

Performance Analysis, Modeling and Prediction of a Parallel Multiblock Lattice Boltzmann Application Using Prophecy System

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

Dazhi Yu and Jacques Richard
Department of Aerospace Engineering
Texas A&M University, College Station, TX
Email: {dzyu, richard}@aero.tamu.edu

Abstract

Recently, the Lattice Boltzmann method is widely used in simulating fluid flows. In this paper, we present the performance analysis, modeling and prediction of a parallel multiblock Lattice Boltzmann application on up to 512 processors on three SMP clusters: two IBM SP systems at San Diego Supercomputing Center (DataStar – p655 and p690) and one IBM SP system at the DOE National Energy Research Scientific Computing Center (Seaborg) using the Prophecy system. By characterizing the performance of the Lattice Boltzmann application as the problem size and the number of processors increase, we can identify and eliminate performance bottlenecks, and predict the application performance. The experimental results indicate that the application with large problem sizes scales well across these three clusters, and performance models using the coupling method are accurate with less than 4.8% average relative prediction error.

1. Introduction

The Lattice Boltzmann method (LBM) is a non-conventional method recently widely used in simulating fluid flows; this method is an extension of the lattice gas algorithms [2]. In contrast to conventional methods in fluid dynamics, which are based on the discretization of macroscopic differential equations, the LBM has the ability to deal efficiently with complex geometries and topologies. It is computationally intensive, and requires significant computing power for large-scale applications. Therefore, it is necessary to use large-scale clusters to solve this class of applications.

In a parallel implementation of the LBM, Yu et al. [13] developed a multiblock LBM method to satisfy the requirement of resolutions in varied regions in the computational domain, and extended to 3D simulations using a parallel multiblock Lattice Boltzmann (PMLB) method [14]. Pohl et al [4] applied a parallel implementation of the LBM to three applications: nanotechnology, turbulence, and metal foams. They used the flops metric to evaluate the performance of these applications on the Hitachi SR8000-F1, the SGI Altix and the NEC SX6. In contrast, our work is focused on scalability, performance modeling and prediction of the PMLB.

In this paper, we use the Prophecy system [8] to investigate the performance of the parallel implementation of the PMLB on three SMP (Symmetric MultiProcessors) clusters: San Diego Supercomputing Center (SDSC) DataStar p655 (P655) and p690 (P690) [6], and DOE National Energy Research Scientific Computing Center (NERSC) Seaborg (Seaborg) [3], and model and predict the performance to aid in estimating the execution time for the PMLB application with larger problem size and number of processors. Experimental results show that the PMLB application with large problem sizes scales well, and the prediction model using the coupling method is accurate with less than 4.8% average relative prediction error.

The remainder of this paper is organized as follows. Section 2 provides an overview of the Prophecy system. Section 3 discusses a parallel multiblock Lattice Boltzmann application and describes the control flow of the application. Section 4 describes three SMP clusters used: P655, P690, and Seaborg. Section 5 addresses the scalability of the application. Section 6 presents the performance model used for application performance predictions. Section 7 summarizes the paper.

2. Prophecy System

The Prophecy system [8], shown in Figure 1, consists of three major components: data collection, data analysis, and three centralized databases. The data collection and database components deal with the retrieval and storage of application and performance information [10]. The data analysis component [7, 11] allows the user to produce an analytical performance model based on data from the performance database [9], model templates from the template database, and system characteristics from the systems database. The interface to Prophecy system is a set of dynamic database-driven web pages [12].

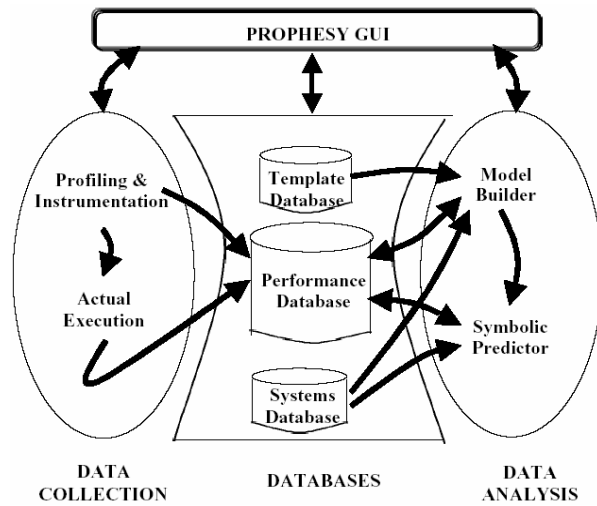


Figure 1. Prophecy system

The kernel coupling prediction method [7, 11], which is available with the Prophecy system, represents the performance of an application in terms of its kernels. A kernel is a unit of computation that denotes a logical entity within the larger context of an application. The unit may be a loop, procedure, or file. When developing performance models of an application, it is extremely useful to understand the relationships between the different functions that compose the application. The coupling method encapsulates this relation into coefficients that can be used to develop analytical models.

3. A Parallel Multiblock Lattice Boltzmann Application

The Lattice Boltzmann method is widely used for simulating fluid dynamics. In this section, we describe a PMLB (parallel multiblock Lattice Boltzmann)

application and its program control flow concerning kernels.

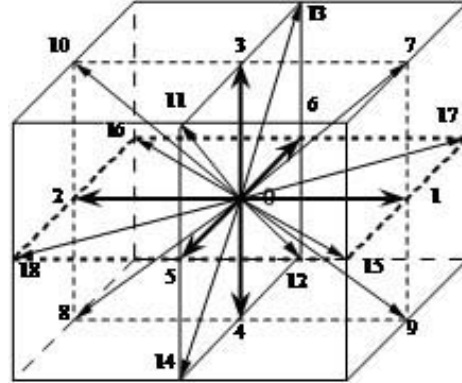


Figure 2. The D3Q19 lattice model

3.1 Lattice Boltzmann Method

The lattice Boltzmann method (LBM) is based on the kinetic theory, which entails a more fundamental level in studying the fluid than Navier-Stokes equations. In the LBM approach, the kinetic equation in the discretized phase space for the particle velocity distribution function $f(\mathbf{x}, \mathbf{e}_\alpha, t)$ ($f_\alpha(\mathbf{x}, t)$) is solved using the following equations:

$$f_\alpha(\mathbf{x} + \mathbf{e}_\alpha \delta t, t + \delta t) - f_\alpha(\mathbf{x}, t) = -\frac{1}{\tau} [f_\alpha(\mathbf{x}, t) - f_\alpha^{(eq)}(\mathbf{x}, t)]$$

where \mathbf{e}_α is the discretized particle velocity vector, \mathbf{x} is the spatial position vector, t is the time, δt is the time step, and τ is the relaxation time. The equilibrium distribution is of the Maxwellian form (expanded with the small Mach number assumption):

$$f_\alpha^{(eq)} = \rho w_\alpha \left[1 + \frac{3}{2} \mathbf{e}_\alpha \cdot \mathbf{u} + \frac{9}{2c^4} (\mathbf{e}_\alpha \cdot \mathbf{u})^2 - \frac{3}{2c^2} \mathbf{u} \cdot \mathbf{u} \right]$$

where w_α is the weighting factor, \mathbf{u} is the fluid velocity, and c is the lattice constant which is unit in our model. In the discretized phase space, the density and momentum fluxes can be evaluated as

$$\rho = \sum_{k=0}^n f_\alpha = \sum_{k=0}^n f_\alpha^{(eq)}$$

and

$$\rho \mathbf{u} = \sum_{k=1}^n \mathbf{e}_\alpha f_\alpha = \sum_{k=1}^n \mathbf{e}_\alpha f_\alpha^{(eq)}$$

where $n+1$ is the total number of discrete particle velocities. There are different 2- and 3-dimensional models discussed in [5]. In our simulations, we use the D3Q19 lattice model (19 velocities in 3D) with collision and streaming operations as shown in Figure 2.

3.2 Parallel Multiblock Lattice Boltzmann Method

Because of the high locality of the collision and streaming operations, domain decomposition is very effective and widely used for the parallel LBM. In the parallel implementation, the entire computational domain is divided into $n_1 \times n_2 \times n_3$ blocks, where n_1, n_2, n_3 are the number of segments in the x -, y -, and z -dimensions, respectively. Each block is assigned to a processor. The grid sizes in each block can be different. For example, Figure 3 shows blocks arrangement and the data communicated between blocks in a 2D case. The dash line is the internal boundary between blocks. The data in the dash line in the current block (0,2) must be transferred from the neighboring blocks. The neighbors of block (0,2) are blocks (0,1), (1,1), and (1,2). As we see in Figure 3, for the PMLB, the neighbors include the blocks connected through the corner c .

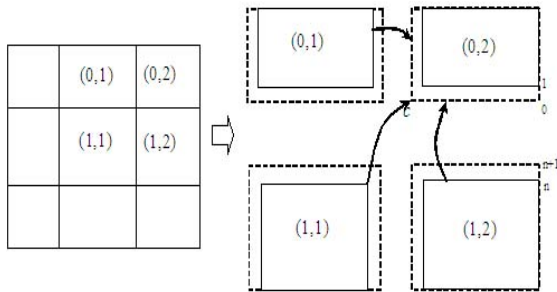


Figure 3. Blocks layout and communication pattern

3.3 Program Control Flow

The PMLB application code is written in C with MPI, and is divided into six kernels:

- *Initialization*: reads input files and sets up the initial parameter values.
- *Collision*: computes the effect of the collisions, which occur during the particle movement.

- *Communication*: communicates the needed data among neighboring blocks.
- *Streaming*: moves particles in motion to new locations along their respective velocities.
- *Physical*: calculates macroscopic variables such as fluid density, which are used in the collision and streaming steps.
- *Finalization*: cleans up the program after everything is done, and outputs the results.

The kernels *Collision*, *Communication*, *Streaming*, and *Physical* are in a loop with the number of iterations of 200. *Initialization* and *Finalization* are executed once.

4. Experiment Platforms

In this section, we describe the three platforms used for our experiments: P655, P690, and Seaborg. Table 1 shows the specifications of three platforms. The three platforms have the same operating system AIX 5.2, scheduler LoadLeveler 3.2, MPI and POE PE3.2, and C/C++ compiler VAC 6.0.

The Seaborg [3] 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. 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.

Table 1. Specifications of three SMP clusters

System	P655	P690	Seaborg
# Nodes	176	7	380
CPUs/Node	8	32	16
CPU type	1.5GHz P4	1.7GHz P4	375MHz P3
CPU Speed	6.0 GFlops	6.8 GFlops	1.5GFlops
Memory/node	16GB	128GB	16-64GB
Network	Federation	Federation	Colony

DataStar [6] is SDSC's largest IBM terascale machine. It has 176 (8-way) P655, and 7 (32-way) P690 compute nodes. The 8-way nodes have 16 GB, while the 32-way nodes have 128 GB of memory. 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. The use of 32-way nodes is shared among users.

We use Intel’s MPI benchmarks [1] to measure point-to-point performance between two processes within the same node (one process per processor) for intra-node performance, or between two nodes (one process per node) for inter-node performance. The performance is measured in Mbytes/s for bandwidth and in units of microseconds for latency. Table 2 presents the minimum latency and the maximum bandwidth of each cluster of SMPs with varying message sizes from 0 to 4MB. For intra-node performance, the bandwidths for P655 and P690 reach the maximum with the message size of 128KB. Seaborg gets the maximum bandwidth with the message size of 2MB. For inter-node performance, when the message size increases to 2MB or 4MB, these three SMP clusters get the maximum bandwidths. Note that the Federation switch used on P655 and P690 has lower latency and higher bandwidth than the Colony switch used on Seaborg.

Table 2. Latency and bandwidth across three SMP clusters using PingPong

Platform	Communication Mode	MPI Latency (us)	MPI Bandwidth (MB/s)
P655	Intra-node	1.68	2862.22
	Inter-node	5.66	1390.60
P690	Intra-node	3.52	2052.89
	Inter-node	6.74	1370.28
Seaborg	Intra-node	9.93	485.39
	Inter-node	26.22	224.24

5. Performance Analysis

In this section, we run the PMLB application on up to 512 processors. The problem sizes (i.e. computational domains) we used are five datasets: 32x32x32, 64x64x64, 128x128x128, 256x256x256, and 512x512x512. Because P690 allows up to only 4 nodes (32 processors per node) to be used for any parallel job submissions, the performance data for P690 corresponds to up to 128 processors. Further, multiple runs were performed for each data point to insure data consistency.

5.1 Problem and Processor Scaling

Because of the memory limitations, the application cannot be performed on one processor for the dataset of 256x256x256, and on less than 32 processors for the dataset of 512x512x512. Therefore, the bases for the relative speedups for the datasets of 256x256x256 and 512x512x512 are 2 processors and 32 processors, respectively.

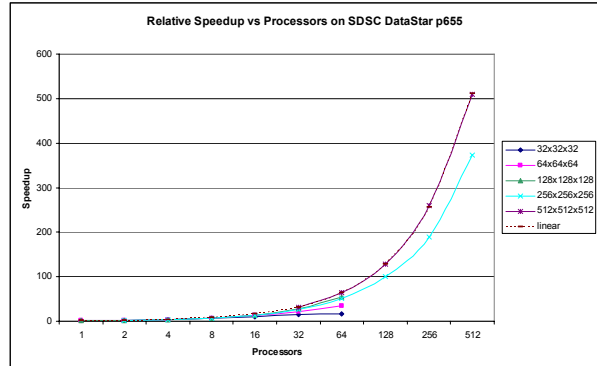


Figure 4. Performance on P655

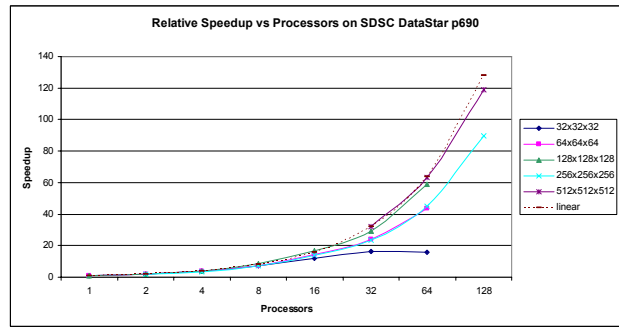


Figure 5. Performance on P690

Figure 4 indicates that the application scales well with increasing problem size and number of processors except the small problem size of 32x32x32 on P655. This is especially the case for the large problem size of 512x512x512, whereby the relative speedup is almost linear. For the small problem size, 32x32x32, the relative speedup is less than 20 on 64 processors because of the very small workload per processor.

Figures 5 and 6 indicate that the application scales well with increasing the number of processors on P690 and Seaborg; this is especially the case for the large problem size of 512x512x512, the relative speedup is almost linear. For the smallest problem size, 32x32x32, the relative speedup for 64 processors also is less than 20 on P690, and less than 10 on Seaborg.

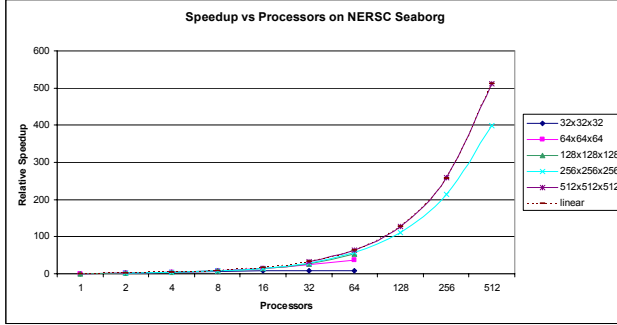


Figure 6. Performance on Seaborg

In summary, the application with large problem sizes scales well on all three architectures. For the large problem size, 512x512x512, these SMP clusters sustained the high relative speedup (e.g., almost linear).

5.2 Constant Workload per Processor

For the fixed workload per processor of size 32x32x32, we investigate how the execution and communication times change with increasing number of processors to further address the application scalability on these clusters.

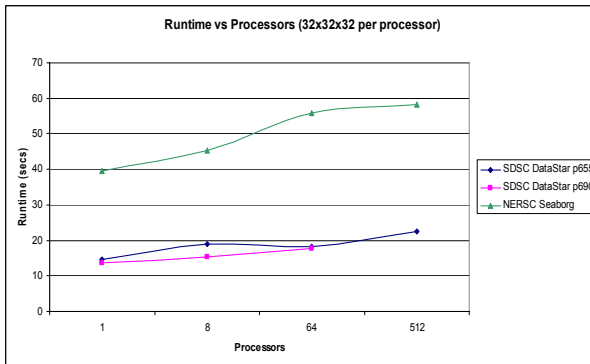


Figure 7. Runtime for the workload 32x32x32 per processor

Figure 7 indicates that the execution times increases slightly with an increase in the number of processors on the P655 and P690. In contrast, the execution time increases significantly on the Seaborg, given that the colony network of the Seaborg has higher latency and lower bandwidth than the federation network of the P655 and P690. This is further illustrated in Figure 8, which gives the communication time on different number of processors. The communication time on the Seaborg is twice as large as the P690 and P655, and for 512 processors, the communication rate is 17.72% on Seaborg and 8.84% on P655. Overall, the

application with constant workload per processor scales well with the increasing the number of processors for P690 and P655.

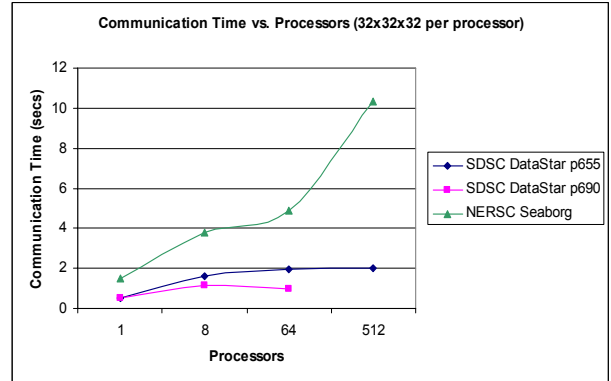


Figure 8. Communication time for the workload 32x32x32 per processor

6. Performance Modeling and Prediction

In this section, we model and predict the application performance across three SMP clusters.

6.1 Performance Modeling

In this section, we model each kernel of the PMLB application, and then use kernel coupling to combine these kernel models to generate the whole application model. As discussed in Section 3.3, the application consists of six kernels: *Initialization*, *Collision*, *Communication*, *Streaming*, *Physical*, and *Finalization*. *Initialization* and *Finalization* are called once; *Collision*, *Communication*, *Streaming*, and *Physical* are all called 200 times. Let performance model P_1 , P_2 , P_3 , P_4 , P_5 , and P_6 for six kernels respectively; let the number of calls N_1 , N_2 , N_3 , N_4 , N_5 , and N_6 for six kernels respectively, too. Then, we use Prophecy system to combine these kernel models to generate the whole application performance model as follows:

$$T = \sum_{i=1}^6 \alpha_i N_i P_i \quad (1)$$

where T denotes the application performance model. The term $\alpha_i (i=1, \Lambda, 6)$ is a coefficient for each term to represent the interaction that each kernel i has with the others, and is a weighted average of all coupling values associated with the given kernel i . For the given application, from Equation 1, we have its performance model

$$T = P_1 + 200 \times \sum_{i=2}^5 \alpha_i P_i + P_6 \quad (2)$$

where that α_1 and α_6 are 1 because initialization and finalization have very little interactions with the other kernels that are in a loop.

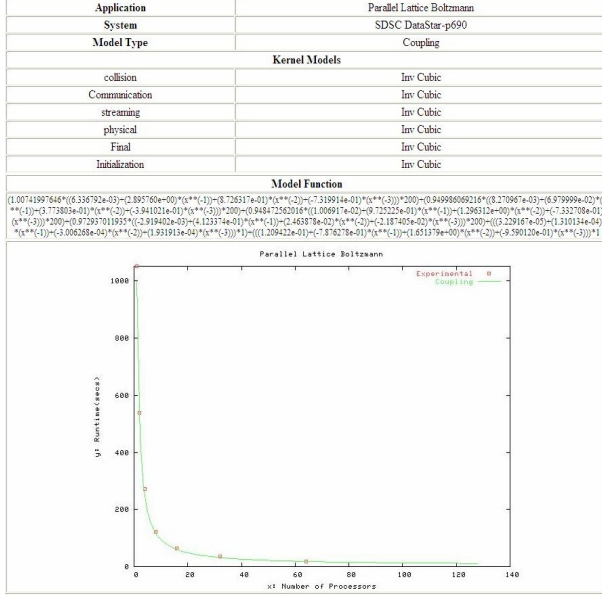


Figure 9. Performance model and its plot on P690

Figure 9 illustrates model development with the Prophecy system. The figure provides the performance model and the resultant plot for the P690; the actual execution data is given as data points in the graph. The inverse cubic curve fitting method is used to model each kernel and the coupling values are used to combine these models.

Table 3 shows actual runtime, predicted execution time using Equation 2, and prediction error rate for the problem size of 128x128x128. On all three SMP clusters, the prediction error rates are less than 9.50% on up to 64 processors, and the average prediction error rate is less than 3.50%.

Table 4 shows actual runtime, predicted execution time using Equation 2, and prediction error rate for the problem size of 256x256x256. On all three SMP clusters, the prediction error rates are less than 10.30% on up to 512 processors; only two predictions have the error rate more than 10%, and the average prediction error rate is less than 4.80%.

Table 5 shows actual runtime, predicted execution time using Equation 2, and prediction error rate for the problem size of 512x512x512. On all three SMP clusters, the prediction error rates are less than 4.90% on up to 512 processors, and the average prediction error rate is less than 1.95%.

In summary, on all three SMP clusters, the prediction error rates using Equation 2 are less than 10.3% on up to 512 processors, and the average prediction error rate is less than 4.8%.

6.2 Performance Prediction

In this section, we use the problem size of 128x128x128 to demonstrate the ability to reuse kernel coupling values from P655 and Seaborg to predict the performance on P690. Table 6 shows the predicted results for P690. The column “P690” is the same as that in Table 3. The column “P655” shows the predicted execution time on P690 using the kernel coupling values from the P655, and prediction error rate. The column “Seaborg” shows the predicted execution time on P690 using the kernel coupling values from the Seaborg, and prediction error rate. For all predictions, the prediction error rates are less than 9.50% on up to 64 processors, and the average prediction error rate is less than 3.93%. Similar results occurred with other problem sizes.

7. Summary

In this paper, we investigated the performance of the PMLB application using the Prophecy system. Experimental results indicate that the application with large problem sizes scaled well across the three SMP clusters, and the prediction model generated using kernel coupling with the Prophecy system was accurate with less than 4.8% average relative prediction error for all cases.

8. Acknowledgements

The authors would like to acknowledge the SDSC for the use of DataStar p655 and p690, and the DOE NERSC for the use of the Seaborg. This work is supported in part by NSF ITR-0086044, ITR-0085952, ANI-0225642, NSF NGS grant EIA-9974960, NASA grant NCC 2-1363, and TeraGrid TG-ASC040036T.

References

- [1] Intel MPI Benchmarks, Users Guide and Methodolgy Description, Version 2.3, <http://www.intel.com/cd/software/products/asmona/eng/cluster/mpi/219848.htm>.
- [2] McNamara G, Zanetti G., Use of the Boltzmann Equation to Simulate Lattice Gas Automata, *Phys Rev Lett* 1988; 61: 2332–2335.

- [3] NERSC Seaborg, <http://www.nersc.gov/nusers/resources/>.
- [4] Pohl, T., Deserno, F., Thurey, N., Rude, U., Lammers, P., Wellein, G., and Zeiser, T., Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures, in *Proc. of the SC2004*, 2004.
- [5] Qian YH, d'Humieres D, Lallemand P., Lattice BGK Models for Navier Stokes Equation, *Europhys Lett* 1992;17:479-484.
- [6] SDSC DataStar, http://www.sdsc.edu/user_services/datatar/.
- [7] Valerie Taylor, Xingfu Wu, Jonathan Geisler, and Rick Stevens, Using Kernel Couplings to Predict Parallel Application Performance, in *Proc. of the HPDC2002*, 2002.
- [8] Valerie Taylor, Xingfu Wu, and Rick Stevens, Prophecy: An Infrastructure for Performance Analysis and Modeling System of Parallel and Grid Applications, *ACM SIGMETRICS Performance Evaluation Review*, Volume 30, Issue 4, March 2003.
- [9] Xingfu Wu, Valerie Taylor, Jonathan Geisler, Xin Li, Zhiling Lan, Rick Stevens, Mark Hereld and Ivan R. Judson, Design and Development of the Prophecy Performance Database for Distributed Scientific Applications, in *Proc. of the 10th SIAM Conference on Parallel Processing for Scientific Computing*, March, 2001.
- [10] Xingfu Wu, Valerie Taylor and Rick Stevens, Design and Implementation of Prophecy Automatic Instrumentation and Data Entry System, in *Proc. of the 13th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS2001)*, 2001.
- [11] Xingfu Wu, Valerie Taylor, Jonathan Geisler, and Rick Stevens, Isocoupling: Reusing Coupling Values to Predict Parallel Application Performance, in *Proc. of the IPDPS2004*, April 2004.
- [12] 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.
- [13] D. Yu, R. Mei, and W. Shyy, A Multi-block Lattice Boltzmann Method for Viscous Fluid Flows, *Int J Numer Meth Fluids* 2002;39:99-120.
- [14] D. Yu, and S. Girimaji, Multi-Block Lattice Boltzmann Method: Extension to 3D and Validation in Turbulence, accepted to be published in *Physica*

Table 3. Predicted performance (seconds) and error rate for problem size of 128x128x128

#Procs	P655			P690			Seaborg		
	Runtime	Prediction	Error	Runtime	Prediction	Error	Runtime	Prediction	Error
1	1003.67	1074.83	7.09%	1051.09	1002.01	4.67%	3018.66	3003.82	0.49%
2	519.89	520.18	0.06%	537.449	560.61	4.30%	1504.66	1511.97	0.49%
4	274.56	276.16	0.58%	271.94	246.13	9.49%	758.00	766.31	1.10%
8	162.85	167.42	2.81%	121.19	121.44	0.21%	396.73	394.88	0.47%
16	78.40	77.11	1.64%	63.12	65.13	3.19%	224.20	217.96	2.78%
32	36.41	36.02	1.07%	36.01	35.59	1.14%	109.12	103.76	4.91%
64	18.16	17.93	1.80%	17.79	17.54	1.43%	55.84	52.96	5.14%
average			2.15%			3.49%			2.20%

Table 4. Predicted performance (seconds) and error rate for problem size of 256x256x256

#Procs	P655			P690			Seaborg		
	Runtime	Prediction	Error	Runtime	Prediction	Error	Runtime	Prediction	Error
2	4210.24	4179.74	0.72%	3828.56	4214.54	10.08%	13087.59	12999.59	0.67%
4	2298.15	2233.01	2.83%	2177.34	2151.97	1.17%	6217.65	6222.58	0.08%
8	1258.04	1336.15	6.21%	1059.49	1009.38	4.73%	3335.74	3316.55	0.58%
16	636.92	634.62	0.36%	556.21	562.02	1.04%	1800.26	1832.24	1.78%
32	326.92	316.33	3.24%	326.48	325.69	0.24%	899.50	905.79	0.70%
64	164.13	159.01	3.12%	169.43	164.49	2.91%	461.33	455.89	1.18%
128	84.06	79.12	5.88%	85.02	85.55	0.62%	235.63	221.42	6.03%
256	44.46	40.12	9.76%	NA	NA		121.95	113.25	7.14%
512	22.53	20.22	10.26%	NA	NA		58.24	53.92	7.43%
average			4.71%			2.97%			2.84%

Table 5. Predicted performance (seconds) and error rate for problem size of 512x512x512

#Procs	P655			P690			Seaborg		
	Runtime	Prediction	Error	Runtime	Prediction	Error	Runtime	Prediction	Error
32	2601.66	2474.42	4.89%	2590.18	2594.54	0.17%	7578.84	7593.77	0.20%
64	1306.55	1275.08	2.41%	1314.47	1312.39	0.14%	3822.76	3778.13	1.17%
128	653.31	657.82	0.69%	685.63	695.98	1.51%	1897.32	1914.27	0.89%
256	319.71	317.90	0.57%	NA	NA		933.59	928.74	0.52%
512	163.44	161.58	1.13%	NA	NA		474.04	474.19	0.03%
average			1.94%			0.61%			0.56%

Table 6. Predicted performance (seconds) and error rate for problem size 128x128x128 for P690

#Procs	P690			Seaborg		P655	
	Runtime	Prediction	Error	Prediction	Error	Prediction	Error
1	1051.09	1002.01	4.67%	1010.62	3.85%	1042.18	0.85%
2	537.449	560.61	4.30%	528.41	1.69%	499.83	7.01%
4	271.94	246.13	9.49%	252.5	7.15%	254.92	6.26%
8	121.19	121.44	0.21%	116.83	3.60%	118.94	1.86%
16	63.12	65.13	3.19%	62.22	1.43%	63.11	0.00%
32	36.01	35.59	1.14%	36.38	1.04%	34.13	5.22%
64	17.79	17.54	1.43%	19.33	8.66%	17.16	3.53%
average			3.49%		3.92%		3.53%