ΠΡΟΣ

1) **Όλα τα μέλη ΔΕΠ του Τμήματος Επιστήμης Υπολογιστών**

2) **Τους εκπροσώπους των Μεταπτυχιακών φοιτητών του Τμήματος Επιστήμης Υπολογιστών**

3) **Την Επταμελή Εξεταστική Επιτροπή**

4) **Όλα τα μέλη της Πανεπιστημιακής Κοινότητας**

Πρόσκληση σε Δημόσια Παρουσίαση της Διδακτορικής Διατριβής του

κ. Ζακκάκ Φοίβου

Την Τρίτη, 25 Οκτωβρίου 2016 και ώρα 16:30 στην αίθουσα Τηλεδιάσκεψης Κ206 του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου Κρήτης στο Ηράκλειο, θα γίνει η δημόσια παρουσίαση και υποστήριξη της Διδακτορικής Διατριβής του υποψηφίου διδάκτορος του Τμήματος Επιστήμης Υπολογιστών κ. Ζακκάκ Φοίβου με θέμα:

“Η Java™ σε Κλιμακώσιμες Αρχιτεκτονικές Μνημών ”

“Java™ on Scalable Memory Architectures”

ΠΕΡΙΛΗΨΗ

Η αρχή της χιλιετίας που διανύουμε σήμανε την ανάγκη αναζήτησης νέων μεθόδων για τη συνέχιση της βελτίωσης της απόδοσης των επεξεργαστών. Η μέθοδος της αύξησης της συχνότητας των επεξεργαστών καθώς τα τρανζίστορ γίνονται ολοένα και μικρότερα δεν είναι πλέον εφικτή, λόγο της ολοένα αυξανόμενης διαρροής ρεύματος και παραγωγής θερμότητας. Στην προσπάθεια τους να διατηρήσουν τους ρυθμούς βελτίωσης της απόδοσης των επεξεργαστών, οι κατασκευαστές επεξεργαστών στράφηκαν προς τον σχεδιασμό πολυπύρηνων επεξεργαστών. Με αυτό τον τρόπο η βιομηχανία εξακολουθεί να εκμεταλλεύεται το ολοένα μικρότερο μέγεθος των

τρανζίστορ, περιλαμβάνοντας περισσότερα επεξεργαστικά στοιχεία στον ίδιο χώρο αντί να αυξάνουν την πολυπλοκότητα των κυκλωμάτων και την συχνότητα εναλλαγής καταστάσεων. Καθώς όμως το πλήθος των πυρήνων αυξάνεται, εμφανίζονται νέες προκλήσεις. Μία τέτοια πρόκληση αφορά την παροχή συνέπειας μνήμης. Η διαχείριση των κρυφών μνημών και η διατήρηση συνεπών αντιγραφών σε αυτές γίνεται ολοένα και πιο περίπλοκη και ενεργειακά μη αποδοτική καθώς το πλήθος των πυρήνων αυξάνεται. Για να αντιμετωπίσουν αυτό το πρόβλημα, οι αρχιτέκτονες υπολογιστών πειραματίζονται με νέες αρχιτεκτονικές που παρέχουν συνέπεια μνήμης μόνο μεταξύ ενός υποσυνόλου των κρυφών μνημών ή και καθόλου. Αυτές οι αρχιτεκτονικές αναθέτουν τη διαχείριση μνήμης στο λογισμικό.

Στη διατριβή αυτή μελετάμε πως μπορούν να τρέξουν γλώσσες προγραμματισμού υψηλού επιπέδου σε τέτοιες αρχιτεκτονικές. Οι γλώσσες προγραμματισμού υψηλού επιπέδου, όπως η Java, είναι σχεδιασμένες να αποκρύπτουν τις λεπτομέρειες της αρχιτεκτονικής από τους προγραμματιστές. Με αυτόν τον τρόπο επιτρέπουν στους προγραμματιστές να επικεντρωθούν στην υλοποίηση των αλγορίθμων τους και όχι στην διαχείριση της εκάστοτε αρχιτεκτονικής. Οι γλώσσες προγραμματισμού υψηλού επιπέδου βασίζονται σε εικονικές μηχανές για να παρέχουν τα ίδια αποτελέσματα σε διαφορετικές αρχιτεκτονικές. Στην εργασία μας επικεντρωνόμαστε στην εικονική μηχανή της Java. Η εικονική μηχανή της Java είναι μία από τις πιο δημοφιλείς και πολυμελετημένες εικονικές μηχανές η οποία υλοποιεί δεκάδες γλώσσες προγραμματισμού, από τις οποίες ξεχωρίζουν κυρίως η Java και η Scala.

Οι υλοποιήσεις εικονικών μηχανών Java πρέπει να συμμορφώνονται στον ορισμό της γλώσσας προγραμματισμού Java (Java language specification) και στο μοντέλο μνήμης της (Java Memory Model). Στην εργασία αυτή μελετάμε το μοντέλο μνήμης της Java και παρουσιάζουμε μία επέκταση του που εκφράζει ρητά τις μεταφορές δεδομένων μεταξύ των κρυφών μνημών. Η επέκταση αυτή παρέχει ρητούς κανόνες που αφορούν τις μεταφορές δεδομένων μεταξύ των κρυφών μνημών οι οποίοι διευκολύνουν την επιχειρηματολογία σχετικά με την συμμόρφωση μίας εικονικής μηχανής Java στο μοντέλο μνήμης της Java. Ακόμη αποδεικνύουμε ότι η επέκταση αυτή είναι συμβατή με το αρχικό μοντέλο μνήμης της Java και επιτρέπει τις ίδιες βελτιστοποιήσεις κώδικα.

Επιπροσθέτως σχεδιάζουμε μία εικονική μηχανή Java για αρχιτεκτονικές με μερική ή μηδενική συνέπεια μνήμης. Ο σχεδιασμός μας στοχεύει στην ελαχιστοποίηση των μεταφορών δεδομένων και ανταλλαγών μηνυμάτων που χρειάζονται για τη συμμόρφωση στο μοντέλο μνήμης της Java. Ο σχεδιασμός μας εκμεταλλεύεται αρχιτεκτονικές με μερική συνέπεια μνήμης χρησιμοποιώντας μερικές κοινές δομές μεταξύ πυρήνων που βρίσκονται στην ίδια συστάδα πυρήνων με συνέπεια μνήμης. Βασιζόμενοι στον σχεδιασμό αυτό υλοποιούμε μία εικονική μηχανή Java και την αξιολογούμε τρέχοντας πειράματα σε έναν εξομοιωτή αρχιτεκτονικής χωρίς συνέπεια

μνήμης. Τα αποτελέσματα δείχνουν ότι η υλοποίηση μας κλιμακώνεται μέχρι και σε 500 πυρήνες και η κλιμακωσιμότητα της είναι συγκρίσιμη με αυτή της πιο πρόσφατης εικονικής μηχανής Java (HotSpot VM) σε μία αρχιτεκτονική με συνέπεια μνήμης.

Τέλος εκφράζουμε με μαθηματικό τρόπο την υλοποίησή μας και αποδεικνύουμε ότι παράγει μόνο σωστές εκτελέσεις, σύμφωνα με το μοντέλο μνήμης και τον ορισμό της γλώσσας.

Επόπτης Διδακτορικής Διατριβής: Καθηγητής Άγγελος Μπίλας

ABSTRACT

The beginning of the new millennium signaled the need for new ways to keep improving the performance of processors. The approach of increasing the frequency as transistors got smaller and smaller is no longer viable due to the increased power-leakage and heat generation. In an effort to maintain the performance increase steady despite the difficulties, processor manufacturers turned to multi-core computing. This way the industry keeps taking advantage of the smaller transistors by packing more compute elements in the same area, rather than making the circuits more complex and increasing the frequency. However, as the number of cores increases new challenges come up. One such challenge regards cache-coherence. Managing hardware caches and keeping the data coherent across them is become more and more complex and energy inefficient as the number of cores grow. To tackle this issue hardware architects are experimenting with modular non cache coherent and partially coherent architectures. Such architectures delegate the memory coherency to the software.

In this thesis we study how high productivity languages can be run efficiently on such architectures. High productivity languages, like Java, are designed to abstract away the hardware details and allow developers to focus on the implementation of their algorithm. Such programming languages rely on process virtual machines, like the Java virtual machine, to provide consistent behavior across different platforms. We focus our work on the Java virtual machine since it is one of the most popular and widely studied process virtual machines on top of which tens of languages are being implemented, with the most distinguishable being Java and Scala.

Java virtual machine implementations need to adhere to the Java language specifications and the Java memory model. In this thesis we study the Java memory model and present an extension of it that exposes explicit memory transfers between

caches. This extension eases the process of arguing about the adherence of a Java virtual machine targeting a non cache coherent architecture by providing explicit rules regarding the ordering of memory transfers in respect to other operations in an Java execution. We prove that our extension is complies to the original Java memory model and allows the same optimizations.

We present a Java virtual machine design targeting non cache coherent and partially coherent architectures. Our design aims to minimize the number of memory transfers and messages exchanged while still adhering to the Java memory model. Our design also takes advantage of partial coherence by sharing some structures across different cores on the same coherence island. Based on our design we implement a Java virtual machine and evaluate it on an emulator of a non cache coherent architecture. The results show that our implementation scales up to 500 of cores and its scalability is comparable to that of the state of the art Java virtual machine running on a cache-coherent architecture.

Last but not least we model our implementation in the operational semantics of a Java core calculus that we define for this purpose. We then prove that these operational semantics produce only well-formed executions according to the Java memory model. Since the operational semantics model our implementation we argue that the latter also produces only well-formed executions, thus it adheres to the Java memory model.

Supervisor: Professor Aggelos Bilas