

THE IMPACT OF IRREGULARITY ON EFFICIENT LARGE-SCALE INTEGER-INTENSIVE COMPUTING

By

Akintayo Holder

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: COMPUTER SCIENCE

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Dr. Christopher Carothers, Thesis Adviser

Dr. Yannick L. Le Coz, Member

Dr. Boleslaw Szymanski, Member

Dr. Mohammed Zaki, Member

Rensselaer Polytechnic Institute
Troy, New York

June 2011
(For Graduation August 2011)

ABSTRACT

Recently, the growth in computing power has been driven by ever increasing levels of parallelism. In supercomputing, this has led to computers with tens of thousands, if not hundreds of thousands, of processors. However, developing large-scale applications that support tens of thousands of processors is a difficult task. This investigation seeks to develop efficient large-scale algorithms for integer-intensive problems. Integer-intensive computing is interesting because these applications dominate computing, but are overlooked in supercomputing. They may also be harder to parallelise because they have fewer heavy floating point operations. Irregularity also limits parallel performance, because irregularity makes it difficult to figure out how to decompose the problem into concurrent tasks.

This thesis claims that a balanced large-scale platform can mitigate the impact of irregularity and allow these applications to scale efficiently. This investigation looks at two instances of irregularity, and how they can be implemented on large-scale computers. This investigation looks at temporal irregularity, an uncertainty in the control flow, and structural irregularity, which is derived from irregular data objects.

The first contribution of this thesis is to demonstrate that temporal irregularity does not prevent applications from running efficiently on large-scale computers. Speculation usually complicates the control flow due to the detection of and recovery from speculative errors, which causes temporal irregularity. This is demonstrated using Time Warp, an optimistic parallel discrete event simulation protocol. This work shows that Time Warp is able to scale to tens of thousands of processors, while retaining the efficiency of a fast shared memory simulator.

Large-scale Time Warp implementations have been limited by difficulty in efficiently synchronising the execution of events across many processors. Using the balanced IBM Blue Gene/L supercomputer, the ROSS implementation described in this thesis demonstrated that a GVT algorithm could be efficiently implemented over the supercomputer's fast collective networks. To the best of the author's knowledge,

the 2.47 billion committed events per second achieved with the PCS model was the first published multi-billion event rate when running a Time Warp kernel on a IBM Blue Gene supercomputer. The 853 million committed events per second achieved with the PHOLD benchmark, was 1.5 times better than the published results for any parallel discrete event simulation synchronisation protocol.

ROSS was improved to address the poor scalability of models with large remote event populations. This was achieved by adding flow control and improving the performance of remote event handling. These changes allowed the PHOLD model to achieve 4 billion committed events per second with a 100 percent remote event population, and 12.26 billion events per seconds with a 10 percent remote event population. To the best of the author's knowledge, this was the first time a Time Warp kernel achieved this level of performance with these PHOLD configurations.

The second contribution is to demonstrate how structural irregularity affects efficient execution on large-scale computers. Structural irregularity is due to unstructured data objects, these objects have a flexible construction that makes them difficult to partition, which in turn complicates problem decomposition. The example used is Static Timing Analysis, a part of a semi-conductor chip design and analysis work flow. Static timing analysis ensures that a circuit design meets its timing constraints and the structure of the circuit being analysed determines the behaviour of the application. Parallel static timing analysis faces the challenge of managing an irregular, closely coupled communication pattern. This prototype demonstrated that interleaving communication and computation helped the timing algorithm to scale with a round robin partitioning algorithm. To the best of the author's knowledge, the static timing analysis prototype described in this thesis was the first result demonstrating scalable STA on the IBM Blue Gene. The prototype achieved about 300 times speedup with two, three million node benchmark circuits, while a billion node circuit showed twelve times speedup as the core count increased from 1024 to 16,384. This work shows that static timing analysis may be able to scale to tens of thousands of processors, if the circuits were large enough.