DISS. ETH No. 22006

Modeling Memory System Performance of NUMA Multicore-Multiprocessors

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
ZOLTÁN MAJÓ
Ing. dipl., Technical University of Cluj-Napoca
born on June 18, 1983
citizen of Hungary and Romania

accepted on the recommendation of
Prof. Dr. Thomas R. Gross, examiner
Prof. Dr. Frank Müller, co-examiner
Prof. Dr. Michael Stumm, co-examiner

2014

Abstract

The performance of many applications depends closely on the way they interact with the computer's memory system: Many applications obtain good performance only if they utilize the memory system efficiently.

Unfortunately, obtaining good memory system performance is often difficult, as developing memory system-aware (system) software requires a thorough and detailed understanding of both the characteristics of the memory system and of the interaction of applications with the memory system. Moreover, the design of memory systems evolves as newer processor generations appear on the market, thus the problem of software–hardware interaction must be revisited to understand the interaction of (already existing) software with newer memory system designs as well.

This thesis investigates the memory system performance of a recent class of machines, multicore-multiprocessors with a non-uniform memory architecture (NUMA). A NUMA multicore-multiprocessor system consists of several processors where each processor integrates multiple cores. Typically, cores of a multicore processor share resources (e.g., last-level caches) and contention for these shared resources can result in significant performance degradations.

NUMA multicore-multiprocessors are shared-memory computers, but the memory space of a NUMA multicore-multiprocessor system is partitioned between processors. Accessing the memory of a local processor takes less time than accessing the memory of other (remote) processors, therefore data locality (a low number of remote memory accesses) is critical for good performance on NUMA machines.

This thesis presents a performance-oriented model for NUMA multicore-multiprocessors. The model considers two application classes, multiprogrammed workloads (workloads that consist of multiple, independent processes) and multithreaded programs (programs that consist of a number of threads that operate in a shared address space). The thesis presents an experimental analysis of memory system bottlenecks experienced by each application class. Moreover, the thesis presents techniques to reduce the performance-degrading effects of these bottlenecks.

We determine (based on experimental analysis) that the performance of multiprogrammed workloads depends on both multicore-specific and NUMA-specific aspects of a NUMA multicore-multiprocessor's memory system. Therefore, a process scheduler must find a balance between reducing cache contention and improving data locality; the N-MASS scheduler presented by the thesis attempts to strike a balance between these, sometimes contradicting, goals. N-MASS improves performance up to 32% over the default setup in current Linux implementations on a recent 2-processor 8-core machine.

Based also on experimental analysis we find that data locality is of prime importance for the performance of multithreaded programs. The thesis presents extensions to two popular parallel programming frameworks, OpenMP and Intel's Threading Building Blocks. The extensions

allow a programmer to express affinity of data and computations, which, if done appropriately, helps to improve data locality and thus performance on NUMA multicore-multiprocessors (by up to 220% on a recent 4-processor 32-core machine). The thesis also shows that adding NUMA support not only to the programmer interface, but also to the underlying runtime system, allows programs to be portable across different architectures as well as to be composable with other programs (that use the same runtime system).

Zusammenfassung

Die Rechenleistung vieler Software-Applikationen hängt von der Interaktion der Applikation mit dem Speichersystem des Computers ab: Viele Applikationen erreichen eine gute Rechenleistung nur wenn sie das Speichersystem des Computers effizient nutzen.

Es ist aber leider oft schwierig Software zu entwickeln, die das Speichersystem effizient benützt, da Programmierer sowohl die Merkmale des Speichersystems wie auch das Zusammenspiel der Software mit dem Speichersystem verstehen müssen, um effiziente Software entwickeln zu können. Darüber hinaus muss häufig das Zusammenspiel (schon existierender) Software erneut analysiert and verstanden werden, wenn neue Computerarchitekturen (eventuell mit einem neuen Typ von Speichersystem) auf den Markt gebracht werden.

Diese Dissertation analysiert das Speichersystem einer neuen Klasse von Rechnern, Multikern-Multiprozessoren mit einer nicht-uniformen Speicherarchitektur (engl.: non-uniform memory architecture (NUMA)). Ein NUMA Multikern-Multiprozessor besteht aus mehreren Prozessoren; jeder Prozessor des Systems besteht aus mehreren Kernen. Die Kerne eines Multikernprozessors teilen in der Regel Ressourcen (z.B. den Cachespeicher des Prozessors) und der gleichzeitige Gebrauch von geteilten Ressourcen kann zu einer Erhöhung der Laufzeit von Applikationen führen (im Vergleich mit dem Fall wenn keine Ressourcen geteilt sind).

Jeder Prozessor eines NUMA Multikern-Multiprozessors hat Zugriff auf alle Speicherstellen des Systems, der Adressraum des Systems ist aber zwischen den Prozessoren partizioniert. Zugriffe auf Speicherstellen des lokalen Prozessors dauern weniger lang als Zugriffe auf Speicherstellen eines entfernten Prozessors, daher ist Datenlokalität (eine niedrige Zahl von Zugriffen auf die Speicherstellen eines entfernten Prozessors) entscheidend für die Rechenleistung vieler Applikationen.

Diese Dissertation beschreibt ein leistungsorientiertes Modell für NUMA Multikern-Multiprozessoren. Das Modell betrachtet zwei Klassen von Software-Applikationen, multiprogrammierte Applikationen, welche aus mehreren unabhängigen Prozessen bestehen, und multithreaded Applikationen, welche aus mehreren Threads bestehen, die miteinander Daten teilen. Die Dissertation identifiziert Engpässe des Speichersystems, welche die Laufzeit von Applikationen beider Klassen negativ beinflussen. Die Dissertation beschreibt auch Methoden um die negativen Auswirkungen der Engpässe zu reduzieren.

Wir stellen fest (mittels experimenteller Analyse), dass sowohl Datenlokalität als auch die gleichzeitige Benutzung geteilter Ressourcen für die Laufzeit multiprogrammierter Applikationen entscheidend ist, und dass der Scheduler des Betriebsystem eine Balance zwischen den beiden, oft miteinander im Konflikt stehenden Faktoren, finden muss. Die Dissertation beschreibt einen neuen Scheduler-Algorithmus, N-MASS. Die Verwendung von N-MASS ergibt eine Verbesserung der Laufzeit multiprogrammierter Applikationen von bis zu 32% (verglichen mit einer Standard Linux-Implementation auf einem modernen NUMA Multikern-Multiprozessor

mit zwei Prozessoren und 8 Kernen).

Ausserdem stellen wir fest (auch mittels experimenteller Analyse), dass Datenlokalität für eine effiziente Ausführung von multithreaded Applikationen unerlässlich ist. Die Dissertation präsentiert Erweiterungen für zwei bekannte Frameworks für parallelle Progammierung, OpenMP und Intel Threading Building Blocks. Mit diesen Erweiterungen können Programmierer die Affinität von Daten und Berechnungen ausdrücken, was, wenn dies in geeigneter Weise getan wird, dazu führt, dass Datenlokalität und somit auch Rechenleistung sich deutlich (bis zu 220% auf einem Rechner mit vier Prozessoren mit 32 Kernen) verbessert. Die Dissertation zeigt auch, dass, wenn nicht nur die Programmierschnittstelle sondern auch das Laufzeitsystem für NUMA Multikern-Multiprozessoren angepasst wird, optimierte Programme portabel sind und ihre Datenlokalität auch dann bewahren, wenn sie mit anderen Programmen zusammengesetzt werden.