Master-Seminar - Hochleistungsrechner: Aktuelle Trends und Entwicklungen
Wintersemester 2015/2016

Many-core-Architekturen (Xeon Phi)

Fabio Gratl
Technische Universität München

Vortrag: 03.02.2016

Inhaltsverzeichnis

1 Einführung zu Many-core **2**
1.1 Die allgemeine Idee . . . 2
1.2 Einsatz von Many-core Systemen und ihr Potential . . . 2

2 Die Xeon Phi Familie **2**
2.1 Hintergrund und Gegenwart . . . 2
2.2 Roadmap . . . 3

3 Hardwareeigenschaften von Xeon Phis **3**
3.1 Kerne . . . 3
3.2 Speicher . . . 3
3.3 Chip oder Beschleuniger? . . . 4

4 Programmierung eines Xeon Phi **4**
4.1 Single-core Parallelisierung . . . 4
4.2 Multi-core-Parallelisierung . . . 5
4.3 Benutzungsmodi . . . 5

5 Performanzvergleich **6**

6 Fazit **8**

Zusammenfassung

Die aktuelle Situation des Prozessormarkts zeigt einen Trend zu immer höherer Parallelität. Große Systeme nutzen zunehmend sogenannte Rechnerbeschleunigerkarten (*Accelerator*) wie GPGPUs, um die Leistung durch hochparallele Hardware stark zu erhöhen ohne im selben Maße Kosten und Energiekonsum anzuheben.
In dieser Arbeit soll ein neuer Typ Beschleuniger namens Intel Xeon-Phi besprochen werden. Es wird der Stand der Hardwaretechnik gezeigt sowie die veröffentlichten Pläne, wohin die Entwicklung geht. Zuletzt werden die Paradigmen besprochen, welche zur Programmierung eines *Xeon Phi* benötigt werden und ein kurzer Leistungsvergleich zu Sandybridge Prozessoren gezogen.

1 Einführung zu Many-core

1.1 Die allgemeine Idee

Bis vor zehn Jahren war die allgemeine Herangehensweise um Prozessoren (CPUs) schneller und damit leistungsfähiger zu machen die Erhöhung ihrer Taktfrequenz. Um das Jahr 2005 stießen Hardwarehersteller allerdings auf das Problem, dass bei Frequenzen über 4 GHz die resultierende Abwärme unverhältnismäßig groß und nicht mehr allgemein sinnvoll abführbar wurde. Dies führte zur Entscheidung, die Performanz einer CPU zu erhöhen indem man mehrere Prozessorkerne auf einem Chip verbaut und somit theoretisch die $n-$fache Leistung erzielen kann. Dies ist vorteilhaft, da das mehr zu verbauende Material kaum Kosten verursacht und der Mehrverbrauch an Strom geringer ist als eine Erhöhung der Taktfrequenz, um dieselbe Leistung zu erzielen [5].
Die Idee, Leistung durch parallele Hardware zu erhöhen, ist nicht neu. Grafikkarten arbeiten schon länger mit vielen Rechenkernen und sogenannte *General Purpose Graphic Processing Units* (GPGPUs) werden auch schon erfolgreich als Beschleuniger im *High Performance Computing* eingesetzt.
Der Vorteil, den Many-core-Systeme bieten, ist leichtere Programmierbarkeit [1]. Dadurch, dass sie dieselben Architekturen wie reguläre CPUs verwenden (z.B. x86), können dieselben Programmier- und Parallelisierungstechniken (z.B. *OpenMP*, MPI) verwendet werden. Auch wird das Portieren von bereits geschriebenem Code sehr vereinfacht.

1.2 Einsatz von Many-core Systemen und ihr Potential

Der wichtigste Akteur auf diesem Gebiet zur Zeit ist Intel mit dem *Intel Xeon Phi*-Projekt, welcher damit in direkte Konkurrenz zu AMD und Nvidia auf dem Markt der Rechnerbeschleuniger tritt. Die Idee sehr viele kleine, niedrig getaktete Kerne zu verwenden bringt Vorteile in der Rechenleistung pro Watt und ist somit interessant für große Rechencluster, da hiermit die enormen Energiekosten gedrückt werden können.
Erste Testsysteme wurden im CERN, Korea Institute of Science and Technology Information (KISTI) und Leibniz-Rechenzentrum (LRZ SuperMIC) errichtet und erfolgreich getestet. Eines der ersten Cluster, welches den *Xeon Phi* einsetzt, ist *Stampede* vom Texas Advanced Computing Center der Universität von Texas. Bei seiner Produktionsinbetriebnahme 2013 rangierte das System auf Platz sechs der Top500[1].
Der zur Zeit (2013-2015) leistungsstärkste Supercomputer der Welt, der Tianhe-2 (auch Milkyway-2) des National Super Computer Center in Guangzhou, erreicht seine Leistung hauptsächlich durch *Xeon Phi*-Karten[2].

2 Die Xeon Phi Familie

2.1 Hintergrund und Gegenwart

Das *Intel Xeon Phi*-Projekt basiert direkt auf den Erkenntnissen des *Larrabee*-Projekts. In diesem Projekt versuchte Intel eine mehrkernige Grafikeinheit zu entwickeln, welche, im Gegensatz zu üblichen Grafikkarten nicht aus GPUs, sondern aus *Pentium*-basierten x86-Prozessoren bestand. Außerdem enthielt jeder Kern eine 512 Bit breite *Vector Processing Unit*, was die mögliche Rechenleistung stark erhöht. Das *Larrabee*-Projekt wurde 2010 beendet ohne ein daraus resultierendes Produkt auf den Markt zu bringen [7].
Der nächste große Schritt der *Many Integrated Core (MIC)*-Architektur war 2010 der Prototyp namens *Knights Ferry*. Diese PCIe-Karte ist ein hoch paralleler Rechnerbeschleuniger mit 32 Universalrechnerkernen basierend auf dem Pentiumdesign [2].
2011 veröffentlichte Intel die erste kommerzielle Produktreihe mit *MIC*-Architektur, die *Knights Corner*. Zur Zeit (Stand 2015) sind drei Ausführungsreihen erhältlich, optimiert auf die beste Rechenleistung pro Dollar (3100 Serie), beste Rechenleistung pro Watt (5100 Serie) und, das Flaggschiff, für universale

[1] http://top500.org/system/177931
[2] http://top500.org/system/177999

Performanz (7100 Serie). [3]. Eine genauere Diskussion der Eigenschaften erfolgt in Abschnitt 3. Knights Corner zeichnet sich aus durch einen 22 nm Fertigungsprozess. Zukünftige Versionen werden in noch kleineren Maßstäben produziert werden.

2.2 Roadmap

Intel hält sich sehr zurück mit Informationen bezüglich zukünftiger Releases, allerdings sind einige wenige, grobe Details bekannt. Die Nachfolgegeneration zu *Knights Corner* wird *Knights Landing*. Sie wird verfügbar sein sowohl als Beschleunigerkarte als auch als eigenständiger Prozessor.
Auch geplant ist eine Erhöhung der Anzahl an Prozessoren, der Rechenleistung, der Speicherbandbreite und eine Verbesserung der Energieeffizienz. Im Gegensatz zu aktuellen Knights Corner-Systemen, welche auf *Pentium*-Kernen aufbauen, sollen zukünftige *Xeon Phi*-Generationen auf Kernen des Typs *Silvermont* aufbauen. Diese Mikroarchitektur wird für Atom- und Celeronprozessoren genutzt, welche spezialisiert sind auf Energieeffizienz.
Des Weiteren ist geplant, AVX-512 weiter zu vervollständigen und in den kommenden Generationen der regulären *Xeon*-Prozessoren (*Skylake*) zu implementieren.
Eine weitere Besonderheit einer Variante der *Knights Landing*-Prozessoren wird die Integrierung eines *Omni-Path Fabric*[4] Controllers in den Prozessor. Fabric ist Intels eigene High-Performance-Netzwerktechnologie für schnelle Datenübertragung und geringe Latenz, basierend auf Infiniband[5]. Eine Integrierung des Controllers eliminiert einen potentiellen Flaschenhals und spart sowohl Geld als auch Energie für eigene Controller-Karten.

3 Hardwareeigenschaften von Xeon Phis

3.1 Kerne

Die aktuelle (Stand 2015) Generation von *Xeon Phi*s ist ausgestattet mit ca. 60 Kernen (genaue Zahlen schwanken mit der Version, siehe Tabelle 1), welche auf dem Design der Pentium-Prozessoren mit x86-Architektur basieren, allerdings 64-Bit unterstützen. Jeder Kern besitzt vier Hardwarethreads, welche mit Round Robin gescheduled werden. Das wohl interessanteste Merkmal ist die 512 Bit breite Vektorinstruktionseinheit. Mit ihr lassen sich sogenannte *Single Instruction Multiple Data (SIMD)*-Operationen mit bis zu 16 32 Bit- oder acht 64 Bit-Zahlen im selben Takt durchführen. Daraus ergeben sich acht *Double Precision Floating Point Operations (DPFLOPs)*. Da *Xeon Phi* über *Fused Multiply-Add (FMA)*, also die Möglichkeit, eine Multiplikation und eine Addition zu einer Operation zu verschmelzen, verfügt, müssen die oben genannten acht *DPFLOPS* mal zwei genommen werden. Mit diesen Werten lässt sich die theoretische Leistung, z.B. für ein Modell der 71xx Reihe, wie folgt berechnen:

$$16\,DPFLOPs \cdot 61\,Cores \cdot 1,238\,GHz \approx 1208\,DPGFLOPs$$

Zu beachten ist weiterhin, dass nicht das volle Potential aller Kerne ausgeschöpft werden kann, da der Coprozessor nicht über ein treiberbasiertes System verfügt, sondern ein eigenes Linux-Betriebssystem hostet. Dieses gibt dem Coprozessor eine eigene IP, kümmert sich um die Kommunikation mit dem Hostsystem und sorgt für ein softwaregesteuertes Scheduling. Des Weiteren erlaubt es dieses System, den Beschleuniger in verschiedenen Benutzungsmodi zu verwenden. Mehr dazu im Kapitel 4.3.

3.2 Speicher

Jeder Kern verfügt über jeweils 32 KB L1-Daten- und 32 KB L1-Instruktionscache. Die L2-Caches sind an einen bidirektionalen Ringbus angeschlossen, der für schnelle Kommunikation auf dem Prozessor und Cachekohärenz sorgt (siehe Abbildung 1). Pro Kern stehen 512 KB vereinter L2-Cache zur Verfügung. Ein über alle Kerne geteilter L3-Cache existiert nicht aufgrund der hohen Bandbreite des auf der PCIe-Karte verbauten Arbeitsspeichers von 352 GB/s bei 2570 MHz[4]. Auch mit diesem Bus verbunden sind das PCIe-Interface und bis zu acht (abhängig von der Version) GDDR5 dual-channel Controller, an die bis zu 16 GB Arbeitsspeicher angeschlossen sind. Daraus resultiert eine relativ kleine Arbeitsspeichergröße pro Kern von ca. 262,3 MB (105,3 MB für 31xx und 133,3 MB für 51xx).

[3] http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html
[4] http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/overview.html
[5] http://www.infinibandta.org/

Version	# Kerne	Taktfrequenz (GHz)	Max. Speicher (GB)	Max. Speicher-bandbreite (GB/s)	Leistung (Double GFLOP)
31xx	57	1,1	6	240	1003
51xx	60	1,053	8	320-352	1011
71xx	61	1,238 (Turbo: 1,333)	16	352	1208

Tabelle 1: Hardwarespezifikationen der *Xeon Phi* Coprozessoren nach dem Intel *Xeon Phi* Product Family: Product Brief.

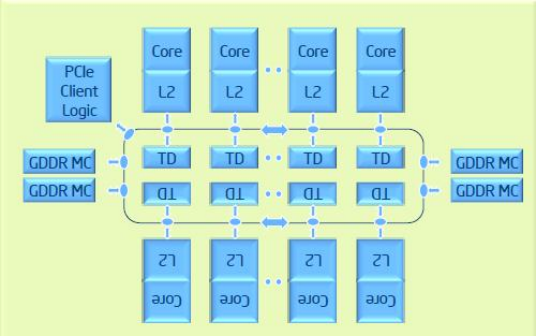

Abbildung 1: Ringbus Verbindung des *Xeon Phi* (Quelle: https://software.intel.com)

3.3 Chip oder Beschleuniger?

Die aktuellen *Xeon Phi*-Modelle sind als PCIe-Rechnerbeschleunigerkarte gebaut, doch bringt dies ein paar Nachteile mit sich, wie zum Beispiel starke Limitierung des Arbeitsspeichers und die Notwendigkeit über den PCIe-Port zu kommunizieren. Das Problem wird illustriert in Abbildung 2. Die linke Grafik zeigt die Kommunikation auf einem Knoten, welcher hier aus zwei Xeon E5-2670 (*Sandybridge*) CPUs und zwei *Xeon Phi* 5110P (Knights Corner) Coprozessoren besteht[6]. Die Kommunikation zwischen *Xeon Phi* und Host über PCIe 2.0 erreicht hier bis zu 6.3 GB/s bei einem theoretischen Maximum von 8 GB/s für 16-lane PCIe 2.0. Sehr starke Bandbreiteneinbrüche können bei direkter Kommunikation zwischen zwei Coprozessoren gemessen werden. Hier ist die maximale erreichte Bandbreite 1 GB/s. Dies liegt unter anderem an der fehlenden Optimierung der *Sandybridge* Prozessoren, die hier als PCIe Brücke verwendet werden müssen und dafür nicht ausgelegt sind. Auf der rechten Seite ist die Kommunikation über mehrere Knoten zu sehen. Wie bereits im Kapitel 2.2 angesprochen wurde, sollen zukünftige Generationen des *Xeon Phi* auch als Hostprozessor verfügbar sein, was diese Probleme lösen würde, da die Kommunikation über den PCIe-Flaschenhals entfällt.

4 Programmierung eines Xeon Phi

4.1 Single-core Parallelisierung

Unter *Single-Core-Parallelisierung* versteht man die parallele Verarbeitung mehrerer Daten auf einem Rechenkern. Dies wird erreicht mit sogenannten *Single Instruction Multiple Data (SIMD)*-Vektorinstruktionen. Die hierfür zuständigen Instruktionssätze sind (in historischer Reihenfolge) *SSE* und *AVX* und insbesondere auf dem *Xeon Phi AVX-512*. Im Falle von Letzterem werden acht 64 Bit-(oder 16 32 Bit)-Werte in jeweils ein sogenanntes Vektorregister geladen. Operationen zwischen zwei Vektorregistern erfolgen dann elementweise, wie z.B. die bekannte Vektoraddition.
Der *Xeon Phi* zieht einen Großteil seiner Leistung aus diesen breiten Vektoreinheiten, weshalb man, wenn man Code für *Xeon Phi* optimieren möchte, den Code so weit wie möglich vektorisieren sollte, da dies für die parallelen Abschnitte theoretisch annähernd linearen Speedup bedeutet, da Vektorisierung kaum Overhead produziert.
Vektorisierung kann auf mehrere Arten implementiert werden. Die simpelste und zugleich ineffizienteste Möglichkeit ist, den Compiler selbst vektorisierbare Abschnitte (hauptsächlich Schleifen) finden zu lassen und diese zu vektorisieren. Dazu führt der Compiler Abhängigkeitsanalysen auf Schleifen durch und zerlegt diese gegebenenfalls. Diese Möglichkeiten werden mit gängigen Compilern wie z. B. dem Gnu C/C++-Compiler (gcc/g++) oder dem Intel C/C++-Compiler (icc/icpc) geboten. Die zweite Möglichkeit ist das Nutzen sogenannter *Intrinsic Functions (Intrinsics)*. Intrinsics sind Wrapper um Assemblerbefehle und bieten dem Programmierer so eine große Kontrolle über das tatsächliche Ver-

[6] https://www.nics.tennessee.edu/computing-resources/beacon/configuration

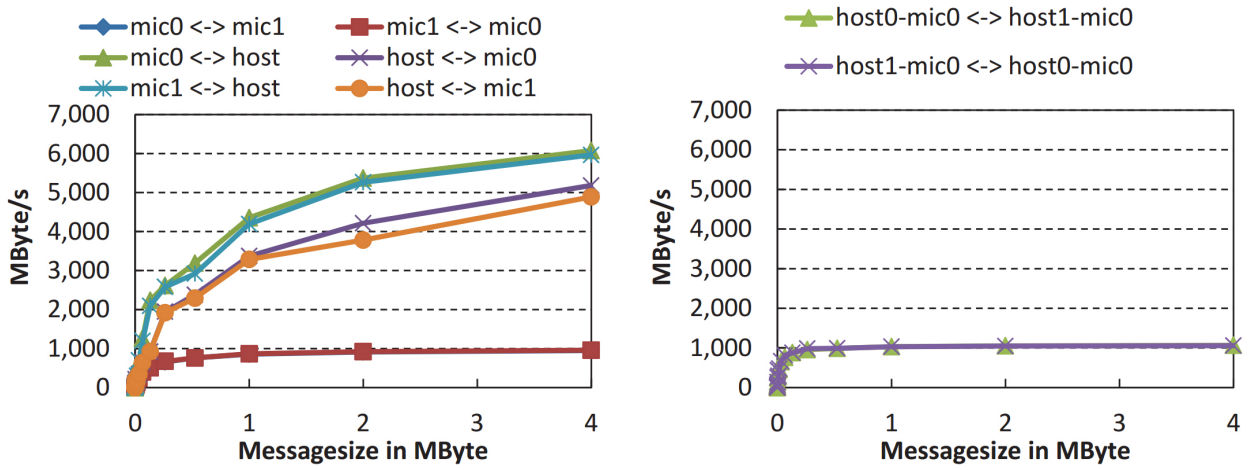

Abbildung 2: Kommunikationsbandbreite durch das PCIe-Interface.
Hardware: host = *Xeon* E5-2670 (Sandybridge), mic0/1 = *Xeon Phi* 5110P (*Knights Corner*) (Quelle: [4])

halten der CPU. Der Vorteil gegenüber direktem Aufruf von Assemblercode ist, dass der Compiler bei der Codeoptimierung die Intrinsics mit berücksichtigen kann. Vektorisierung mittels Intrinsics bedeutet oft einen gewissen Implementierungsaufwand, führt dafür aber oft zu sehr guten Ergebnissen.

4.2 Multi-core-Parallelisierung

Multi-core-Parallelisierung ist die parallele Ausführung von Code auf mehreren Rechenkernen auf einem Rechenknoten. Dies ist besonders wichtig für *Xeon Phi*s, da diese über eine für einzelne Knoten hohe Zahl an Kernen verfügen. Der große Vorteil des *Xeon Phi*s ist hier die x86-Architektur und Unterstützung von gängigen Programmiertechniken wie *OpenMP*. Dadurch kann Code, der bereits mit derselben Technik für andere CPUs parallelisiert wurde, direkt auf dem *Xeon Phi* parallel ausgeführt werden.
OpenMP wird durch sogenannte *Compiler Pragmas* genutzt. Pragmas sind spezielle Methoden, welche dem Compiler zusätzliche Informationen geben wie er ein Programm kompilieren soll. Der Programmierer kann einzelne Durchläufe von Schleifen auf Prozessorkerne verteilen (omp parallel (for)), oder auch ganze Codeblöcke über die Kerne verteilen (omp task).

4.3 Benutzungsmodi

Auf jedem *Xeon Phi* läuft ein vollständiges Linux-Betriebssystem, was ihn autonom agieren lässt. Da aktuelle und teilweise auch zukünftige *Xeon Phi*-Modelle PCIe-Rechnerbeschleunigerkarten sind, ergeben sich verschiedene Möglichkeiten der Benutzung.
Die simpelste Möglichkeit ist der sogenannte *native* Modus. Hier wird das Programm zuerst speziell für den Beschleuniger durch das Übergeben eines Compiler-Flags kompiliert. Führt man das Programm aus, wird es erst in den Speicher des *Xeon Phi*s geladen und dann nur dort lokal ausgeführt. Es wird nur die gepufferte Ausgabe zum Host zurück kopiert.
Eine weitere Benutzungsmöglichkeit ist die Ausführung mittels *Offload*. In diesem Modus wird das Programm sowohl für den Hostprozessor als auch für den Beschleuniger kompiliert. Zur Ausführung wird das Programm auf dem Hostsystem gestartet. Trifft der Programmfluss auf eine Region, die als Offload mit Pragmas markiert ist, wird diese Region mit aktuellen Variablenwerten auf den Beschleuniger kopiert, dort ausgeführt und anschließend nur vom Programmierer definierte Werte zurück kopiert. Die Ausführung auf dem Hostsystem wartet auf den Abschluss dieses gesamten Vorgangs. Dieser Modus eignet sich sehr gut, um große, rechenintensive Teile des Codes, die keinen I/O enthalten, zu beschleunigen.
Der sogenannte *symmetrische* oder *synchrone* Modus bildet die dritte Möglichkeit zur Benutzung eines *Xeon Phi*s. Dieser funktioniert ähnlich zum Offload-Modus, jedoch wird die Ausführung auf dem Hostprozessor nicht blockiert und die Arbeitslast manuell verteilt. Hier gilt es eine gute Verteilung zu finden, da der *Xeon Phi* zwar eine enorme Rechenleistung hat, aber zuerst der PCIe-Flaschenhals überwunden werden muss, wohingegen der Hostprozessor direkt mit der Berechnung starten kann.

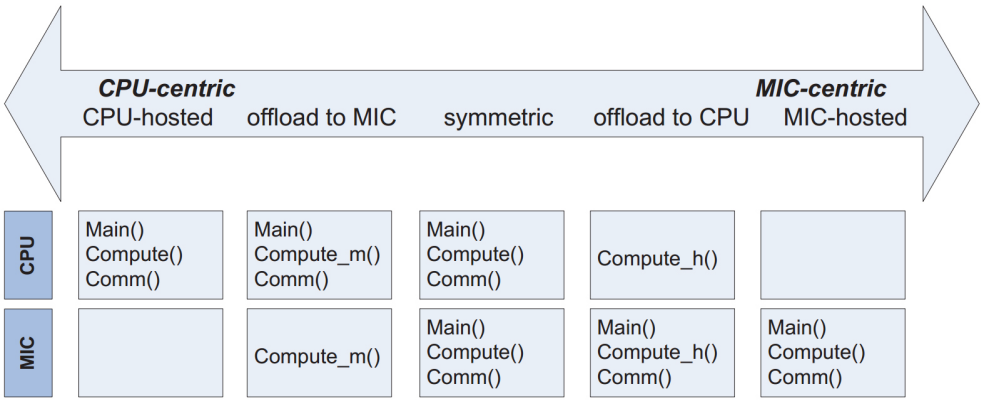

Abbildung 3: Benutzungsmodi einer *Xeon Phi* PCIe-Karte (Quelle: [4])

Aufgrund des oben erwähnten eigenen Betriebssystems des *Xeon Phi* sind beliebige Kombinationen möglich oder sogar ein Offload zum Hostprozessor aus dem nativen Modus. Eine Übersicht dazu gibt Abbildung 3. Intels *Math Kernel Library (MKL)* wurde ebenfalls für den Einsatz mit *Xeon Phi*s optimiert. Sie kann mit allen drei Modi aufgerufen werden und verteilt je nach Konfiguration die Rechenlast auf die verfügbaren Ressourcen.
Weitere Nutzungsoptionen ergeben sich mit dem *Message Passing Interface (MPI)*. MPI wurde von Anfang an für die Kommunikation in heterogenen Systemen entwickelt. Da jeder *Xeon Phi* sein eigenes Linux-Betriebssystem hostet, eigenen Speicher besitzt und IP-adressierbar ist, wirkt er von außen sehr ähnlich wie ein Rechenknoten. Daher ist eine natürliche Herangehensweise, ihn als eigenen MPI-Rank zu nutzen. Dies ist ohne weitere Modifikationen am Code möglich, jedoch sollten eventuell Lastverteilungsüberlegungen angestellt werden, da der *Xeon Phi* relativ wenig Speicher pro Prozessor zur Verfügung hat.

5 Performanzvergleich

Trotz der relativ simplen und langsamen *Pentium*-Kerne sollte mit seiner großen Anzahl an Kernen ein *Xeon Phi* signifikant mehr Leistung bringen als ein Prozessor ähnlicher Generation. In [4] werden unter anderem Implementierungen von Matrixmultiplikationen verglichen, welche bewusst die vorliegende Hardware ausnutzen. Es wird angemerkt, dass für den *Xeon Phi* nur eine Reimplementierung der Kernels nötig war, um die in dieser Version des *Xeon Phi* verwendeten Vektorinstruktionen auszunutzen. Wie in den Abbildungen 4 und 5 zu sehen ist, kann ein einzelner *Xeon Phi* hier mit der 3,75-fachen Anzahl an Kernen fast die dreifache Leistung erbringen wie zwei Sandybridge-Prozessoren, welche ca. die 2,5-fache Taktfrequenz haben. Das niedrige Leistungsmaximum für TifaMMy wird in der referenzierten Arbeit durch fehlende Möglichkeiten zur Parallelisierung mittels *OpenMP* begründet. Trotzdem wird eine parallele Effizienz von über 98% sowohl für die MKL als auch die TifaMMy Implementierung angegeben[4].

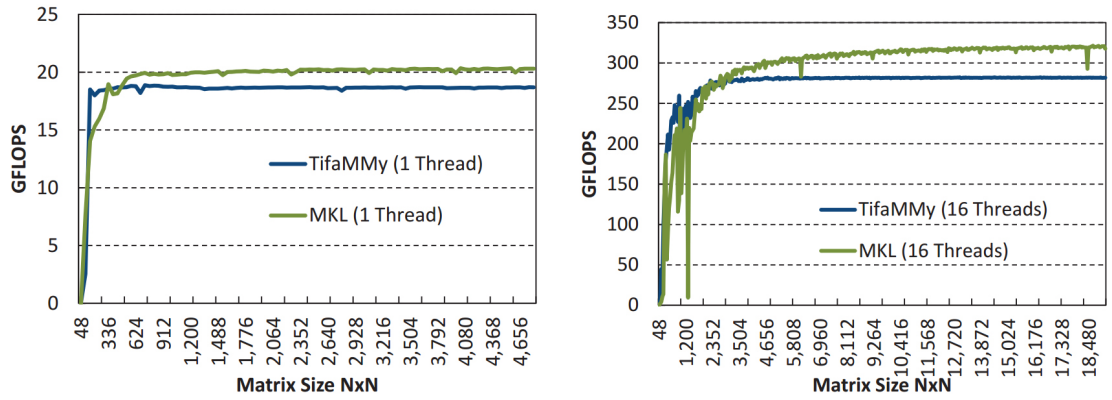

Abbildung 4: Matrix Multiplikation auf zwei Xeon E5-2670 (Sandybridge)
Links: ein Kern, rechts: 16 Kerne (Quelle: [4])

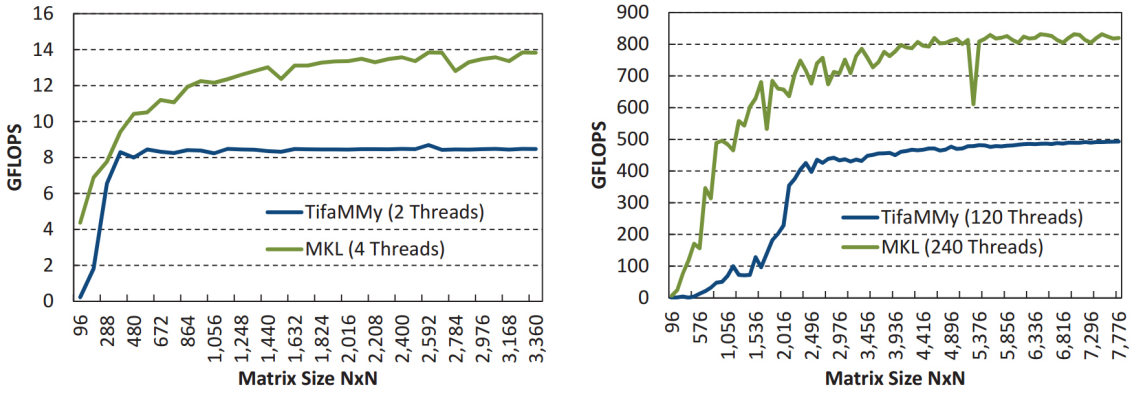

Abbildung 5: Matrix Multiplikation auf einem *Xeon Phi* 5110P (Knights Corner)
Links: ein Kern, rechts: 60 Kerne (Quelle: [4])

6 Fazit

Der Trend zu hochparalleler Hardware ist klar erkennbar. Intels *Xeon Phi*-Projekt ist noch jung, aber interessante Resultate im Bezug auf Rechengeschwindigkeit, Energieeffizienz und einfacher Programmierbarkeit konnten bereits geliefert werden. Es wurde gezeigt, dass es sehr einfach ist, existierenden Code auf *Xeon Phis* zum Laufen zu bringen, allerdings doch noch einige hardwarenahe Optimierungsschritte nötig sind, um das volle Potential auszunutzen. Dieses Potential liegt eindeutig in den breiten Vektoreinheiten, verbunden mit einer großen Anzahl an Threads. Zukünftige *Xeon Phi*-Versionen, welche nicht nur als Beschleuniger, sondern auch als Hostprozessor verfügbar sein werden, werden das Problem des PCIe-Flaschenhalses lösen und dadurch Leistung sowie Benutzbarkeit signifikant verbessern.

Alle angegebenen Websites wurden zuletzt am 20.12.2015 aufgerufen.

Literatur

[1] George C Caragea, Fuat Keceli, Alexandros Tzannes, and Uzi Vishkin. General-purpose vs. gpu: Comparison of many-cores on irregular workloads. *Proc. HotPar*, 2010.

[2] Alejandro Duran and Michael Klemm. The intel® many integrated core architecture. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 365–366. IEEE, 2012.

[3] Alexander Heinecke, Karthikeyan Vaidyanathan, Mikhail Smelyanskiy, Alexander Kobotov, Roman Dubtsov, Greg Henry, Aniruddha G Shet, Grigorios Chrysos, and Pradeep Dubey. Design and implementation of the linpack benchmark for single and multi-node systems based on intel® xeon phi coprocessor. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 126–137. IEEE, 2013.

[4] Alexander Friedrich Heinecke. *Boosting scientific computing applications through leveraging data parallel architectures*. Verlag Dr. Hut, 2013.

[5] James Jeffers and James Reinders. *Intel Xeon Phi coprocessor high-performance programming*. Newnes, 2013.

[6] Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Performance evaluation of sparse matrix multiplication kernels on intel xeon phi. In *Parallel Processing and Applied Mathematics*, pages 559–570. Springer, 2014.

[7] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, et al. Larrabee: a many-core x86 architecture for visual computing. In *ACM Transactions on Graphics (TOG)*, volume 27, page 18. ACM, 2008.