Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Τίτλος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

MARIOS

Επιβλέπων : test
test

Αθήνα, Ιανουάριος 1111

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Τίτλος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

MARIOS

Επιβλέπων : test
test

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Ιανουαρίου 1111.

.....	.....	.....
test test	test test	test test

Αθήνα, Ιανουάριος 1111

marios

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Λέξεις κλειδιά

Abstract

Key words

Ευχαριστίες

marios,

Αθήνα, 1η Ιανουαρίου 1111

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

When back in April 1965 Gordon E. Moore stated the following

> "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer."[3]

had no idea that he had actually started a race among the academia and the industry to overcome or at least abide the this law.

At first, since the technology was premature, the evolution in VLSI technology went hand in hand with the evolution in computer architecture. The more and faster transistors resulted in achievements in instruction level parallelism (ILP). From 1975 to 2005 the endeavour put in computer architecture resulted in technological advances varying from deeper pipelines and faster clock speeds to superscalar architectures. But in around 2005 the ILP wall was hit. Transistors could not be utilized to increase serial performance, logic became too complex and performance attained was very low compared to power consumption. This lead to the creation of multicore systems and entered the programmers to the jungle of parallel software. So far the evolution was almost in accordance with the famous law. However, in around 2009 to 2011, it was the power wall's time to be hit. The famous power equation $P = cV^2 f$ along with the CPU to memory gap (eikona) led to the technological burst of distributed and cloud computing.

In 2009 Amazon.com introduced the Elastic Compute Cloud and since then the term 'cloud' is one of the hottest buzzwords not only among the industry and academia but also among everyday people that take advantage of the 'power of cloud'. Although the term may be vague, the definition of cloud computing, according to NIST (National Institute of Standards and Technology), is the following:

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics ,three service models, and four deployment models."[4]

In the previous brief computer chronology, I kept describing bottlenecks and walls to be overcome. However, it not clear how these bottlenecks become obvious and how scientists can be sure that they have reached one's technology's limits before moving on to the next one. The answer to the previous questions has always been given through tracing. Tracing is a process recording information about

a program's execution, while it is being executed. These information may be low level metrics like performance counters or time specific metrics in order to evaluate system's latencies and throughput. Tracing data are mostly useful for developers and can be used for debugging, performance tuning and performance evaluation. From the single-cpu, integrated computer to the hundreds-node cloud infrastructure, trace and performance engineers face challenging problems that vary from platform to platform, but in any case play a vital role the system's design and implementation.

Cloud and distributed computing provided trace engineers with more challenging problems. The system scale is now much greater and program execution is far from deterministic and can take place in any cluster node. So each program execution is not bounded to a specific context. Other problems that needed solving was data and time correlation between the different computing nodes. Also, unlike single chip platforms that can be individually traced and evaluated, cloud infrastructures need to be traced with full-load under production conditions. This set more restrictions concerning the overhead that tracing adds to the application. Finally, tracing is notorious about the amount of data that produces. So distributed and cloud tracing demands the use of distributed data storage systems and processing methods like distributed NOSQL databases and Map-Reduce frameworks.

So to sum up, as described by any design model, the system verification consists a major part of a system's implementation and working process. Verification is achieved through monitoring and tracing. Depending on the system's nature tracing and monitoring process and the tools used may vary. Picking the right tracing tools that will reveal the system's vulnerabilities and faults can be very demanding and the performance engineer for bringing them to light, respecting all the prerequisites set by the system.

1.1 Thesis motivation

The motivation behind this thesis emerged from concerns about the storage performance of the Synnefo [1] cloud software, which powers the ~**okeanos** [2] public cloud service [1]. I will briefly explain what ~**okeanos** and Synnefo are in the following paragraphs.

~**okeanos** is an IaaS (Infrastructure as a Service) that provides Virtual Machines, Virtual Networks and Storage services to the Greek Academic and Research community. It is an open-source service that has been running in production servers since 2011 by GRNET S.A. [3]

Synnefo [2] is a cloud software stack, also created by GRNET S.A., that implements the following services which are used by ~**okeanos** :

- *Compute Service*, which is the service that enables the creation and management of Virtual Machines.
- *Network Service*, which is the service that provides network management, creation and transparent support of various network configurations.
- *Storage Service*, which is the service responsible for provisioning the VM volumes and storing user data.
- *Image Service*, which is the service that handles the customization and the deployment of OS images.

[1] www.synnefo.org/
[2] https://okeanos.grnet.gr/
[3] Greek Research and Technology Network, https://www.grnet.gr/

- *Identity Service*, which is the service that is responsible for user authentication and management, as well as for managing the various quota and projects of the users.

Synnefo provides each virtual machine with at least one virtual volume provisioned by the Volume Service called Archipelago[**?**] and will be furthered detailed in Chapter . This thesis' purpose is to provide the developer or the system administrations with a cross-layer representation accompanied with the equivalent metrics and time information of an I/O request's route within the infrastructure from the time it is created inside the virtual machine till it is finally served by the storage backend. The design and implementation has to be done respecting the following two prerequisites:

- The tracing information should be gathered and processed in real-time from every node participating in the request serving.
- The tracing infrastructure should add the least possible overhead to the instrumented system, which should continued working properly production-wise

After the end of the tracing infrastructure implementation, the developer should be able to identify the distinct phases and the duration of each that an IO request passes through, measure communication latencies between the different layers and collect all the necessary information (chosen by him) that would help him understand the full context under which this specific request was served. All these information can be used for software faults detection and performance tuning as well as hardware malfunctions and faults like disk or network failures that would be difficult to detect otherwise.

The novelty of this thesis consists in combining live cross-layer, multi-node data aggregation, which is typical for monitoring but not for tracing, with the precision and accuracy of tracing, respecting a hard prerequisite of low overhead. Previous tracing infrastructures offered only partial solutions. Some of them would separate the tracing from the working phase because of the great added overhead, others provided no mechanism for data correlation, while the traditional monitoring systems did not meet our low-level tracing needs.

The proposed system is called *B*lkKin. It is designed respected the aforementioned prerequisites and make use of the latest tracing semantics and infrastructures employed by great tech companies like Google and Twitter.

1.2 Thesis structure

This thesis is structured as follows:

Bibliography

[1] Vangelis Koukis, Constantinos Venetsanopoulos, and Nectarios Koziris. okeanos: Building a cloud, cluster by cluster. IEEE Internet Computing, 17(3):67–71, May 2013.

[2] Vangelis Koukis, Constantinos Venetsanopoulos, and Nectarios Koziris. Synnefo: A complete cloud stack over ganeti. ;login, 38(5):6–10, October 2013.

[3] Gordon E. Moore. Cramming more components onto integrated circuits. In Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, editors, Readings in computer architecture, pages 56–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[4] National Institute of Standards and Technology. The NIST definition of cloud computing. Special Publication 800-145, September 2011.