國 立 清 華 大 學

資訊工程研究所

碩士論文

Docker容器在swarm叢集中的遷移

Docker containers migration in Docker swarm

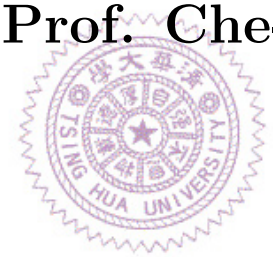
研究生：103062622 黃晟豪（Cheng-Hao Huang）

指導教授：李哲榮 教授（Prof. Che-Rung Lee）

中華民國一零五年七月

Docker containers migration in Docker swarm

Student: Cheng-Hao Huang
Advisor: Prof. Che-Rung Lee



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

July 2016

中文摘要

容器技術自 2013 年 Docker 發表後在全世界迅速竄紅，Container 解決了維護人員在伺服器進行大量部屬時的痛點，使得環境部屬只需要建立完容器映像檔後就可以進行大量部屬，並對每一個容器環境進行隔離。

在這篇論文中，我們提出了在 Docker swarm 叢集中，將容器在多個節點中相互搬移。另外，可以針對特定的容器定期設定 checkpoint 儲存至雲端儲存空間，若叢集中的節點遇到不正常的離線時，可以及時回復最近的容器狀態到健康的節點上。

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

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This is Chapter **??**, introduction.

Chapter 2

Background

2.1 Docker

Docker is a open-source project container engine. It provides an additional layer of abstraction and automation of operating-system-level virtualization on Linux. Docker engine include Docker client and Docker daemon.

2.1.1 Docker client

Docker is typical Client/Server architecture application. Docker client uses Docker command to send and receive requests to Docker daemon. Also, Docker supports remote RESTful API to send and receive HTTP requests to Docker daemon, it has been implemented by more than 10 programming languages.

2.1.2 Docker daemon

Docker daemon is a daemon that runs as system service. It has two the most importance features:

- receive and handle Docker client's requests.
- manage containers.

When docker daemon is running, it will run a server that receives requests from Docker clients or remote RESTful API. After receives requests, server will pass requests by router to find handler to handle the requests.

2.2 Docker Swarm

Docker Swarm is native clustering for Docker. It gathers several docker engines together into one virtual docker engine. Docker Swarm serves standard Docker API, so it can be connected by Dokku, Docker Machine, Docker Compose, Jenkins, DockerUI, Drone, etc. And it also support Docker client of course.

2.2.1 Discovery services

2.2.2 Advanced Scheduling

2.2.3 High availability

2.3 criu

?? and cite test [?, ?, ?].

Chapter 3

Design and Implementation

3.1 Design

3.1.1 Docker client

3.1.2 Docker daemon

3.1.3 Docker container migration

3.2 Docker Swarm rescheduling policy

Chapter 4

Conclusion

This is conclusion.