UNIVERSITÄT ZU LÜBECK

Benchmarking eines HPC-Clusters zur Genomanalyse am Beispiel des Lübecker OMICS-Clusters

Benchmarking of an HPC cluster for genom analysis on the example of the OMICS-Cluster in Luebeck

Bachelorarbeit

verfasst am
Institut für Softwaretechnik und Programmiersprachen

im Rahmen des Studiengangs
Bachelor Informatik
der Universität zu Lübeck

vorgelegt von
Frank Rühlemann

ausgegeben und betreut von
Prof. Dr. Martin Leucker

mit Unterstützung von
Malte Schmitz

Die Arbeit ist mit freundlicher Unterstützung des OMICS-Clusters der Universität zu Lübeck durch die Bereitstellung von Rechenressourcen entstanden.

Lübeck, den 26. August 2020

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Frank Rühlemann

Zusammenfassung

Das OMICS-Cluster ist ein HPC-Rechencluster an der Universität zu Lübeck, welches primär für die Genomanalyse eingesetzt wird. Da hierbei die Verarbeitung großer Datenmengen mit meist mäßiger Parallelisierung im Vordergrund steht, schlägt sich dies auch im technischen Aufbau nieder. Entsprechend liegt der Schwerpunkt auf der Bereitstellung einer sehr großen Storage-Kapazität und sehr potenter Rechenknoten.
In dieser Arbeit wird beleuchtet, ob das Cluster diesen Anforderungen gerecht werden kann. Dafür wurden die verschiedene ausgestatteten Rechenknotentypen mit je drei Benchmarks vermessen und an Hand der Ergebnisse zu ihrer Eignung für die Genomanalyse beurteilt.

Abstract

The OMICS-Cluster is an HPC compute cluster at the University of Luebeck which is primarly used for genome analysis. The main purpose is the processing of large data files with algorithms that often have a medium grade of parallelization. This is represented by the structure of the cluster. Therefore is the main focus on high storage capacity and more powerful compute nodes.
This thesis will examine wether the cluster meets the requirements. Therefore three benchmarks were run on the different compute node types and on the basis of the results the suitability of these nodes for genome analysis was evaluated.

Inhaltsverzeichnis

1

Einleitung

Während früher vor allem mathematische, physikalische oder chemische Berechnungen auf HPC-Systemen durchgeführt wurden, haben mit der Zeit auch andere wissenschaftliche Disziplinen Möglichkeiten zur Nutzung dieser großen Maschinen gefunden. So wurde zum Beispiel das OMICS-Cluster an der Universität zu Lübeck explizit für die Genomanalyse vorgesehen. Unter dem Stichwort Big Data steht hierbei vor allem die Verarbeitung großer Datenmengen im Vordergrund. Daher ergeben sich verschiedene Fragen darüber, wie aussagekräftig die üblichen Benchmarks wie Linpack oder HPCG für die Genomanalyse sind, da sie ursprünglich für die Beurteilung eines HPC-Systems zur Eignung bei anderen Problemen entwickelt wurden. Diesen Fragen wird in dieser Arbeit auf den Grund gegangen.

1.1 Wissenschaftliche Fragestellungen

Die Beschaffung und genaue Ausgestaltung eines HPC-Clusters gestaltet sich schwierig, da bereits die Vielzahl der möglichen CPU-Modelle erschlagend wirken kann. Hinzu kommen die möglichen RAM-Konfigurationen und anderen Ausstattungsoptionen für jede einzelne Maschine, die jeweils Einfluss auf die Performance hat.

Da das Cluster über die letzten Jahre hinweg immer wieder mit verschiedensten Rechenknoten erweitert wurde, ergibt sich nun die Möglichkeit, um den Einfluss verschiedenster Faktoren auf die Eignung zur Genomanalyse zu evaluieren. Mit den insgesamt sechs verschiedenen Intel-Prozessortypen in vier Generationen stellt sich die Frage, welche sich besonders bewähren und wie sie sich im Verhältnis zu dem neuen, deutlich günstigeren AMD EPYC-Prozessor verhalten. Ähnlich verhält es sich mit der Größe des Arbeitsspeichers. Einige Rechenknoten besitzen deutlich mehr RAM und sind entsprechend kostenintensiver. Daher soll im Rahmen dieser Arbeit auch geklärt werden, ob diese Mehrkosten auch tatsächlich eine messbare Mehrleistung bringen.

Im HPC-Umfeld werden einige Standard-Benchmarks immer wieder zum Vergleich herangezogen, welche allerdings ursprünglich die Eignung für andere mathematische Problemstellungen evaluieren sollen. Im Rahmen dieser Arbeit wurden daher auch die beiden geläufigsten Vertreter Linpack und HPCG durchgefhrt und dann deren Aussagekraft für die Genomanalyse untersucht.

Im Zuge dieser Arbeit sollen daher folgende Fragen durch die genutzten Benchmarks

geklärt werden:

1. Welche der im OMICS-Cluster verbauten CPUs ist besonders für die Genomanalyse geeignet?
2. Sind neuere CPU-Generationen generell besser geeignet?
3. Wie wirkt sich die RAM-Größe auf die Performance aus?
4. Welchen Einfluss hat die RAM-Größe auf die Eignung zur Genomanalyse?
5. Welche Aussagekraft hat ein hoher GFLOP/s-Wert in Linpack und HPCG auf die Genomanalyse?
6. Wie verhalten sich Rechenknoten mit AMD EPYC-CPUs im Vergleich zu ähnlich ausgestatteten Maschinen mit Intel-Proesszoren?

Zur Beantwortung dieser Fragen werden die in Kapitel 3.3 vorgestellten Benchmarks genutzt. Diese werden jeweils nochmal einzeln genauer vorgestellt. Im Anschluss werden die gewonnen Ergebnisse präsentiert und interpretiert.

1.2 Aufbau dieser Arbeit

Zu Beginn dieser Arbeit werden die wichtigsten Begriffe definiert. Hierzu werden Definitionen anderer Arbeiten verglichen. Es folgt eine Einordnung des OMICS-Clusters. Abgeschlossen wird der erste Teil durch eine Beschreibung des Aufbaus und des schrittweisen Ausbaus. Dies soll verdeutlichen, warum die verschiedenen Rechenknotentypen in ihrer jetzigen Konfiguration vorhanden sind.

In Kapitel 3 folgt dann eine genauere Erläuterung, was ein Benchmark ist, welche ausgesucht wurden und warum diese für die Betrachtung des OMICS-Clusters relevant sind. Hierzu gehört auch eine genauere Beschreibung der Testbedingungen.

Nach diesen eher theoretischen Abschnitten werden die Benchmarks einzeln betrachtet, ihre Ergebnisse vorgestellt und ausgewertet. Hierzu findet sich ein eigener Abschnitt pro Benchmark.

Abgerundet wird die Arbeit mit einer übergeordneten Betrachtung der Benchmarkergebnisse und Empfehlungen für einen zukünftigen Ausbau.

Die genauen Jobscripte, die zur Testdurchführung genutzt wurden, finden sich im Anhang A.

2

Das OMICS-Cluster

2.1 Einordnung

Definition: HPC

Zunächst einmal stellt sich die Frage nach der Definition eines HPC-Clusters. Hierbei sind zwei Elemente zu beachten: *HPC* (High Performance Computing) und *Cluster*. Während sich ein Cluster als *Rechnerverbund*, also dem Zusammenschluss verschiedener Einzelmaschinen, noch recht leicht erklärt, sieht dies beim *High Performance Computing* anders aus. Hier muss sowohl die zeitliche Entwicklung, als auch der technische Stand zum jeweiligen Zeitpunkt, betrachtet werden.

Von Bauke und Mertens wird ein HPC-Cluster in *Cluster Computing*, S. 15 aus 2006 so beschrieben:

> Cluster Computing ist dabei, die Welt des wissenschaftlichen Rechnens zu revolutionieren. Dabei ist die Idee bestechend einfach: man nehme eine Handvoll Rechner, verbinde sie mit handelsüblicher Netztechnik und installiere frei verfügbare Software, die aus den vernetzten Rechnern einen Parallelrechner macht. Damit haben Sie ein System, das für ein Minimum an Kosten ein Maximum an Rechenleistung bietet.

Dies verdeutlicht zwar die grobe Richtung, ist allerdings zu stark vereinfacht, da moderne Cluster mit Infiniband und ähnlichen Technologien teils weit weg von „handelsüblicher Netztechnik“ sind. Außerdem trifft „eine Handvoll Rechner“ weder bei modernen Clustern wie dem Fugaku [30] mit 158.976 Rechenknoten [22] noch beim OMICS-Cluster mit seinen 34 Rechenknoten sinnvoll zu. Die Maschinen sind jenseits von dem, was einen Standard-PC ausmacht, denn es sind speziell für den Anwendungszweck zusammengestellte Server, die auf die jeweiligen Bedürfnisse möglichst genau zugeschnitten sind oder gar für diese Cluster eigens entwickelt wurden.

Arora kommt in *Conquering Big Data with High Performance Computing*, S. 3 von 2016 [1] zu dieser Definition:

> HPC is the use of aggregated high-end computing resources (or supercomputers) along with parallel or concurrent processing techniques (or algorithms) for solving both compute- and data-intensive problems. These problems may or may not be memory-intensive. The terms HPC and supercomputing are often used interchangeably.

Diese Beschreibung geht deutlich besser auf die für HPC typischen Eigenschaften ein. HPC-Systeme sind in der Regel „high-end“, also mit sehr neuer Technik ausgestattet und stellen in großen Clustern oft das aktuelle Maximum an finanzierbarer Rechenleistung dar. Gleichzeitig werden auch direkt die beiden Hauptanwendungsfälle angesprochen: „compute- and data-intensive problems“, also rechen- oder datenintensive Problemstellungen. Letztere zielt bereits auf den Schwerpunkt Big Data, der am Ende diesen Abschnitts näher betrachtet wird. Eher nebenbei wird noch der Unterschied zwischen einem Cluster und einem Supercomputer erwähnt: Während Cluster eine Zusammenstellung („aggregated“) von Rechenkapazität durch Zusammenschaltung von Einzelsystemen erzeugen, sind Supercomputer besonders große Einzelsysteme. In einem Cluster läuft also ein Betriebssystemkernel pro Rechenknoten, während auf einem Supercomputer ein einziger Kernel die kompletten Ressourcen verwaltet. Entsprechend stellt ein Supercomputer eine besonders große Shared-Memory-Maschine dar. Dieser Unterschied wirkt sich deutlich auf die Nutzung aus.

Daher erscheint die Definition von Arora deutlich besser auf moderne HPC-Cluster zu passen. Aber auf dessen Grundlage folgt nun eine eigene Definition:

Definition 2.1 (HPC-System). Ein HPC-System stellt seinen Nutzern eine Rechenleistung bereit, die weit über die der gängigen Destktop-PCs hinaus geht. Dies kann je nach Anforderung der betreibenden Einrichtung mit einem Schwerpunkt auf CPU-Core-Anzahl, RAM-Größe oder Speicherkapazität liegen. HPC-Systeme können entweder durch eine besonders große Maschine (Supercomputer), den Zusammenschluss mehrerer Rechenknoten zu einem HPC-Cluster oder einer Kombination aus beidem erstellt werden. In der Regel werden rechen- oder datenintensive Probleme verarbeitet, wobei üblicherweise auf eine hohe Parallelisierung durch verschiedene Techniken geachtet wird.

Definition: Genomanalyse

Die Genomanalyse bezeichnet die Analyse und wissenschaftliche Auswertung von Genomdaten. Sie setzt sich aus der DNA-Sequenzierung und der DNA-Sequenzanalyse zusammen.

Auf die *Sequenzierung* wird hier nicht näher eingegangen, da dies die Digitalisierung von Basenpaaren des DNA-Stranges mittels biochemischer und technischer Verfahren ist. Dieser Schritt dient der Abtastung des genetischen Materials und findet in entsprechenden Anlagen außerhalb des Rechenclusters statt. Als Endprodukt entstehen entsprechend lange Strings der Basenpaare T,C,G und A, die zur Weiterverarbeitung genutzt werden.

Nachdem die Basenpaarreihenfolge in digitaler Form vorhanden ist, findet die *Sequenzanalyse* mittels Computern statt. Hierbei geht es um die genauere Bestimmung der enthaltenen Genome. In großen Datenbanken, wie GenBank, finden sich Millionen von Einträge zu den verschiedensten Genomen. Zur Bestimmung werden Stringalgorithmen genutzt. Zu den üblichsten Verfahren gehören der FASTA- und der BLAST-Algorithmus. Letzterer wird im Bereich des NCBI-BLAST-Benchmarks 4.4 näher erläutert.

Um diese großen Suchen zu beschleunigen, werden viel RAM und schnelle Datenträger benötigt, um möglichst schnelle Zugriffe auf die Datenbanken zu ermöglichen. Da diese großen Datenbanken entsprechend vorgehalten und dauerhaft neben den zu analysierenden Daten abgelegt werden müssen, kommt auch dem Storage-System ein entscheidender Faktor zu.

Definition: Big Data

Zusätzlich sollte im Zusammenhang von der Genomanalyse noch der Begriff *Big Data* betrachtet werden, der in dem Zusammenhang immer wieder genannt wird. Dieser ist leider schwierig zu definieren, da sich durch den technischen Fortschritt das Verständnis von großen Datenmengen ständig ändert.

Arora beschreibt ihn in *Conquering Big Data with High Performance Computing*, S. 3 so:

> Recent advancements in the field of instrumentation, adoption of some of the latest Internet technologies and applications, and the declining cost of storing large volumes of data, have enabled researchers and organizations to gather increasingly large and heterogeneous datasets. Due to their enormous size, heterogeneity, and high speed of collection, such large datasets are often referred to as "Big Data".

Hierbei wird direkt auf die Hauptaspekte eingegangen: Bereithaltung und Verarbeitung immer größerer Datenmengen. Dabei stellt die Größe den für diese Arbeit relevantesten Faktor dar, da die Heterogenität der Daten eher ein inhaltliches Problem ist, welches im Zuge der konkreten Verarbeitungspipeline beachtet werden muss. Im Rahmen dieser Arbeit wird daher auf diesen wichtigen Punkt nicht näher eingegangen, da er stark von den konkreten Datenquellen und wissenschaftlichen Berechnungen abhängt. Allerdings spricht diese Definition noch nicht die eigentlichen Probleme von Big Data an: Storagekosten und Rechenkapazität. Trotz sinkender Preise für Datenträger, ist Beschaffung und Betrieb des entsprechend dimensionierten Datenspeichers ein merkbarer Kostenfaktor. So müssen die entsprechenden Maschinen durch Abnutzung regelmäßig (üblicherweise nach 5 Jahren) ausgetauscht werden. Gleichzeitig muss die Storage-Lösung performant genug sein, um schnell die benötigten Daten liefern zu können.

Eine Definition könnte daher sein:

Definition 2.2 (Big Data). Big Data bezeichnet verschiedene Aspekte, Potentiale und Probleme bei der Verarbeitung ungewöhnlich großer Datenmengen. Hierbei übersteigen die Datenmengen oder die Auswertungskomplexität deutlich die Rechen- und Speicherkapazität typischer Desktop-PCs. Daher werden oft HPC-Systeme zur Verarbeitung der Datenmengen in akzeptabler Zeit benötigt. Auch fallen bei der Beschaffung und dem Betrieb der großen Datenspeicher trotz sinkender Preise meist hohe Kosten an, da die Daten auch schnell abrufbar sein müssen.

Einordnung

Klassische wissenschaftliche Disziplinen wie die Physik und Mathematik nutzen HPC-Systeme oft mittels MPI-Jobs. Dies sind gut parallelisierbare Algorithmen, die gegen MPI-Bibliotheken entwickelt und kompiliert werden, um die Rechenkapazität mehrerer Rechenknoten für eine Berechnung zusammenschalten zu können.

Dem gegenüber stehen die Lebenswissenschaften, wie die Genomanalyse, deren Softwaretools oft nur mäßig oder gar nicht über Rechenknotengrenzen hinweg parallelisierbar sind. Daher wird hier oft die Parallelisierung durch mehrere voneinander unabhängige Jobs realisiert, die jeweils einen Teil der Daten verarbeiten.

Entsprechend stellt das OMICS-Cluster der Universität zu Lübeck ein HPC-Cluster dar, welches spezielle für die Big-Data-Anwendungen der Genomanalyse konstruiert wurde.

2.2 Aufbau

In diesem Abschnitt soll nun einmal der genaue Aufbau des OMICS-Clusters betrachtet werden. Da es aus vielen verschiedenen Komponenten besteht, werden diese getrennt voneinander beschrieben, obwohl sie sich natürlich aufeinander auswirken.

Nachfolgend werden zuerst die Rechenknoten, die die eigentlichen Berechnungen durchführen, vorgestellt. Dem folgt das GlusterFS-System als Hauptdatenspeicher und das Netzwerk als Verbindungselement zwischen all diesen Komponenten.

Auf eine detailierte Beschreibung der Verwaltungsmaschinen wird aus Übersichtsgründen verzichtet. So dient zum Beispiel der Headnode primär den Nutzern als Einsprungpunkt und zur Verwaltung ihrer Jobs, hat allerdings keinen Einfluss auf die Leistungsfähigkeit der Rechenknoten.

Rechenknoten

Das OMICS-Cluster besteht aus Rechenknoten für verschiedene Aufgaben und aus verschiedenen Beschaffungszeiträumen. Diese Heterogenität erschwert den Vergleich, ermöglicht im gleichen Zuge allerdings auch eine höhere Flexibilität für die Nutzer des OMICS-Clusters. Es unterscheidet sich daher von klassischen HPC-Clustern, die in der Regel eine Vielzahl identischer beziehungsweise sehr ähnlicher Rechenknoten besitzen.

In der Tabelle 2.7 wurden die verschiedenen Rechenknoten aufgelistet und in Typen unterteilt. Hierbei liegt das Hauptaugenmerk in der Generation der Maschinen, der CPU und der RAM-Ausstattung. In den meisten Rechenknoten wurden aus Kostengründen Festplatten als lokaler Datenspeicher genutzt, nur einige wenige besitzen SSDs. Diese lokalen Datenspeicher beherbergen jeweils das Betriebssystem des Rechenknotens, die installierte wissenschaftliche Software und einen größeren Bereich für den Scratch-Speicher */scratch*. Dieser Scratch-Speicher steht den Nutzern während der Joblaufzeit zur Verfügung und ist deutlich reaktionfreudiger als die Netzwerkdateisysteme. Damit eignet er sich besonders für temporäre Zwischenergebnisse. Vor Beginn des Jobs wird für diesen in der Regel ein Unterordner in /scratch automatisch vorbereitet und der Pfad dahin über die Variable $SCRATCH in das Jobscript gereicht. Nach Abschluss der Nutzerbefehle und damit am Jobende wird der Bereich $SCRATCH automatisiert wieder entfernt, um Platz für den nächsten Job frei zu machen. Diese Automatismen wurden über Prolog- und Epilog-Scripte [29] realisiert.

Unter den Rechenknoten nehmen die Maschinen des Typs *SM7371-512* eine Sonderstellung ein, da sie mit AMD-Prozessoren ausgestattet wurden. Diese sind in der Beschaffung merkbar günstiger, allerdings existierte im IT-Service-Center zum Zeitpunkt der Beschaffung keine Erfahrung mit diesen neuen EPYC-Prozessoren.

Dateispeicher

Wie die Definitionen von *Big Data* und die Einordnung in 2.1 gezeigt haben, kommt den Datenspeichern im OMICS-Cluster eine besondere Bedeutung zu. Nach der Nutzung von zwei großen Storage-Systemen vom Typ Huawei S2200T mit einer Kapazität von zusammen ca. 300TB wurde dieses Speicherkonzept wegen des Nadelöhrs der FibreChannel-Kapazität und der schwierigen Erweiterbarkeit durch eine GlusterFS-Umgebung abgelöst. Diese wurde

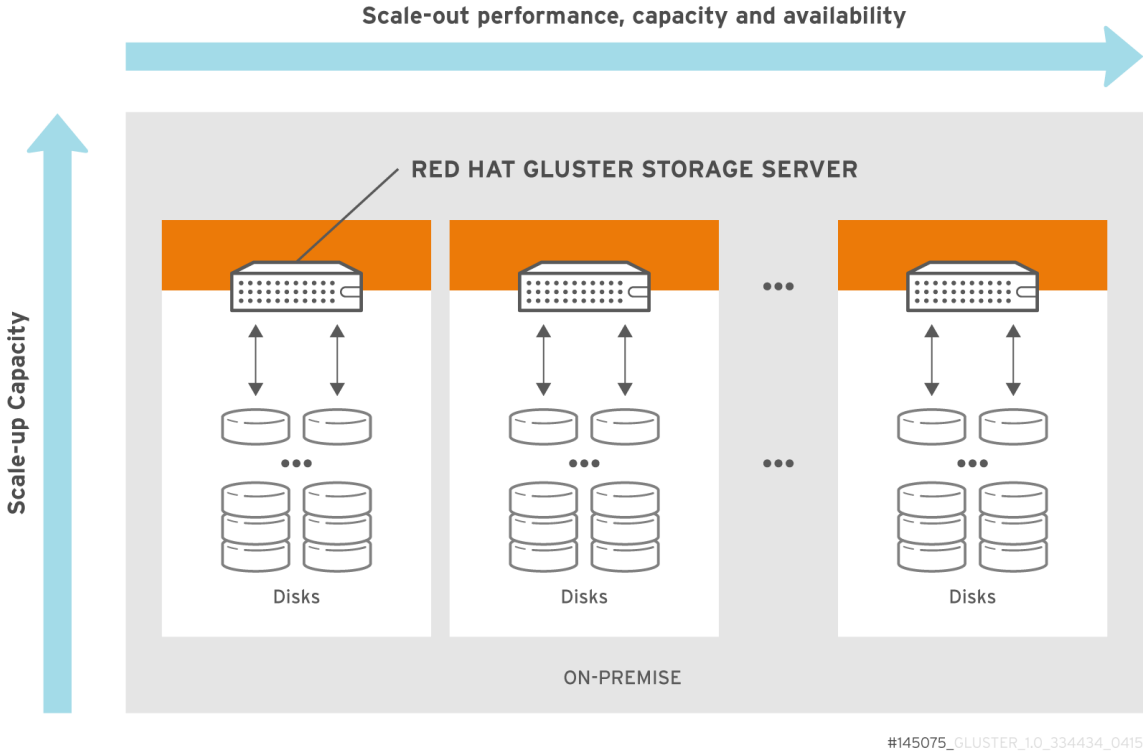

Abbildung 2.1: Schematische Übersicht der Architektur eines GlusterFS-Clusters [27]

dann für eine höhere Stabilität und leichtere Erweiterbarkeit intern umstrukturiert und den Bedingungen angepasst, damit der Speicher besser an die Anwendungsbedingungen angepasst werden konnten.

Auf Grund der verschiedenen Anforderungen kommen im OMICS-Cluster inzwischen verschiedene Techniken zum Einsatz. So wird GlusterFS für die Projektverzeichnisse */data* und die Home-Verzeichnisse genutzt, während */work* auf NFS basiert.

Das *GlusterFS* basiert im Gegensatz zu klassischen Storage-Systemen, wie der anfangs verwendeten Huawei S2200T, nicht auf einzelnen proprietären Komponenten. Stattdessen werden mehrere Server zu einem Cluster verbunden, die gemeinsam den Datenspeicher darstellen und verwalten. Diese horizontale Skalierbarkeit wird in Abbildung 2.1 deutlich. Eine Speichererweiterung erfolgt im OMICS-Cluster nun typischerweise durch die Einfügung weiterer Server. So ist das GlusterFS-System des OMICS-Clusters nach mehreren Jahren inzwischen auf 12 Server gewachsen, die gemeinsam über 700TB an Speicherplatz in */data* und ca. 3,8TB in */home* nutzbare Kapazität bereitstellen.

Um den unterschiedlichen Anforderungen von */home* und */data* gerecht zu werden, wurden zwei Volumes[1] aufgebaut: ein Distributed Dispersed [24, S. 92] für */data* und ein Distributed Replicated [24, S. 77] für */home*. Die Wahl wurde primär mit der Abwägung zwischen finanziellen Kosten und Performance getroffen. Ein GlusterFS-Server in der aktuell verbauten Konfiguration kostet ca. 12.000€ und stellt dem GlusterFS 12 Bricks[2] mit je

[1] Volumes sind in der GlusterFS-Terminologie ein Zusammenschluss von Bricks zu einem mountbaren Verbund. Volumes kommen in verschiedenen Typen vor. [24, S. 11]

[2] Bricks stellen im GlusterFS das kleinste nutzbare Element dar und sollte in der Regel einem (ggf.

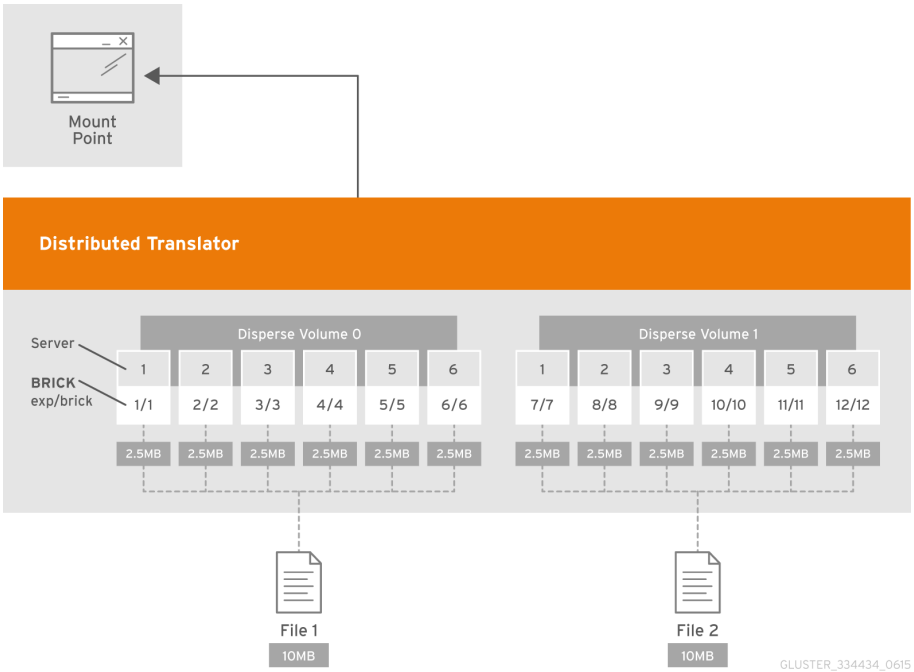

Abbildung 2.2: Schematischer Aufbau eines GlusterFS-Volumes des Typs Distributed Dispersed
Gut sichtbar ist die Aufteilung der Datei mittels Erasure Codings und der um ca. 50% höhere Speicherplatzverbrauch der Dateien im Backend. [25]

8TB zur Verfügung. Beide Volumes bauen auf Dreiergruppen von Servern auf, sodass immer in Vielfachen von drei Servern erweitert werden muss. Eine Erweiterung oder abnutzungsbedingte Ablösung muss also entsprechend auch immer in Dreiergruppen von zusammen ca. 36.000€ teuren Servern erfolgen. Dies stellt eine gut planbare Größenordnung dar. Auch wird eine kostspielige Ersetzung des gesamten Storage-Systems nach Ende der Wartungsverträge vermieden, welche sonst bei klassischen Storages anfällt, da bei Technologiewechseln oft einzelne Teile (wie der Controller) nicht oder nur sehr teuer weiter verlängert werden können.

Aus Performancesicht, insbesondere in Bezug auf die Small-File-Performance von sich häufig ändernden Daten, wäre ein reines Distributed-Volume am sinnvollsten, da hierbei die wenigsten Operationen im Backend der Gluster-Server notwendig ist. Wie in Tabelle 2.4 zu sehen ist, ist es in Bezug auf nutzbare Speichergröße auch die günstigste Volumeart, da alle Bricks voll genutzt werden können. Allerdings sind dann auf Grund fehlender Redundanz bei Ausfall eines Bricks oder Servers dessen Daten nicht mehr für die Clients (z.B. die Rechenknoten) verfügbar oder bei Defekten gar dauerhaft verloren. [24, S. 73]

Da Redundanz sehr wichtig ist, um auch den Ausfall einzelner Festplatten oder ganzer Server verkraften zu können, stehen also nur Distributed Replicated und Distributed Dispersed zur Auswahl. Allerdings fallen bei einem Distributed Replicated auch die höchsten Kosten an, da von jeder Datei mehrere Kopien vorgehalten werden müssen. Um Probleme nach einem Verbindungsausfall zwischen den Servern (Splitbrain) zu vermeiden sind es

virtuellen) Datenträger entsprechen. [24, S. 10]

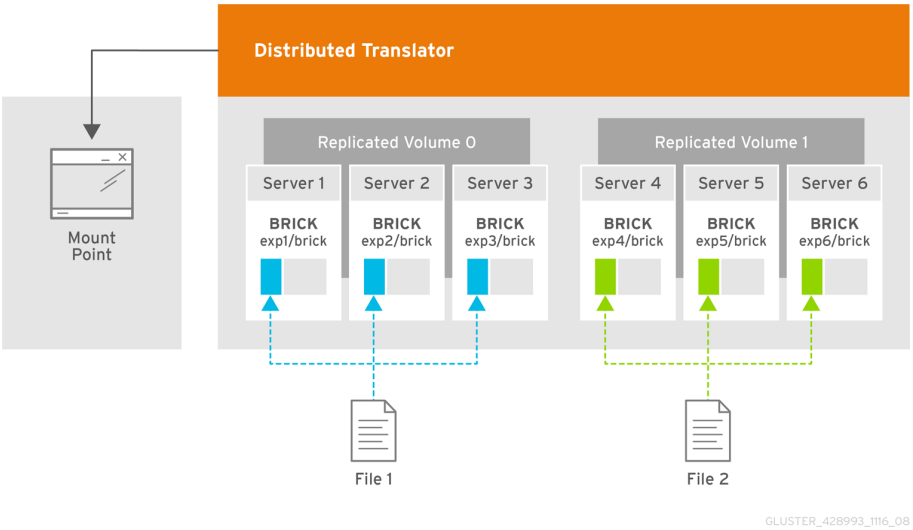

Abbildung 2.3: Schematischer Aufbau eines GlusterFS-Volumes des Typs Distributed Replicated mit der empfohlenen Replication 3 zur Vermeidung von Problemen bei Splitbrain-Konstellationen [24, S. 78]
Die Dateien belegen durch die beiden 1:1-Kopien im Backend 300% ihrer eigentlichen Größe. [26]

in der Regel mindestens drei Kopien. [24, S. 76] Ein Distributed Dispersed-Volume bietet zwar einen besseres Verhältnis von Speicherplatz-zu-Kosten, ist allerdings merkbar langsamer bei häufigen Änderungen, da ständig das Erasure-Coding berechnet werden muss und dies aufwändiger als eine 1:1-Kopie ist. Grafik 2.2 zeigt hierbei anschaulich, wie die Dateien aufgespalten und durch das Erasure Coding auf ca. 150% Größe im Backend aufgeblasen werden. Hierbei ergibt sich jedoch noch immer eine deutlich Platz- und Kostenersparnis im Vergleich zum Distributed Replicated, bei dem jede Datei auf ca. 300% im Backend wächst, wie es in Grafik 2.3 gut ersichtlich ist. Zur besseren Veranschaulichung wurden in Tabelle 2.4 alle drei möglichen Konfigurationen und ihre nutzbare Größe aufgelistet, wobei immer 12 Server mit 12 Bricks zu je 8TB als Rechengrundlage genutzt wurde.

Da bei */data* primär die Kosten im Vordergrund stehen, um die enormen Datenmengen mit vertretbarem finanziellen Aufwand speichern zu können, wurde hier ein Volume vom Typ Distributed Dispersed gewählt.

Bei */home* handelt es sich hingegen um sehr viele kleine Config-Files der jeweils genutzten Programme, die oft geändert werden und bei jedem Job gelesen werden. Daher wurde hier auf ein Distributed Replicated-Volume gesetzt, welches zur weiteren Beschleunigung auf SSDs basiert.

GlusterFS bietet durch die Verwendung von Standard-Servern eine Herstellerunabhägigkeit, wie sie mit den Huawei-Storages nicht gegeben war. Gleichzeitig findet mit weiteren Bricks eine automatische Lastverteilung statt, da der Datentransfer ohne zentrale Instanz direkt zwischen Client und Brick abgewickelt wird.

Das zentrale GlusterFS-System und die lokalen Scratch-Platten werden um den *NFS*-Speicher */work* ergänzt. Auf Grund der Performance-Probleme des Distributed Dispersed-Volumes bei kleinen Dateien, kam nutzerseitig der Wunsch nach einem weiteren Speicher auf.

Dieser sollte auf Small-File-Performance optimiert und im Gegensatz zum Scratch-Speicher knotenübergreifend zur Verfügung stehen. Daher fiel die Wahl auf einen NFS-Speicher mit ca. 20TB nutzbarer Kapazität auf SSDs. NFS bietet den Vorteil, dass Dateien innerhalb des Mounts serverseitig bewegt werden können, ist allerdings dafür weniger gut skalierbar als GlusterFS, da in der Regel keine Synchronisation über Server hinweg stattfindet.

Der NFS-Speicher /work wurde nutzerweise strukturiert. Jeder Nutzeraccount hat hier einen eigenen Ordner, welcher über die Umgebungsvariable *$WORK* exportiert wird. Die Environment-Module von R, Python und Perl biegen den Pfad für nutzerseitig installierte Zusatzmodule standardmäßig auf ein Unterverzeichnis von /work um, damit diese auf allen Rechenknoten direkt zur Verfügung stehen und von der Beschleunigung durch den NFS-Speicher profitieren können.

Volumetyp	Speichergröße	Hinweise
Distributed Dispersed (4 aus 6)	12 Server×12 Bricks×8 TB×(4 aus 6) = 12×12×8TB×$\frac{2}{3}$ = 768TB	Je Dreierbund kann 1 Server ohne Datenverlust ausfallen.
Distributed Replicated (Redundanz 3)	12 Server×12 Bricks×8 TB×(Redundanz 3) = 12×12×8TB×$\frac{1}{3}$ = 384TB	Je Dreierbund können bis zu 2 Server ohne Datenverlust ausfallen.
Distributed	12 Server×12 Bricks×8 TB = 12×12×8TB = 1152TB	Keine Redundanz. Bei jedem Ausfall sind Daten nicht verfügbar bzw. verloren.

Tabelle 2.4: Nutzbare Größe eines Volumes bei 12 Servern mit je 12 Bricks (Festplatten) zu jeweils 8TB bei verschiedenen Volumetypen eines GlusterFS-Systems
Das Distributed Dispersed-Volume stellt einen preisgünstigen Kompromiss zwischen dem Distributed Replicated-Volume und dem reinen Distributed-Volume ohne Redundanz dar und wurde daher für */data* im OMICS-Cluster gewählt.

Netzwerk

Da das Netzwerk die einzelnen Komponenten (Storage, Rechenknoten, Verwaltungsmaschinen) untereinander und mit der Außenwelt verbindet, spielt es eine entscheidende in der Leistungsfähigkeit des Gesamtsystems. Während anfänglich auf 1 Gigabit Ethernet gesetzt wurde, konnte dieses inzwischen flächendeckend zu 2×10 Gigabit Ethernet ausgebaut werden, die meisten GlusterFS-Server sind darüber hinaus auf 4×10 Gigabit Ethernet vorbereitet. Hiervon sind nur die drei Altsysteme SM5650-64 und SM5650-48 ausgenommen. Aus Kostengründen wurde auf 10 Gigabit Ethernet gesetzt, da so bereits auf eine gut ausgebaute Netzwerkstruktur in den Rechenzentren des IT-Service-Centers zurück gegriffen werden konnte. Allerdings steht das OMICS-Cluster mit dieser Wahl nicht allein da. Wie Abbildung 2.5 zeigt, nutzten im November 2019 ca. 19,6% der Top500-Systeme 10 Gigabit Ethernet als primären Interconnect, wobei dies allerdings primär HPC-Systeme am unteren Ende der Rangliste betrifft, da auf diese nur ca. 9,5% der Gesamtleistung entfällt (Abbildung 2.6).

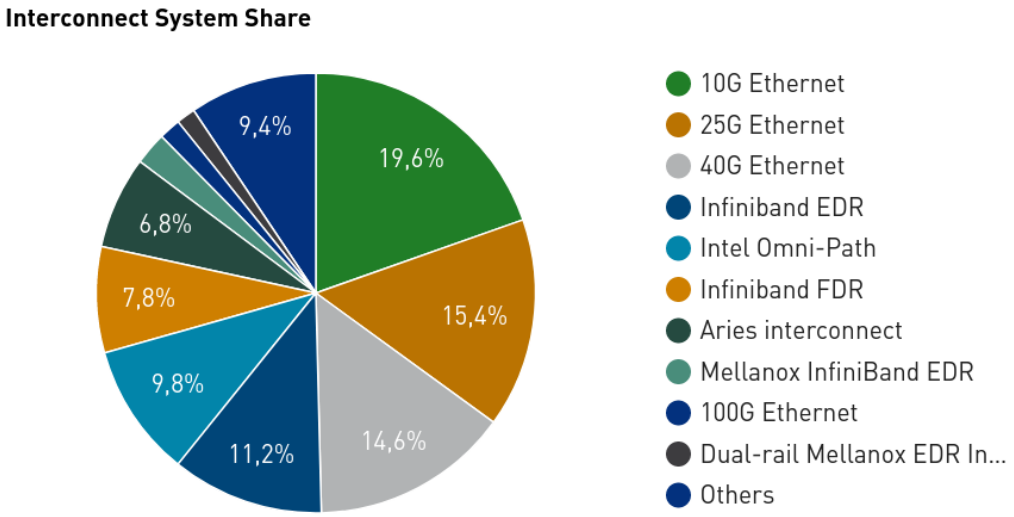

Abbildung 2.5: Prozentualer Anteil an HPC-Systemen in den Top 500 im November 2019, nach Interconnect [31]

2.3 Schrittweiser Ausbau

Das OMICS-Cluster wurde in den letzten 6 Jahren seines Betriebes mehrfach erweitert und teils tiefgreifend umgebaut. Zum besseren Verständnis wurden die Rechenknoten daher in Generationen eingeteilt, die sich primär an der CPU-Generation und dem Beschaffungszeitpunkt orientieren.

Generation 0

Bereits vor dem OMICS-Cluster gab es am Lübecker Institut für Experimentelle Dermatologie (LIED) ein deutlich kleineres Rechencluster: die Derma-Cloud. An dieser Stelle ist sie nur so weit interessant, dass einige der Hardwareelemente in das OMICS-Cluster übernommen wurde. Dies geschah 2015, als das OMICS-Cluster den produktiven Betrieb aufnahm und die Derma-Cloud außer Betrieb genommen wurde. Nach der Verschrottung der restlichen Maschinen der Derma-Cloud blieben nur noch drei Maschinen (Typ SM5650-48 und SM5650-64 in Tabelle 2.7) für die Debug-Partition, in der primär kleinere interaktive Jobs und Testläufen stattfinden. Damit können Nutzer die Lauffähigkeit ihrer Jobscripte evaluieren. Zwar besitzen sie die ältesten CPUs und den wenigsten Arbeitsspeicher im Cluster, übernehmen allerdings Aufgaben, für die sonst leistungsstärkere Maschinen belegt werden würden.

Generation 1

Der Planung des OMICS-Clusters begann im Jahr 2014. Aus Geldern des Exzellenzclusters für Entzündungsforschung sollte ein Rechencluster gebaut werden, welches von den beteiligten Professoren gemeinsam genutzt werden kann.

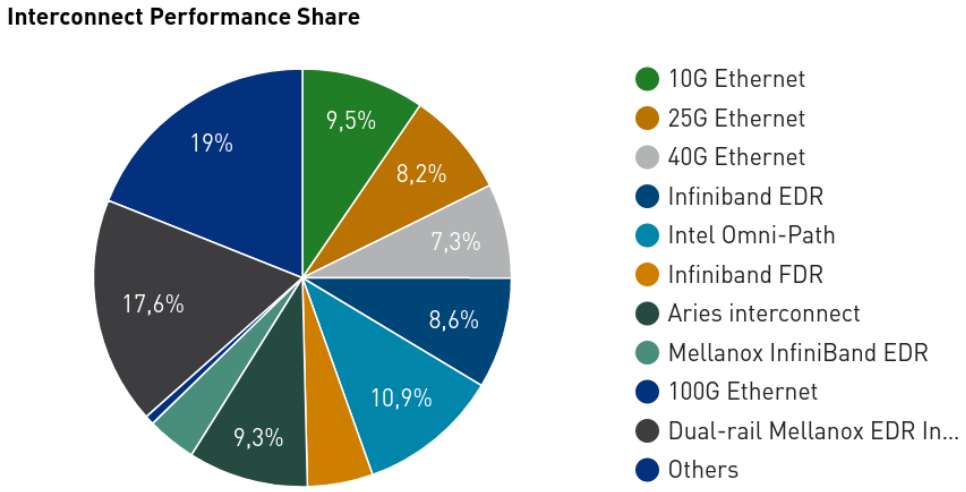

Abbildung 2.6: Prozentualer Anteil an HPC-Performance in den Top 500 im November 2019, nach Interconnect [31]

In dieser ersten Ausbaustufe lieferte ein Huawei-Storage vom Typ OceanStor S2200T ca. 160TB nutzbaren Speicherplatz.

Für die benötige Rechenleistung der Analysen sorgten 16 kleine Rechenknoten (Typ *HU2658-192* in Tabelle 2.7) mit je 24 Cores / 48 Threads und 192 GB RAM, die von einem Bignode (Typ *HU2699-512* in Tabelle 2.7) mit 36 Cores / 72 Threads und 512 GB RAM ergänzt wurden. Während der ersten Betriebsjahre wurden diese Rechenknoten noch mit den Original-Festplatten (3× 2 TB) betrieben und besaßen so nur eine Scratchgröße von ca. 5 TB. Als 2018 das erste Storage außer Betrieb genommen wurde, konnten diese Maschinen mit den 4 TB-Festplatten auf 12TB bzw. sogar 32TB Speicherkapazität gesteigert werden.

Nach einer Vorbereitungszeit von ca. 1 Monat ging das Cluster in einen ersten Testbetrieb mit stark eingeschränktem Nutzerkreis.

2015 erfolgte dann die 1. Erweiterung des Clusters erfolgen, welches primär weiteren Datenspeicher bringen sollte. Hierbei wurde aus verschiedenen Gründen auf das GlusterFS-System gewechselt.

Im Bereich der Rechenknoten wurden drei weitere kleine Rechenknoten beschafft, dabei wurden zwei mit etwas mehr RAM (Typ *HU2658-256* in Tabelle 2.7) bestückt.

Generation 2

2017 wurde dann der erste Rechenknoten mit Skylake-Prozessor (Typ *SM4110-384* in Tabelle 2.7) beschafft und integriert.

Parallel dazu konnte mit einem dritten GlusterFS-Server das Storage von einem Distributed Volume auf ein Distributed Dispersed Volume umgestellt werden, um die zuvor genannten Vorteile nutzen zu können. Dies legte den Grundstein für den aktuellen Aufbau des GlusterFS-Storages. Dieser Aufbau wurde dann in 2018, 2019 und 2020 mit weiteren GlusterFS-Servern fortgesetzt.

Generation 3

Alle Rechenknoten der Generation 3 (Tabelle 2.7) erweiterten dann 2019 das OMICS-Cluster auf seine aktuelle Größe. Dies beinhaltet insbesondere die Rechenknoten des Typs SM7371-512 mit den AMD EPYC-CPUs, auf die im Laufe dieser Arbeit noch näher eingegangen wird.

Rechenknotentyp	SM5650-64	SM5650-48	
Generation	0	0	
Anzahl	2	1	
Hersteller	SuperMicro	SuperMicro	
CPU	2× Intel Xeon X5650	2× Intel Xeon X5650	
RAM	64 GB	48 GB	
Scratch-Speicherplatz	ca. 5 TB	ca. 15 TB	
Hinweise	*Altsysteme aus der Derma-Cloud für Debug-Zwecke mit geringerer Netzwerkanbindung von nur 2×1Gbit/s und Westmere-CPUs*		
Rechenknotentyp	HU2658-192	HU2658-256	HU2699-512
Generation	1	1	1
Anzahl	16	2	1
Hersteller	Huawei	Huawei	Huawei
CPU	2× Intel Xeon E5-2658 v3	2× Intel Xeon E5-2658 v3	2× Intel Xeon E5-2699 v3
RAM	192 GB	256 GB	512 GB
Scratch-Speicherplatz	ca. 11 TB	ca. 5 TB	ca. 28 TB
Hinweise	*Haswell-Rechenknoten*		
Rechenknotentyp	SM4110-384	SM5217-384	D5217-384
Generation	2	3	3
Anzahl	1	5	1
Hersteller	SuperMicro	SuperMicro	DELL
CPU	2× Intel Xeon Silver 4110	2× Intel Xeon Gold 5217	2× Intel Xeon Gold 5217
RAM	384 GB	384 GB	384 GB
Scratch-Speicherplatz	ca. 11 TB	ca. 11 TB	ca. 11 TB
Hinweise	*Skylake-Rechenknoten*		
Rechenknotentyp	F6246-768	SM7371-512	
Generation	3	3	
Anzahl	2	2	
Hersteller	Fujitsu	SuperMicro	
CPU	2× Intel Xeon Gold 6246	1× AMD EPYC 7371	
RAM	768 GB	512 GB	
Scratch-Speicherplatz	ca. 20 TB	ca. 11 TB	
Hinweise	*Skylake-Rechenknoten*	*AMD-EPYC-Rechenknoten*	

Tabelle 2.7: Übersicht der verbauten Rechenknoten geordnet nach Typen, Generation und Hersteller

3

Benchmarking

3.1 Allgemeines

Benchmarks dienen in erster Linie dem Vergleich ähnlicher Systeme untereinander. Geeignete Benchmarks helfen allen an einem Computersystem beteiligten Parteien bei dessen Einordnung. So können sich Hersteller darüber vergleichen und mit ihren besonders performanten Systemen neue Kunden und Märkte erschließen. Betreiber von diesen Systemen können prüfen, ob sie geliefert bekamen, was sie bestellt haben und Nutzer können besser abschätzen, auf welche Laufzeiten sie sich einstellen müssen.

Besonders im HPC-Umfeld beinhalten Ausschreibungen für sehr große Systeme häufig, dass das Zielsystem in vorgegebenen Benchmarks eine bestimmte Leistung erreichen müssen. So wird sicher gestellt, dass die Forscher und Betreiber ein System bekommen, welches für den jeweiligen Zweck geeignet ist. Gleichzeitig stellt die Veröffentlichung der Ergebnisse auch einerseits heraus, dass der Hersteller in der Lage ist ein solches System zu erstellen und andererseits zeigt es, dass sich die betreibende Einrichtung eine entsprechende Maschine leisten kann. Dies hilft dann auch der Einrichtung (z.B. Universität) bei der Anwerbung von Professoren, die auf diesen Systemen rechnen möchten.

Die Forscher Dongarra, Luszczek und Petitet zielten ursprünglich darauf ab, besser einschätzen können, wie lange ihre linearen Gleichungssysteme wahrscheinlich bei der Lösung auf den verfügbaren Computersystemen benötigen werden („[…] providing information on execution times required to solve a system of linear equations.“ [10]). Dafür wurden dann verschiedenste vorhandene Systeme vermessen. Daraus ergibt sich automatisch eine entsprechende Rangliste, die – wie oben beschrieben – auch wieder als Aushang genutzt wurde und wird.

Für das OMICS-Cluster mit seinen vielen verschiedenen Rechenknotentypen sollen nun drei verschiedene Benchmarks einen Vergleich zwischen den verschiedenen Typen ermöglichen.

Prinzipiell belastet ein Benchmark eine oder mehrere Maschinen mit einer vorher genau definierten Aufgabe und misst währenddessen verschiedene Parameter. Aus diesen Messungen und den vorher definierten Parametern lässt sich nun eine Leistungsbewertung nach verschiedenen Kriterien errechnen. Welche Komponenten in diese Bewertung wie stark gewichtet werden, unterscheidet sich zwischen den Benchmarks.

3.2 Ziele

Da das OMICS-Cluster in erster Linie der Genomanalyse dient und die meisten der dort verwendeten Tools kein MPI unterstützen, wird der primäre Fokus auf die Leistungsbeurteilung der einzelnen Rechenknotentypen gelegt. Die Gesamtleistung des Clusters ergibt sich hierbei durch eine Parallelisierung auf Job-Ebene, indem Verarbeitungsschritte mit verschiedenen Datengruppen in eigenen Jobs unabhängig voneinander Ressourcen anfordern. Der einzelne Rechenknoten ist daher wichtiger als das Zusammenspiel in kleineren und größeren Gruppen.

Die Beantwortung der Fragen in Kapitel 1.1 sollen mit passenden Benchmarks möglichst umfassend beantwortet werden.

Daher wurden drei Benchmarks ausgewählt: Linpack, HPCG und NCBI-Blast. Warum sich diese besonders für die Betrachtung eignen, wird in Kapitel 3.3 erläutert. Zusätzlich findet sich zu jedem Benchmark ein eigenes Kapitel, in dem er jeweils vorgestellt wird und die technische Umsetzung genauer beschrieben werden. Im Anschluss folgt eine Präsentation der Ergebnisse und der daraus ableitbaren Erkenntnisse. Eine Zusammenführung der Einzelergebnisse zu einer gemeinsamen Auswertung befindet sich im Anschluss in Kapitel 5.

3.3 Ausgewählte Benchmarks

Linpack, genauer Linpack-HPL, stellt durch die Nutzung in den Top500 den Standardtest im HPC-Umfeld dar und wird im weltweiten Vergleich von HPC-Systemen genutzt. Auch ist er Teil der Student Supercomputing Challenge und der Sammlung des HPC Challenge Benchmark.

Durch seinen Fokus rein auf die Lösung eines linearen Gleichungssystems wird die CPU unverhältnismäßig stark gewertet. So schreiben die Entwickler von HPCG unter [14]: „The result is that HPL tends to represent a very optimistic performance prediction that can only be matched by a small number of real scientific applications.“ und zeigen auf, dass Linpack oft zu optimistische Werte liefert.

HPCG wird oft im Zusammenhang mit Linpack betrachtet und stellt den zweiten Standardtest im HPC-Umfeld dar. Auch dieser kann entsprechend zur Einordnung genutzt werden. Auf Grund der eher pessimistischen Bewertung von HPCG, liefert er in dieser Betrachtung eher eine untere Grenze der Leistungsfähigkeit.

In Jobs von Nutzern ist daher eher eine Leistung zwischen diesen beiden Werten zu erwarten. Der genaue Wert ist aber natürlich stark abhängig von der Optimierung der Anwendung und den genauen Befehlen, die ausgeführt werden.

Um die eher synthetischen Tests Linpack und HPCG um Werte aus der Genomanalyse zu ergänzen, wurde als dritten Benchmark Blast von NCBI ausgewählt. Er soll der Evaluation dienen, in wie weit die Ergebnisse von Linpack und HPCG mit denen von NCBI-Blast übereinstimmen. Wie in Kapitel 1.1 bereits genannt, soll geklärt werden, ob alle Benchmarks die Rechenknotentypen ähnlich bewerten oder jeweils einzelne Konfigurationen bevorzugen.

Eine ausführliche Beschreibung der einzelnen Tests befindet sich in den entsprechenden Kapiteln.

4

Durchgeführte Benchmarks

4.1 Allgemeines

Die Benchmarks wurden alle auf dem OMICS-Cluster der Universität zu Lübeck durchgeführt. Hierfür wurde das vorhandene Batch-System Slurm[28] genutzt. Die jeweiligen Scripte der Jobs finden sich im Anhang A.

Das Cluster konnte für die Ausführung der Tests leider nicht exklusiv genutzt werden und so könnten die parallelen Aktivitäten anderer Nutzer durch Seiteneffekte unerwünschte Auswirkungen auf die Testergebnisse gehabt haben. Um diese möglichst gering zu halten, wurden die meisten Tests auf den lokalen Datenträgern der jeweiligen Rechenknoten durchgeführt. Diese lokalen Datenträger sind, wie in Kapitel 2.2 bereits erwähnt, den Jobs unter `$SCRATCH` für Rechenjobs verfügbar und werden entsprechend in den Scripten genutzt.

Zusätzlich wurden die Rechenknoten immer durch die Angabe des Slurm-Parameters `--exclusive` exklusiv angefordert, um die Beeinflussung der genutzten Rechenknoten durch Aktivitäten anderer Nutzer auf den gleichen Maschinen zu unterbinden. Allerdings mussten natürlich gemeinsame Ressourcen weiterhin geteilt werden. Hierbei hat das gemeinsame Netzwerk das größte Potential für Störfaktoren, welches sich leider nicht vermeiden ließ.

Durch die Vermessung einzelner Jobs wirken sich diese Störfaktoren primär beim Zugriff auf die Netzwerkspeicher /data und /work aus.

4.2 Linpack

Über diesen Test

LINPACK (genauer Linpack-HPL) gilt als der Standardtest im HPC-Umfeld und wurde 1979 zum ersten Mal im LINPACK User's Guide[11] mit Vergleichswerten als Benchmark veröffentlicht.

Die Grundidee ist, dass eine zufällig gefülltes lineares Gleichungssystem in Form einer Matrix mit einem vorgegebenen Algorithmus gelöst werden muss. An Hand der vorher berechenbaren Anzahl an Operationen kann dann zusammen mit der gemessenen Laufzeit die Anzahl der Floating-Point-Operationen pro Sekunde – kurz FLOPS – als Leistungsmerkmal bestimmt werden.

Mathematisch beschrieben muss $Ax = b$ gelöst werden[10]. Um hierbei eine Vergleichbarkeit zu ermöglichen, ist die Nutzung von 64Bit-Floating-Point-Arithmetik vorgeschrieben. Gleichzeitig muss der Algorithmus je nach Test einige Bedingungen erfüllen.

Während der klassische LINPACK-Benchmark eigentlich aus mehreren Tests besteht, ist heute nur noch der **LINPACK HPL** relevant. Die Tests **LINPACK 100**[10] mit einer 100×100-Matrix und **LINPACK 1000**[10] mit einer 1000×1000-Matrix verlieren durch die immer schnelleren Prozessoren zunehmend an Bedeutung. Beide Tests haben weitere Einschränkungen, so dass z.B. LINPACK 100 nur durch den Compiler und nicht händisch optimiert werden darf. Bei LINPACK HPL darf hingegen fast nach Belieben optimiert werden, solange der eigentliche Rechenweg nicht geändert wird, um die FLOPs weiterhin zuverlässig berechnen zu können.

Entsprechend wurde im Rahmen dieser Arbeit auch nur LINPACK HPL durchgeführt, wie er im HPC Challenge Benchmark[13] vorgesehen ist. Bei der Umsetzung wurde sich an den Empfehlungen von *The Student Supercomputer Challenge Guide*[2, Chapter 11] orientiert. Eine genauere Beschreibung der technischen Umsetzung findet sich im nächsten Abschnitt.

Die weltweit besten Ergebnisse aus dem LINPACK-Benchmark werden zweimal im Jahr von der University of Tennessee gesammelt und veröffentlicht. Die daraus resultierende Liste ist als „Top500“ bekannt und gibt so einen guten Überblick über die leistungsstärksten HPC-Systeme weltweit. Hierbei entfallen auf Grund der freiwilligen Einreichungen natürlich Systeme, die der Geheimhaltung unterliegen oder deren Benchmarking zu kostenintensiv wäre.

Bei den Testläufen wurden automatisch verschiedene Parameter durchgetestet, die so abgeschätzt wurden, dass ca. 80% des RAMs gefüllt war. Von den verschiedenen Ergebnissen wurde das mit dem höchsten GFLOP/s-Wert ausgewählt und dessen Werte entsprechend im Abschnitt Ergebnisse aufgelistet. Die weiteren Ergebnisse wurden der Übersicht halber nicht aufgeführt.

Technische Umsetzung

Der Benchmark setzt grundsätzlich auf drei besonders wichtigen Komponenten auf: Compiler, MPI-Bibliothek und BLAS-Bibliothek. Um eine bessere Vergleichbarkeit zu ermöglichen, wurden möglichst viele Komponenten aus den offiziellen Debian-Repositories genutzt.

Als *Compiler* wurde entsprechend ein GCC 6.3.0 verwendet, wie er in den Debian-Repositories verfügbar [12] ist.

Da es Probleme im Zusammenhang mit der von Debian bereitgestellten OpenMPI-Version gab, wurde als MPI-Bibliothek Intel-MPI in der Version 2019.5 ausgewählt. Die *MPI-Bibliothek* ist maßgeblich für die Kommunikation zwischen den einzelnen MPI-Threads zuständig.

Die *BLAS-Bibliothek* beinhaltet die eigentlichen Algorithmen zur Lösung der Matrix-Multiplikationen und sollte daher einen deutlich sichtbaren Einfluss auf die Leistungsfähigkeit haben. Hierbei wurden zwei Bibliotheken genutzt, die in den Debian-Repositories verfügbar sind: libblas und OpenBLAS. Entsprechend wurde gegen beide Bibliotheken getrennt voneinander kompiliert und getestet, um deren Auswirkungen sehen zu können. Die Bibliothek libblas lag in Version 3.7.0 vor [16, 15]. Zum Vergleich wurden auch Messungen

mit der BLAS-Bibliothek OpenBLAS in Version 0.2.19 aus den Debian-Repositories [17] durchgeführt.

Die hpcc-Binaries wurden jeweils auf der entsprechenden Hardware kompiliert und systematisch unter /work abgelegt, um eine Zuordnung zu ermöglichen. Dafür wurden zwei build-Scripte geschrieben, welche als A.1 und A.2 am Ende im Appendix einsehbar sind.

Die Messungen selbst wurden dann mittels Jobscripte durchgeführt, die allerdings in Bezug auf die Ressourcenanforderungen im Bereich der #SBATCH-Optionen auf den jeweiligen Rechenknotentypen angepasst werden müssen. Entsprechend findet sich unter A.3 ein Beispiel für den Typ SM7371-512. Die Scripte für die anderen Typen sind vergleichbar.

Ergebnisse

Knotentyp	**GFLOP/s**	**N**	**NB**	**P**	**Q**
SM5650-64	4.92	28000	80	8	1
SM5650-48	9.74	28000	80	8	1
HU2658-192	7.51	36000	80	8	1
HU2658-256	8.24	36000	80	8	1
HU2699-512	7.92	42000	80	8	1
SM4110-384	9.61	32000	80	8	1
SM5217-384	12.96	32000	80	8	1
D5217-384	12.50	32000	80	8	1
F6246-768	12.32	36000	80	8	1
SM7371-512	9.35	42000	80	8	1

Tabelle 4.1: Übersicht der Leistung von Linpack einzelner Rechenknotentypen für Linpack mit der *libblas*

Knotentyp	**GFLOP/s**	**N**	**NB**	**P**	**Q**
SM5650-64	17.41	42000	80	8	1
SM5650-48	9.81	28000	80	8	1
HU2658-192	21.29	42000	80	8	1
HU2658-256	31.74	54000	80	8	1
HU2699-512	40.25	72000	80	8	1
SM4110-384	49.81	72000	80	8	1
SM5217-384	36.20	48000	80	8	1
D5217-384	37.45	48000	80	8	1
F6246-768	31.74	84000	80	8	1
SM7371-512	32.37	72000	80	8	1

Tabelle 4.2: Übersicht der Leistung von Linpack einzelner Rechenknotentypen für Linpack mit der *openblas*

Die Ergebnisse der Benchmarks wurden in der Tabelle 4.1 für die Läufe mit der libblas [16] und in der Tabelle 4.2 für OpenBLAS [17] dargestellt.

Die wichtigsten Testparameter wurden hierbei angegeben, sind aber bei den verschiedenen Läufen teils sehr unterschiedlich. Dies ergibt sich daraus, dass für die einzelnen Ma-

schinen möglichst hohe Werte erreicht werden sollten und so hängt die Matrixgröße N stark von der jeweiligen RAM-Größe ab. Diese schwankt, wie in Tabelle 2.7 ersichtlich, zwischen 60GB und 768GB. Zusätzlich wurde NB angegeben, welches die Größe der Blöcke spezifizert, in die Linpack die Matrix unterteilt. Auch die Größe der Prozessorgrids P und Q sind aufgeführt.

Auswertung

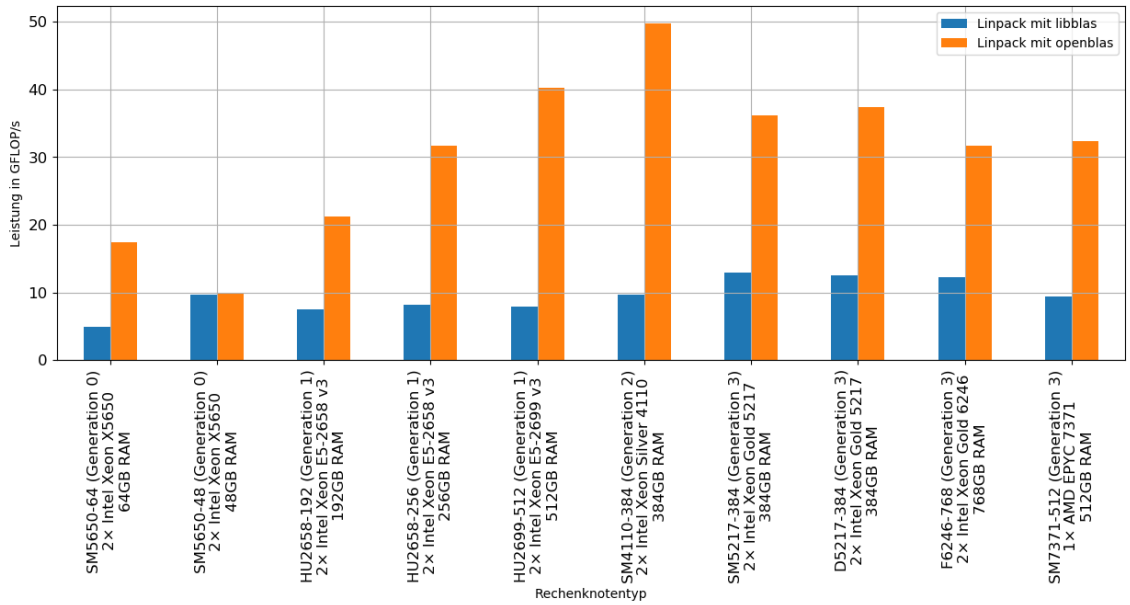

Abbildung 4.3: Leistung von Linpack in GFLOP/s der besten gemessenen Linpack-Ergebnisse mit den beiden genutzten BLAS-Blibliotheken nach Rechenknotentyp , wie in Tabelle 2.7 gelistet.
Höhere Werte sind besser.

Abbildung 4.3 visualisiert die gemessene Leistung pro Rechenknotentyp. Hierbei fällt zunächst auf, dass die Ergebnisse sehr entscheidend von der BLAS-Bibliothek abhängt. So sind die Ergebnisse bei Verwendung von OpenBLAS durchweg höher, was auf eine bessere Optimierung innerhalb der Bibliothek deutet. Unterschiede zwischen den Bibliotheken wurden zwar erwartet, aber ein Faktor von fünf wie bei SM4110-384 überrascht.

Um die Leistung der einzelnen CPU-Cores besser betrachten zu können, wurde in Abbildung 4.4 die gemessene Leistung durch die CPU-Core-Anzahl (ohne Hyper-Threading-Cores) der Maschine geteilt. Der Einfluss der RAM-Größe wird in die Abbildung 4.5 dadurch deutlich, dass der GFLOP/s-Wert durch die RAM-Größe in GB geteilt wurde.

Die Rechenknoten der Generation 0 zeigen in Abbildung 4.3, wie erwartet, die niedrigsten Ergebnisse. Dass sie in Leistung pro Core laut Abbildung 4.4 mit den Maschinen der Generation 1 mithalten können, wurde jedoch nicht erwartet. Der Spitzenwert in Abbildung 4.5 erklärt sich am ehesten durch die sehr sparsame RAM-Ausstattung von 48GB und 64GB der Maschinen.

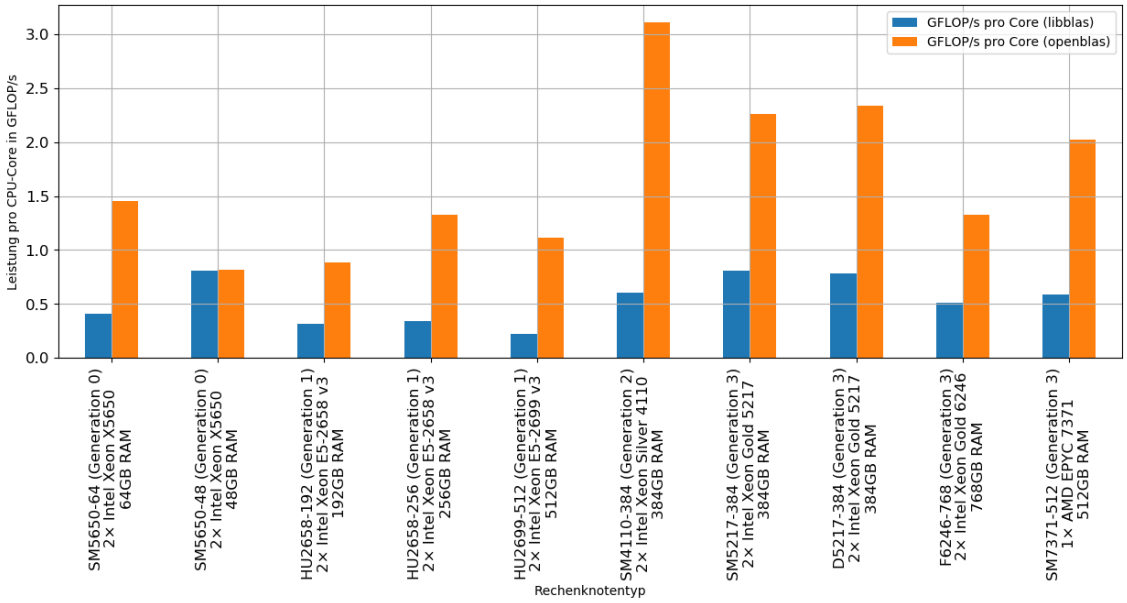

Abbildung 4.4: Leistung von Linpack in GFLOP/s geteilt durch die Anzahl an verfügbarer CPU-Cores (ohne Hyper-Threadingcores) im jeweiligen Rechenknotentyp.
Hierdurch wird die Leistung der einzelnen Prozessoren deutlich besser sichtbar.
Höhere Werte sind besser.

Sehr unterschiedliche Ergebnisse zeigen die Rechenknoten der Generation 1. So schnitten die Maschinen vom Typ HU2658-192 am schlechtesten von den Rechenknoten, die explizit für das OMICS-Cluster beschafft wurden, ab.[3] Nur der Typ HU2699-512 kann in Abbildung 4.3 mit den Knoten der neueren Generationen mithalten. Dies verdankt er allerdings sehr wahrscheinlich nur der Summe von 36 CPU-Cores, da die Leistung pro Core in Abbildung 4.4 nicht mithalten kann.

Der Rechenknoten der Generation 2 landet überraschenderweise mit einem überragenden Ergebnis sowohl in Bezug auf Gesamtleistung (Abbildung 4.3) als auch in Pro-Core-Leistung (Abbildung 4.4) bei den Messungen mit OpenBLAS. Eigentlich wurde erwartet, dass er mit seinen Intel Xeon Silver-CPUs eher im Mittelfeld landet, wie es sich auch mit der libblas zeigt.

Bei den Rechenknoten der Generation 3 landen durchweg auf den oberen Plätzen und stellen somit in Hinblick auf Linpack eine deutliche Leistungssteigerung gegenüber den Rechenknoten der Generation 1 dar. Dies zeigt sich sowohl mit der libblas als auch mit OpenBLAS.

Die Rechenknoten mit den AMD EPYC-Prozessoren (SM7371-512) sind am ehesten vergleichbar mit den Maschinen vom Typ SM5217-382. Hierbei schneiden die AMD-Rechenknoten nur minimal schlechter als diese ab.

[3] Zur Erinnerung Generation 0-Rechenknoten stammen aus dem Vorgänger, der Derma-Cloud.

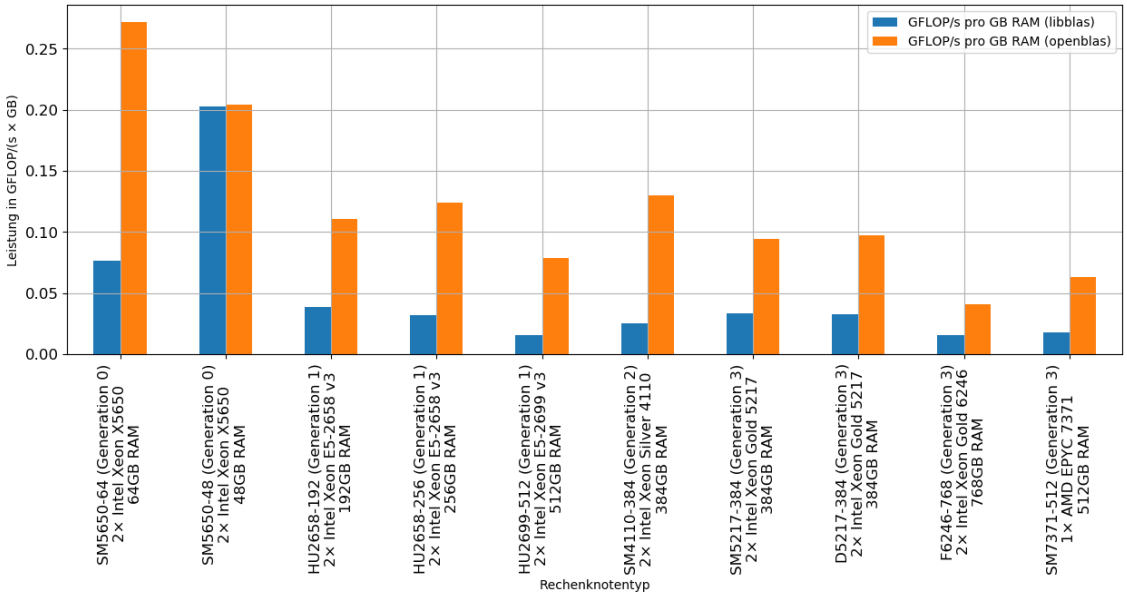

Abbildung 4.5: Leistung von Linpack in GFLOP/s geteilt durch die Anzahl an verfügbarem RAM (in GB) im jeweiligen Rechenknotentyp.
Hierdurch wird der Einfluss der Arbeitsspeicherverfügbarkeit besser sichtbar.
Höhere Werte sind besser.

4.3 HPCG

Über diesen Test

HPCG ist der zweite große Standardtest für HPC-Systeme. Im Gegensatz zu Linpack, der oft etwas zu optimisitische Werte ausgibt, erzielen die getesteten Systeme in HPCG oft etwas zu pessimistische Ergebnisse. Mit beiden zusammen ergibt sich für interessierte Forscher also ein besseres Bild, was sie von der Leistungsfähigkeit des Clusters erwarten können.

Bei HPCG wird wieder ein lineares Gleichungssystem $Ax = b$ gelöst. Im Unterschied zu Linpack erfolgt dies allerdings unter Nutzung des konjugierten Gradienten-Verfahren, dessen ausführliche Erklärung den Rahmen dieser Arbeit sprengen würde. [8]

HPCG zielt in eine andere Richtung als Linpack, in dem es statt der Peak Performance lieber die Gesamtleistung des Systems bewerten möchte. Die theoretische und Peak Performance von Linpack wird bei realen Anwendungen selbst mit starken Optimierungen nur sehr selten erreicht, da oft der Arbeitsspeicher eine deutlich merkbare Bremse darstellt. Dies wird durch Linpack leider nicht deutlich genug repräsentiert und an diesem Punkt setzt HPCG an. [14]

Es provoziert durch seinen Aufbau [8] deutlich mehr unregelmäßige RAM-Zugriffe als Linpack und erreicht damit ganz bewusst deutlich niedrigere Gflop/s-Werte. So erreichte das System Summit des Oak Ridge National Laboratory mit seinen 2.414.592 Power9-Cores offiziell eine maximale Performance von 148,600 Pflop/s [20], während es auch die HPCG-Liste mit einem Wert von 2,926 Pflop/s anführte. Allerdings landet in den gleichen Listen das Trinity-System des Los Alamos National Laboratory mit seinen 979.072 Xeon-Haswell-

& Xeon-Phi-Cores bei Linpack auf Platz 7 mit 20,159 Pflop/s und bei HPCG auf Platz 3 mit 0,546 Pflop/s, deutlich vor dem Sunway TaihuLight des National Supercomputing Center von Wuxi (Platz 3 in Linpack, Platz 6 bei HPCG).

Allerdings repräsentiert HPCG oft ein deutlich realistischeres Bild von der Rechenleistung, die normale Anwender erwarten können, da es eher ausgeglichene Systeme bevorteilt, während Linpack am besten von spezialisierten Systemen profitiert. Während die Linpack-Werte für bestimmte Anwendungen (besonders im Bereich Physik) ein sehr gutes Maß darstellt, scheint HPCG aussagekräftigere Werte für die Genomanalyse zu bieten, da deren Anwendungen meist größere Mengen RAM benötigen.

Technische Umsetzung

Auch dieser Benchmark wird wieder stark von den den besonders wichtigen Komponenten beeinflusst: Compiler und MPI-Bibliothek. Um eine bessere Vergleichbarkeit zu ermöglichen, wurden möglichst viele Komponenten aus den offiziellen Debian-Repositories genutzt.

Als *Compiler* wurde entsprechend wieder der GCC 6.3.0 genutzt, da dieser über die Debian-Repositories bereitgestellt [12] wird.

Als *MPI-Bibliothek* konnte OpenMPI in Version 2.0.2 aus den Debian-Repositories [21] genutzt werden.

Die hpcg-Binaries wurden wieder jeweils auf der entsprechenden Hardware kompiliert und entsprechend unter /work abgelegt, um eine Zuordnung zu ermöglichen. Dafür wurde ein build-Script geschrieben, welche als A.4 im Appendix einsehbar sind.

Die Messungen selbst wurden dann wieder mittels Jobscripten durchgeführt, die sich primär in den Ressourcenanforderungen im Bereich der #SBATCH-Optionen und den entsprechenden Pfaden unterscheiden. Entsprechend findet sich unter A.5 ein Beispiel für den Typ SM7371-512. Die Scripte für die anderen Typen sind vergleichbar. Hierbei wurde nur eine Beispielkonfiguration eines Laufes mit den Werten nx=ny=nz=208 und rt=60 aufgeführt.

Ergebnisse

Knotentyp	**GFLOP/s**	**NX**	**NY**	**NZ**	**RT**
SM5650-64	1.24173	416	416	208	60
SM5650-48	1.28283	208	208	208	60
HU2658-192	0.64591	416	416	416	60
HU2658-256	1.28308	208	208	208	30
HU2699-512	0.650797	416	416	208	60
SM4110-384	1.48969	208	208	208	60
SM5217-384	1.69198	208	208	208	60
D5217-384	1.52618	416	416	208	60
F6246-768	1.39002	416	416	208	30
SM7371-512	0.79687	416	416	208	30

Tabelle 4.6: Übersicht der Leistung einzelner Rechenknotentypen für HPCG

Die Ergebnisse für die HPCG-Läufe finden sich in Tabelle 4.6. Hierbei fallen zunächst,

wie erwartet, die deutlich niedrigeren GFLOP/s-Werte im Vergleich zu Linpack (Tabelle 4.2) auf.

Zusätzlich wurden die Größen NX, NY und NZ angegeben. Diese definieren die Matrixgröße der lokal zu berechnenden Teilprobleme je MPI-Prozess und wurden so dimensioniert, dass sie zusammen den RAM zu ca. 80% des Rechenknotens ausfüllen. [9]

Auswertung

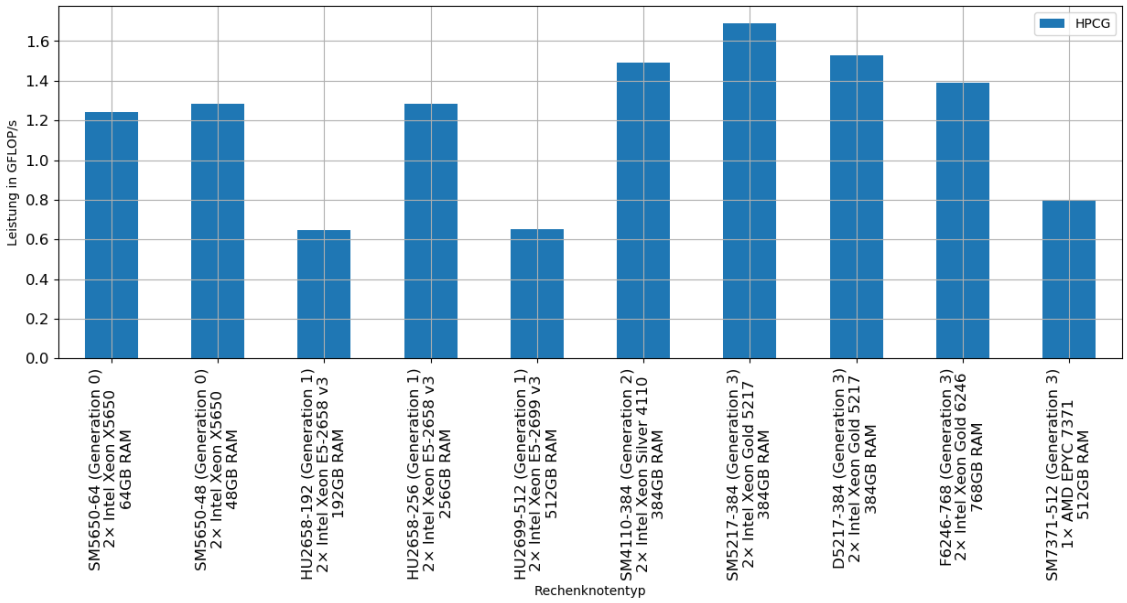

Abbildung 4.7: Leistung von HPCG in GFLOP/s pro Rechenknotentyp.
Höhere Werte sind besser.

Auch in Abbildung 4.8 liegen die erreichten GFLOP/s-Werte von HPCG deutlich unter denen von Linpack. Sie liegen sogar noch deutlich unter denen der langsamen libblas. So ist der höchste erreichte Wert bei HPCG ca. 1,7GFLOP/s (SM5217-384 in Tabelle 4.6), während bei Linpack kein Rechenknoten mit unter 4,92 (SM5650-64 in Tabelle 4.1) abschnitt. Dies war aber, wie bereits erwähnt, zu erwarten.

Überraschend ist, dass die sehr ähnlichen Maschinen SM5217-384 und D5217-384 unterschiedlich abschneiden. Die Dell-Maschine erreicht nur ca. 91% der Leistung der SuperMicro-Maschine, obwohl beide gleich viel RAM und mit den gleichen CPUs ausgestattet sind.

Am schlechtesten schneidet mit HU2658-192 ein Rechenknoten der Generation 1 ab. Generell zeigt sich bei HPCG unter Betrachtung der Abbildung 4.9 ein sehr durchwachsenes Ergebnis der Leistung pro CPU-Core. Neuere Prozessoren scheinen hier nicht zwingend gleichwertig oder leistungsstärker.

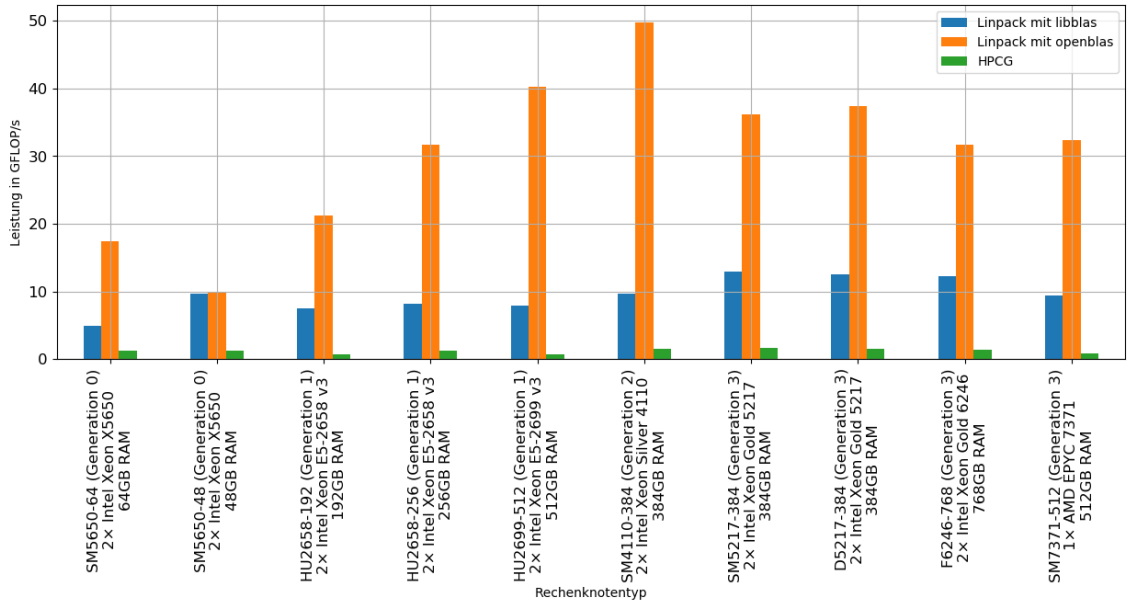

Abbildung 4.8: Leistung von HPCG und Linpack in GFLOP/s pro Rechenknotentyp. Höhere Werte sind besser.

4.4 NCBI Blast

Blast ist eine Sammlung von Programmen für den Abgleich von Sequzenzen innerhalb einer Proteinsequenz [18]. Hierfür bringt es verschiedene Tools mit, die Teilsuchen und -vergleiche auf der von BLAST generierten Datenbank ermöglichen. Viele der genutzten Algorithmen basieren auf heuristischen Verfahren, um mit den riesigen Datenmengen umgehen zu können. Für die Datenbankerstellung verarbeitet BLAST Genome beispielsweise in Form von FASTA-Dateien, die Nukleinsäuren in Textform enthalten.[7]. Die Daten können aber teils auch aus den öffentlich zugänglichen Datenbanken geholt werden, die von der NCBI bereitgestellt werden.

Damit bildet es die Grundlage einer Vielzahl von anderer Analysesoftware, wie z.B. qiime [23] und Matlab [19], die auf dem OMICS-Cluster installiert sind. Die Verarbeitungszeit von Blast wirkt sich daher direkt auf die Laufzeit vieler weiterer Software aus und bietet sich entsprechend als Metrik für einen Benchmark an. An der University of California [5] wurde ein entsprechender Benchmark entwickelt, der an dieser Stelle verwendet wird.

Während unter `https://blast.ncbi.nlm.nih.gov/Blast.cgi` eine einfache, aber im Umfang eingeschränkte, Suche über die Maschinen der NCBI möglich ist, wird für größere Datenbestände und komplexe Anfragen auf die Standalone-Version verwiesen, die die Suche auf lokaler Hardware ermöglicht. Entsprechend wird diese hier genutzt. Damit wurde bewusst die Nutzung der vorkompilierten Version gewählt, wie sie unter `https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/` verfügbar ist, um eine bessere Vergleichbarkeit zu erreichen.

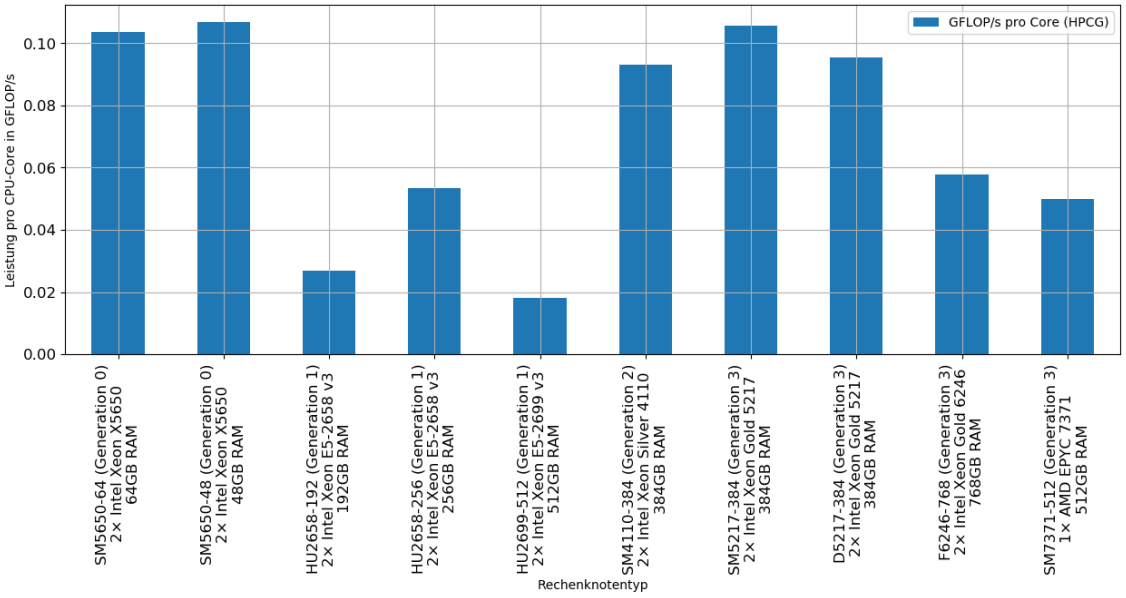

Abbildung 4.9: Leistung von HPCG in GFLOP/s geteilt durch die Anzahl an verfügbarer CPU-Cores (ohne Hyper-Threadingcores) im jeweiligen Rechenknotentyp.
Hierdurch wird die Leistung der einzelnen Prozessoren deutlich besser sichtbar.
Höhere Werte sind besser.

Über diesen Test

Dieser Benchmark nutzt einen Standard-Datensatz und wendet auf diesem typische Analysen an, wobei von der University of California verschiedene Auswertungen modelliert wurden, bei denen die unterschiedlichen Teilprogramme von BLAST aufgerufen werden [6]. Gemessen wird währenddessen die Laufzeit. Eine kürzere Laufzeit ist entsprechend besser und deutet auf eine höhere Leistungsfähigkeit der genutzten Hardware hin. Hierbei wird allerdings nicht nur die CPU oder der RAM besonders beansprucht, sondern das Gesamtsystem aus beiden Komponenten und den vorhandenen Datenträgern muss für eine hohe Leistung zusammen passen.

Wie in Kapitel 2.1 erklärt, führt dieser Benchmark eine String-Suche innerhalb eines Testdatensatzes mit einer mitgelieferten Testdatenbank durch.

Gemessen wird die Zeit der Ausführung. Daher wird versucht in den Ergebnissen möglichst niedrige Werte zu erzielen. Dies muss bei der Auswertung beachtet werden.

Technische Umsetzung

Für die Benchmarks wurde die vorkompilierte Version 2.10.1 [4] von Blast genutzt, um eine bessere Vergleichbarkeit zu ermöglichen.

Hierbei wird die Parallelisierung nicht wie bei Linpack und HPCG über ein MPI-Framework realisiert, sondern über den Start mehrerer Prozesse durch das make-System. Dies wird oft bei der Genomanalyse genutzt, da die zugehörigen Programme häufig nicht über Rechenknotengrenzen hinweg arbeiten können.

Die Messungen selbst wurden dann wieder durch Jobscripte und das Slurm-Batch-System durchgeführt, wobei die Ressourcenanforderungen im Bereich der #SBATCH-Optionen an den jeweiligen Rechenknotentypen angepasst werden müssen und sich natürlich die entsprechenden Pfade unterscheiden. Im Appendix findet sich unter A.6 ein Beispiel für den Typ SM7371-512. Die Scripte für die anderen Typen sind vergleichbar. Gut sichtbar ist der Befehl `/usr/bin/time` zur Zeitmessung.

Um den Einfluss des Dateisystems auf die Laufzeit zu bestimmen, musste jeder Rechenknotentyp verschiedene Läufe auf den drei großen Dateisystemen /data, /work und /scratch erledigen. Das Dateisystem, auf dem der Lauf stattfindet, wird im angefügten Jobscript A.6 durch den Befehl `cd $SCRATCH` (Zeile 16) definiert.

Ergebnisse

Knotentyp	**TotalCPU**
SM5650-64	44:04
SM5650-48	44:32
HU2658-192	36:36
HU2658-256	41:11
HU2699-512	30:59
SM4110-384	36:19
SM5217-384	26:04
D5217-384	25:56
F6246-768	20:51
SM7371-512	24:46

Abbildung 4.10: Laufzeit *TotalCPU* der Testläufe der einzelnen Rechenknotentypen für NCBI-Blast auf */data*.

Knotentyp	**TotalCPU**
SM5650-64	43:46
SM5650-48	44:25
HU2658-192	36:18
HU2658-256	41:26
HU2699-512	30:47
SM4110-384	36:18
SM5217-384	26:06
D5217-384	25:57
F6246-768	20:34
SM7371-512	24:37

Abbildung 4.11: Laufzeit *TotalCPU* der Testläufe der einzelnen Rechenknotentypen für NCBI-Blast auf */scratch*.

Knotentyp	**TotalCPU**
SM5650-64	43:53
SM5650-48	44:20
HU2658-192	36:12
HU2658-256	41:31
HU2699-512	30:54
SM4110-384	36:28
SM5217-384	25:50
D5217-384	25:56
F6246-768	20:43
SM7371-512	24:32

Abbildung 4.12: Laufzeit *TotalCPU* der Testläufe der einzelnen Rechenknotentypen für NCBI-Blast auf */work*.

Abbildung 4.13: Die Tabellen 4.10, 4.11 und 4.12 liefern eine Übersicht der Laufzeit *TotalCPU* der Testläufe auf den einzelnen Rechenknotentypen für NCBI-Blast. Hierbei repräsentiert jede der Tabellen die Läufe auf einem der Dateisysteme.

In den Tabellen der Abbildung 4.13 wurden alle gemessenen Laufzeiten der besten gewerteten Testläufe zum NCBI-Blast-Benchmark aufgelistet. Die Zeit versteht sich als Laufzeit für einen Benchmark-Durchlauf in in Minuten. Angegeben wurde wieder die Zeit des besten erfolgreichen Laufes.

Auswertung

In Abbildung 4.14 fällt direkt deutlich auf, dass zwar Unterschied zwischen den einzelnen Rechenknotentypen gemessen werden konnten, die verschiedenen Dateisysteme jedoch nur

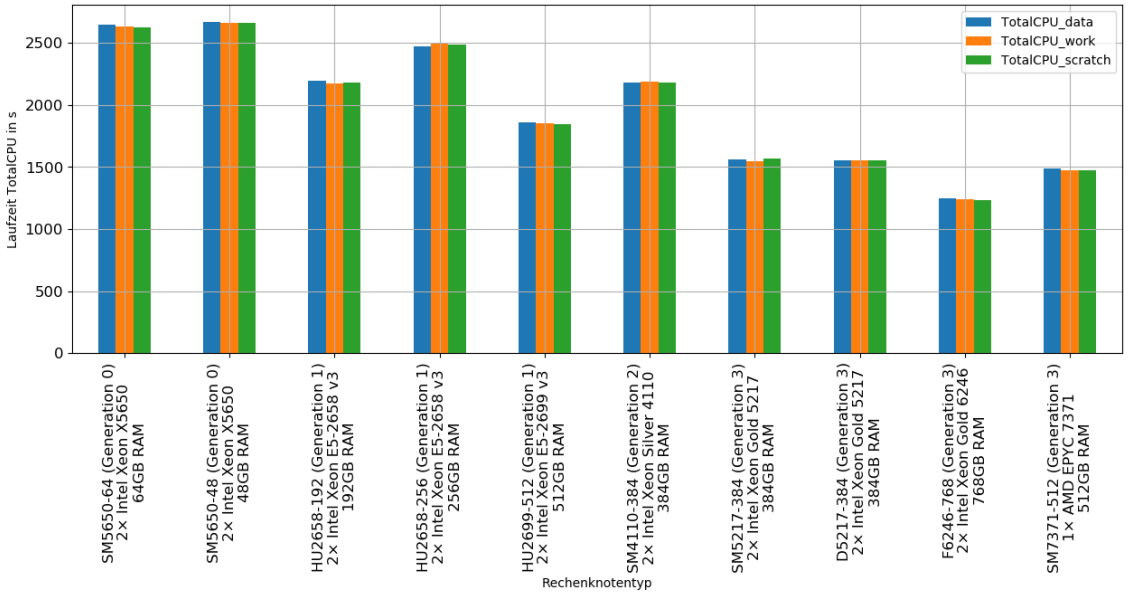

Abbildung 4.14: Lauftzeit von NCBI-Blast *TotalCPU* in Sekunden pro Rechenknotentyp. Kleinere Werte sind besser.

minimalste Schwankungen in der Laufzeit ergaben. Das Dateisystem scheint also überraschenderweise keinen sonderlich großen Einfluss auf diesen Benchmark zu haben. Hier wurde ein sichtbarer Unterschied zumindest zwischen /data und /scratch erwartet, der das lokale Dateisystem /scratch hätte sichtbar schneller erscheinen lassen. So schwankt die Laufzeit z.B. bei SM5650-64 nur zwischen 44:04 für /data und 43:46 für /scratch. Diese 18 Sekunden Unterschied sind bei einer Gesamtlaufzeit von ca. 44 Minuten verschwindend gering. Eine Erklärung dafür könnte sein, dass die Datenbank komplett im RAM gehalten werden konnte und daher keinen Einfluss erkennen lässt.

Nachfolgend werden daher zur Vereinfachung nur die Werte von den Läufen auf /scratch betrachtet, wenn dies nicht explizit anders angegeben wurde.

Die Laufzeit des Benchmarks in Abbildung 4.14 zeigt eine deutliche Beschleunigung neuerer Generationen. Dies betrifft in besonderem Maße die Rechenknoten der Generation 3, welche durchweg die besten Werte erreichen. So brauchte F6246-768 mit 20:34 im Vergleich zu den 43:46 des SM5650-64 weniger als die Hälfte der Zeit.

Die Leistungssteigerung der einzelnen CPU-Cores zeigt sich in Abbildung 4.15. So schneiden die alten CPUs der Generation 0 mit deutlichem Abstand besonders schlecht ab. Aber interessanterweise lieferten die Maschinen mit den meisten CPU-Cores (HU2699-512 und F6246-768) auch die höchste Leistung pro Core.

Den Einfluss der RAM-Größe zeigt sich in Abbildung 4.16 sehr gut. Die Laufzeit pro GB RAM ist um so kürzer, je mehr Arbeitsspeicher in der Maschine verbaut ist. So schneidet F6246-768 auch hier am besten ab.

Beim Vergleich der AMD EPYC-Rechenknoten SM7371-512 mit den ähnlich ausgestatteten Maschinen auf Intel-Basis HU2699-512 und SM5217-384 schneidet der AMD-Rechenknoten mit einer kürzeren Laufzeit in Abbildung 4.14 besser ab.

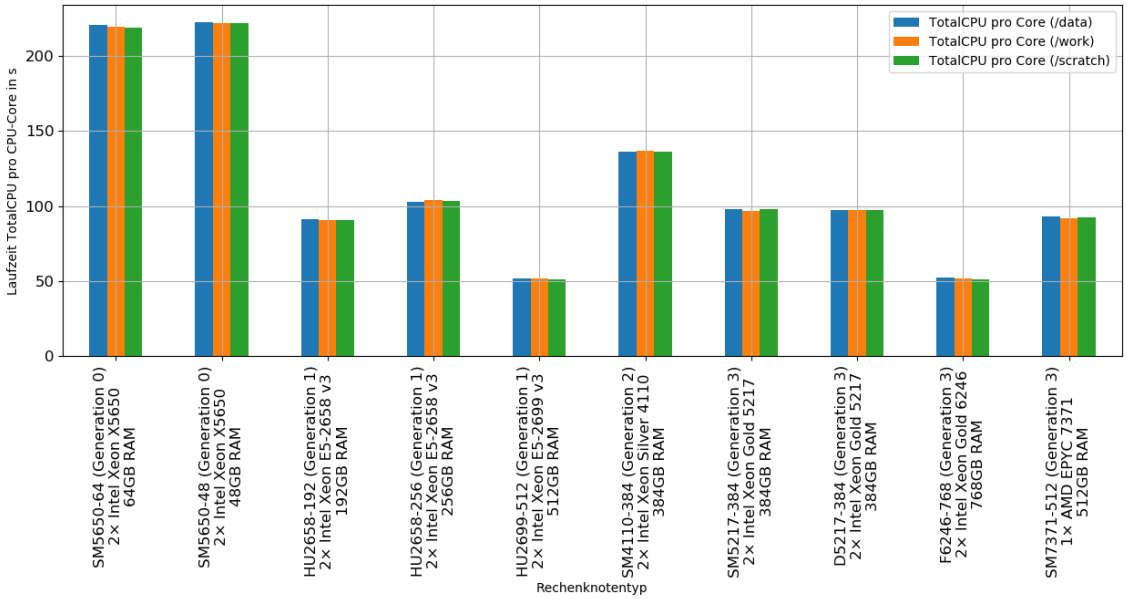

Abbildung 4.15: Lauftzeit von NCBI-Blast *TotalCPU* in Sekunden pro CPU-Core (ohne Hpyerthreading).
Kleinere Werte sind besser.

4.5 Zusammenführung der Ergebnisse

Unter Berücksichtigung der in den Benchmarks erreichten Ergebnisse sollen nun die in Kapitel 1.1 aufgestellten Fragestellungen beantwortet werden.

Welche der im OMICS-Cluster verbauten CPUs ist besonders für die Genomanalyse geeignet?

Im NCBI-Blast-Benchmark lieferten die CPUs der Generation 3 die besten Ergebnisse. Diese schnitten auch bei Linpack ab. In besonderem Maße scheinen sich CPus mit besonders vielen Cores zu eignen, hierzu zählen der Intel Xeon Gold 6246 des F6246-768 und der Intel Xeon E5-2699 v3 des HU2699-512.

Einen deutlicheren Einfluss scheint jedoch die RAM-Ausstattung zu besitzen.

Sind neuere CPU-Generationen generell besser geeignet?

Für Linpack lässt sich dies mit Blick auf Abbildung 4.4 lässt sich dies ganz klar bejahen, da eine deutliche Leistungssteigerung bei den Rechenknotentypen der Generation 2 und 3 erkennbar ist.

Dies trifft auch für NCBI-Blast zu, da Abbildung 4.14 erkennen lässt, dass die Laufzeit der Maschinen der Generation 3 durchweg schneller als die Rechenknoten der Generation 1 waren.

Bei HPCG ergibt sich hierzu leider kein so klares Bild. Unter Berücksichtigung der Abbildung 4.7 schneiden zwar die Rechenknoten der Generation 3 durchweg am besten ab,

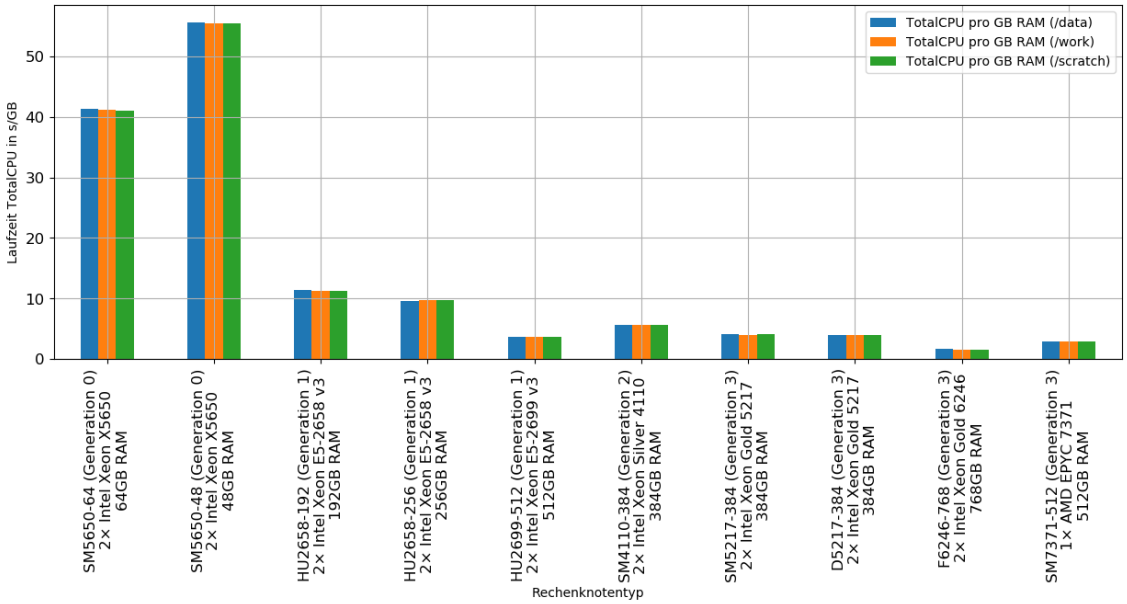

Abbildung 4.16: Lauftzeit von NCBI-Blast *TotalCPU* in Sekunden pro GB RAM. Kleinere Werte sind besser.

die Maschinen mit AMD EPYC-Prozessoren erreichten allerdings nur Werte auf dem Niveau der Generation 1-Maschinen. Bei der Leistung pro Core lieferten sogar überraschenderweise die alten Rechenknoten der Generation 0 die höchsten Werte, wie Abbildung 4.9.

Wie wirkt sich die RAM-Größe auf die Performance aus?

Während Linpack und HPCG nicht sonderlich von einer hohen Menge Arbeitsspeicher profitieren zu scheinen, sieht dies bei NCBI Blast anders aus. Hier schneiden die beiden Rechenknotentypen HU2699-512 mit 512GB und F6246-768 mit 768GB jeweils am besten in ihrer Generation ab.

Welchen Einfluss hat die RAM-Größe auf die Eignung zur Genomanalyse?

Bei Betrachtung der Ergebnisse des NCBI-Blast-Benchmarks scheint die RAM-Größe neben der CPU-Core-Anzahl einen entscheidenden Einfluss auf die Eignung zur Genomanalyse zu haben. Die Ergebnisse lassen erahnen, dass sich dies bei größeren Datensätzen fortsetzen wird.

Welche Aussagekraft hat ein hoher GFLOP/s-Wert in Linpack und HPCG auf die Genomanalyse?

Die typischen HPC-Benchmarks Linpack und HPCG haben eine mäßige Aussagekraft über die Eignung zur Genomanalyse. Durch sie lässt sich die Leistungsfähigkeit der CPU beurteilen. Durch den geringen Einfluss der RAM-Größe auf diese beiden Benchmarks fehlt jedoch eine große Komponente in deren Betrachtung.

Auch dürfte deren Beurteilungsstärke in der Betrachtung von Berechnungen liegen, die Rechenknotengrenzen überschreiten. Da dies aber von vielen Tools der Genomanalyse noch nicht unterstützt wird, ist dies für das OMICS-Cluster aktuell nebensächlich. Dies könnte sich ändern, falls auch die Genomanalyseprogramme gegen MPI-Bibliotheken entwickelt werden.

Wie verhalten sich Rechenknoten mit AMD EPYC-CPUs im Vergleich zu ähnlich ausgestatteten Maschinen mit Intel-Proesszoren?

Die AMD EPYC-Rechenknoten zeigten bei NCBI-Blast vergleichbare Ergebnisse zu den ähnlichen Rechenknoten mit Intel-CPUs. Obwohl sie in HPCG deutlich schlechter als die Intel-basierten Rechenknoten abschnitten, könnten sie eine ernsthafte Alternative darstellen, da sie deutlich kostengünstiger sind.

5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

Die Ergebnisse zeigen, dass die im HPC-Umfeld typischerweise genutzten Benchmarks Linpack und HPCG nur eine beschränkte Aussage zur Eignung der Maschinen zur Genomanalyse zulassen. Allerdings können sie als einen ersten Anhaltspunkt dienen, um die Leistungsstärke der Prozessoren beurteilen zu können. Zusätzlich muss allerdings dann noch die RAM-Ausstattung betrachtet werden.

Generell scheint die Arbeitsspeichergröße einen nicht zu vernachlässigenden Faktor bei der Genomanalyse zu spielen. In besonderem Maße zeigen dies die Abbildung 4.16 und 4.14, wo die Maschinen mit vielen CPU-Cores und hoher Arbeitsspeicherverfügbarkeit innerhalb ihrer Generation am besten abschneiden.

Wenn zusätzlich noch berücksichtigt wird, dass sich viele Programme für die Genomanalyse nicht mittels MPI knotenübergreifend parallelisiert werden können, sollte der Schritt weiter in Richtung potenter Rechenknoten gehen. Dafür sollten die Maschinen großzügig mit CPU-Cores und RAM ausgestattet werden, da diese Maschinen beim NCBI-Blast-Benchmark besonders gut abschnitten.

Die AMD EPYC-Rechenknoten stellten sich als sinnvolle Alternative zu denen mit vergleichbaren Intel-CPUs dar und sollten bei zukünftigen Beschaffungen weiter betrachtet werden.

5.2 Ausblick

Die Benchmarks könnten auch in Zukunft auf Rechenknoten ausgeweitet werden, die noch zum OMICS-Cluster hinzukommen werden. Damit würde sich eine größere Datenbasis ergeben. Hierzu könnten auch weitere, noch stärker auf die Genomanalyse zugeschnittene Benchmarks betrachtet werden, für die leider im Rahmen dieser Arbeit keine Zeit war.

Linpack und HPCG könnten auch auf Rechenknotengruppen und gegebenenfalls das komplette Cluster ausgeweitet werden, um auch Forschern aus anderen Disziplinen als der Genomanalyse einen Anhaltspunkt über die Leistungsfähigkeit des OMICS-Clusters zu ermöglichen. Dies war leider im Rahmen dieser Arbeit nicht möglich, da dann das Cluster für die mehrtägige Dauer des Benchmarks nicht anderweitig zur Verfügung gestanden hätte.

Mit selbst kompilierten BLAS-Bibliotheken und anderen BIOS-Einstellungen könnten eventuell höhere Werte bei Linpack oder HPCG erreicht werden.

Literatur

[1] Arora, R. *Conquering Big Data with High Performance Computing*. Switzerland: Springer-Verlag, 2016. ISBN: 978-3-319-33742-5.

[2] ASC Community Inspur (Beijing) Electronic Information Industry Co., L. *The Student Supercomputer Challenge Guide*. Science Press und Springer Nature Singapore Pte Ltd., 2018. ISBN: 9789811037313.

[3] Bauke, H. und Mertens, S. *Cluster Computing*. Berlin Heidelberg: Springer-Verlag, 2006. ISBN: 13 978-3-540-42299-0.

[4] *Blast 2.10.1*. https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.1/. Zugegriffen am 18.06.2020.

[5] *Blast Benchmark*. https://fiehnlab.ucdavis.edu/staff/kind/collector/benchmark/blast-benchmark. Zugegriffen am 16.06.2020.

[6] *Blast Benchmark Präsentation*. ftp://ftp.ncbi.nih.gov/blast/demo/benchmark/BLAST_benchmarks.ppt. Zugegriffen am 17.06.2020.

[7] *Blast Input Formats*. https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp. Zugegriffen am 17.06.2020.

[8] Dongarra, J., Heroux, M. A. und Luszczek, P. *HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems*. 2015.

[9] Dongarra, J., Luszczek, P. und Heroux, M. A. *HPCG Technical Specification*. 2013.

[10] Dongarra, J. J., Luszczek, P. und Petitet, A. *The LINPACK benchmark: Past, Present, and Future*. 2002.

[11] Dongarra, J., Gorra, D., Dongarra, J., Industrial, S. for, Mathematics, A., Bunch, J., Moler, C., Moler, G. und Stewart, G. *LINPACK Users' Guide*. Miscellaneous Bks. Society for Industrial und Applied Mathematics, 1979. ISBN: 9780898711721.

[12] *GCC GNU-C-Compiler*. https://packages.debian.org/stretch/gcc. Zugegriffen am 07.06.2020.

[13] *HPC Challenge Benchmark*. https://icl.utk.edu/hpcc/. Zugegriffen am 09.05.2020.

[14] *HPCG, HPL, STREAMS*. https://www.hpcg-benchmark.org/custom/index.html?lid=158&slid=281. Zugegriffen am 23.08.2020.

[15] *LAPACK Linear Algebra PACKage*. http://www.netlib.org/lapack/. Zugegriffen am 21.08.2020.

[16] *libblas-dev Basic Linear Algebra Subroutines 3, statische Bibliothek*. https://packages.debian.org/stretch/libblas-dev. Zugegriffen am 07.07.2020.

[17] *libopenblas-dev Optimierte Bibliothek für Lineare Algebra*. https://packages.debian.org/stretch/libopenblas-dev. Zugegriffen am 21.08.2020.

[18] Madden, T. *The BLAST Sequence Analysis Tool*. 2013.

[19] *MATLAB blastlocal*. https://www.mathworks.com/help/bioinfo/ref/blastlocal.html. Zugegriffen am 17.06.2020.

[20] *November 2019 Top500 Supercomputer Sites*. https://www.top500.org/lists/2019/11/. Zugegriffen am 09.02.2020.

[21] *Open MPI Leistungsstarke Message-Passing-Bibliothek - Programmdateien*. https://packages.debian.org/stretch/libopenmpi-bin. Zugegriffen am 07.06.2020.

[22] *Outline of the Development of the Supercomputer Fugaku*. https://www.r-ccs.riken.jp/en/fugaku/project/outline. Zugegriffen am 08.07.2020.

[23] *Qiime Blast Interface*. http://qiime.org/scripts/blast_wrapper.html. Zugegriffen am 17.06.2020.

[24] *Red Hat Gluster Storage 3.5 Administration Guide*. https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.5/pdf/administration_guide/Red_Hat_Gluster_Storage-3.5-Administration_Guide-en-US.pdf. Zugegriffen am 16.07.2020.

[25] Red Hat, Inc. *GlusterFS Illustration of a Distributed Dispersed Volume*. https://access.redhat.com/webassets/avalon/d/Red_Hat_Gluster_Storage-3.5-Administration_Guide-en-US/images/a7dcb285bdf18f46e85d61df2fbeb4a0/distributed_dispersed.png. Zugegriffen am 17.07.2020. 2017.

[26] Red Hat, Inc. *GlusterFS Illustration of a Three-way Distributed Replicated Volume*. https://access.redhat.com/webassets/avalon/d/Red_Hat_Gluster_Storage-3.5-Administration_Guide-en-US/images/90b08c44901d695d2509438d949ee85c/RH_Gluster_Storage_diagram_334434_0715_JCS_AdminGuide-08.png. Zugegriffen am 17.07.2020. 2017.

[27] Red Hat, Inc. *GlusterFS Red Hat Gluster Storage for On-premise Architecture*. https://access.redhat.com/webassets/avalon/d/Red_Hat_Storage-3.1-Administration_Guide-en-US/images/19f4d091b238a40d8a5f322e71c8fd38/RH_Gluster_Storage_diagrams_334434_0415_JCS_2.png. Zugegriffen am 16.07.2020. 2017.

[28] *SchedMD Slurm*. https://www.schedmd.com/. Zugegriffen am 22.01.2020.

[29] *Slurm Prolog and Epilog Guide*. https://slurm.schedmd.com/prolog_epilog.html. Zugegriffen am 16.07.2020.

[30] *Top500 Supercomputer Fugaku*. https://www.top500.org/system/179807/. Zugegriffen am 08.07.2020.

[31] *Top500 Supercomputer Sites*. https://www.top500.org/. Zugegriffen am 17.05.2020.

A

Jobscripte

A.1 Linpack

Programmtext A.1: Build-Script für Linpack mit libblas: build.sh

```
1  #!/bin/bash
2  #SBATCH --mem=20G
3  #SBATCH --cpus-per-task=8
4  #SBATCH --exclusive
5  #SBATCH --time=4:0:0
6
7  export RESULTDIR="$WORK/hpcc_intelmpi/$SLURM_JOBID"
8  export RESULTLOG="$RESULTDIR/build.log"
9
10 export HPCCTAR="$WORK/hpcc/source/hpcc-1.5.0.tar.gz"
11 export HPCCMAKE="$WORK/hpcc/source/Make.nodenative"
12 export HPCCLOG="$WORK/hpcc_intelmpi/$(hostname)/build.log"
13 export HPCCPATH="$WORK/hpcc_intelmpi/$(hostname)_native"
14
15 mkdir $RESULTDIR
16 mkdir --parents $HPCCPATH
17
18 source /etc/profile.d/modules.sh
19 shopt -s expand_aliases
20 module purge
21
22 module load intel-mpi/2019.5
23
24 cd $SCRATCH
25
26 tar xf $HPCCTAR
27 cd hpcc-1.5.0
28 cp $HPCCMAKE hpl/
29 make arch=nodenative |& tee --append $HPCCLOG
30
31 cp hpcc $HPCCPATH
32 echo "Done." |& tee --append $HPCCLOG
```

Programmtext A.2: Build-Script für Linpack mit OpenBLAS: build.sh

```
#!/bin/bash
#SBATCH --mem=20G
#SBATCH --cpus-per-task=8
#SBATCH --exclusive
#SBATCH --time=4:0:0

export RESULTDIR="$WORK/hpcc_intelmpi_openblas/$SLURM_JOBID"
export RESULTDIR2="$WORK/hpcc_intelmpii_openblas/$(hostname)"
export RESULTLOG="$RESULTDIR/build.log"

export HPCCTAR="$WORK/hpcc/source/hpcc-1.5.0.tar.gz"
export HPCCMAKE="$WORK/hpcc/source/Make.node18"
export HPCCMAKE2="$WORK/hpcc/source/Make.nodenative"
export HPCCMAKE3="$WORK/hpcc/source/Make.nodeopenblas"
export HPCCLOG="$WORK/hpcc_intelmpi_openblas/$(hostname)/build.log"
export HPCCPATH="$WORK/hpcc_intelmpi_openblas/$(hostname)"
export HPCCPATH2="$WORK/hpcc_intelmpi_openblas/$(hostname)_native"

mkdir $RESULTDIR
mkdir --parents $HPCCPATH
mkdir --parents $HPCCPATH2

source /etc/profile.d/modules.sh
shopt -s expand_aliases
module purge

module load intel-mpi/2019.5

cd $SCRATCH

tar xf $HPCCTAR
cd hpcc-1.5.0
cp $HPCCMAKE3 hpl/
make arch=nodeopenblas |& tee --append $HPCCLOG

cp hpcc $HPCCPATH2
echo "Done." |& tee --append $HPCCLOG
```

Programmtext A.3: Benchmark-Script für Linpack am Beispiel von node34 (SM7371-512): linpack_node34.sh

```
#!/bin/bash
#SBATCH --mem=480G
#SBATCH --cpus-per-task=32
#SBATCH --exclusive
#SBATCH --time=24:0:0

export REFERENZNODE="node34"
export REFERENZNODE2="node34_native"
export REFERENZNODE3="node34_native"
export RESULTDIR1="$WORK/hpcc_intelmpi/$SLURM_JOBID/native"
export RESULTDIR2="$WORK/hpcc_intelmpi/$SLURM_JOBID/native_openblas"
```

```
12 export HPCCCFG="$WORK/hpcc_intelmpi/$REFERENZNODE/hpccinf.txt"
13 export HPCCBIN1="$WORK/hpcc_intelmpi/$REFERENZNODE2/hpcc"
14 export HPCCBIN2="$WORK/hpcc_intelmpi_openblas/$REFERENZNODE3/hpcc"
15 export HPCCCPU="$(nproc)"
16 # Choose binaries:
17 export TESTATLASNATIVE=""
18 export TESTOPENBLASNATIVE="yes"
19 source /etc/profile.d/modules.sh
20 shopt -s expand_aliases
21 module purge
22 module load intel-mpi/2019.5
23
24 if [ -z $TESTATLASNATIVE ]
25 then
26         echo "### Test native atlas skipped ###"
27 else
28         mkdir $RESULTDIR1
29         cd $RESULTDIR1
30         cp $HPCCBIN1 .
31         cp $HPCCCFG .
32         /usr/bin/time --verbose mpirun -np $HPCCCPU ./hpcc |& tee --append mpiexec.log
33
34         echo ""
35         echo "### Result naitve ###"
36         cat $RESULTDIR1/hpccoutf.txt
37 fi
38
39 if [ -z $TESTOPENBLASNATIVE ]
40 then
41         echo "### Test native openblas skipped ###"
42 else
43         mkdir $RESULTDIR2
44         cd $RESULTDIR2
45         cp $HPCCBIN2 .
46         cp $HPCCCFG .
47         /usr/bin/time --verbose mpirun -np $HPCCCPU ./hpcc |& tee --append mpiexec.log
48
49         echo ""
50         echo "### Result naitve openblas ###"
51         cat $RESULTDIR2/hpccoutf.txt
52 fi
```

A.2 HPCG

Programmtext A.4: Build-Script für HPCG: build.sh

```
1 #!/bin/bash
2 #SBATCH --mem=20G
3 #SBATCH --cpus-per-task=8
4 #SBATCH --exclusive
5 #SBATCH --time=4:0:0
```

```
6
7  export RESULTDIR="$WORK/hpcg/$SLURM_JOBID"
8  export RESULTDIR2="$WORK/hpcg/$(hostname)"
9  export RESULTLOG="$RESULTDIR/build.log"
10 export SOURCETAR="$WORK/hpcg/source/hpcg-3.1.tar.gz"
11 mkdir $RESULTDIR
12
13 source /etc/profile.d/modules.sh
14 shopt -s expand_aliases
15 module purge
16
17 cd $SCRATCH
18 tar xf $SOURCETAR
19 mkdir hpcg-3.1/build
20 cd hpcg-3.1/build
21 ../configure GCC_OMP |& tee --append $RESULTLOG
22 make |& tee --append $RESULTLOG
23
24 cp bin/xhpcg $RESULTDIR
25 cp -r "$RESULTDIR" "$RESULTDIR2"
26
27 echo "Done." |& tee --append $RESULTLOG
```

Programmtext A.5: Benchmark-Script für HPCG am Beispiel von node34 (SM7371-512): node34.sh

```
1  #!/bin/bash
2  #SBATCH --mem=300G
3  #SBATCH --cpus-per-task=32
4  #SBATCH --exclusive
5  #SBATCH --time=12:0:0
6
7  export RESULTDIR="$WORK/hpcg/$SLURM_JOBID"
8  export HPCGEXE="$WORK/hpcg/node34/xhpcg"
9  export RESULTLOG=$WORK/hpcg/$SLURM_JOBID/run.log
10
11 mkdir $RESULTDIR
12
13 source /etc/profile.d/modules.sh
14 shopt -s expand_aliases
15 module purge
16
17 cd $SCRATCH
18 cp $HPCGEXE .
19
20 export nx=208
21 export ny=208
22 export nz=208
23 export rt=60
24 srun /usr/bin/time --verbose $HPCGEXE $nx $ny $nz $rt |& tee --append $RESULTLOG
25
26 mv ./HPCG-Benchmark_3.1_* $RESULTDIR
27 echo "Done."
```

A.3 NCBI-Blast

Programmtext A.6: Benchmark-Script für NCBI-Blast am Beispiel von node34 (SM7371-512) und $SCRATCH: node34.sh

```
1  #!/bin/bash
2  #SBATCH --mem=480G
3  #SBATCH --cpus-per-task=32
4  #SBATCH --exclusive
5  #SBATCH --time=1-00:00:00
6
7  export RESULTDIR="$WORK/ncbi-blast/$SLURM_JOBID"
8  export RESULTTAR="$WORK/ncbi-blast/$SLURM_JOBID/$SLURM_JOBID.tar.gz"
9  export BLASTSOURCE="$WORK/ncbi-blast/benchmark2013.tar.gz"
10
11 source /etc/profile.d/modules.sh
12 shopt -s expand_aliases
13 module purge
14 module load ncbi-blast/2.10.1
15
16 cd $SCRATCH
17
18 cp $BLASTSOURCE .
19 tar xf benchmark2013.tar.gz
20 cd benchmark
21
22 srun /usr/bin/time --verbose make -j$(nproc) |& tee --append $RESULTDIR/make01.log
23 tar czf $RESULTTAR output/ queries/
24 echo "Done."
```