NVIDIA CUDA Architecture-Based Parallel SAT Solver

Master Project Proposal prepared by

Yandong Wang

Department of Computer Science
Rochester Institute of Technology
Rochester, NY, USA
Feb-18-2010

Project committee:

Chair: Alan Kaminsky
Reader: Stanisław Radziszowski
Observer: James Heliotis
Abstract

The SAT problem is the first NP-complete problem. So far there is no algorithm that can solve it in polynomial time. Over the past decade, the development of efficient and scalable algorithms has dramatically leveraged the ability of solving SAT problem instances involving tens of thousands of variables and millions of constraints. But as industry demand is increasing, a faster SAT solver is needed, so people begin to exploit parallel computing power. Recently, GPU has shown an appealing potential of parallel computing capability and its versatility allows it to be used for non-graphic algorithms. In addition, GPU manufacturers are actively improving the architecture of graphics cards to make them adapt to the general purpose parallel computation, to complete them with on-board shared memory and the general-purpose APIs for parallel programming. There is reason to believe that in the near future, GPU architecture might be able to be applied to problems where traditional multi-processors parallel architecture is being used. So, applying new GPU architecture to the SAT solver might bring a breakthrough in performance.

Key Words: SAT problem, SAT Solver, Parallel computing, GPU computing

1 Introduction

The SAT problem is the shorthand of the satisfiability problem which refers to the question that: Given a boolean expression, decide if there exists some assignments of TRUE or FALSE to variables that will make the entire expression true. Since the Cook-Levin theory proved in 1971 that the SAT problem is the NP-complete problem, this problem has puzzled numerous computer scientists from generation to generation, but there is still no algorithm that can solve it in polynomial time. So the aim of the SAT solver is to solve the SAT problems efficiently in reasonable time.

But why do we care about the performance of the SAT solver? First and foremost, there are real-world applications that depend on finding the solutions to the SAT problem such as electronic design automation, automatic test pattern generation, formal verification and artificial intelligence, all of which are the traditional domains to which SAT solvers have been applied. In addition, the SAT solvers have been successfully applied to cryptanalysis of hash functions. With the improvement of the performance of SAT solvers, more sophisticated hash functions have to be invented. Second, since all the NP problems can be reduced to the SAT problem in polynomial time, if there exists an efficient SAT solver that can solve SAT problems quickly, the same solution might be able to be applied to all other NP problems.

Over the past several decades, computer scientists have never stopped pursuing inventing better SAT solvers, and their performance is also constantly evolving. Recently, with the rise of the new GPU computation power, it is a novel computation model that attracts more and more focus from researchers; no doubt, it has a brilliant future. But can this new computation model be applied to
design parallel SAT solver and achieve satisfactory performance? This will be the main goal that my thesis is going to explore. Although GPU manufacturers have ameliorated the architecture of graphics cards to make them adapt to general purpose parallel computation, due to the complexity of programming in GPU architecture and its difference from the traditional multi-processors architecture, it still will be a challenge to implement a new parallel SAT solver based on the GPU architecture.

2 Project Plan

Problem Explanation: In my project, I will study the problem: Can NVIDIA CUDA architecture be applied to parallelize incomplete SAT Solver algorithm and achieve linear speedup? In order to investigate this problem, I plan to partition my work into four parts:

Prerequisite: In order to successfully finish my project, I need to arm myself with the knowledge of existing complete and incomplete SAT algorithms and potential algorithms that can be applied to SAT solvers. This will be the work that I call: "Learning Phase". In this phase, I will do deep research on three or four papers[10][9][5] that are related to the GPU architecture programming and incomplete SAT solvers. This phase needs to be done before the other three phases that I will mention next. The GPU architecture I will use in my project is "NVIDIA Tesla C870", which has 16 processors and 1.5 GB memory, and each processor has 8 cores.

Phase 1: First, I need to show NVIDIA CUDA architecture can be applied to implement a incomplete SAT solver algorithm. I call this phase: "Probing Phase". The objective of this phase is to investigate if it is feasible to implement a incomplete SAT solver on NVIDIA CUDA architecture regardless of the running time and the size of SAT problems. The test cases for this phase can be any size of SAT problems, but because this implementation probably will be very slow when it is being applied to large SAT problems, relative small SAT problems will be tested.

Phase 2: "Kernel Phase". In this phase, I will apply NVIDIA CUDA architecture to parallelize incomplete SAT solver based on my knowledge of incomplete SAT algorithms. The objective of this phase is to give out the concrete NVIDIA CUDA architecture-based parallel implementation of incomplete SAT solver and the measurement of different SAT benchmarks that I will collect from the previous SAT competitions[3] and SAT Live web site[2]. I will use different types and sizes of SAT problems for testing. The size of SAT problems will probably be between 360 and
26000 variables, they can be any K-SAT instances, though typically K will be 3, 5, or 7. "Probably SAT problem" and "Unknown SAT problem" will also be included in the test cases. In addition, I will measure these benchmarks by using different number of coprocessors to check what kind of speedup my implementation is able to achieve.

**Phase 3:** Once I successfully go through the above two phases, I will start the last phase which I call: "Analyzing Phase". The objective of this phase is to find the advantage and disadvantage of my entire work by comparing the data that I will collect from my measurement to the data that other parallel SAT solvers, such as ManySAT [6], have produced.

**Goal:** All in all, the goal of my project is to give a concrete implementation and measurement to demonstrate that NVIDIA CUDA architecture can be applied to parallelize the incomplete SAT solver, and this new parallel SAT solver can achieve ideal performance.

3 Deliverables

1. Program source code.  
   *This will be the C language implementation of parallel incomplete SAT solver algorithm which is based on NVIDIA CUDA architecture.*

2. A user’s manual for my program.  
   *This will include the information about how to configure the environment, how to run the program, input and output specification and so on.*

   *This will be the SAT benchmarks collected from previous SAT competitions[2].*

4. The analysis of measurement.  
   *This will be the support to approve or disapprove the problem that I will study in my project.*

5. Technical paper.  
   *This will be my project paperwork.*

4 Schedule


2. *Mar-26-2010 ~ Apr-5-2010* Probing Phase
3. **Apr-6-2010 ~ May-5-2010 Kernel Phase**
   - **Apr-6-2010 ~ Apr-30-2010 Algorithm Implementation**
   - **May-1-2010 ~ May-5-2010 Algorithm Measurement**

4. **May-6-2010 ~ May-10-2010 Analyzing Phase**

5. **May-11-2010 Submission**

6. **May-30-2010 Defense**

**References**


