Annotated Bibliography

Amy Le, Emily Bertelson, Kristina Entzel, Long Tat

CSCI 4576/5576

November 18, 2014

References

[1] K.B. Athreya and Jack Dai. Random logistic maps. *Journal of Theoretical Probability*, 13(2):595–608, 2000.

Athreya and Dai explore the concept of a time-varying logistic map. In a sense, they lay out the general groundwork for exploring a spatially-varying logistic map. They find a theoretical explanation for their observations, expressed as probabilities. These findings imply that (and this is beyond the scope of the project) there could be some interesting observations from the spatially-random logistic map, and the observations can be quantified as probabilities. This project's goal is to characterize the map in terms of the average number of period p orbits in any given realization, and also to find the set-valued bifurcation diagram that describes this system.

[2] Center for Computational Research. MPI and parallel computing, 2004–2014. http://www.buffalo.edu/ccr/support/UserGuide/AdvancedTopics/mpi.html.

This article and corresponding site explain MPI programmimg with a problem focused context when it comes to parallel computing. It contains tutorials for parallel programs and bash scripts involving the slurm workload manager, tutorial slides for the different practical problems that will be solved with MPI and the issues that arise with them.

[3] Rohit Chandra. *Parallel programming in OpenMP*. Morgan Kaufmann, 2001.

The authors of this book were originally SGI engineers who were involved in the design and implementation of OpenMP. The main information avaible about OpenMPI can be found at www.openmp.org. Although full specification of OpenMP is appropriate and complete, it is not a very accessible format for programmers wishing to use OpenMP for developing parallel application. This book tries to fullfill the needs of these programmers. This book can serve as a complete reference guide, also can be the mean tool for exploring options to improve performance on the OpenMP section of the project(removing dependency, load balancing between threads).

[4] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, AD '11, pages 36–47, New York, NY, USA, 2011. ACM.

As the title suggests, this paper gives an overview of the functionality provided by HDF5, speaking at a higher, more conceptual level than documentation. Particularly applicable is a section on improving I/O performance.

[5] The HDF Group. HDF5 user's guide, 2014. http://www.hdfgroup.org/HDF5/doc/UG/index.html.

This user's guide covers HDF5 usage in C and Fortran. It is comprehensive and well-organized. A listing of available dataset functions appears to be an especially practical reference.

[6] Jeroen S.W. Lamb, Martin Rasmussen, and Christian S. Rodrigues. Topological bifurcations of minimal invariant sets for set-valued dynamical systems. *Proceedings of the American Mathematical Society*, 2013.

Lamb, et. al explore the concept of set-valued bifurcations as an extension of the more common single-valued bifurcation. The kind of problems the authors are interested in are random dynamical systems, such as the Random Logistic Map. This paper serves as a theoretical underpinning for our simulation, and also as a reference for the set-valued bifurcation diagram we plan to generate.

[7] S. Olivier and J. Prins. Scalable dynamic load balancing using UPC. In *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pages 123–131, Sept 2008.

Olivier and Prins implement an asynchronous work-stealing dynamic load balancer with Unified Parallel C (UPC). They evaluate the performance of their balancer with the Unbalanced Tree Search (UTS) benchmark, which is a synthetic tree-structured search space that is highly imbalanced. They observe parallel efficiency of 80% using 1024 processors performing over 85,000 total load balancing operations per second continuously. An additional finding is that the careful use of one sided reads and writes is necessary to minimize the communication overhead. The authors' findings indicate that we should minimize the number of read and write operations as we compute solutions to the fixed point equations in order to keep the communication overhad low.

[8] Mitsuhisa Sato, Toshihiro Hanawa, Matthias S Müller, Barbara Chapman, and Bronis R de Supinski. *Beyond Loop Level Parallelism in OpenMP: Accelerators, Tasking and More*, volume 6132. Springer, 2010.

Chapter one of this book, Enabling Low-Overhead Hybrid MPI/OpenMP Parallelism with MPC, introduce a new module to MPC framework handling a fully 2.5-compliant OpenMP runtime completely intergrated to an MPI1.3 implementation. This chapter review strategy how to target oversubscribing capabilities and the possibility to run hybrid MPI/OpenMP application with a limited overhead. This can be really useful for us when implementing our strategy as we are most likely will face the same problem, ie how to optimize performance when implement the hybrid system using limited sharing resources.

[9] Various contributers: The Open MPI Project. Open MPI: Open Source High Performance Computing, 2014. http://www.open-mpi.org/faq/.

This article goes through frequently asked questions for new MPI users, like most of us in this class. It covers general information and usability, running, and tuning for the best performance. It also suggests performance and analysis tools besides just manual walltime measurement and MFLOP/second measurement, which may give us measurements without crowded walltime syntax.

[10] Marc H. Willibeek-LeMair and Anthony P. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed Computing*, 4(4), September 1993.

Willibeek-LeMair and Reeves discuss five strategies for dynamic load balancing: sender initiated diffusion, receiver initiated diffusion, hierarchical balancing model, gradient model, and dimension exchange method. The authors consider tasks such as: processor load evaluation, load balancing profitability, task migration, and task selection. Given the trade off between accuracy and increased time for communication, they conclude that the receiver initiated diffusion (RID) is the best method for dynamically load balancing. It is the method that scales the best with the number of processors and requires the least amount of communication overhead. Dynamic load balancing is key for solving systems whose solutions are defined recursively, such as a fixed point iteration. We will use the findings of this paper to guide our program design such that it is the best suited for RID.