, compared to 2,048 cores. A similar increase in MPI_Wait time was also observed on the XT4, when comparing 2,048 with 4,096 cores, and the combined system when comparing 16,384 with 20,000 cores.

As mentioned in the introduction, S3D uses non-blocking sends and receives to communicate ghost-zone information between the nearest neighbors in

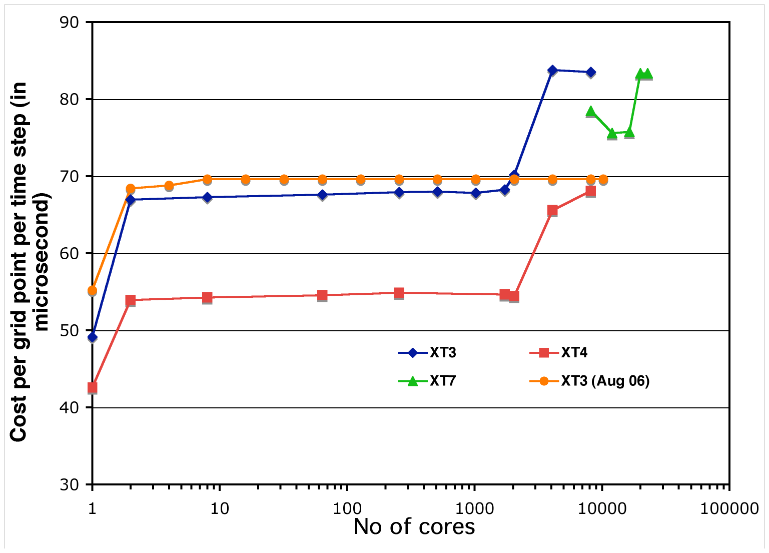

Figure 1: S3D parallel scaling on the Cray XT3/XT4 at NCCS

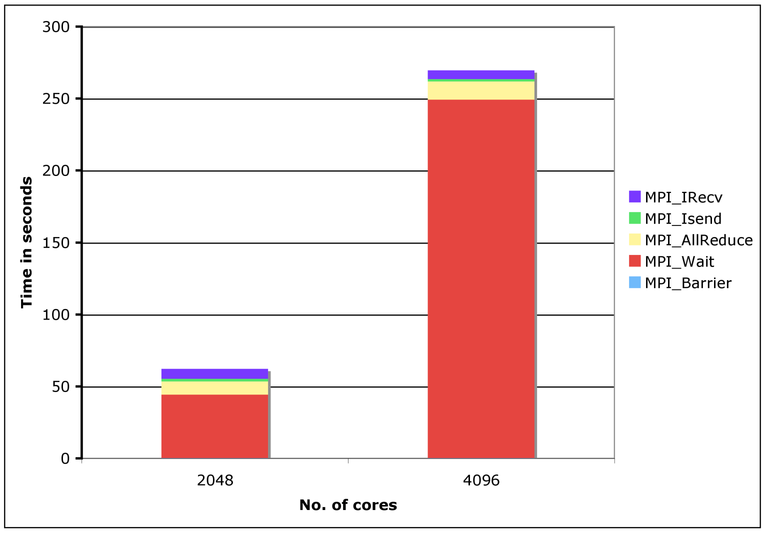

Figure 2: Average time spent in MPI calls on 2048 and 4096 XT3 cores.

```
! get ghost cells from neighbor on (-x) side
call MPI_IRecv(...)
! send ghost cells to neighbor on (-x) side
call MPI_ISend(...)

! get ghost cells from neighbor on (+x) side
call MPI_IRecv(...)
! send ghost cells to neighbor on (+x) side
call MPI_ISend(...)

!Compute derivatives at the interior points
[computation]

!Wait to receive ghost cells from the (-x) side
call MPI_Wait(...)

!Compute derivatives at the (-x) boundary
[more computation]

!Wait to receive ghost cells from the (+x) side
call MPI_Wait(...)

!Compute derivatives at the (+x) boundary
[more computation]

!Wait to complete any pending sends before exiting the routine
call MPI_Wait(...)
```

Figure 3: Pseudo-code of derivative operation in the x-direction, showing the context of MPI_Waits and non-blocking communication calls.

Problem Size	MPI mode	XT3		XT4	
		wall time seconds	cost μseconds per grid point per time step	wall time seconds	cost μseconds per grid point per time step
50×50×50	yod -np 1 -SN	617	49 (98)	532	43 (86)
100×50×50	yod -np 1 -SN	1206	48(96)	1041	42 (84)
100×50×50	yod -np 2 -SN	614	49(98)	532	43 (86)
100×50×50	yod -np 2 -VN	838	67	674	54

Table 1: Comparison of single core and multi-core performance of S3D on XT3 and XT4. Numbers in brackets show the cost of execution when time is charged for both cores on a socket irrespective of one or both cores being used.

the 3D topology. The non-blocking communication calls, MPI_Isend and MPI_IRecv, and MPI_Wait are interleaved with computation as shown in Figure 3. The amount of computation in between the communication calls and the MPI_Wait calls determines the amount of time actually spent waiting. Therefore, the observed MPI_Wait behavior cannot be deduced from standard MPI benchmark data alone and requires experimentation with S3D code to reproduce the exact results.

It is suspected that the non-perfect scaling might be related to some pathological placement issue. This could be tested with dedicated access to the entire machine and then using specific sockets with regard to placement. In order to pursue this strategy, we need to fully understand the 3D torus layout and the interprocess wiring. The latter is not simple as cabinets that are physically next to each other are not necessarily wired together.

Another suggestion was that the packet aging setting could be adjusted to alleviate the problem. In particular, the priority of packets could be raised based on the time that the packet is on the interconnect. This supposedly will help improve network performance with jobs that physically span a large area of the machine.

5 Multi-core performance

In Figure 1, it was found that the cost of execution on two cores was higher than on a single core on the XT3/XT4 system. Here we systematically compare the performance of S3D on a single core and dual cores of XT3 and XT4. The results are shown in Table 1. It is noted here that the "-SN" option to yod executes the code using one core per socket only, while the "-VN" option makes use of both cores. It is seen that the cost of execution is higher in the "VN" mode than in the "SN" mode. "VN" mode is only more expensive when comparing the normalized cost metric (time per grid point per time step). However, in terms of absolute time to solution, for a fixed problem size (100$\times$50$\times$50) executing on both cores gives a faster turn-around time than when executing on one core only.

Profiles of the benchmark problem using the CrayPat tool [10], Tau [11] and HPC Toolkit [12] show that the portions of the code that operate on large five-dimensional arrays increased in cost when executing in dual-core mode. Therefore, the increase in cost when executing on dual cores can be attributed to the contention for memory bandwidth between the two MPI threads.

Also note that the difference in performance when comparing runs on the XT3 and XT4 can be attributed to the contention for memory bandwidth. The XT4 nodes provide a much higher memory bandwidth (as mentioned in section 4); thus, as seen in Figure 1, the performance of S3D on the XT4 shows to be approximately 20% faster than on the XT3.

6 Conclusions

S3D is a massively parallel DNS solver that solves the full compressible Navier-Stokes, total energy, species and mass continuity equations coupled with detailed chemistry. It is based on a high-order accurate, non-dissipative numerical scheme. The governing equations are solved on a conventional three-dimensional structured Cartesian mesh. S3D shows good parallel performance on several architectures and can make effective use of a large fraction of the Office of Science leadership computing platforms [4].

Large-scale S3D runs in August 2006 on ORNL's XT3 demonstrated perfect scaling. However, after an upgrade to combine 68 XT4 cabinets to the previously existing 56 XT3 cabinets, S3D no longer exhibited perfect scaling on the XT platform. Some as-yet-determined scaling problem presents itself when using 4K or more cores. The hybrid XT system is fairly recent at the NCCS and was made available to users only in the first week of April this year. The performance results presented here were collected during the first month on the system. We expect further improvements in performance as more understanding of the system is gained and possible causes for performance degradation are identified. It is suspected that the scaling issue might be related to some pathological placement issue. Adjusting a system level parameter with regard to packet aging was also suggested as a fix, but has yet to be tested. These potential explanations will be investigated in the near future.

7 Acknowledgments

This research was sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, US Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

This research used resources of the National Center for Computational Sciences (NCCS) at Oak Ridge National Laboratory (ORNL), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The work at SNL was supported by the Division of Chemical Sciences, Geosciences and Biosciences, the Office of Basic Energy Sciences (BES), the U.S. Department of Energy (DOE) and also by the U.S. DOE, BES, SciDAC Computational Chemistry program.

8 About the Authors

Ramanan Sankaran is a computational scientist in the National Center for Computational Sciences at Oak Ridge National Laboratory. Ramanan received a M.S. in mechanical engineering from University of Tennessee and a PhD in mechanical engineering from University of Michigan. He can be reached at Oak Ridge National Laboratory, PO. Box 2008 MS6008, Oak Ridge, TN 37831-6008, Email: sankaranr@ornl.gov

Mark R. Fahey is a senior Scientific Application Analyst in the Center for Computational Sciences at Oak Ridge National Laboratory. He is the past CUG X1-Users SIG chair and current CUG Treasurer. Mark has a PhD in mathematics from the University of Kentucky. He can be reached at Oak Ridge National Laboratory, P.O. Box 2008 MS6008, Oak Ridge, TN 37831-6008, E-Mail: faheymr@ornl.gov.

Jacqueline H. Chen is a distinguished member of technical staff in the Combustion Research Facility at Sandia National Laboratories (Livermore). Jackie received a M.S. in mechanical engineering from University of California and a PhD in mechanical engineering from Stanford University. She can be reached at Sandia National Laboratories, PO Box 969 MS9051, Livermore, CA 94551-0969, Email: jhchen@sandia.gov

References

1. Evatt R. Hawkes, Ramanan Sankaran, James C. Sutherland, and Jacqueline H. Chen. Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models. *Journal of Physics: Conference Series*, 16:65–79, 2005.

2. James C. Sutherland. *Evaluation of mixing and reaction models for large-eddy simulation of non-premixed combustion using direct numerical simulation*. PhD thesis, University of Utah, 2004.

3. Christopher A. Kennedy, Mark H. Carpenter, and R. Michael Lewis. Low-storage, explicit runge-kutta schemes for the compressible navier-stokes equations. *Applied Numerical Mathematics*, 35(3):177–219, 2000.

4. Ramanan Sankaran, Evatt R. Hawkes, Jacqueline H. Chen, Tianfeng Lu, and Chung K. Law. Direct numerical simulations of turbulent lean premixed combustion. *Journal of Physics: Conference Series*, 46:38–42, 2006.

5. Ramanan Sankaran, Evatt R. Hawkes, Jacqueline H. Chen, Tianfeng Lu, and Chung K. Law. Structure of a spatially developing turbulent lean methane-air bunsen flame. *Proceedings of the Combustion Institute*, 31(1):1291–1298, 2007.

6. Evatt R. Hawkes, Ramanan Sankaran, James C. Sutherland, and Jacqueline H. Chen. Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal co/h2 kinetics. *Proceedings of the Combustion Institute*, 31(1):1633–1640, 2007.

7. Chun S. Yoo, Jacqueline H. Chen, and R. Sankaran. Direct numerical simulation of a turbulent lifted hydrogen/air jet flame in an autoignitive heated coflow. In *2nd ECCOMAS thematic conference on computational combustion*, 2007.

8. Innovative and Novel Computational Impact on Theory and Experiment (INCITE). `http://hpc.science.doe.gov/`.

9. William Gropp and Kristopher Buschelman. Fast profiling library for mpi. `htttp://www-unix.mcs.anl.gov/fpmpi/WWW/`, 2007.

10. Cray Inc. *Using Cray performance analysis tools*,

2006. S-2376-31.

11. Tuning and analysis utilities. `http://www.cs.uoregon.edu/research/tau/home.php`.
12. HPCToolkit. `http://www.hipersoft.rice.edu/hpctoolkit/`.