國 立 清 華 大 學

資訊工程研究所

碩士論文

硬體線程級投機效能分析

Hardware Thread-Level Speculation Performance Analysis

研究生：102062557 王瀅捷（Ying-Chieh Wang）

指導教授：李哲榮 教授（Prof. Che-Rung Lee）

中華民國一零四年一月

Hardware Thread-Level Speculation Performance Analysis

Student: Ying-Chieh Wang
Advisor: Prof. Che-Rung Lee

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, 30013, R.O.C.

January 2015

中文摘要

線程級投機是一種平行的架構。線程級投機可以免除直接編譯平行程式時所需的問題分析，且這對於程式開發者建立平行程式是很有幫助的。然而，對於平行程式來說，效能才是最重要的問題。因此，我們分析 IBM Blue Gene/Q 電腦上硬體線程級投機的效能。

此篇論文會展示在 IBM Blue Gene/Q 電腦上硬體線程級投機的效能模型。此模型有很好的效能預測，也經由實驗驗證過。這個模型能夠幫助我們了解利用特殊目的的線程級投機可以讓需要以單一執行來避免記憶體衝突的程式得到多少潛在的效能加速。基於分析和測量線程級投機的運作和成本，我們推出一個能夠幫助我們利用這個硬體的特色的策略。除此之外，我們比較了硬體線程級投機和 OpenMP 的效能。基於效能分析，我們給了一個方向，幫助我們決定硬體線程級投機和 OpenMP 哪個效能較好。這個結果不但可以幫助使用者利用程級投機，同時也能提供未來線程級投機架構設計改進的方向。

Abstract

Thread-Level Speculation (TLS) is one of the parallel frameworks. TLS can avoid the analysis problem of compiler-directed code parallelization and this is helpful for programmers to generate parallel programs. However, the performance is the most important issue for parallel programs. Therefore, we analyse the performance of hardware Thread-Level Speculation (TLS) in the IBM Blue Gene/Q computer.

This paper presents a performance model for hardware Thread-Level Speculation (TLS) in the IBM Blue Gene/Q computer. The model shows good performance prediction, as verified by the experiments. The model helps to understand potential gains from using special purpose TLS hardware to accelerate the performance of codes that, in a strict sense, require serial processing to avoid memory conflicts. Based on analysis and measurements of the TLS behavior and its overhead, a strategy is proposed to help utilize this hardware feature. Furthermore, we compare the performance of hardware Thread-Level Speculation and OpenMP. Based on the performance analysis, we give a direction for deciding between this two parallel frameworks. And the results can not only help users to utilize the TLS but also suggest potential improvement for the future TLS architectural designs.

Acknowledgements

There are many people giving assistance on my thesis work. I am very grateful to each of them. Most of all, I would like to show my deepest gratitude to my advisor, Prof. Che-Rung Lee. Prof. Lee teaches me how to define problems, map out schedule of validation, analyse the result of experiments and do great presentation. While discussing problems with Prof. Lee, I learned to use mathematical method to solve problems and proofed the results. In addition, Prof. Lee always helps me find the right direction to overcome difficulties. Therefore, I got more progress in study and also be more delighted on doing research. Except researching, Prof. Lee provided me the chance to train the junior students to attend Student Cluster Competition. I learned a lot and been inspired while I was teaching and discussing with them.

I would like to express thanks to Dr. I-Hsin Chung, a cooperative partner in IBM, he gave me advises on project and help me learn professional knowledge about my research. Except working, Dr. Chung also teached me the importance of relationship and provided me a chance to get a job in IBM. And My lab mates also support me on my way of research during the two years. They gave me favors anytime I needed helps. I really appreciate scope lab.

My family always encourage me to achieve my best. I am very grateful to my parents and my brother. Thanks to all the people around me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Chapter 2

Background

Now we do face a revolution in data processing as internet information size that we can collect grows quickly, currently there are many solutions developed to analyze and mine valuable treasure in diverse and useless garbage data. Hadoop [6] and MapReduce [3] are famous and effective solutions, they give programmer user-friendly and compact batch processing module.

Mapreduce mainly involves batch processing technique on shared and parallel cluster meaning it works statically and event triggered, But MapReduce cannot provide low latency response when it process continous and real-time unbounded data stream [2]. To overcome this disadvantage, several stream processing system have been proposed, e.g. Storm [1] and S4 [5]. The abvious difference between MapReduce and Storm is that a MapReduce work will finally finish, while Storm work will keep on working until be stoped by user.

Storm

Storm attributes distributed realtime stream processing framework with scalable, reliable and fault tolerant computation, it can act excellently on many use cases like: realtime analytics, online machine learning, ETL, and more. For these surprising features, Many companies such as Twitter choose Storm as data processing System.

Components of Storm

A Storm cluster is superficaly similar to a Hadoop cluster, there are two kinds of nodes on a Storm clster: the master node and worker nodes. Storm executes a daemon called *Nimbus* on master node as Hadoop runs *JobTracker* on it, Nimbus mainly manages task assignment, work code dirtribution and failure recovery on cluster. There will be a daemon called *Supervisor* on each worker node, Supervisor's mission is listening job assignment from Nimbus and manages worker processes on it for assigned works. To achieve coordination and fault tolerancy, Storm use ZooKeeper [4] as communication interface between Nimbus and Supervisors, which is illustrated in Fig 2.1. Additionally, Nimbus and Supervisor daemon are fail-fast and stateless; all Strom daemon state is kept in Zookeeper or local disk. It means if Nimbus or Supervisor are broken, they can be start up like nothing happened.

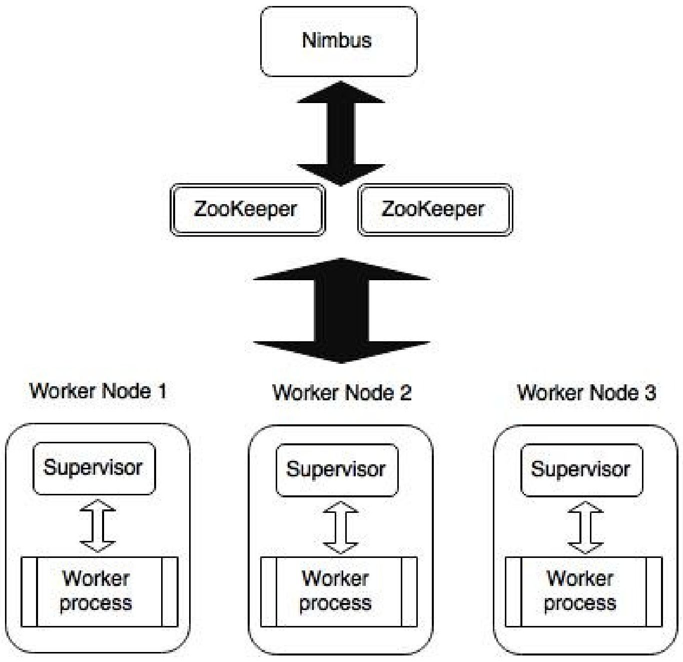

Figure 2.1: the physical views of Storm

Storm Topology

Although MapReduce and Storm are similiar physically, there are still some differences between them in logical view. For example, we use *MapReduce job* to describe MapReduce work, but in Storm it use another name *topology* as alias for Storm work. A standard Storm topology is described as a directed acyclic graph which usually includes two kinds of components: *spout* and *bolt*. Spout is a source of data stream in topology, spout can generate data itself or receive it from

Chapter 3

Conclusion

This is conclusion.

Bibliography

[1] Leonardo Aniello, Roberto Baldoni, and Leonardo Querzoni. Adaptive online scheduling in storm. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, pages 207–218, New York, NY, USA, 2013. ACM.

[2] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM.

[3] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[4] Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, and Benjamin Reed. Zookeeper: Wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[5] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 170–177, Dec 2010.

[6] K. Shvachko, Hairong Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10, May 2010.