Moving Execution - Motion

Sascha Uhrig

Fakultät für Elektrotechnik und Informationstechnik
TU Dortmund
sascha.uhrig@tu-dortmund.de

Abstract: Das Moving Execution Projekt erforscht eine neuartige Technik Programme auf einem Many-Core Prozessor auszuführen, sowie eine hierzu passende Prozessorarchitektur. Ziel ist es, die Kommunikationslast zwischen den Prozessorknoten und zwischen Knoten und Speicher zu minimieren und dadurch die Leistung solcher Systeme zu optimieren. Im Vordergrund steht hierbei die Vermeidung von Optimierungswerkzeugen und speziellen Programmiertechniken.

1 Einleitung

Die weitere Erhöhung der Taktfrequenz aktueller Rechnersysteme wird im Wesentlichen durch Probleme bei der Stromversorgung, der Kühlung und auch der Zuverlässigkeit der Systeme beschränkt. Mehrkernsysteme, die als Lösungsansatz für das Frequenzproblem verstanden werden, leiden verstärkt unter dem Flaschenhals der Speicherschnittstelle. Dieser Flaschenhals wird zwar durch Caches entschärft, sind aber lokale Caches vorhanden so muss die Konsistenz der Speicherzugriffe sichergestellt werden. Mit steigender Anzahl an Prozessorkernen in einem Rechnersystem wird es einerseits immer aufwändiger diese Konsistenz zu erhalten und andererseits steigt auch die Anzahl der Zugriffe auf den Speicher. Moderne Many-Core Systeme erfordern daher eine spezielle Art der Programmierung, welche die explizite Verwendung von privaten Speichern erlaubt. Ein solcher privater Speicher kann entweder nur einem Prozessorkern alleine zur Verfügung stehen, oder aber von mehreren Kernen, die zu einem Cluster zusammengefasst sind, verwendet werden. Ziel ist hierbei die Minimierung der Zugriffe auf den globalen Speicher, die aufgrund der großen Distanz zum Prozessorkern und der hohen Zahl an konkurrierenden Zugriffen sehr ineffizient sind.

Ziel des **Mo***ving Execu***tion** (Motion) Projekts ist es, ein Many-Core System mit mehreren 100 Kernen derart zu entwerfen, dass herkömmliche Standard- und Legacy-Software ohne zusätzlichen Aufwand effizient ausgeführt werden kann.

2 Stand der Technik

Ein massiv-paralleles Single-Chip-System steht mit dem Intel Single Chip Cloud Computer (SCC) zur Verfügung [Int10]. Dieses Ein-Chip-System enthält 48 Pentium P54C Prozessoren, die in einem zweidimensionalen Feld von 6x4 Tiles mit je zwei Prozessoren angeordnet sind. Die Kommunikation untereinander und mit dem gemeinsamen Speicher erfolgt über Nachrichten, die über ein internes Kommunikationssystem (Mesh) transportiert werden. Ein Vorgänger des Intel SCC war der Intel Polaris [Int] mit insgesamt 80 Prozessorkernen, die aber keinen Intel Standardbefehlssatz unterstützten. Weitere Many-Core Systeme stammen von Tilera deren aktuelle Prozessorfamilie die TILE-Gx Serie darstellt [Til]. Diese ist mit bis zu 100 VLIW Prozessoren ausgestattet und für Netzwerkanwendungen optimiert.

Im Bereich der Forschung beschäftigt sich der DFG Sonderforschungsbereich/Transregio 89 *Invasive Computing* mit zukünftigen Rechnerarchitekturen und geeigneten Programmierparadigmen [Tei08].

Verschiedene Techniken zur Beschleunigung wissenschaftlicher Untersuchungen mit Hilfe paralleler Systeme werden von Weber et al. [WGHP11] verglichen. Hierbei wird neben der Rechenleistung auch der nötige Entwicklungsaufwand betrachtet.

3 Moving Execution

Alle bekannten Rechnersysteme besitzen die Eigenschaft, dass ein Programm über längere Zeit an einen ausführenden Prozessor gebunden ist. Die bei der Ausführung benötigten Daten werden, mehr oder weniger aufwändig, zum entsprechenden Prozessor transportiert. Der grundlegende Gedanke des Motion Systems ist es hingegen, die Programmausführung dort durchzuführen, wo sich die Daten zur gegebenen Zeit befinden. Die vorhandene Bindung eines Ausführungskontextes an einen Prozessorkern wird hierdurch aufgeweicht. Vielmehr entsteht eine Bindung der Daten an einen Prozessorkern.

Voraussetzung hierfür sind Prozessorkerne mit lokalem Speicher. Abbildung 1 zeigt beispielhaft eine mögliche Motion Systemarchitektur. Ein Prozessorkern wurde hierbei durch eine Speicherschnittstelle ersetzt, so dass Zugriffe auf den externen Speicher hierüber möglich sind.

Zur Verdeutlichung der Funktionsweise sei eine objektorientierte Software bestehend aus einer Vielzahl von Klassen, Objekten und Ausführungskontexten gegeben. Zwei Ausführungskontexte arbeiten sowohl mit privaten Daten als auch mit gemeinsamen Daten, die jeweils durch Objekte repräsentiert werden.

In einem Standard-Mehrkernprozessor würde jeder Ausführungskontext auf einem Prozessor ausgeführt. Konkurrierende Zugriffe auf die gemeinsamen Objekte hätten einen fortwährenden Austausch der Daten dieser Objekte zwischen den lokalen Caches der beteiligten Kerne zur Folge. Insbesondere die Synchronisation erfordert den mehrfachen Austausch der Schlossvariable, selbst wenn ein anderer Ausführungskontext das Objekt

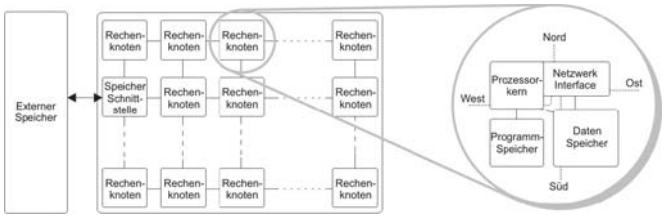

Abbildung 1: Beispiel einer Motion Systemarchitektur

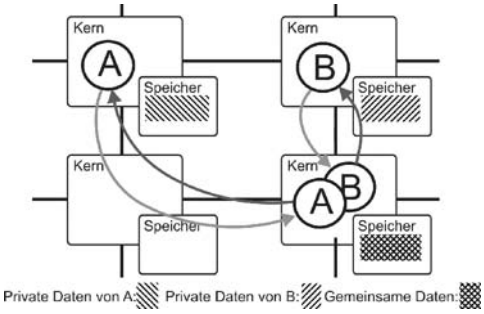

Abbildung 2: Beispielhafte Ausführung zweier Kontexte mit privaten und gemeinsamen Daten.

bereits besitzt und ein Zugriff auf das Objekt derzeit nicht möglich ist. Neben diesem notwendigen Datenaustausch muss bei jedem Datenzugriff, unabhängig davon auf welches Objekt er sich bezieht, sichergestellt werden, dass die gelesenen Daten auch die Aktuellen sind. Hierfür ist ein nicht unerheblicher Hardware- und Zeitaufwand nötig. All diese Probleme umgeht die Motion Architektur, indem jedes Objekt fest an einen Prozessorkern gebunden wird. Sind Operationen auf den Daten dieses Objekts nötig, so werden diese auf dem entsprechenden Kern ausgeführt.

Die beiden Ausführungskontexte des Beispiels sind in Abb. 2 in den beiden oberen Kernen, zusammen mit ihren privaten Daten, beheimatet und werden parallel ausgeführt. Erfolgt ein Zugriff auf die gemeinsamen Daten, so wird die Ausführung des entsprechenden Kontextes auf den Kern mit den gemeinsamen Daten (im Beispiel unten rechts) verschoben. Anschließend wechselt die Ausführung wieder zum original Kern. Während der Abwesenheit der Kontexte A bzw. B von den oberen Kernen können diese andere Kontexte ausführen, die z. B. als Ausführungsaufträge anderer Kerne vorliegen.

Da keine zeitlichen Beschränkungen existieren, können theoretisch beliebige Strecken für die Übermittlung von Ausführungsaufträgen zurückgelegt werden. Daher ist es auch möglich, mehrere Prozessorchips oder ganze Rechnersysteme zu verbinden. Im Gegensatz zu Cluster-, Cloud-, oder Gridsystemen bilden alle miteinander verbundenen Motion Systeme eine Einheit mit einem gemeinsamen Adressraum.

Um die Verschiebung der Ausführung von einem Kern zu einem anderen zu ermöglichen, müssen neben dem Befehlszeiger auf das Ziel, die Rücksprungadresse, der Stackzeiger

und die Funktionsparameter übergeben werden. Befinden sich die Funktionsparameter auf dem Stack, so muss auch ein Teil des Stacks mit verschoben werden. Der Registersatz ist mit Ausnahme des Stackpointers nur dann notwendig, wenn die Funktionsparameter in Registern übergeben werden. Eine spezielle Prozessorerweiterung mit Zugriff auf die Prozessorregister erlaubt das effiziente Übertragen der benötigten Werte. Sie sendet diese, z. B. in Form einer Nachricht über ein Network-on-Chip (NoC), an einen anderen Prozessorkern. Dort wird diese Nachricht ggf. in einer Warteschlange abgelegt bis der Prozessorkern bereit ist die Anforderung auszuführen. Senderseitig überprüft der Prozessorkern nach erfolgtem Abschicken der Nachricht seine eigene Warteschlange und führt ggf. andere Ausführungsaufträge aus. Demnach muss jeder Prozessorkern um eine Sende- und Empfangseinheit erweitert werden, die für die Erteilung und Abwicklung von Ausführungsaufträgen zuständig ist. Im Gegenzug entfällt aber jegliche Logik, welche die Konsistenz von Speicherzugriffen sicherstellt.

4 Zusammenfassung und Ausblick

Der Moving Execution (*Motion*) Gedanke beschreibt eine mögliche Technik herkömmliche Anwendungsprogramme unverändert auf einem Many-Core System mit hunderten von Prozessorkernen auszuführen. Hierzu werden die Prozessorkerne mit einer zusätzlichen Sende- und Empfangseinheit ausgestattet, die Ausführungsaufträge an andere Kerne vergeben bzw. deren Aufträge annehmen kann. Die Basis stellt hierbei die feste Zuordnung von Daten zu einem Prozessorkern dar, was die Replizierung von Daten in mehreren Caches ausschließt und somit die Einhaltung der Konsistenz von Speicherzugriffen obsolet macht. Die Umsetzung und Evaluierung des Systems wird mit Hilfe des MANJAC Emulators durchgeführt werden [BSUU11].

Literatur

[BSUU11] Christian Bradatsch, Sebastian Schlingmann, Theo Ungerer und Sascha Uhrig. MANJAC - Ein Many-Core Emulator auf Multi-FPGA-Basis. In *PARS Workshop*, 2011.

[Int] Intel. The Teraflops Research Chip. http://techresearch.intel.com/ProjectDetails .aspx?Id=151, last download Feb. 23, 2011.

[Int10] Intel. The SCC Platform Overview. http://techresearch.intel.com/spaw2/uploads /files/SCC_Platform_Overview.pdf, last download Feb. 23, 2011, May 2010.

[Tei08] Jürgen Teich. Invasive Algorithms and Architectures. *it - Information Technology*, 50(5):300–310, 2008.

[Til] Tilera. TILE-Gx Processor Family. http://www.tilera.com/products/processors/ TILE-Gx_Family, last download Feb. 23, 2011.

[WGHP11] Rick Weber, Akila Gothandaraman, Robert J. Hinde und Gregory D. Peterson. Comparing Hardware Accelerators in Scientific Applications: A Case Study. *IEEE Trans. Parallel Distrib. Syst.*, 22:58–68, January 2011.