

Performance Analysis and Modeling of the SciDAC GTC Code on Four Large-scale Computer Systems

Xingfu Wu and Valerie Taylor
Department of Computer Science, Texas A&M University
Email: {wuxf, taylor}@cs.tamu.edu

Abstract

The Gyrokinetic Toroidal code (GTC) (version 2) is a 3D particle-in-cell application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic fusion. In this paper, we analyze the performance of the GTC code on four large-scale systems: ORNL Jaguar, UNC RENCIBL BlueGene/L, SDSC DataStar, and NERSC Seaborg. We discuss the scalability of the GTC code with processor and problem scaling, use processor partitioning to investigate the performance impacts of the application components, and identify possible performance optimizations for further work. Then we use Prophesy to generate performance models for GTC on each platform so that we can use these models to predict the performance on larger number of processors.

1. GTC code

The Gyrokinetic Toroidal code (GTC) (version 2) [LE02] is a 3D particle-in-cell application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic fusion. GTC is currently the flagship SciDAC fusion microturbulence code. Figure 1 shows a visualization of potential contours of microturbulence for a magnetically confined plasma using GTC. The finger-like perturbations (streamers) stretch along the weak field side of the poloidal plane as they follow the magnetic field lines around the torus [SD06].

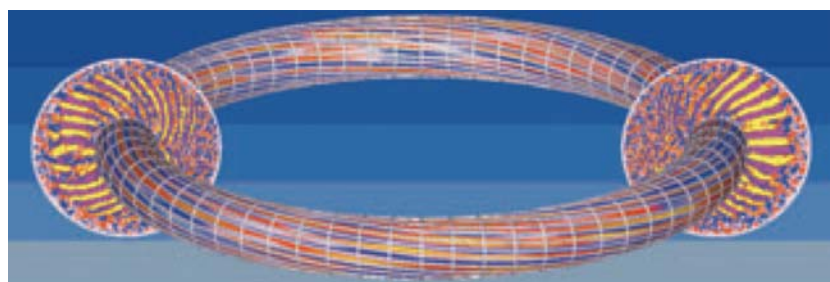
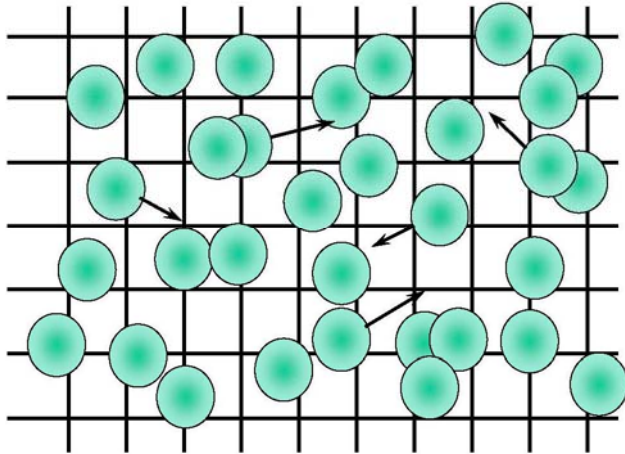


Figure 1. Potential contours of microturbulence for a magnetically confined plasma [SD06].



The PIC Steps

- “**SCATTER**”, or deposit, charges on the grid (nearest neighbors)
- Solve Poisson equation
- “**GATHER**” forces on each particle from potential
- Move particles (**PUSH**)
- Repeat...

Figure 2. Particles in cell (PIC) steps [ES05]

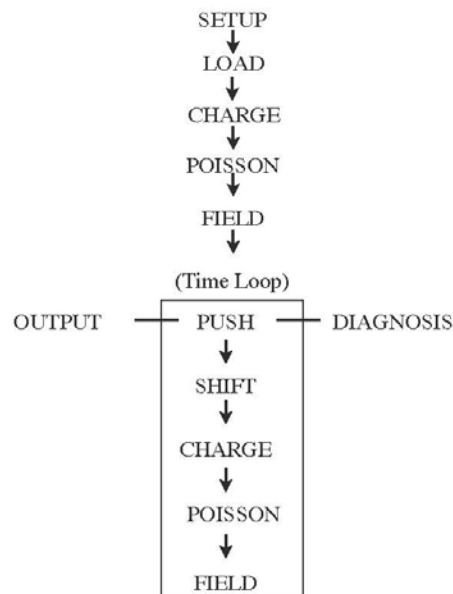


Figure 3. Program control flow of GTC

2. Execution Platforms

In this section, we describe the specifications of four large-scale systems: Cray XT3/4 Jaguar [ORNL] at ORNL (Oak Ridge National Laboratory), IBM POWER3 SMP cluster Seaborg [NERS] at NERSC (National Energy Research Scientific Computing center), IBM BlueGene/L system [RENC] in RENCi at the University of North Carolina, and IBM POWER4 SMP cluster DataStar [SDSC] at SDSC (San Diego Supercomputing Center). Table 1 indicates the configurations of the four large-scale systems.

Jaguar is a Cray XT containing a combination of XT3 and XT4 systems, and has a total of 11,706 processor nodes. Of those, 11,508 are configured as compute nodes and the remainder

provides I/O and login services. Each of the compute nodes contains 2.6 GHz dual-core AMD Opteron processors and 4 GB of memory. The service nodes consist of a 2.6 GHz dual-core AMD Opteron processor with 8 GB of memory. Each node is connected to a Cray Seastar router through Hypertransport, and the Seastars are all interconnected in a 3D-torus topology. The system provides an aggregate peak performance of over 119 Teraflops with approximately 46 terabytes of aggregate memory.

UNC RENCIBlueGene/L is designed to achieve high performance for low cost and with low power consumption. The BG/L has the following configuration: *Compute* - 1024 dual 700 MHz PowerPC 440 nodes with 1GB memory per node; *Storage* - 11 TB of clusterwide disk storage; *Network* - IBM BlueGene Torus, Global Tree network (2.1 GB/s) and Global Interrupt.

Table 1. Specification of four large-scale clusters

| System Name | RENCIBlueGene/L | DataStar | Seaborg | Jaguar |
|-----------------|------------------------|----------------------|------------------|-----------------------|
| Number of Nodes | 1024 | 272 | 380 | 11508 |
| CPUs per Node | 2 | 8 | 16 | 2 |
| CPU type | 700 MHz PowerPC 440 | 1.5-1.7GHz POWER4 | 375MHz POWER3 | 2.6GHz AMD Opteron |
| Memory per Node | 1GB | 16-32GB | 16-64GB | 4GB |
| Network | Torus | Federation | Colony | Torus |
| OS | Linux | AIX | AIX | Linux/Catamount |

The NERSC Seaborg machine is a distributed memory computer with 6,080 processors available to run scientific computing applications. The processors are distributed among 380 compute nodes with 16 processors per SMP node. Processors on each node have a shared memory pool of between 16 and 64 GB (312 nodes with 16GB, 64 nodes with 32GB, and 4 nodes with 64GB). The compute nodes are connected by the IBM Colony switch. The use of 16-way nodes is exclusive: only one user is allowed to use a node at any given time, regardless of the number of CPUs one needs on that node.

The DataStar is SDSC's largest IBM terascale machine. It has 176 (8-way) P655 compute nodes with 1.5GHz POWER4 and 16 GB memory per node, and 96 (8-way) compute nodes with 1.7GHz POWER4 and 32 GB memory per node. The nodes are connected by the IBM Federation switch. The use of 8-way nodes is exclusive: only one user is allowed to use a node at any given time, regardless of the number of CPUs one needs on that node.

3. Performance Analysis

In this section, we report on performance analysis of the GTC code on the four large-scale systems, and we especially discuss the scalability of the GTC code on up to 2048 processors. We used Prophecy [WT01, TW02] to instrument the code to collect performance data for our analysis.

ORNL Jaguar has 2 processors per node, UNC RENCIBlueGene/L has 2 processors per node, SDSC DataStar has 8 processors per node, and NERSC Seaborg has 16 processors per node. Note that the total execution time for the application means the time it takes from the beginning of the application program to the end, which includes I/O time.

3.1 Datasets for Problem and Processor Scaling

The problem sizes for the GTC code are listed in Table 2, where micell is the number of ions per grid cell, mecell is the number of electrons per grid cell, mzetamax is the total number of toroidal grid points, and npartdom is the number of particle domain partitions per toroidal domain.

Table 2. Datasets with scaling the number of processors

| #Procs | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|----------|-----|-----|-----|-----|-----|-----|-----|------|------|
| micell | 100 | 100 | 100 | 100 | 200 | 400 | 800 | 1600 | 3200 |
| mecell | 100 | 100 | 100 | 100 | 200 | 400 | 800 | 1600 | 3200 |
| mzetamax | 8 | 16 | 32 | 64 | 64 | 64 | 64 | 64 | 64 |
| npartdom | 1 | 1 | 1 | 1 | 2 | 4 | 8 | 16 | 32 |

For the problem datasets, with increasing numbers of processors, the workload per processor remains the same, i.e. 100 ions per grid cell and 100 electrons per grid cell.

3.2 Weak Scaling

In this section, we use the execution time on 8 processors as a baseline, and define that the relative slowdown is the execution time on p processors divided by the execution time on 8 processors. Given the fixed workload per processor, the relative slowdown quantifies how much the application performance degrades with increasing the number of processors.

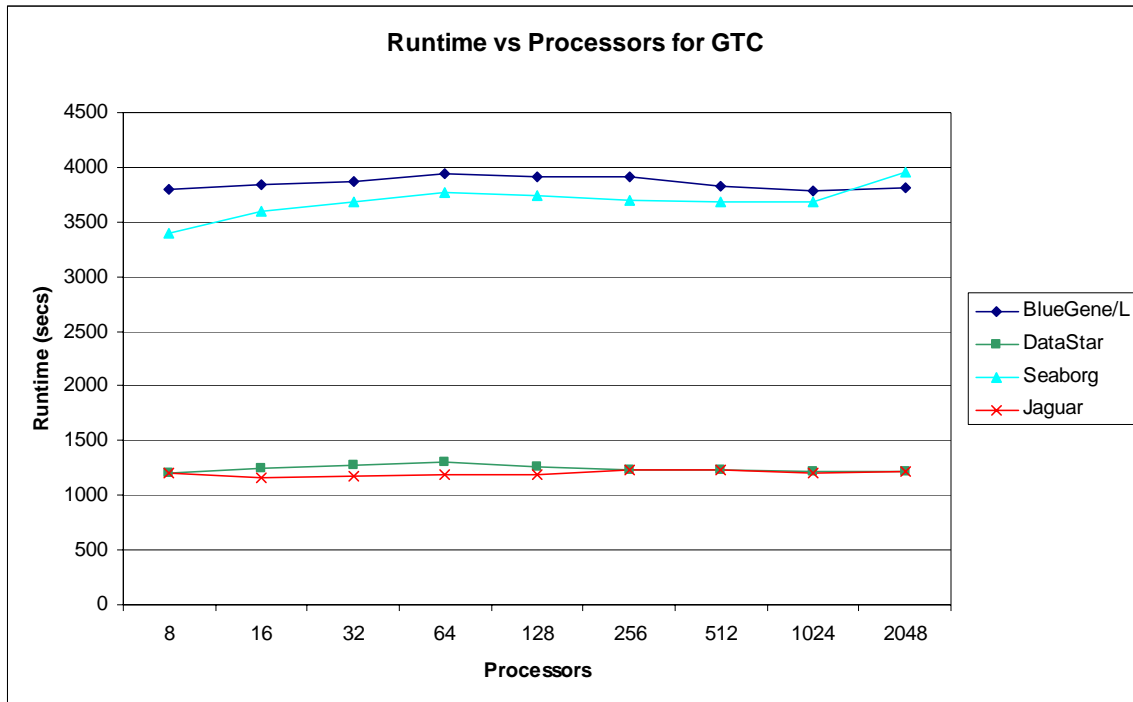


Figure 4. Performance comparisons for GTC code on four systems

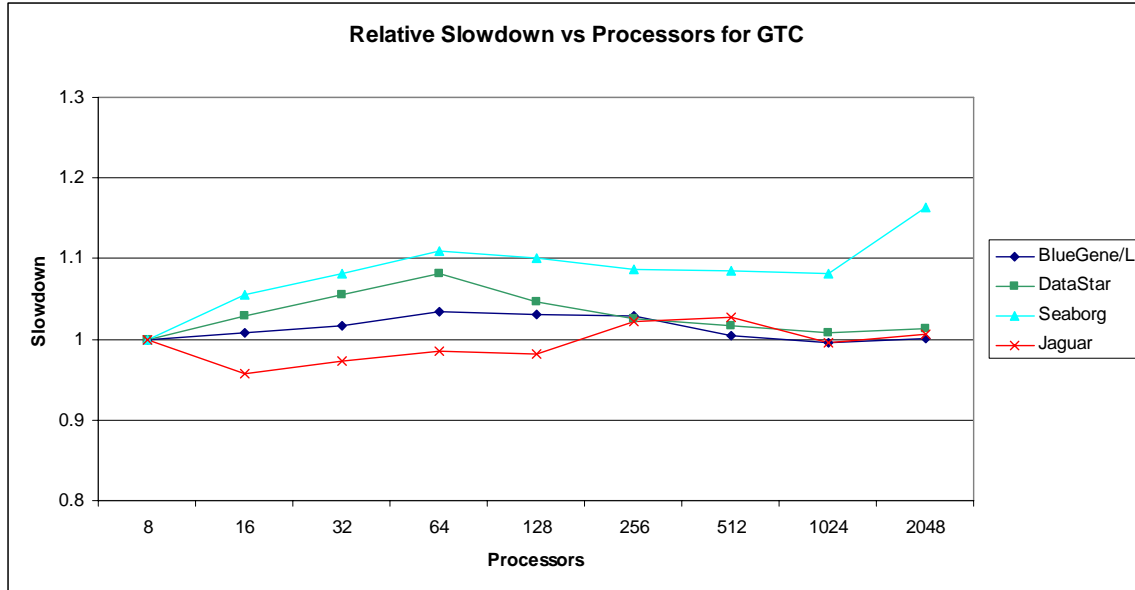


Figure 5. Relative slowdowns for GTC code on four clusters

Overall, the GTC code scales the best scalability on Jaguar with weak scaling because its slowdown is less than 1.03 for all cases shown in Figures 4 and 5. It is interesting to see that the execution times on BlueGene/L are bigger than that on Seaborg except that on 2048 processors, although the processor speed (700MHz) on BlueGene/L is much faster than (375 MHz) on Seaborg. For Seaborg, the slowdown increases significantly from 1024 processors to 2048 processors because of the slow communication system on Seaborg. For Jaguar, the relative slowdown is less than 1 on up to 128 processors because of the combination of XT3 and XT4.

3.3 Application Sensitivity Analysis Using Processor Partitioning

We define processor partitioning as how many processors per node to be used for an application execution. A processor partitioning scheme $N \times M$ means N nodes with M processors per node. We can use processor partitioning to quantify the time difference among different processor partitioning schemes (PPS) and to investigate how the application and its components are sensitive to different communication patterns and memory access patterns for different PPS. For example, given the PPS $N \times M$, when $M=1$, it means that the communication pattern is internode only, and only one processor on each node uses the whole memory. When $M > 1$, it means that there are internode and intranode communications, and M processors on each node competes the whole memory.

In this section, we investigate how processor partitioning impacts the performance of the application and its components. We use GTC on 64 processors on the four systems as an example to conduct our experiments. Runtime means the total application execution time which includes any I/O time.

Because the GTC code cannot be executed on UNC RENCIBLueGene/L with 2 processors per node (in VN mode), we cannot show any data for processor partitioning on that machine.

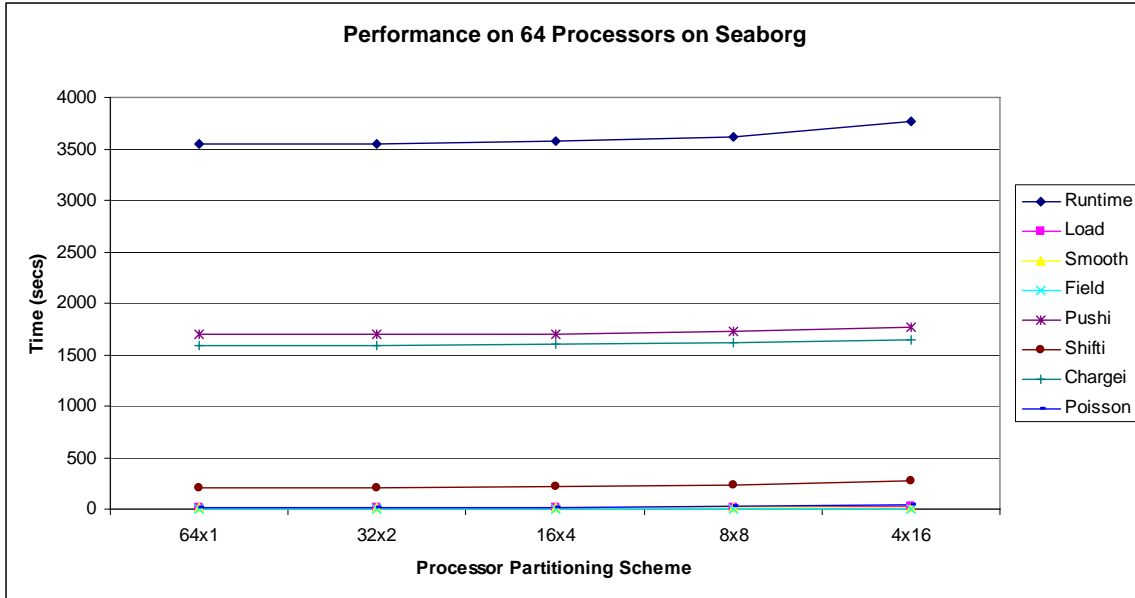


Figure 6. Performance comparison for different PPS on 64 processors on NERSC Seaborg

Table 3. Performance for different PPS on Seaborg

| PPS | 64x1 | 32x2 | 16x4 | 8x8 | 4x16 |
|----------------|----------|----------|----------|----------|----------|
| Runtime | 3545.084 | 3549.409 | 3574.918 | 3617.465 | 3773.425 |
| Load | 13.1776 | 13.45061 | 13.64874 | 14.23223 | 29.77986 |
| Smooth | 7.456567 | 7.815145 | 8.94311 | 8.291781 | 8.208343 |
| Field | 3.616935 | 4.078701 | 4.690424 | 3.996767 | 4.232996 |
| Pushi | 1700.804 | 1702.468 | 1705.315 | 1725.157 | 1771.744 |
| Shifti | 202.9492 | 204.6154 | 217.4198 | 227.7181 | 269.4511 |
| Chargei | 1594.857 | 1594.425 | 1602.154 | 1610.215 | 1647.227 |
| Poisson | 17.89159 | 18.03115 | 18.36418 | 23.33593 | 37.49189 |

Figure 6 and Table 3 show the performance comparison for different processor partitioning schemes on 64 processors on NERSC Seaborg. The time difference between the scheme 64x1 and 4x16 is 6.44%, and the time difference for Shifti is 32.77%. But the time difference is 125.99% for Load, and 109.55% for Poisson.

Table 4. Performance for different PPS on DataStar

| PPS | 64x1 | 32x2 | 16x4 | 8x8 |
|----------------|----------|----------|----------|----------|
| Runtime | 1203.32 | 1253.83 | 1266.49 | 1305.74 |
| Load | 4.461061 | 4.676765 | 4.846003 | 5.176585 |
| Smooth | 2.222829 | 2.33928 | 2.352441 | 2.609333 |
| Field | 1.215625 | 1.251001 | 1.326334 | 1.687492 |
| Pushi | 617.5766 | 636.7238 | 638.1365 | 659.9568 |
| Shifti | 48.42188 | 63.59431 | 71.81874 | 76.36723 |
| Chargei | 521.0852 | 536.8003 | 539.3956 | 549.691 |
| Poisson | 6.9856 | 7.061641 | 7.322136 | 8.891254 |

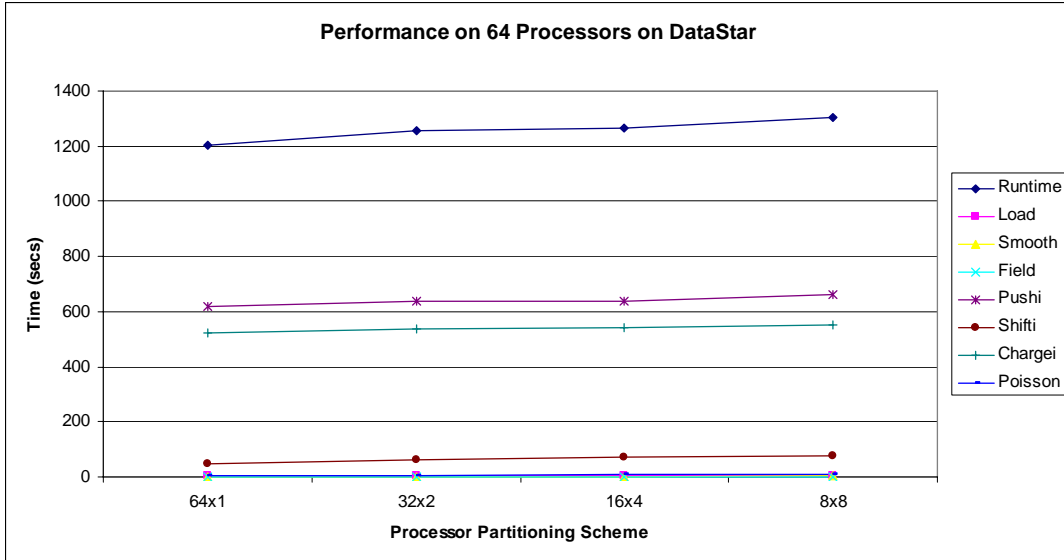


Figure 7. Performance comparison for different PPS on 64 processors on SDSC DataStar

Figure 7 and Table 4 indicate the performance comparison for different processor partitioning schemes on 64 processors on SDSC DataStar. The time difference between the scheme 64x1 and 8x8 is 8.51%, and the time difference for Shifti is 57.71%. But the time difference is only 16.04% for Load, and 27.28% for Poisson.

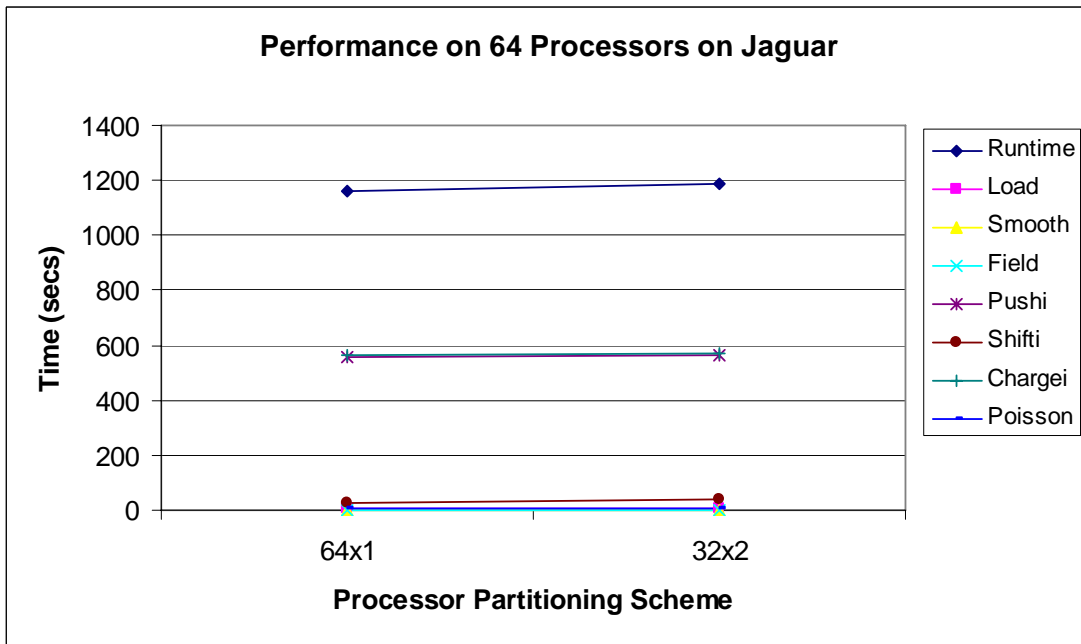


Figure 8. Performance comparison for different PPS on 64 processors on ORNL Jaguar

Table 5. Performance for different PPS on Jaguar

| PPS | 64x1 | 32x2 |
|---------|--------|---------|
| Runtime | 1161.5 | 1188.92 |
| Load | 5.503 | 5.74 |
| Smooth | 2.366 | 2.56 |

| | | |
|----------------|--------|-------|
| Field | 0.9394 | 1.169 |
| Pushi | 556.9 | 562.5 |
| Shifti | 26.85 | 39.39 |
| Chargei | 562 | 570.2 |
| Poisson | 6.285 | 6.779 |

Figure 8 and Table 5 indicate the performance comparison for different processor partitioning schemes on 64 processors on ORNL Jaguar. The time difference between the scheme 64x1 and 32x2 is 2.36%, the time difference for Shifti is 46.7%, 4.31% for Load, and 7.86% for Poisson.

4. Performance Modeling Using the Prophecy System

The Prophecy system [TW02] provides the PAIDE automatic performance instrumentation and data entry tool [WT01]. PAIDE provides the automatic instrumentation of codes at the level of basic blocks, procedures, or functions. The default mode consists of instrumenting the entire code at the level of basic loops and procedures. A user can specify that the code be instrumented at a finer granularity than that of loops or identify the particular events to be instrumented. The resultant performance data is automatically uploaded into the Prophecy database at the end of application execution by PAIDE, and is used to produce performance models and predict application performance via Prophecy web-based interfaces [WT06].

For the GTC code, we used Prophecy to illustrate the modeling process as follows. Figures 9 through 12 illustrate the performance data entry model and predicted performance on 4096 processors based on the performance data on 8 to 2048 processors.

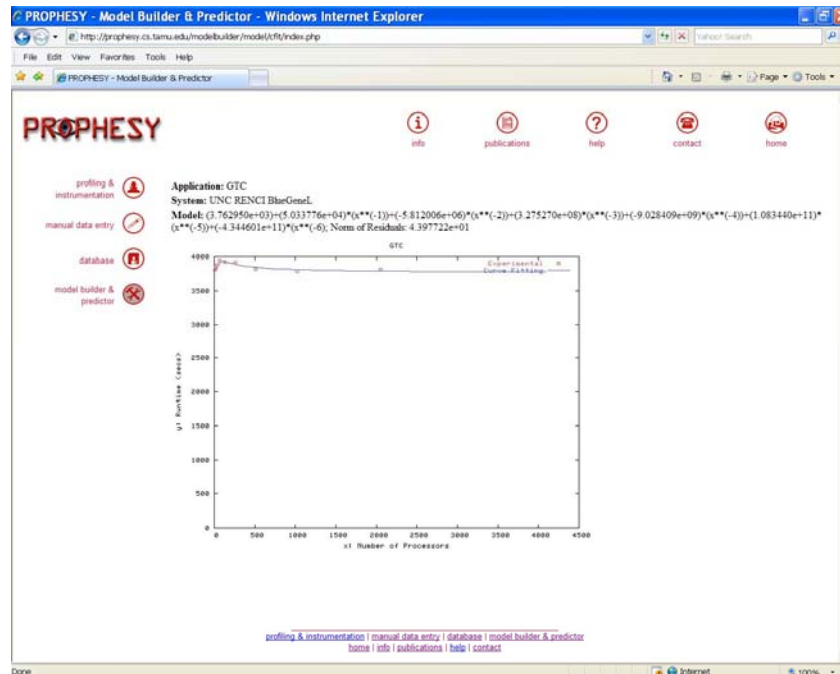


Figure 9. Performance model for GTC on RENC1 BlueGene/L

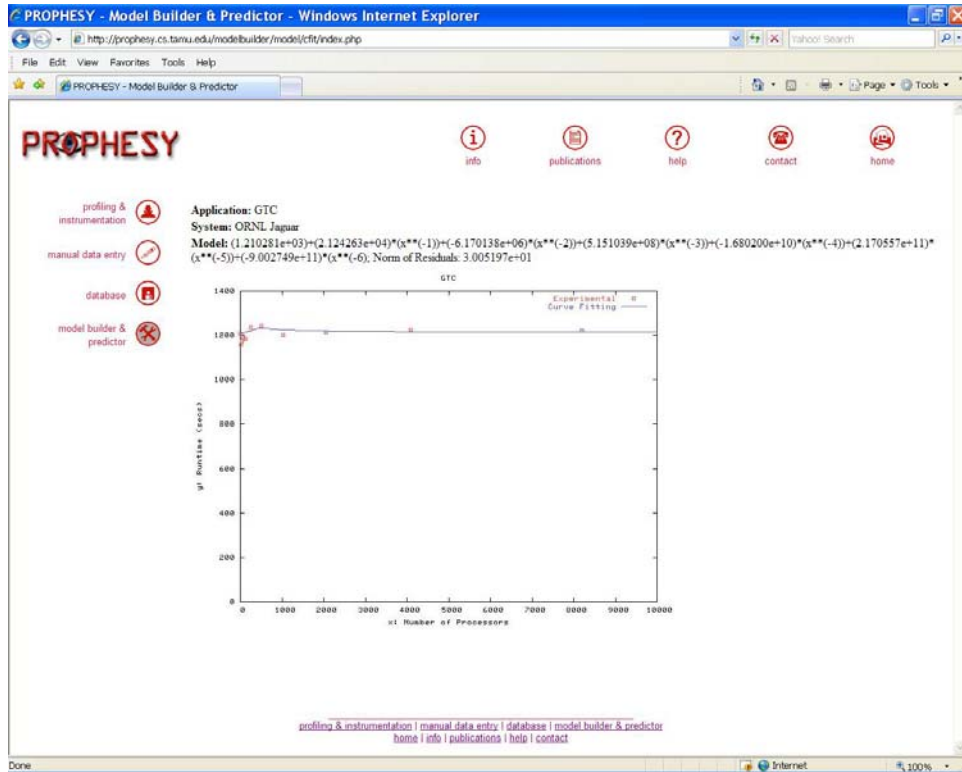


Figure 10. Using Prophecy to generate a performance model for GTC, and to predict the performance of GTC on 10,000 processors based on the performance model.

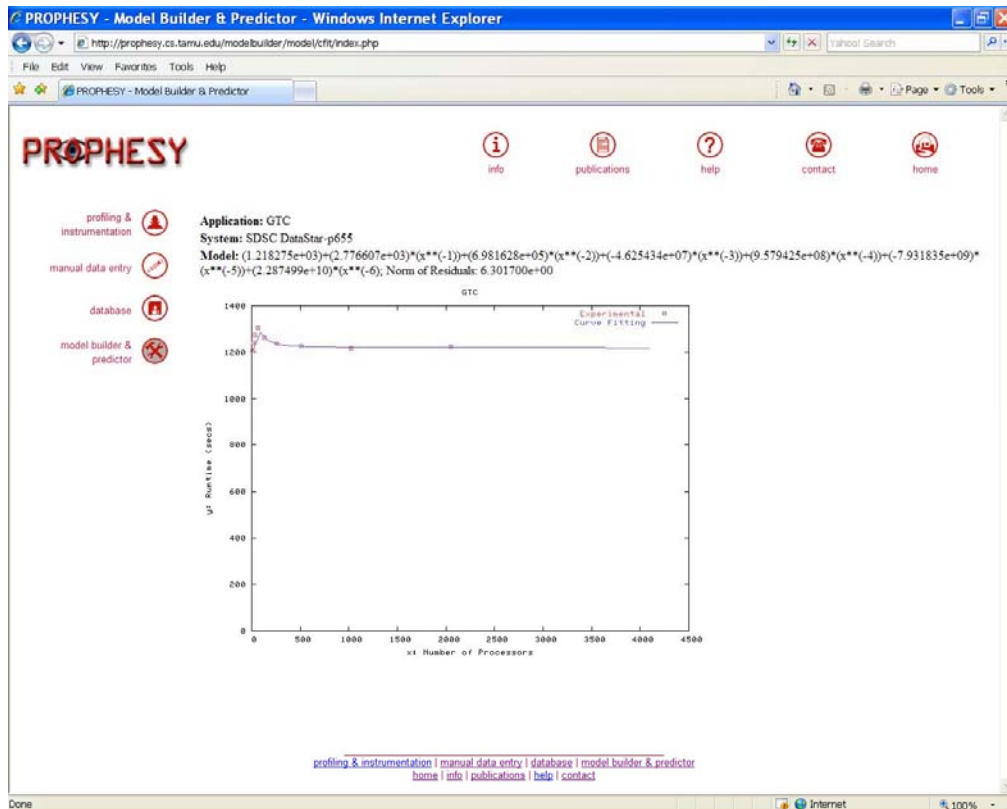


Figure 11. Performance model for GTC on SDSC DataStar

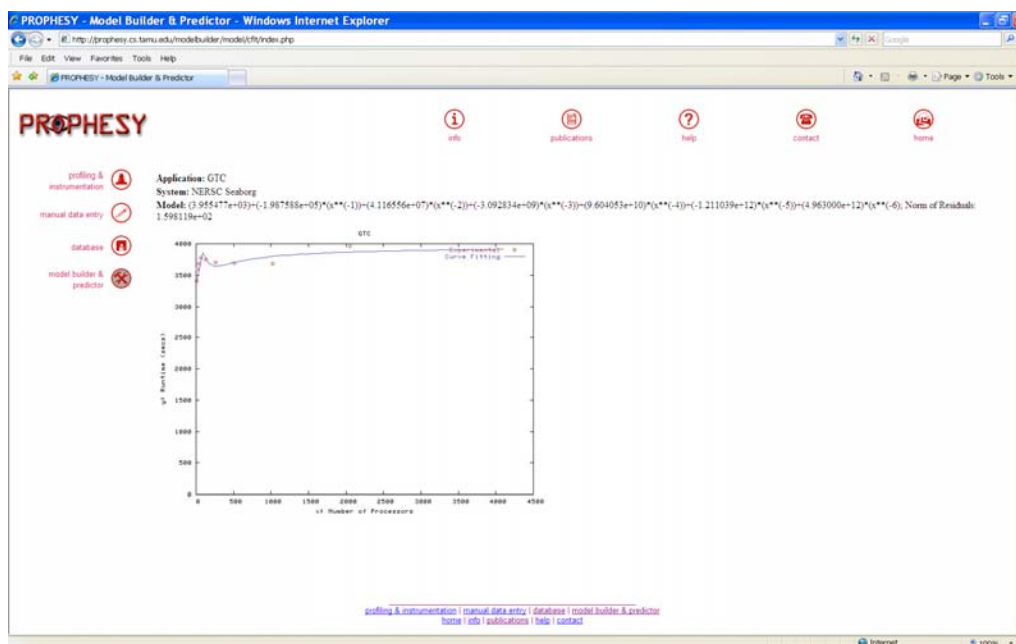


Figure 12. Performance model for GTC on NERSC Seaborg

Figures 9-11 indicate more accurate performance models on BlueGene/L, Jaguar and Datastar, where the norm of residuals is less than 2% of the total execution time on 8 processors. But the performance model in Figure 12 is not accurate, where the norm of residuals is 4.7% of the total execution time on 8 processors.

Acknowledgements

We would like to thank Stephane Ethier from Princeton Plasma Physics Laboratory for providing initial GTC code and datasets, and thank Shirley Moore from University of Tennessee for useful comments about this document.

Summary

We have carried out the following tasks:

- Compared the performance of the GTC code on four large-scale systems
- Discussed which platforms perform the best for the GTC code, in other words, which platforms are most suitable for the SciDAC GTC application
- Investigated how processor partitioning impacts the performance of GTC
- Modeled the performance of GTC and predicted the performance on a larger number of processors

References

[ES05] S. Ethier, First Experience on BlueGene/L, *BlueGene Applications Workshop*, ANL, April 27-28, 2005. http://www.bgl.mcs.anl.gov/Papers/GTC_BGL_20050520.pdf.

- [LE02] Z. Lin, S. Ethier, T.S. Hahm, and W.M. Tang, Size Scaling of Turbulent Transport in Magnetically Confined Plasmas, *Phys. Rev. Lett.* 88, 2002.
- [RENC] UNC RENC BlueGene/L, <http://www.renci.org/about/computing.php>
- [NERS] NERSC Seaborg, <http://www.nersc.gov/nusers/resources/SP/>.
- [ORNL]ORNL Jaguar, <http://info.nccs.gov/resources/jaguar/>.
- [SDSC] SDSC DataStar, http://www.sdsc.edu/user_services/datastar/.
- [WT01] Xingfu Wu, Valerie Taylor and Rick Stevens, Design and Implementation of Prophecy Automatic Instrumentation and Data Entry System, *Proc. of the 13th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2001)*, Anaheim, CA, August 2001.
- [TW02] Valerie Taylor, Xingfu Wu, and Rick Stevens, Prophecy: An Infrastructure for Performance Analysis and Modeling of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, Volume 30, Issue 4, March 2003.
- [WT06] Xingfu Wu, Valerie Taylor, and Joseph Paris, A Web-based Prophecy Automated Performance Modeling System, *the IASTED International Conference on Web Technologies, Applications and Services (WTAS2006)*, July 17-19, 2006, Calgary, Canada.
- [SD06] Scientific Discovery, A progress report on the US DOE SciDAC program, 2006.