PISL: A Large-Scale In Silico Experimental Framework for Agent-Directed Physiological Models

Sunwoo Park, Glen E. P. Ropella, and C. Anthony Hunt
The BioSystems Group, University of California, 513 Parnassus Ave., San Francisco, CA, 94143-0446
spark4@itsa.ucsf.edu gepr@tempusdictum.com hunt@itsa.ucsf.edu

Keywords: systems Biology, biosystems, Swarm, agent modeling, distributed simulation

Abstract

We present in silico, physiologically-based rat liver models that are designed for large-scale and high-performance experimentation and analysis. We refer to the framework used as the parallel in silico liver (PISL). The models include agents. The simulations, intended for exploratory experimentation, are agent-directed. To facilitate reaching our goals the models use multi-scale parallelism, parallel parameter sweeping, parallel partitioning and batch processing, a posteriori analysis, all of which is contained within a high-performance computing infrastructure. The system includes a fully automated simulation and analysis life cycle management component that enables conducting large numbers of in silico experiments under various experimental and model configurations. Here, we present and discuss the main characteristics of these capabilities.

1. INTRODUCTION

We are developing a large-scale, high-performance framework for in silico experimentation and analysis of biological systems that is based on the Functional Unit Representation Method (FURM). FURM is a constructive modeling paradigm that supports iterative experimentation on multi-scale in silico biological systems [1], [2]. The framework used provides for running and using multiple different models of the same system. It has sufficient flexibility to represent different aspects of biological systems.

The behavior space of biological systems is large. However, to date in silico experiments have often only explored a small subset of the space, and the behaviors contained are often experimentally constrained. As a consequence, the experiments typically fail to reproduce realistic biological phenomena. To make progress in overcoming such limitations, multiple, modular, and multi-scale models of a biological system and its components are required to more adequately represent the system's structural and behavioral properties. In silico biomimetic devices are models that represent aspects of anatomic structure combined with selected behaviors of functional units of a biological system [3], [4]. The devices and their components logically map to their biological counterparts. Their components can be plugged together in different ways to represent different mammalian tissues and organs. Our biomimetic devices are designed to be reusable, revisable, modular, and adaptable. For scientific experimentation it is also necessary to support multi-scale, multi-resolution descriptions of these systems and their behaviors.

The In Silico Liver (ISL) is the first class of biological system models being developed based on FURM [3]. A purpose of the ISL is to mimic structural and behavioral properties of the hepatic lobule, the primary functional unit of livers in both rat and man. Validation involves comparing outflow profiles of compounds administered to perfused rat livers [5] with in silico outflow profiles of objects (intended to represent a specific compound) "administered" to an ISL following a protocol that mimics that of the original perfused liver experiments. The similarity of the two outflow profiles, simulated and experimental, is compared using a similarity measure.

Agent-based models have advantages when attempting to understand and simulate phenomena produced by complex adaptive systems. Because all biological systems are complex and adaptive, we design our biomimetic devices to use agents wherever appropriate. We also employ agents outside of the biological system model, but within the computational framework, to play some of the analytical, observational, and data analysis roles preformed by the researcher conducting the original experiments. We use Swarm as an agent M&S framework [6].

Models of the biological systems of interest, such as the ISL, can easily involve thousands of interacting objects and agents within multiple spaces. The potential scope of the systems behavior space can quickly become very large. A given experiment may need to run for many cycles – dozens to hundreds or more – in order to detect the presence or absence of the behavior of interest. Furthermore, because most events are stochastic, it may be necessary to independently repeat an experiment many times to confirm a consistent behavior or to obtain the quality of data needed for application of a similarity measure. Clearly, a systematic exploratory analysis of model behaviors will be computationally intensive and somewhat slow if confined to a single or dual processor computer. The parallel ISL (PISL) described here makes progress in addressing all of

theses potential limitations. It is a new version of the ISL that enables a researcher to take advantage of the high performance capabilities and thereby conduct large-scale experiments and analyses. The PISL incorporates advanced M&S features into the ISL framework: multi-scale parallelism, parallel parameter sweeping, parallel batch processing and partitioning, and a posteriori analysis. The PISL provides users with a fully automated simulation and analysis life cycle management system. It also enables executing and analyzing large numbers of different ISL experiments under various experimental and model configurations.

2. THE IN SILICO LIVER (ISL)

The mammalian liver has several lobes. The hepatic functions of each lobe are located in hundreds of essentially identical polyhedral functional units called lobules [7]. The incoming portal vein and arterial blood supplies branch several times ending at adjacent faces and corners of the polyhedral lobules. From there the blood supply spreads out over the adjacent faces forming portal vein tracts (PV) that feed flow into an interconnected network of sinusoids. Sinusoids are lined with a monolayer webwork of primarily endothelial cells that are separated from plates of hepatocytes by a narrow space of Dissé. The hepatocytes, the repository of most hepatic functions, are flanked on both sides by a space of Dissé. The portal and arterial blood supplies mix together at the outer edges of lobules and then percolate through the merging sinusoids into a common central vein (CV) near the lobule's center. Hepatocytes closer to the PV exhibit a different range of functions than those close to the CV, and those differences are typically associated with zones, as illustrated in Fig. 1.

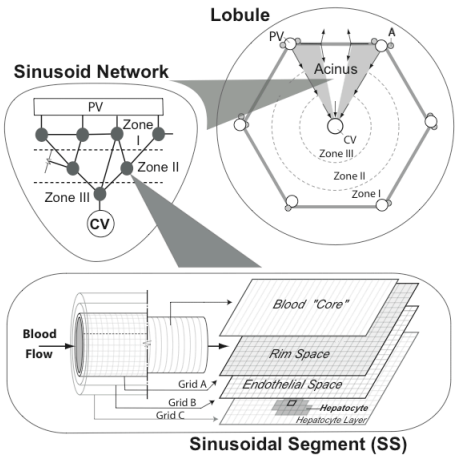

Figure 1. Lobule, sinusoid network, and sinusoid segment of ISL

When dosed with objects representing sucrose, an appropriately parameterized ISL can produce outflow profiles that are indistinguishable from those obtained from *in situ* experiments. The ISL framework contains three models: *DatModel*, *RefModel*, and *ArtModel*. *DatModel* represents all of the outflow data from a given *in situ* perfusion experiment. *RefModel* is a traditional mathematical model of hepatic solute outflow. We currently use the extended convection-dispersion model [7]. *ArtModel* is an articulated, functional unit model that represents an acinus*, the shaded region in Fig. 1.

As illustrated in Fig. 1, a directed graph is used to represent *ArtModel's* interconnected network of sinusoidal flow paths. Several independently assembled ArtModels are needed per ISL. The PV is the source of and the CV is the sink for sucrose objects. The *ArtModel's* directed graph is subdivided into three zones (I, II, and III) and requires at least one node in each zone. One of two types of sinusoidal segment (SS) is placed at each node. The graph dictates the source of objects entering a SS and the target for objects exiting a SS. Each SS is an agent that consists of a perfusate tunnel and three layered spaces as pictured in Fig. 1. The perfusate tunnel (Core) is a vascular passage through which solutes move. The innermost layer of the SS (Grid A) represents the rim of the tunnel. The outer layer wrapped around the Grid A (Grid B) is the endothelial layer. The outmost layer wrapped around the grid B (Grid C) is the space of Dissé, hepatocytes, and bile canaliculi.

The ISL uses a set of parameters to specify a particular experimental setup. The set is divided into four subgroups; *experimental*, *in silico, reference*, and *data*. The subgroup of experimental parameters specifies aspects of system external to *ArtModel*. The other three parameter subgroups denote model configurations: the in silico, reference and data subgroups for *ArtModel*, *RefModel*, and *DatModel*, respectively. All parameters are organized together in a parameter file. Constraints are placed on the allowable parameter values using published data [8]. The framework performs an in silico experiment by reading the parameter file and executing all three models, *ArtModel, RefModel,* and *DatModel*, in parallel.

Validation has been successfully conducted over specific ISLs using published data [5]. In the case of sucrose we have a series of sucrose outflow profiles: four profiles from the same perfused rat liver and one profile from each of six different livers. Validation has been conducted by measuring the similarity [9] between simulated sucrose outflow profiles and those from the *in situ* experiments. Searches of both model and parameter spaces are required to locate a model configuration and parameter vector that validates. Using these values, the ISL will produce results that are experimentally indistinguishable from those of the *in situ* experiments.

* An acinus is comprised of portions of two adjacent lobules fed by a common portal vein tract.

3. PARALLEL IN SILICO LIVER (PISL)

3.1. Multi-Scale Parallelism

Multi-scale parallelism is a collection of parallelisms in which each is bound to a particular system granularity. A parallelism refers to the degree of independent and simultaneous execution of multiple tasks. Multi-scale parallelism permits exploitation of different parallelisms depending on the configuration of the in silico experiment along with the structural and functional levels of ISL. The current PISL implementation supports multi-scale parallelism at two different levels: *Group Level Parallelism* and *Experiment Level Parallelism*. Group Level Parallelism enables executing multiple experiments in parallel by dividing them into groups and allowing each group to run simultaneously. It escalates the execution speed and reduces total execution time for large numbers of in silico experiments. Parallel batch processing and local simulation result analyses are performed in this mode. Experiment Level Parallelism allows one to conduct a large-scale in silico experiment in parallel. It parallelizes the collection of independent, repetitive, and homogeneous fine-grain experimental steps. A single ISL experiment is actually a combination of multiple, independent Monte Carlo simulation runs. Those runs are parallelized in Experiment Level Parallelism.

PISL specifies two additional fine-grain parallelisms: *Validation and Verification Level Parallelism* and *Component Level Parallelism*. Verification Level Parallelism helps parallelize the multiple models that characterize FURM and that may be involved in validation and verification of a particular ISL. In the PISL, *ArtModel, RefModel,* and *DatModel* can be parallelized and can be executed independently during Verification Level Parallelism.

The three parallelisms described above are associated with batch processing, experiment control, and the validation and verification process, rather than with the model itself. Component Level Parallelism allows components of a model to be run in parallel. *ArtModel* can be parallelized based on the anatomic or functional level of the decomposed ISL.

3.2. Parallel Parameter Sweeping

Experimentation on an ISL uses large numbers of parameters, and experiments can be carried out over a wide range of values. It is impractical to perform tasks that need to explore the entire parameter space (e.g., searching for a global maximum). Global search and optimization techniques that are based on heuristics explore the assigned space until a solution is found. However, in general, they do not guarantee that the solutions found are globally unique because the space has not been exhaustively explored. Parameter sweeping is an alternative approach that explores only parameter subspaces seeking solutions over those spaces. Parameter sweeping is a very useful technique for conducting large numbers of different in silico experiments and performing sensitivity analyses over experimental results. The spaces are identified by searching all specified values associated with a given partial set of parameters over the space. For example, suppose a parameter space contains paired objects, and that each pair consists of a parameter name and a value as illustrated in Figure 2. We construct a parameter sweeping space by choosing some parameters from the entire parameter space (e.g., $\{p_i, p_j, p_k\}$), and then search all pairs that contain those named parameters over the space (e.g., (p_i, v_a), (p_i, v_b), (p_j, v_c), (p_j, v_d), (p_j, v_e), (p_k, v_f), (p_k, v_g)). Because the parameter sweeping space is exhaustively searched, we can identify results that meet or exceed a specified similarity measure. In general, it is recommend that one carefully select a small number of parameters so that their sweeping space becomes reasonably smaller than the entire parameter space.

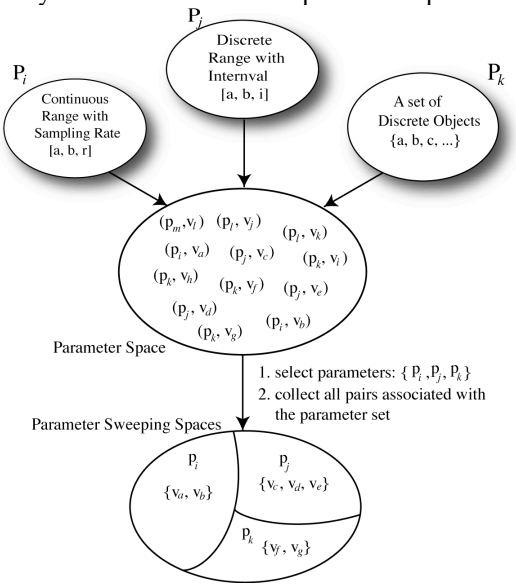

Figure 2. Parameter sweeping space construction

PISL provides an efficient parameter sweeping mechanism by constructing non-linear parameter sweeping spaces in parallel. Along with other features such as parallel batch processing, distributed and parallel agent-directed simulation, and a posteriori analysis, the PISL provides fully automated in silico experiments and analyses over these spaces.

3.3. Parallel Partitioning and Batch Processing

Parallel partitioning is the process of distributing system resources to a set of partition blocks in parallel to improve overall computational performance of in silico experiments. It improves the efficiency of system resource

management and the degree of runtime parallelism in large-scale in silico experiments. From a computational perspective, every processor simultaneously determines its computational resource needs with or without the cooperation of other processors. The system resources are parameter files in Group Level Parallelism and Monte Carlo runs in Experiment Level Parallelism. A set of parameter files, which can be produced by the parameter sweeping construction process or other parameter space generation processes, is partitioned in Group Level Parallelism. A collection of multiple Monte Carlo runs is partitioned in Experiment Level Parallelism. The improvements lead to overall performance gains. The current PISL implementation supports a simple but effective partitioning scheme that tries to minimize the difference between partition block lengths, where a partition block contains at least one parameter file. Application of a more advanced partitioning scheme [10], one that is expected to produce even better partitioning results, may be included in future versions of the PISL.

Parallel batch processing is the process of independently performing multiple in silico experiments in a sequential order using multiple processors. In PISL, parallel batch processing is accomplished when the PISL is configured to Group Level Parallelism. Each processor automatically loads multiple parameter files and runs in silico simulations with those files in a sequential order. It reads necessary parameter files from a particular location in parallel, whereas the ISL reads only a prefixed single parameter file. With the support of a file filtering feature, the PISL also loads only designated, interesting files instead of all available parameter files.

3.4. Distributed and Parallel Agent-Directed Simulation

The PISL provides a distributed and parallel, agent-directed simulation environment for large-scale ISL experimentation. The PISL provides a simulation coordinator that controls overall simulation behaviors during experiments. Along with other system components such as the parameter sweeping space generator, batch processor, partitioner, and a component to deploy and resolve simulation tasks, the coordinator effectively manages the large-scale ISL simulations. It leads the PISL to support a fully automated high-performance agent-directed simulation environment.

We built the simulation environment using Swarm and MPI on a small-scale in-house, eight-node Beowulf cluster machine. MPI provides both programming and runtime environment for distributed and parallel agent-directed simulation [11]. Beowulf clustering provides a high performance computing infrastructure [12]. We designed and implemented the PISL simulation environment so that it can be ported to other high performance computing environments.

3.5. A Posteriori Analysis

A posteriori analyses perform analytic work on collections of experimental results after all simulations are complete. It permits one to automatically identify the best or most interesting outcomes from large numbers of experimental results. Currently, the PISL automatically determines the best experiment by evaluating similarity measure values of all experimental results after all have been successfully completed. Sensitivity analysis can be implemented in the a posteriori analysis phase.

4. EXPERIMENTS AND RESULTS

We conducted a series of in silico experiments using various model configurations and experimental settings. Based on the specified parameter sweeping information (Appendix), the PISL constructs, partitions, and simulates 36 parameter files. After all simulations have been completed, the PISL locates the parameter files that produced the best experimental results: the one with the maximum similarity measure value with respect to *in vivo* experimental data. We repeated the same experiments under different parallelism modes and using different numbers of processors. We also measured and analyzed the simulation execution time of each experiment.

We confirmed that the PISL achieved high performance gains over the sequential ISL. Table 1 shows the simulation execution time of both the ISL and the PISL, and the relative speedup of the PISL over the sequential ISL during the first three experiments. Eight processors were used in both Group Level Parallelism (GLP) and Experiment Level Parallelism (ELP) modes.

Table 1. Simulation execution time of ISL and PISL

Parameter File	Simulation Execution Time (in seconds)			Speedup (over ISL)	
	ISL	GLP	ELP	GLP	ELP
Param-0.scm	1217.00	150.54	144.62	8.08	8.42
Param-1.scm	13723.00	276.64	270.84	49.72	50.67
Param-2.scm	1900.00	230.93	226.41	8.26	8.39

As shown in Table 2, the PISL experimental results also show the characteristic of linear speedup as the total number of processors increases. The PISL experiments were conducted over 36 parameter files (i.e., the entire parameter sweeping space) by changing the number of processors from four to eight in both the Group and Experiment Level Parallelism modes. Simulation execution time and speedup for each parameter file in Group Level Parallelism mode are listed in the Appendix.

Table 2. Linear speedup of PISL experiments in GLP mode

		Number of Processors				
		4	5	6	7	8
Simulation time (in seconds)	AVG	709.82	553.06	451.41	417.05	352.26
	STDEV	378.29	331.09	245.72	256.63	181.36
Speedup (over 4 processors)	***Ideal***	***1.00***	***1.25***	***1.50***	***1.75***	***2.00***
	AVG	***1.00***	***1.36***	***1.63***	***1.80***	***2.01***
	STDEV	0.00	0.37	0.35	0.36	0.28

AVG: Average, STDEV: standard deviation

5. CONCLUSION

We developed a parallized in silico isolated liver model and framework (PISL) to conduct large-scale in silico experiments and analyses in high performance computing environments. The PISL extends and improves the ISL by making available a collection of new features: multi-scale parallelism, parallel parameter sweeping, parallel partitioning and batch processing, and a posteriori analysis. The PISL also provides a fully automated high performance agent-directed simulation capability. Using those features, the PISL can efficiently perform large numbers of in silico experiments and analyses under various experimental conditions and model configurations. The PISL has been successfully validated and verified against in situ experimental data.

ACKNOWLEGMENT

This research has been supported in part by CDH Research Foundation Grant No. R21-CDH-00101, "Fundamentals of Agent-Based Models for Biological Systems."

BIOGRAPHIES

Sunwoo Park is a postdoctoral fellow of Biopharmaceutical Sciences and Bioengineering at UCSF. His research areas include distributed/parallel modeling and simulation, discrete event systems, systems biology and computational biology, and theoretical computing.

C. Anthony Hunt has served as a Professor of Biopharmaceutical Sciences and Bioengineering at the University of California, San Francisco (UCSF) for over 20 years. As Director of the BioSystems group, three of his areas of interest are Systems Biology, Computational Biology and the Pharmaceutical Sciences.

Glen E. P. Ropella is a systems engineer who has worked on various types of simulations for approximately 13 years. He's been working in the domain of fine-grained Agent-Based Modeling for the past 7 years.

REFERENCES

[1] Biosystems Group. 2004. "Functional Unit Representation Method (FURM)." University of California at San Francisco, available at http://biosystems.ucsf.edu/Research/furm/index.html.

[2] Noble, D. 2002. "The Rise of Computational Biology." *Nat. Rev. Mol. Cell. Bio. 3*:459-463.

[3] Hunt, C.A.; G.E.P. Ropella; M. S. Roberts; L. Yan. 2004. "Biomimetic In Silico Devices." *Computational Methods in Systems Biology 2004 (CMSB)* (Paris, France, May 26-28).

[4] B. Falkenhainer and K. D. Forbus. 1999. "Compositional Modeling: Finding the Right Model for the Job." *Artificial Intelligence 51*:95-143.

[5] Hung, D.Y.; P. Chang; M. Weiss; M. S. Roberts. 2001. "Structure-Hepatic Disposition Relationships for Cationic Drugs in Isolated Perfused Rat Livers: Transmembrane Exchange and Cytoplasmic Binding Process." *J. Pharmacol. Exper. Therap. 297*:780-789.

[6] Swarm wikki. "SwarmWiki, the agent-based modeling resource." http://wiki.swarm.org/.

[7] Roberts, M.S. and Y. G. Anissimov. 1999. "Modeling of Hepatic Elimination and Organ Distribution Kinetics with the Extended Convection-Dispersion Model." *J. Pharmacokin. Biopharm. 27*:343-382.

[8] Teutsch, H.F.; D. Schuerfeld; E. Groezinger. 1999. "Three-Dimensional Reconstruction of Parenchymal Units in the Liver of the Rat." *Hepatology 29*: 495-505.

[9] Ropella, G.; D. Nag; C.A. Hunt. 2003. "Similarity Measures for Automated Comparison of In Silico and In Vitro Experimental Results." *Proceedings of the 25th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC03)*, (Cancun, Mexico, September 17-21). IEEE, Piscataway, NJ, 2933-2936.

[10] Park, S. and B.P. Zeigler. 2002. "Distributing Simulation Work Based on Component Activity: A New Approach to Partitioning Hierarchical DEVS Models." *CLADE 2003* (June 2003).

[11] MPI Forum. "The Message Passing Interface (MPI) standard." http://www-unix.mcs.anl.gov/mpi/.

[12] Sterling,T.; D.J. Becker; D. Savarese. 1995. "Beowulf: A Parallel Workstation for Scientific Computation." *Proceedings of the 24th International Conference on Parallel Processing (ICPP) 1* (August)*:*11-14.

APPENDIX

A. PISL Sweep specification file

The PISL builds a parameter sweeping space based on a user-provided specification file. The specification file that we used contains the following information.

```
inputs/parameters/liverBase.scm
cycleLimit 100 200 400
artCoreFlowRate 1 2 3
artSinusoidTurbo 0.20D0 0.40D0
monteCarloRuns 8 16
```

During the parameter sweeping space construction phase, the PISL extracts all parameter information from the file, *inputs/parameters/liverBase.scm,* and stores it in an internal

parameter repository (in our experiments, we considered only 59 parameters and their values). The PISL constructs a parameter sweeping space by dynamically changing the contents of the repository and storing them in new files. We use four parameters, *artCoreFlowRate, artSinusoidTurbo, cycleLimit,* and *monteCarloRuns* to update the repository. As a result, the PISL produces 36 different parameter files starting from *param-0.scm* to *param-35.scm*. Each parameter file contains all of the parameter information listed in *liverBase.scm* except for that of the above four parameters. The values of these four parameters were determined based on parameter values listed in the parameter sweeping specification file. For example, *1, 0.220D, 100,* and *8* are assigned to *artCoreFlowRate, artSinusoidTurbo, cycleLimit,* and *monteCarloRuns,* respectively, in *Param-0.scm.*

B. PISL simulation execution time result in GLP mode

Table 3. PISL simulation execution time over various numbers of processors in GLP mode

	Number of Processors				
	4	**5**	**6**	**7**	**8**
param-0	299.25	262.68	261.92	263.61	150.54
param-1	665.12	548.30	388.82	394.96	276.64
param-2	406.84	381.30	413.87	464.30	230.93
param-3	884.35	870.39	731.31	592.84	398.27
param-4	1132.19	716.99	736.02	765.83	378.36
param-5	1520.66	1459.59	1028.99	1100.25	657.72
param-6	299.63	295.68	261.21	133.20	145.47
param-7	582.08	374.17	379.32	374.80	241.81
param-8	513.62	418.83	433.84	196.87	217.02
param-9	750.62	591.82	576.89	667.35	423.88
param-10	614.42	641.33	783.10	370.35	372.77
param-11	1349.83	1077.01	908.17	691.71	713.37
param-12	254.95	259.49	144.51	151.30	144.80
param-13	534.56	373.12	376.37	235.65	245.32
param-14	411.47	383.31	217.13	233.98	222.08
param-15	825.80	583.85	639.18	388.17	417.42
param-16	724.44	717.29	384.97	461.71	382.87
param-17	1434.44	966.13	864.67	724.50	722.48
param-18	261.95	302.41	144.20	133.68	143.81
param-19	574.22	393.67	338.07	315.90	253.79
param-20	392.44	422.28	218.15	219.44	222.10
param-21	789.68	615.03	559.74	351.21	356.35
param-22	757.22	341.18	389.97	374.24	384.22
param-23	1336.95	989.02	910.30	704.25	710.37
param-24	278.40	126.46	143.30	157.27	143.27
param-25	593.35	488.53	259.01	254.34	261.27
param-26	387.29	183.20	221.58	195.29	222.68
param-27	802.72	605.67	446.59	459.30	443.61
param-28	674.62	379.47	378.48	335.76	389.15
param-29	1381.70	877.71	649.76	679.95	597.55
param-30	255.68	146.70	145.46	144.98	144.47
param-31	520.73	402.12	298.70	300.74	283.59
param-32	414.54	226.12	227.03	218.14	228.93
param-33	753.16	631.02	340.44	376.78	406.90
param-34	720.97	364.26	401.72	398.67	388.15
param-35	1453.69	1494.10	647.88	1182.54	759.35
AVG	**709.82**	***553.06***	**451.41**	**417.05**	**352.26**
STDEV	**378.29**	***331.09***	**245.72**	**256.63**	**181.36**

AVG: Average, STDEV: standard deviation; unit: second

C. Linear speedup of PISL in GLP mode

Table 4. Linear speedup of PISL over various numbers of processors in GLP mode

	Number of Processors				
	4	**5**	**6**	**7**	**8**
param-0	1	1.14	1.14	1.14	1.99
param-1	1	1.21	1.71	1.68	2.40
param-2	1	1.07	0.98	0.88	1.76
param-3	1	1.02	1.21	1.49	2.22
param-4	1	1.58	1.54	1.48	2.99
param-5	1	1.04	1.48	1.38	2.31
param-6	1	1.01	1.15	2.25	2.06
param-7	1	1.56	1.53	1.55	2.41
param-8	1	1.23	1.18	2.61	2.37
param-9	1	1.27	1.30	1.12	1.77
param-10	1	0.96	0.78	1.66	1.65
param-11	1	1.25	1.49	1.95	1.89
param-12	1	0.98	1.76	1.69	1.76
param-13	1	1.43	1.42	2.27	2.18
param-14	1	1.07	1.90	1.76	1.85
param-15	1	1.41	1.29	2.13	1.98
param-16	1.00	1.01	1.88	1.57	1.89
param-17	1.00	1.48	1.66	1.98	1.99
param-18	1.00	0.87	1.82	1.96	1.82
param-19	1.00	1.46	1.70	1.82	2.26
param-20	1.00	0.93	1.80	1.79	1.77
param-21	1.00	1.28	1.41	2.25	2.22
param-22	1.00	2.22	1.94	2.02	1.97
param-23	1.00	1.35	1.47	1.90	1.88
param-24	1.00	2.20	1.94	1.77	1.94
param-25	1.00	1.21	2.29	2.33	2.27
param-26	1.00	2.11	1.75	1.98	1.74
param-27	1.00	1.33	1.80	1.75	1.81
param-28	1.00	1.78	1.78	2.01	1.73
param-29	1.00	1.57	2.13	2.03	2.31
param-30	1.00	1.74	1.76	1.76	1.77
param-31	1.00	1.29	1.74	1.73	1.84
param-32	1.00	1.83	1.83	1.90	1.81
param-33	1.00	1.19	2.21	2.00	1.85
param-34	1.00	1.98	1.79	1.81	1.86
param-35	1.00	0.97	2.24	1.23	1.91
AVG	**1.00**	***1.36***	**1.63**	**1.80**	**2.01**
STDEV	**0.00**	***0.37***	**0.35**	**0.36**	**0.28**

AVG: Average, STDEV: standard deviation