Neueste Trends in HPC

Exascale Computing & Resilienz

Von Pierre Bouchard

Abstract

Exascale ist eine neue Zielgröße in der Entwicklung von Hochleistungsrechnern.

Diese Ausarbeitung beschreibt die Gründe für die Entwicklung von Hochleistungsrechnern und deren Architekturmodellen. Weiterhin wird auf damit verbundene Herausforderungen und Lösungsansätze eingegangen. Am Ende wird gesondert das Thema Resilienz von Hochleistungsrechnern betrachtet.

Inhaltsverzeichnis

1 Gründe für leistungsfähigere Supercomputer 3
1.1 Vorhersagbarkeit von Naturkatastrophen 3
1.2 Fähigkeiten von Nachrichtendiensten 3
1.3 Simulationen in der Forschung 3
1.3.1 Moleküldynamik 3
1.3.2 Astrophysik 3
2 Definition Exascale Computing 4
3 Existierende und geplante Hochleistungssysteme 4
3.1 Sunway TaihuLight 4
3.2 Aurora 5
3.3 Tianhe-3 5
3.4 Planung der EU 5
4 Funktionsweise 5
4.1 Shared Memory Computing 6
4.2 Distributed Memory Computing 8
4.3 Hybrid Distributed-Shared Memory Computing 9
5 Strategie zur Erreichung der Exascale Marke 9
5.1 Lösungsansatz 9
5.2 Verbleibende Probleme 10
5.3 Bestehende Problemlösungsansätze 10
5.3.1 Dreidimensionale Positionierung der Transistoren 10
5.3.2 Wärmereduktion 11
5.3.3 Energieverbrauch-Minderung 11
5.3.4 Identifikation stark energieverbrauchender Komponenten 11
5.3.5 Systemsoftware in Exascale Systemen 12
6 Resilienz 12
7 Quellen 14
8 Abbildungsquellen 15

1 Gründe für leistungsfähigere Supercomputer

1.1 Vorhersagbarkeit von Naturkatastrophen

Naturkatastrophen, wie Hurrikans, Erdbeben, Tsunamis oder Vulkanausbrüche, fordern vor allem sehr viele Menschenleben, verursachen jedoch auch immense Sachschäden und damit verbundene Kosten. Diese Naturkatastrophen werden schon heutzutage von Hochleistungsrechnern versucht vorherzusagen. Jedoch reicht die Rechenleistung dieser Systeme noch nicht aus, um hinreichend genaue Vorhersagen zu treffen. Um diese präziser und vor allem früher zu detektieren, um Gegenmaßnahmen ergreifen zu können, müssen die Hochleistungssysteme immer leistungsstärker werden. Der Schritt zu den Exascale Rechnern ist hierbei entscheidend, um die Vorhersagegenauigkeit zu erhöhen. [1]

1.2 Fähigkeiten von Nachrichtendiensten

Nachrichtendienste, wie zum Beispiel die NSA, zeigen großes Interesse an Supercomputern, welche die Exascale Marke überschreiten können. In den geleakten Dokumenten von Edward Snowden ist die Rede von der Arbeit an einem Quantencomputer. Dieser soll primär zum Hacken von, zum Beispiel Banken, Regierungen sowie anderen Geheimdiensten, aber auch von Forschungseinrichtungen genutzt werden. Jedoch auch zur Abwehr dieser Hackingattacken soll solch ein geplantes Exascale System zur Verfügung stehen. Gegenspieler sind hier die EU und die Schweiz, welche auch in Supercomputer investieren, die dann den Geheimdiensten zur Verfügung stehen. Die Hochleistungssysteme brauchen in diesem Bereich die Leistungssteigerung, um immer komplexere Hackingangriffe durchzuführen oder abzuwehren zu können. Somit steigt auch die Erfolgschance für einen solchen Hackingangriff immens an. [2]

1.3 Simulationen in der Forschung

Der Schritt zum Exascale System ist auch für sehr viele Forschungsgebiete wichtig. Im Folgenden werden beispielhaft die Auswirkung auf die Moleküldynamik und die Astrophysik behandelt.

1.3.1 Moleküldynamik

Im Bereich der Biologie, speziell in der Moleküldynamik, spielt die Berechnung von Simulationen eine große Rolle. Der Sprung über die Exascale Grenze ist sehr wichtig, zum Beispiel für die Berechnungen für das Verhalten von Proteinen auf molekularer Ebene. Hier kann das Verhalten bisher nur für einige Mikrosekunden berechnet werden, jedoch ist hier ein größerer Berechnungszeitraum von Nöten, um das Verhalten verstehen zu können.

1.3.2 Astrophysik

Auch in diesem Forschungsgebiet werden Themen, wie die Entstehung des Universums, von dem Überschreiten der Exascale Marke profitieren. Mit Rechnern dieser Größenordnung sind Simulationen möglich, welche man durch ihre hohe Genauigkeit direkt mit der Realität verknüpfen kann. [3]

Weitere Beispiele, welche die Wichtigkeit des Exascale Computings verdeutlichen, sind in [3] zu finden.

2 Definition Exascale Computing

Exascale Computing steht für eine Größenordnung der Leistungsfähigkeit eines Computersystems. Hierbei ist „Exa“ der entscheidende Schlüsselfaktor. Exa steht für die Zahl 10^{18} und in diesem Zusammenhang geht es um 10^{18} Flops, mit denen ein Hochleistungsrechensystem rechnen kann. Flops steht für **Flo**ating **P**oint Operations Per **S**econd, also Gleitkommazahlen pro Sekunde, mit denen ein solches System rechnen kann. Jedoch existiert bis jetzt kein Hochleistungsrechner, der mit 10^{18} Gleitkommazahlen pro Sekunde rechnen kann. [4]

3 Existierende und geplante Hochleistungssysteme

3.1 Sunway TaihuLight

Dieser Hochleistungsrechner steht unter den Top 500 auf Platz eins und ist somit der leistungsstärkste, gelistete Hochleistungsrechner. [5]
Das System wurde komplett in China fabriziert. Der Grund hierfür liegt in dem amerikanischen Handelsembargo, wodurch Unternehmen, wie Intel, keine Prozessoren für Hochleistungsrechner mehr nach China exportieren dürfen. Dieses tat der Entwicklung in China jedoch keinen Abbruch, weswegen sie jetzt den Platz eins der Top 500 belegen. [6]

Das System wurde im National Supercomputing Center in Wuxi entwickelt und wird vom National Research Center of Parallel Computing Engineering & Technology hergestellt. Das System wird sehr vielseitig eingesetzt, wie zum Beispiel für Ölexplorationen, Biowissenschaften und Wettervorhersagen. Es wird aber auch für das industrielle Design oder für die Pharmazie genutzt. [7]

Die Rechnerarchitektur des Sunway TaihuLight ist in Abbildung 1 zu sehen (wird auf den auf Seite 8-9 noch genauer erklärt) und nennt sich Hybrid Distributed-Shared Memory Computing. [8]

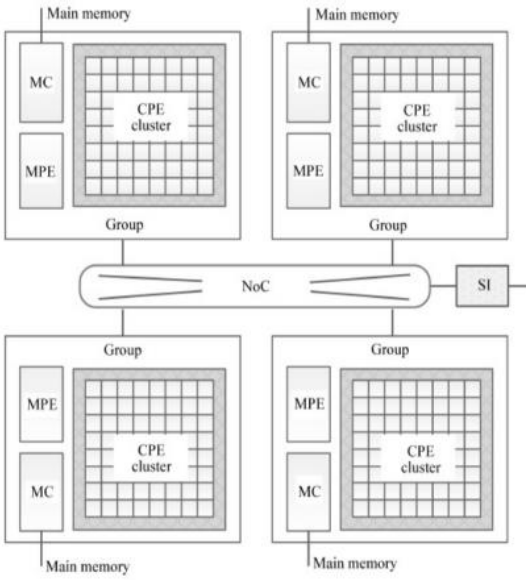

Abbildung 1: Rechnerarchitektur eines Rechnerknotens des Sunway TaihuLight [A1]

Kennzahlen des Systems sind in Tabelle 1 zu finden:

Tatsächliche Rechenleistung	93,015 PetaFlops
Theoretische Spitzenleistung	125,439 PetaFlops
Energieverbrauch	15,371 MegaWatt

Flops pro Watt	6,02 GigaFlops pro Watt
Speichergröße	1,31 Petabytes [7]

Tabelle 1: Kennzahlen des Sunway TaihuLight-Systems

3.2 Aurora

Die Antwort auf den Sunway TaihuLight von den Amerikanern stammt von Intel und wird im Argonne National Laboratory in den USA entwickelt. Die erste Version soll 2018 fertig gestellt werden und eine Leistung von 180 PetaFlops erzielen. Die nächste Ausführung soll schon 450 PetaFlops erlangen. Trotz der viel stärkeren Leistung soll der Energieverbrauch auf 13 MegaWatt schrumpfen, somit ist auch das Verhältnis von Watt zu Flops um einiges besser, sie beträgt rund 13 GigaFlops pro Watt. Mit mehr Daten, die erzeugt werden können und mit denen gerechnet werden kann, steigt auch die Speichergröße von Aurora auf sieben Petabytes an. [9]
Die Rechnerarchitektur ist auch hier Hybrid Distributed-Shared Memory Computing. [10]

3.3 Tianhe-3

Das System, welches als erstes die Exascale Marke überschreiten soll, ist der Tianhe-3. Der Hochleistungsrechner besteht, wie der Sunway TaihuLight, vollkommen aus chinesischen Komponenten und soll 2020 eine Rechenleistung von 10^{18} erreichen. 2018 soll schon ein Prototyp auf den Markt kommen. Das System soll allgemein für Forschungseinrichtungen verfügbar sein und soll zum Beispiel zur Berechnung der Smog-Verteilung in China genutzt werden. [11]

3.4 Planung der EU

Auch die EU Kommission plant den Bau eines Exascale Rechners. Dieser soll bis 2022/2023 fertiggestellt werden. Die Planung dafür soll bis Ende 2017 abgeschlossen sein. Dafür wurde ein Finanzierungspaket in Höhe von 5 Milliarden Euro bereitgestellt. [12]

4 Funktionsweise

Wie schon im vorherigen Kapitel angekündigt folgen auf den nächsten Seiten nun die genaueren Erläuterungen der Rechnerarchitekturen. Bei dem Sunway TaihuLight wird Hybrid Distributed-Shared Memory Computing genutzt. Wie der Name schon sagt, ist diese Rechnerarchitektur ein Hybrid aus dem Distributed und des Shared Memory Computing. Die verschiedenen Funktionsweisen werden im Folgenden erläutert.

4.1 Shared Memory Computing

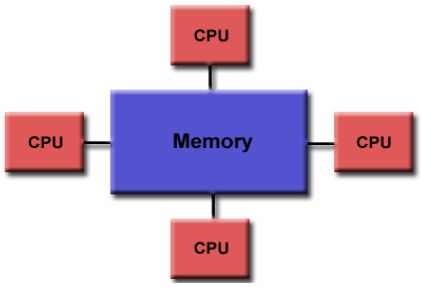

Abbildung 2: Schematischer Aufbau vom Shared Memory Computing [A2]

Die Abbildung 2 veranschaulicht den schematischen Aufbau der CPU/Speicherarchitektur eines Hochleistungsrechners im Modell „Shared Memory Computing“. Alle Prozessoren haben Zugriff auf den gleichen Speicher. Dieser shared memory access ermöglicht dem Programmierer eine leichte Dekomposition der Aufgaben und Daten. Somit ist der Programmieraufwand weniger komplex.

Für das weitere Verständnis ist der Begriff Threading wichtig. Threads sind Aktivitätsträger, welche einzelne Rechenoperationen lösen. Hochleistungsprozessoren, wie zum Beispiel der Xeon-Phi von Intel, haben durch die Manycore-Architektur über 200 dieser Threads. Jeder Thread hat seine eigene Adresse, sowie seinen eigenen Stack. Auf die Threads werden nun die Prozesse verteilt. In einem (Hochleistungs-) Rechner können dann alle vorhandenen Threads parallel gelöst werden.

Eine Programmierschnittstelle, über welchen der eben genannte Prozess läuft, heißt OpenMP. Diese Schnittstelle ist dafür verantwortlich, alle Threads innerhalb eines Systems effizient zu nutzen.

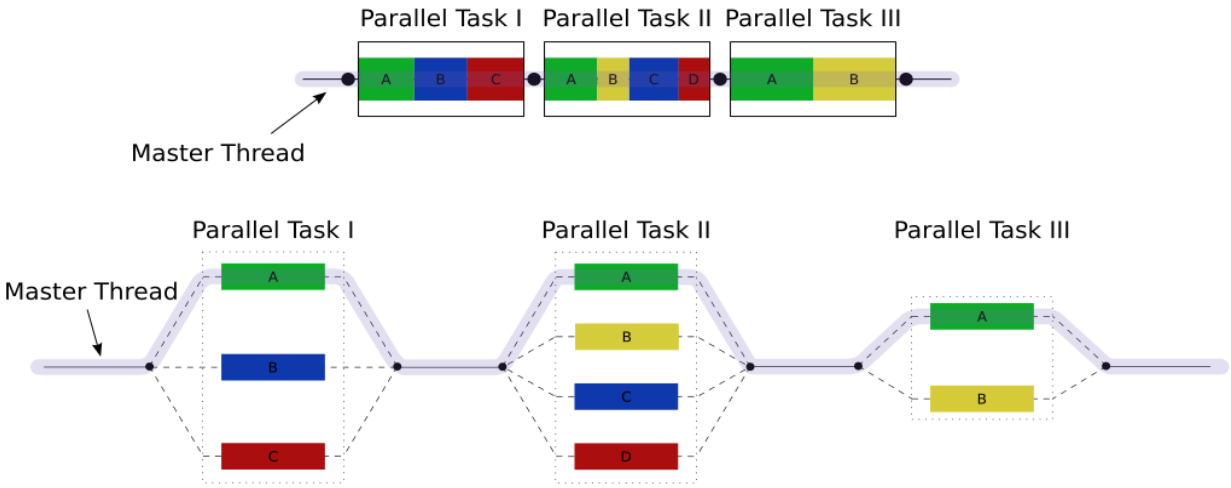

Abbildung 3: Funktionsweise Fork-Join-Modell [A3]

OpenMP funktioniert nach dem Fork-Join-Modell. Zum besseren Verständnis eignet sich die Abbildung 3 sehr gut. Die Aufgabe (Master Thread), die in diesem Beispiel ausgerechnet werden soll, besteht aus drei Aufgabengruppen, die parallel gelöst werden können. Die Aufgabengruppen werden automatisch vom OpenMP auf Threads verteilt, welche dann parallel rechnen. Dieser Prozess des Aufspaltens der Aufgabe in Teilaufgaben wird als der Fork-Abschnitt des Modells bezeichnet. Solche Aufgaben sind zum Beispiel for-Schleifen, die nun nicht mehr hintereinander ausgeführt werden müssen. Nach dem Abschluss aller parallel ausgeführten Threads werden diese über den Join-Teil des Modells wieder zusammengeführt.

Die Abbildung 3 zeigt ein sehr leichtes Fallbeispiel, da die parallel gelösten Threads nicht noch einmal in parallel zu lösende Aufgaben gespalten werden. Wenn dies der Fall ist, werden über den Fork-Teil

noch einmal die Aufgaben in mehrere Threads aufgeteilt. Jedoch werden alle Threads im Laufe des Prozesses wieder zu einem Masterthread zusammengeführt.

Diese Aufgaben werden vom OpenMP automatisch in Threads zerlegt, welches den Programmieraufwand stark senkt und das Benutzen des Hochleistungsrechners leichter macht.

Ein weiterer Vorteil an OpenMP ist der ressourcenschonende Umgang mit Threads. Da diese nicht nach jedem Rechenprozess zerstört werden und somit nicht vor jedem neuen Rechenprozess neu erzeugt werden müssen.

Außerdem ist das Spektrum an Programmiersprachen, welche kompatibel sind mit C, C++ und FORTRAN, relativ groß.

Eine andere Programmierschnittstelle ist Intel's Threading Building Blocks, kurz TBB. Sie ist nur mit C++ kompatibel, hat jedoch noch zusätzliche Vorteile gegenüber OpenMP. TBB hat viele schon vorprogrammierte Features, welche das Programmieren erleichtern. Trotz dieser vielen Features ist die Benutzeroberfläche sehr übersichtlich und einfach gestaltet.

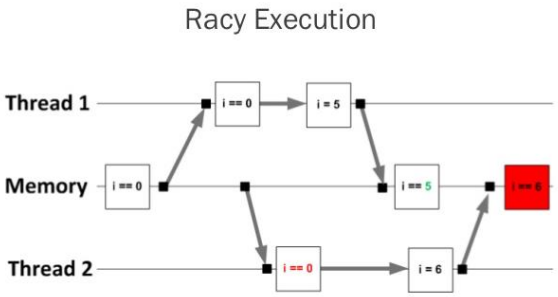

Abbildung 4: Veranschaulichung von Datarace [A4]

Shared Memory Computing hat neben den vielen Vorteilen jedoch auch einige schwerwiegende Probleme.

Das Problem, welches am häufigsten auftreten kann, ist Datarace (Abbildung 4). Da ein Shared Memory Computer nur einen Speicher hat, kann es passieren, dass mehrere Threads auf dieselbe Speicheradresse zugreifen wollen. Dass Thread 1 auf die Speicheradresse i zugreift, auf der der Wert i == 0 gespeichert ist, ist noch kein Problem. Nun addiert Thread 1 zu dem Wert i == 0, 5 dazu, wodurch das Ergebnis i == 5 ist. Thread zwei sollte mit diesem Ergebnis (i == 5) rechnen, greift jedoch zu früh auf die Speicheradresse i zu. Zu diesem Zeitpunkt ist auf i noch 0 gespeichert und nicht 5. Addiert Thread 2 nun zu i == 0, 6 dazu wird i zu i == 6. Nun überschreibt Threat 2 das Ergebnis von Thread 1. Somit ist das Endergebnis i == 6 und nicht i == 11 (5+6). Datarace kann jedoch mittels Barrieren eingeschränkt werden, indem die Bedingung aufgestellt wird, dass erst Thread 1 das Ergebnis auf den Speicherplatz i geschrieben haben muss, bevor andere Threads auf i zugreifen können.

Allerdings entstehen hierdurch zwei neue Probleme. Zum einen der Deadlock, welcher entsteht, wenn sich zwei oder mehrere Barrieren blockieren, wodurch der Rechenprozess nicht fortgeführt werden kann. Wenn dies jedoch nicht passiert, kann es durch die Barrieren auch zu einem großen Zeitverlust kommen, da alle Threads, die auf diese Speicherardesse zugreifen, solange nicht arbeiten können, bis die Barriere aufgelöst wird. Und da es bei Hochleistungsrechnern, vorallem beim Schritt zum Exascale System darum geht, mit so viele Flops wie möglich zu rechnen, können solche Barrieren diese Kennziffer drastisch senken. [13]

4.2 Distributed Memory Computing

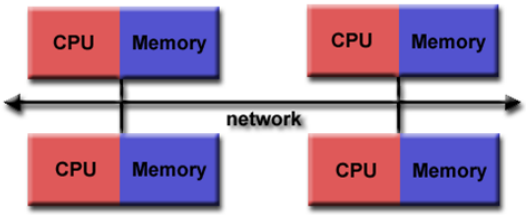

Abbildung 5: Schematischer Aufbau vom Distributed Memory Computing [A2]

Die Abbildung 5 zeigt wieder den schematischen Aufbau, jedoch diesmal von einem Distributed Memory Computer.

Der Unterschied zum Shared Memory Computing ist der, dass das System aus vielen Einheiten besteht, die jeweils aus einem Prozessor und einem Speicher zusammengesetzt sind. Innerhalb einer Einheit können die Prozessoren über den direct access sehr schnell auf den lokalen Speicher zugreifen. Die Einheiten kommuniziernen über ein Netzwerk, zum Beispiel MPI. MPI steht für **M**essage **P**assing **I**nterface und gewährleistet ein aktives Versenden und Nutzen von Daten, die nicht im lokalen Speicher einer Einheit stehen.

Bei der Benutzung vom MPI wird die Dekomposition anders als im Shared Memory Computing vom Programmierer, nicht vom Programm übernommen. Dennoch bietet die Benutzerschnittstelle einige Features zur Beschleunigung des Rechenprozesses, wie zum einen das one-side data movement. Dieses Feature gewährleistet einen Datentransfer, ohne dass eine Bestätigung erfolgt, dass die Daten vollständig sind.

Zum andern gibt es im MPI noch den remote direct memory access. Dieser erlaubt es dem Rechenprozess direkt auf jeden Speicher außerhalb der Einheit zuzugreifen, ohne die Daten erst anfragen zu müssen.

Diese beiden Features beschleunigen den Rechenprozess immens. Jedoch wird der daraus gewonnene zeitliche Vorteil durch den Zeitverlust des erhöhten Programmieraufwandes kompensiert.

Abgesehen von der erhöhten Vorarbeit, welche nötig ist, ist durch die zwei Speicherzugriffsarten, direct memory access und remote memory access, die Speicherorganisation komplizierter und komplexer geworden. Ein weiteres Problem, welches mit dem Speichern im Zusammenhang steht, ist der Nachteil, welcher aus der nicht einheitlichen Zugriffsdauer auf die Speicher einhergeht. Dieser verkompliziert die Programmierung noch einmal.

Ein zusätzlicher Nachteil lässt sich leicht anhand eines Beispieles (Abbildung 6) verdeutlichen:

Abbildung 6: Schematische Darstellung einer Temperatur-verteilung in einem Eisenstab

In diesem soll die Temperatur eines Eisenstabes im Verlauf der Zeit dargestellt werden. An einem Ende befindet sich eine wärmende und an dem anderen eine kühlende Quelle. Der Stab wird der Einfachheit halber nur in drei Abschnitte unterteilt. Jedem Abschnitt wird eine Einheit aus dem Computer zugeordnet.

Damit die Einheit 1 die Temperatur aus ihrem Abschnitt berechnen kann, wird die Temperatur aus der Quelle, die eigene Temperatur und die Temperatur des angenzenden Abschnittes des Stabs benötigt.

Diese müssen in Speicher 1 gespeichert werden, damit die Temperatur für diesen Abschnitt berechnet werden kann.

Dasselbe gilt für die anderen beiden Speicher, wodurch jeder Speicher die dreifache Menge an Daten speichern muss, im Vergleich zum Shared Memory Computing.

Daraus resultiert ein sehr großer Speicherbedarf. [13]

4.3 Hybrid Distributed-Shared Memory Computing

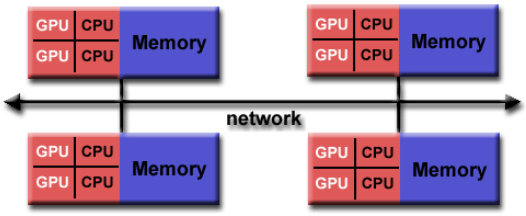

Abbildung 7: Schematischer Aufbau vom Hybrid Distributed-Shared Memory Computing [A2]

Diese Rechnerarchitektur wird schon im Sunway TaihuLight verwendet und soll in Zukunft auch in Aurora angewandt werden. Daher ist das ein Aufbau, der Zukunftspotenzial hat. Das liegt vorallem auch daran, dass durch die neue Architektur viele Probleme der Vorgänger (Distributen Memory Computing und Shared Memory Compuing) durch die Kombination eleminiert werden.

Wie schon in der Abbildung 7 zu sehen ist, ähnelt der schematische Aufbau in seinen Grundzügen an das Distributen Memory Computing. Es exististieren Einheiten, welche über ein Netzwerk, wie MPI, kommunizieren. Jede Einheit besteht wieder aus einem Speicher. Die Besonderheit jedoch ist, dass es jetzt nicht nur einen Prozessor gibt, sondern mehrere. Diese sind innerhalb der Einheit mit dem einen Speicher, wie beim Shared Memory Computing, verbunden. Dadurch können einzelne Prozesse komplett innerhalb einer Einheit gelöst werden, ohne dass die Probleme vom MPI diesen behindern. Die Wahrscheinlichkeit, dass ein Datarace auftritt, wird durch die Eigenschaft des MPI jedoch gesenkt, da es jetzt mehrere Speicher gibt.

Hybrid Distributed-Shared Memory Computing wird mit einem dynamischen [13] Hybridcode aus MPI und OpenMP betrieben. Sofern OpenMP 4.0 oder eine neuere Version genutzt wird, können, wie auch in der Abbildung 7 zu sehen ist, Grafikprozessoren mit eingebunden werden. [15]
Grafikprozessoren sind besser im parallelen Ausführen von Aufgaben als normale Prozessoren, weswegen diese zur Leistungssteigerung genutzt werden. [14]

5 Strategie zur Erreichung der Exascale Marke

Da die grundlegende Funktionsweise geklärt ist, werden nun die Lösungsvorschläge zum Erreichen der Exascale Marke im Folgenden behandelt.

5.1 Lösungsansatz

Eine Grundidee, mit welcher die Hochleistungsrechner immer leistungsfähiger gemacht wurden, ist die Beschleunigung der Prozessoren. Dieses kann zum einen durch das Einsetzen und Nutzen von mehr Kernen erfolgen, zum anderen aber auch durch die größere Speicherkapazität von Caches erreicht werden. Außerdem werden schon Grafikprozessoren benutzt, welche die Leistungsfähigkeit noch einmal steigern und weiterentwickelt werden. Ebenso werden die Vektoreinheiten immer breiter, wodurch mehr Prozesse parallel betrieben werden können. [1]

5.2 Verbleibende Probleme

Die Gründe wieso mit den eben genannten Lösungsansätzen nicht einfach die Exascale Marke geknackt werden kann sind folgende:

Zum einen ist auf den Prozessoren nicht mehr genügend Platz, um ohne weiteres immer mehr Kerne mit einzubinden, zum anderen braucht ein Hochleistungsrechner immer mehr Platz, wenn mehr Prozessoren dem System hinzugefügt werden. Außerdem würde mit der steigenden Anzahl leistungsfähiger Prozessoren der Stromverbrauch massiv steigen. Zusätzlich steigt auch die Wärmeentwicklung an [16], wodurch wiederum der Energieverbrauch der Kühlsysteme steigen würde. [17]

Ein zusätzlicher Nachteil entsteht durch die Einbindung von Grafikprozessoren. Unter anderem sorgen diese für heterogene Systeme, welche zur Folge haben, dass die benötigten Programme sehr komplex sind und die komplette Ausnutzung der gesamten Rechenleistung somit nicht möglich ist (siehe Sunway TaihuLight, tatsächliche und theoretische Spitzenleistung). [13]

Ein weiteres Problem bei den Hochleistungsrechnern ist die Systemsoftware, wie Linux, welche weiterhin genutzt wird, obwohl sie die tatsächliche Rechenleistung eines Hochleistungsrechners mindert. [1]

5.3 Bestehende Problemlösungsansätze

5.3.1 Dreidimensionale Positionierung der Transistoren

Der Platzmangel, die Energieknappheit, sowie die Wärmeentwicklung sollen unter anderem durch die dreidimensionale Positionierung von Transistoren reduziert werden.

Die Forscher setzen an den Leiterbahnen an, welche zum Beispiel die Transistoren in einem Chip verbinden, da diese sehr viel Strom verbrauchen. Dieses liegt daran, dass ein permanenter elektrischer Widerstand an den Leiterbahnen aufrechterhalten werden muss. Dieser permanente Stromverbrauch ist 10-mal höher als der Stromverbrauch, der durch das Schalten der Transistoren verbraucht wird. Somit entstehen 99% des Energieverlustes durch die Leiterbahnen.

Der einfachste Weg, hierbei Strom einzusparen, wäre es, die Leiterbahnen zu verkürzen. Hierdurch würde die Strecke auf der der Widerstand aufrechterhalten werden muss, kleiner werden, wodurch dann auch der Stromverbrauch sinken würde.

Jedoch geht dieses bei der herkömmlichen zweidimensionalen Rechnerarchitektur nicht mehr effizient, da die Transistoren schon so eng aneinander positioniert liegen, dass ein zusätzliches Verkürzen nur schwer möglich ist.

Daher haben sich einige Forscher damit beschäftigt, eine dreidimensionale Rechnerarchitektur zu entwickeln. In dieser sind die Transistoren dreidimensional verteilt. Hierdurch kann der Energieverbrauch um das 100-fache und das Computervolumen durch die kompaktere Bauweise um das 1000-fache reduziert werden.

Ein weiterführender Schritt wäre es, den gesamten Hochleistungsrechner in einer geordneten fraktalen Struktur zu bauen. Hierfür könnte zum Beispiel das Gehirn als Vorlage dienen, da es trotz der immer noch viel stärkeren Rechenleistung viel weniger Wärme, Energie und Platz benötigt. Würde man das nun auf einen Hochleistungsrechner übertragen und nicht nur die Transistoren, sondern das

gesamte System so aufbauen, würde man zusätzlichen den Stromverbrauch um das 30-fache und das Volumen nochmal um das 1000-fache senken. [16]

5.3.2 Wärmereduktion

Die Computerchips können nur bis zu einer Temperatur von 100 Grad Celsius ihre bestmögliche Leistung erreichen. Da die Chips immer leistungsfähiger werden, verbrauchen sie auch immer mehr Energie, womit eine stärkere Wärmeentwicklung einhergeht.

Der erste Lösungsansatz, um dem entgegenzuwirken, ist die **Flüssigkeitskühlung direkt auf dem Chip.** Hierfür wird ein nichtwässriges, nichtreaktives, flüssiges Kühlmittel benutzt. Durch diese Eigenschaften kann es nicht zu einem Kurzschluss kommen, gleichzeitig kann direkt die Oberfläche der Chips gekühlt werden, ohne dass eine isolierende Luftschicht dieses verhindert. Diese Technik wird schon heute im SuperMUC von IBM genutzt, hier werden Fluorkarbone als Kühlmittel eingesetzt.

Ein anderer Ansatz ist es, **Wasser als Elektrolyt-Treibstoff** zu nutzen. Das Ziel hierbei ist die Rückgewinnung von Energie, wodurch dem System weniger Energie zugeführt werden muss, was als Konsequenz bedeutet, dass die Wärmeentwicklung sinkt. Diese Energierückgewinnung findet in redox-flow-Zellen statt. Das Prinzip funktioniert jedoch erst durch die Tatsache, dass das Wasser, welches als Kühlmittel in der Nähe der Computerchips fließt, die von den Chips abgestrahlten Ionen aufnimmt. Mithilfe von diesen Ionen kann dann der Strom, der durch die redox-flow-Zellen erzeugt wird, direkt zum Betrieb von Mikroprozessoren genutzt werden. [16]

5.3.3 Energieverbrauch-Minderung

Eine Studie hat ergeben, dass ein Exascale System maximal 20 Megawatt verbrauchen darf, um aus Sicht des Umweltschutzes vertretbar zu bleiben. Zum Vergleich: ein kleines Kernkraftwerk, erzeugt um die 40 bis 50 Megawatt. Aus dieser Kennziffer lässt sich errechnen, dass ein Hochleistungsrechner mit 50 GigaFlops/Watt rechnen müsste. Der Sunway TaihuLight ist davon jedoch mit ca. 6 GigaFlops/Watt noch weit entfernt. Auch Aurora ist mit 13 GigaFlops/Watt noch weit davon entfernt, jedoch ist das schon einmal ein Schritt in die richtige Richtung.

Werden die Hochleistungsrechner jedoch immer leistungsstärker und komplexer, dann müssen auch die Kühlsysteme immer besser und größer werden. Um den damit einhergehenden, steigenden Energieverbrauch entgegenzuwirken wird an dem Ansatz gearbeitet, mit der Abwärme die Umgebung über Blockheizkraftwerke zu beheizen. Dadurch profitieren die beheizten Gebäude, sowie das Kühlsystem, da weniger zusätzliche Kühlelemente benötigt werden. [17]

5.3.4 Identifikation stark energieverbrauchender Komponenten

Mit einer im Verhältnis zur Konkurrenz preiswerten Lösung wartet das europäische Projekt Exa2Green auf. Im Rahmen dieses Projektes wurde ein Gerät entwickelt, welches so klein ist, dass es direkt in den Hochleistungsrechner integriert werden kann. Es dient der Lokalisierung von Komponenten mit einem sehr hohen Energieverbrauch innerhalb eines solchen Systems. Sofern das geschehen ist, kann die Systemsoftware gezielt angepasst werden. Zusätzlich können diese Komponenten auch hardwaretechnisch neu entworfen und entwickelt werden. Somit kann der gesamte Energieverbrauch eines Hochleistungsrechners deutlich gesenkt werden. [18]

5.3.5 Systemsoftware in Exascale Systemen

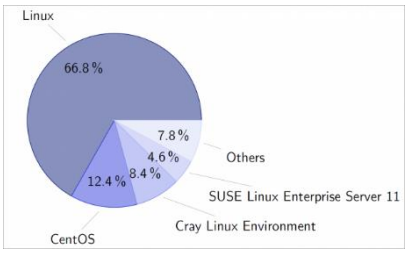

Abbildung 8: Diagramm zur prozentualen Nutzung von Systemsoftware in Hochleistungsrechnern [A5]

In dem Kreisdiagramm in Abbildung 8 ist zusehen, dass der Großteil der verwendeten Systemsoftware von Linux ist. Linux wird vor allem wegen der guten Handhabung und der guten Benutzeroberfläche genutzt. Dennoch hat Linux bezogen auf Hochleistungssysteme eine Komplexität, welche die Laufzeitvorhersage fast unmöglich macht. Außerdem steigt die Anfälligkeit für Fehler und durch die Komplexität entstehen häufig Seiteneffekte, welche den eigentlichen Rechenvorgang stören.

Eine Alternative bieten die Lightweight Kernels. Sie bestehen zum Beispiel nur aus einem 5.000 Zeilen C++ Code. Hierdurch ist das Laufzeitverhalten wieder kalkulierbar. Es werden die eben genannten Seiteneffekte vermieden und es ist nicht mehr so anfällig für Fehler. Jedoch wird diese Möglichkeit nicht häufig genutzt, was man auch in der Abbildung 8 erkennen kann. Das liegt an dem sehr hohen Programmieraufwand, welcher damit einhergeht.

Die Lösung dieses Problems scheint Systemsoftware, wie Zeptos-OS oder ein Multikernel, zu sein. Zeptos-OS ist eine abgespeckte Version von Linux, welche somit nicht so komplex ist und einige Nachteile von Linux somit ausmerzt. Die Idee von einem Multikernel basiert darauf, dass die guten Eigenschaften von einem Fullweight Kernel Linux, wie die Benutzeroberfläche, mit denen von einem Lightweight Kernel kombiniert werden. Dazu sind die Betriebssysteme auf verschiedene Kerne verteilt, welche dann unterschiedliche Aufgaben bekommen. Somit entsteht eine sehr gute Kombination, mit Vorteilen aus beiden Arten. [1]

6 Resilienz

Ein weiterer Aspekt bei der Entwicklung von leistungsstärkeren Hochleistungsrechnern ist die Resilienz.

Resilienz steht für die Widerstandsfähigkeit, d.h. die Fähigkeit, mit Problemsituationen beim Rechnen umgehen zu können. [19]

Ein großer Nachteil der Leistungssteigerung eines Hochleistungsrechners ist, dass das System und die Software komplexer und größer werden, wodurch die Fehlerwahrscheinlichkeit steigt. Dieses liegt daran, dass mehr Komponenten in einem solchen System existieren, welche ausfallen können. Genauso wie an den darauf laufenden Programmen, müssen bzw. können diese immer komplexer werden. Dadurch wird aber im Gegenzug die Wahrscheinlichkeit größer, dass ein Fehler in das Programm einprogrammiert wurde oder bei der Ausführung ein Fehler passiert.

Die Zeit, die gebraucht wird, um diese Fehler zu lokalisieren, steigt mit der der steigenden Komplexität ebenso an. [20]

Fehler-**To**lerante Umgebung für Petascale MPI **L**öser, abgekürzt „FeTol“, ist der Name für ein deutsches Forschungsprojekt, welches sich mit der Resilienz der Supercomputer beschäftigt. Das Ziel ist die Ausfallsicherheit für Petascale Systeme zu erhöhen. Sobald Exascale Systeme auf dem Markt sind, wird die Funktionsweise auch auf größere Hochleistungsrechner übertragbar sein.

FeTol funktioniert nach dem devide-and-conquer Prinzip. Das heißt, dass der Gesamtprozess in einzelne Prozesse aufgeteilt wird und dass ähnliche Prozesse in Prozessbündeln (BP) gruppiert werden. Innerhalb eines Prozessbündels kommunizieren die Prozesse über ein MPI, wohingegen die Prozesse aus unterschiedlichen Prozessbündeln über ein eigens dafür entwickeltes Multi-Agent-System kommunizieren. Dieses heißt BOND.

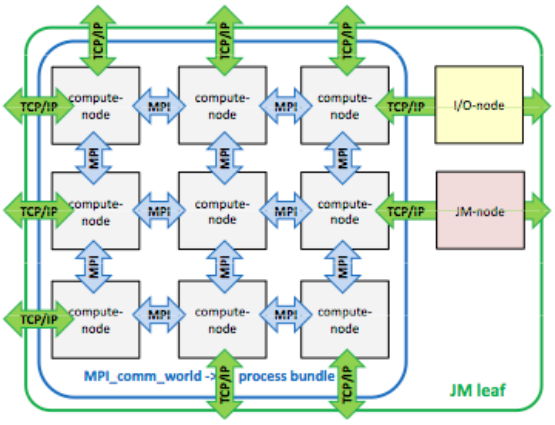

Abbildung 9: Funktionsschema eines JM leafs mit FeToL [A6]

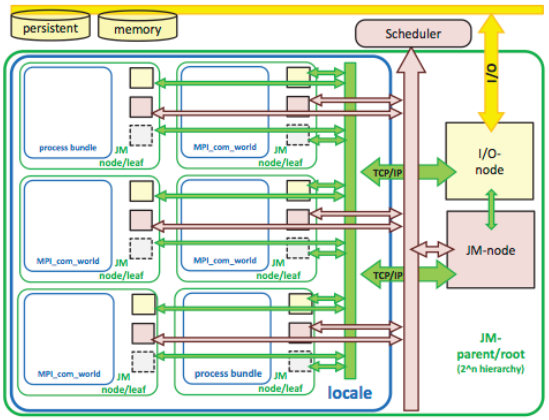

Abbildung 10: Funktionsschema Verknüpfung von JM leafs durch FeToL [A6]

In der Abbildung 9 ist der Aufbau eines Jobmanager leafs zu sehen. Dieses besteht aus zwei Verbindungspunkten zur Steuer- und zur Speichereinheit. Außerdem beinhaltet er noch einen Prozessbündel. Innerhalb dieses Prozessbündels befinden sich Computerknoten, welche über ein MPI verbunden sind und eine Verbindung (Bond) zu anderen Jobmanager leafs bzw. Prozessbündeln haben. Diese sieht man in der Abbildung 10 noch einmal deutlicher.

Fallen nun ein oder mehrere Prozesse aus, können diese Daten wiederhergestellt werden, ohne das gesamte Programm neu zu starten. Hierzu ist nur der Neustart eines Prozessbündels notwendig. Die fehlenden Daten werden mithilfe von Checkpointdaten aus dem Memory gelesen und mithilfe von Daten der Prozessnachbarn ergänzt. Das funktioniert, da durch das MPI Prozessnachbarn ihre Daten gegenseitig sichern. Dadurch können die Daten rekonstruiert werden, die in dem ursprünglichen Prozess verloren gegangen sind.

Somit ist FeTol eine sehr gute Möglichkeit, die Resilienz von Hochleistungsrechner drastisch zu verbessern, ohne dabei den Rechenprozess zu verlangsamen oder Unmengen von Speicher zu benötigen. [20]

7 Quellen

[1]: https://www.golem.de/news/supercomputer-wie-die-exaflop-marke-geknackt-werden-soll-1610-122823.html 21.05.2017

[2]: http://www.spiegel.de/netzwelt/netzpolitik/nsa-will-super-computer-zum-ausspaehen-entwickeln-a-941604.html 21.05.2017

[3]: https://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf 21.05.2017

[4]: https://de.wikipedia.org/wiki/Floating_Point_Operations_Per_Second 21.05.2017

[5]: https://www.top500.org/lists/2016/11/ 21.05.2017

[6]: http://www.spiegel.de/netzwelt/gadgets/chinas-sunway-taihulight-ist-schnellster-supercomputer-a-1098599.html 21.05.2017

[7]: https://www.top500.org/system/178764 21.05.2017

[8]: https://www.nextplatform.com/2016/06/20/look-inside-chinas-chart-topping-new-supercomputer/ 21.05.2017

[9]: http://www.intel.com/content/www/us/en/high-performance-computing/aurora-fact-sheet.html 21.05.2017

[10]: https://webdocs.cs.ualberta.ca/~paullu/Aurora/aurora.html 21.05.2017

[11]: http://www.chinadaily.com.cn/china/2017-02/20/content_28259294.htm 21.05.2017

[12]: https://m.heise.de/newsticker/meldung/EU-will-bei-Exascale-Computing-aufholen-3710281.html?wt_ref=android-app%3A%2F%2Fcom.google.android.googlequicksearchbox%2Fhttps%2Fwww.google.com&wt_t=1494494495705396 21.05.2017

[13]: https://software.intel.com/en-us/articles/hybrid-parallelism-parallel-distributed-memory-and-shared-memory-computing 21.05.2017

[14]: https://computing.llnl.gov/tutorials/parallel_comp/#Whatis 21.05.2017

[15]: https://en.wikipedia.org/wiki/OpenACC 21.05.2017

[16]: http://www.spektrum.de/news/kampf-gegen-die-hitze/1180336 21.05.2017

[17]: http://www.cmcc.it/wp-content/uploads/2012/05/rp0121-sco-12-2011.pdf 21.05.2017

[18]: https://www.steinbeis-europa.de/files/transfer3-2015_exa2_green.pdf 21.05.2017

[19]: http://resilienz-freiburg.de/index.php/was-ist-resilienz/definition-und-merkmale 21.05.2017

[20]: https://www.tu-braunschweig.de/irmb/forschung/abgeschlosseneprojekte/fetol 21.05.2017

8 Abbildungsquellen

[A1]: https://www.nextplatform.com/2016/06/20/look-inside-chinas-chart-topping-new-supercomputer/ 21.05.2017

[A2]: https://computing.llnl.gov/tutorials/parallel_comp/Whatis 21.05.2017

[A3]: https://en.wikipedia.org/wiki/Fork%E2%80%93join_model#/media/File:Fork_join.svg 21.05.2017

[A4]: https://www.google.de/imgres?imgurl=https%3A%2F%2Fimage.slidesharecdn.com%2Fdrdsecrfinal3-121106080213-phpapp01%2F95%2Fdynamic-data-race-detection-in-concurrent-java-programs-5-638.jpg%3Fcb%3D1352191909&imgrefurl=https%3A%2F%2Fwww.slideshare.net%2FDevexperts%2Fdynamic-data-race-detection-in-concurrent-java-programs&docid=SSwmwgSXulLQKM&tbnid=HhCf5uZ9AjNkuM%3A&vet=10ahUKEwjvm5jfjsTTAhVGAxoKHYOvDGsQMwgpKAQwBA..i&w=638&h=479&bih=702&biw=1536&q=data%20race&ved=0ahUKEwjvm5jfjsTTAhVGAxoKHYOvDGsQMwgpKAQwBA&iact=mrc&uact=8#spf=1 21.05.2017

[A5]: https://www.golem.de/news/supercomputer-wie-die-exaflop-marke-geknackt-werden-soll-1610-122823-3.html 21.05.2017

[A6]: https://www.tu-braunschweig.de/irmb/forschung/abgeschlosseneprojekte/fetol 21.05.2017