Statement of Purpose for a Postdoctoral Fellow - Computational Science Department-81936 at Lawrence Berkeley Lab

Rahulkumar Gayatri *March 25, 2016*

Multi-core processors are a norm today. Most of the new generation computing systems in the market are at-least a quadcore with hyperthreading. And this trend is on a rise with 8 and 16 core processors already into the commercial market. But in-order to completely utilize the computing power of these machines, parallel applications and softwares should be developed, that they can completely utilize all the cores. This implies we need compilers and runtime environents that make parallel programming easier.

1 Past Research

My Doctoral thesis was done at Barcelona SuperComputing Center (BSC). It focussed on using STM to synchronize multiple threads in StarSs, a task-based programming Model. Synchronization of threads in a shared memory muti-core processor is necessary during the critical memory updates and to maintain sequential consistency of the parallel code. Speculative synchronization using Software Transactional Memory (STM) based approach was adopted instead of the traditional locks, mutexes and barriers to achieve concurrency control.

The first part of my thesis concentrated on integrating a Software Transactional Memory (STM) library into the StarSs framework. After that, I extended the StarSs framework with compiler directives and the associated runtime support for STM-based concurrency control among multiple threads. We ported applications such as NQueens and Specfem which update critical memory onto StarSs. A performance comparison between lock-based synchronization and STM-based concurrency control were performed. The results showed an increase in performance of applications wherein, the contention for locks were higher. Also the Optimistic nature of STM can exploit higher degrees of parallelism from an application.

In the second part of my thesis, we extended the speculative memory updates to speculative execution of tasks. I extended the StarSs framework such that tasks can be scheduled before their presence in the execution flow can be confirmed. The framework was extended with compiler directives that a programmer can use to annotate code blocks, where speculative task execution can improve performance. Runtime support was added so that the speculatively generated tasks can be executed as transactions whose updates are rolled back in case the speculation fails. To avoid the overhead associated with an STM-framework, such as conflict detection and concurrency control, I implemented a lighweight rollback mechanism which can be used to undo the updates of tasks. The idea of speculative task execution, wherein the tasks are scheduled even before their validity can be ascertained has given an average of 20% performance benefits. This is a highly positive result, since the performance improvement was achieved on an already parallelized code.

Additionally at BSC, I also worked on porting applications using SMPSs, StarSs implementation for SMPs onto Symmetric Multiprocessors (SMPs). I built applications that show the performance benefits, ease of use and portability of using the StarSs framework. The applications are a part of th StarSs application repository. My background in mathematics helps me develop novel ideas for parallelization of algorithms and efficient ways to implement them. In my Mtech thesis, I designed and implemented a Breadth First Search algorithm, that optimises the use of low memory available in the Synergistic Processing Ellement(SPE) of IBM's Cell.B.E processor. The strong point of the implementation was the design of a data structure to represent node. This data structure helped in increasing memory locality.

2 Current Research

After my graduation, I have continued my work in the field of HPC. My current work involves, parallelization of *MOOSE*, a software that simulates the behavior of neurons in a human brain. The cells in a human body interact via electrical and chemical exchanges between them. *Moose* simulates this interactions using linear solvers such as kinetic solver. The multi-scale characteristic of this project allows us to exploit both shared and distributed memory parallelism. Interactions between different cells can performed on different nodes. Each cell is divided into subcells and each of them are processed parallely by the multiple cores available

on each node. I have successfully implemented the shared-memory parallelism using OpenMP and Pthreads on the Kinetic solver. With this we achieved a speedup of 4.5X with 8 threads. Currently I am focussed on using MPI to parallelize the processing of various cells. The implementation is being done on the 64-node infiniband cluster available with the client.

3 Future Research