HUMBOLDT-UNIVERSITÄT ZU BERLIN

Exposé zur Bachelorarbeit

Implementierung eines NGS-Workflows unter Verwendung von MapReduce und Hadoop

Autor: Carsten Lipka

Tutoren: Prof. Dr. Ulf Leser
Marc N. Bux
Jörgen Brandt

Ziel dieser Bachelorarbeit ist die Implementation eines NGS-Workflows mit Hilfe des MapReduce-Programmiermodells unter Verwendung verschiedener Erweiterungen von Hadoop. Die Arbeit soll Aufklärung darüber schaffen, wo die Schwierigkeiten darin liegen, einen solchen Workflow mit dem MapReduce-Paradigma umzusetzen und welche Technologien zur Lösung des Problems am besten geeignet sind.

1 Einführung

Die Sequenzierung eines menschlichen Genoms, d. h. die Ermittlung der Abfolge der Basenpaare innerhalb einer DNA, kann auf unterschiedliche Arten erfolgen. Neben der traditionellen Sequenzierungsmethode nach Sanger haben sich in den letzten Jahren mehrere Verfahren etabliert, die unter dem Begriff Next-Generation-Sequencing (NGS) zusammengefasst werden. Mit der Weiterentwicklung der ursprünglichen Methode sank die benötigte Zeit für die Sequenzierung eines Genoms. Dadurch stieg die Menge der produzierten Daten (mehrere GB pro sequenziertem Genom) in gleichem Zeitraum deutlich an [6]. Ergebnis einer Sequenzierung sind Daten in Form von Zeichenketten (Reads) sowie eine zusätzliche Einschätzung der Sequenzierungsqualität pro Base. Mit einer Länge von 30 bis 500 Basenpaaren, gegenüber den über drei Milliarden Basenpaaren des gesamten menschlichen Genoms, sind diese Reads deutlich kürzer. Eine einzelne Sequenz eines Reads ist in Abbildung 1 dargestellt.

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

Abbildung 1: Die Ausgabe einer einzelnen Sequenz eines Sequenzierungsgerätes im FASTQ-Format.

NGS wird in drei Analysephasen gegliedert [5, 10]. In der primären Phase ermitteln Sequenzierungsgeräte aus dem verfügbaren DNA-Material die oben genannten Reads und zugehörige Qualitätskennzeichnungen. Die sekundäre Phase beinhaltet das Alignment und das Variant Calling. Beim Alignment handelt es sich um den Vergleich zwischen Reads und einem Referenzgenom und der damit verbundenen Positionsbestimmung der Reads bezüglich der DNA-Sequenz. Das Variant Calling bezeichnet den Vorgang der Ermittlung von Abweichungen einzelner Basen in den Reads bezogen auf das Referenzgenom.

Die tertiäre Phase beendet den Vorgang mit der Auswertung, z. B. mit einem Vergleich der Sequenz der DNA von Tumorgewebe mit der DNA gesunden Gewebes. Um die Übersichtlichkeit über diese Vorgänge zu erhöhen, werden Scientific Workflows verwendet. Ein Scientific Workflow bezeichnet einen gerichteten, azyklischen Graphen (DAG), dessen Knoten Aufgaben und dessen Kanten Datenflüsse zwischen diesen Aufgaben beschreiben. Damit gibt dieser DAG weiterhin Aufschluss über alle Datenabhängigkeiten. In Abbildung 2 ist ein Scientific Workflow dargestellt, der in der primären Phase entstandene Daten aufnimmt und in der sekundären und tertiären Phase verarbeitet [11]. Ziel eines solchen Workflows ist z. B. die Detektion von Biomarkern. Für die Auswahl einer adäquaten Behandlung wird dabei die Eigenschaft genutzt, Mutationen finden zu können, die für einen Tumor typisch sind. War ein Medikament bei der Behandlung eines bestimmten Tumors erfolgreich, könnte es möglicherweise auch erfolgreich zur Behandlung eines anderen Tumors mit ähnlichen Mutationen eingesetzt werden.

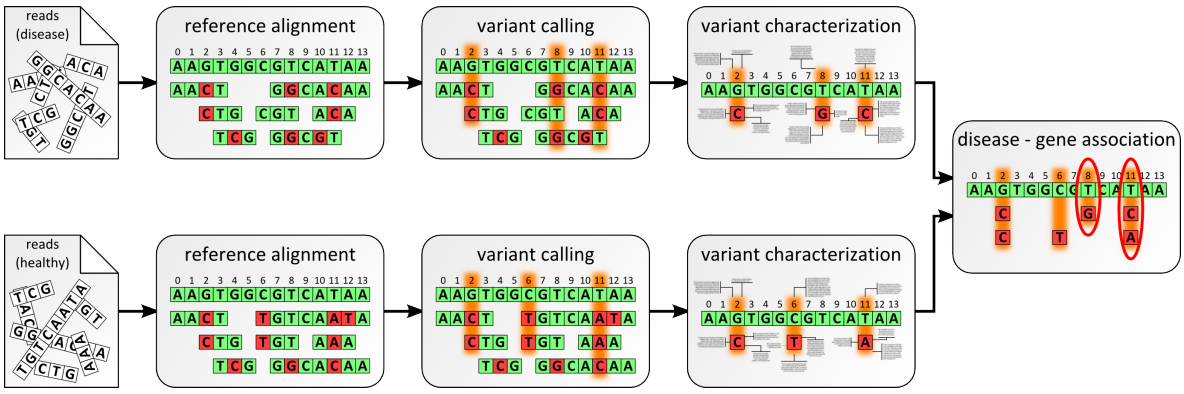

Abbildung 2: NGS-Workflow zur Anordnung und anschließender Auswertung zweier Read-Pools sowie abschließender, vergleichender Assoziationsuntersuchung. Bild entnommen aus [11].

Um die bei der primären Phase entstehenden Datenmengen verarbeiten zu können, muss eine sehr große Rechenleistung zur Verfügung gestellt werden [7]. Da diese Daten bei Lebenswissenschaftlern grundsätzlich nur sporadisch anfallen, wäre der Kostenaufwand für einen Supercomputer unverhältnismäßig hoch. Zur Lösung dieses stetig anwachsenden Problems, das in der Abbildung 2 im Paper [8] gut zu erkennen ist, wird immer häufiger die Berechnung auf Commodity-Hardware-Clustern diskutiert. Diese Art von Cluster verbindet eine genügend große Anzahl von günstigen Commodity-Hardware-Rechnern zur gewünschten Berechnungsleistung.

Ein Programmiermodell, das zur parallelen Berechnung genutzt wird, ist MapReduce [9]. MapReduce wurde von Google entwickelt, um auf hunderten oder tausenden von handelsüblichen Computern, Daten parallel verarbeiten zu können. Es lässt sich jedoch ebenso auf einem einzelnen Computer nutzen, indem die Verteilung nicht auf Computerebene, sondern auf CPU-Ebene stattfindet. Dabei werden große, zu verarbeitende Datenmengen aufgeteilt, an verschiedene Knoten einer Rechenarchitektur geschickt und parallel verarbeitet (Map-Phase) sowie deren Zwischenergebnisse im Anschluss erst sortiert und an weitere Knoten der Rechenarchitektur verteilt. Im Abschluss (Reduce-Phase) werden die Daten, die sich nach Sortieren mittels Key-Value-Paaren einen Schlüssel teilen, weiter parallel verarbeitet. Dabei greifen Map- sowie Reduce-Phase bei der Verarbeitung der Daten auf ein verteiltes Dateisystem zu. Abbildung 3 beschreibt diesen Vorgang noch deutlicher.

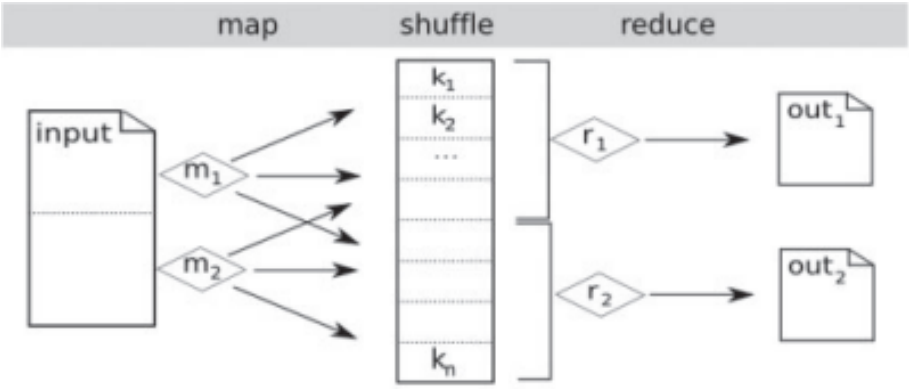

Abbildung 3: MapReduce-Schema. Die Eingabedateien werden abhängig von Größe und gewünschter Anzahl von Map-Instanzen automatisch in Blöcke zerlegt. Jede Map-Instanz (hier dargestellt durch m_1 und m_2) führt eine Funktion auf dem Eingabe-Block aus und erzeugt Key-Value-Paare. In der Shuffle-Phase wird eine Liste aus mit jedem Key (k_1, k_2 und k_n) verbundener Values erstellt. Die Reduce-Instanzen (r_1 und r_2) führen eine Funktion für ihre jeweilige Teilmenge von Keys und verbundener Liste von Values aus und erzeugen die Menge der Ausgabedateien. Bild entnommen aus [16].

Eine Open-Source-Implementation von MapReduce, die weit verbreitet zur Verarbeitung sehr großer Datenmengen angewendet wird, ist Hadoop [1, 2, 3]. Hadoop-Programme lassen sich überall dort ausführen, wo das Java-basierte Hadoop-Framework installiert ist. Dies bedeutet ebenfalls, dass die Programme unabhängig von der Hadoop zugrundeliegenden Plattform funktionieren. Dabei kann wie bei Googles MapReduce sowohl ein einzelner Computer als auch ein Cluster von Maschinen oder ein Supercomputer zum Einsatz kommen. Hadoop verwendet ein eigenes Dateisystem, das als Hadoop Distributed File System (HDFS) bezeichnet wird. Dateien werden in Form von Blöcken (standardmäßig 128 MB) gespeichert, welche redundant in der Standardkonfiguration auf mindestens drei Maschinen des Hadoop-Clusters aufbewahrt werden. Hadoop übernimmt die Konfiguration, d.h. die Verwaltung und Zusammenarbeit der verteilten Ressourcen.

2 Ziele

Ziel dieser Bachelorarbeit ist die Implementation eines NGS-Workflows mit Hilfe des MapReduce-Programmiermodells unter Verwendung verschiedener Erweiterungen von Hadoop. Die Arbeit soll weiterhin Aufklärung darüber geben, wo die Schwierigkeiten darin liegen, einen solchen Workflow mit dem MapReduce-Paradigma umzusetzen und welche Technologien zur Lösung des Problemes am besten geeignet sind. Weiterhin sollen Ergebnisse darüber geliefert werden, welche Phasen des Workflows besonders rechen- bzw. speicherintensiv sind. Benchmarks sollen Aufschluss darüber geben, welche Güte bezüglich Skalierbarkeit die Implementation besitzt.

3 Herangehensweise

Die erste Schwierigkeit bei der Implementierung ist die Überführung des DAGs des Scientific Workflows aus der Aufgabenstellung in das MapReduce-Modell, das Hadoop verwendet. Bei den Knoten dieses DAGs handelt es sich um Map- oder Reduce-Schritte in Form von Kommandozeilen-Tools, bei den Kanten um den Datenfluss zwischen den Tools sowie den damit verbundenen Abhängigkeiten.

Der Teil des Workflows, der das Reference Alignment und das Variant Calling umfasst, lässt sich in MapReduce definieren. Der gesamte Workflow ist jedoch nicht direkt in einem MapReduce-Schritt umsetzbar, da sich nach der Reduce-Phase direkt noch ein weiterer Reduce-Schritt anschließt. Eine Lösung für dieses Problem bietet TEZ [4], eine Erweiterung des MapReduce-Modells, die es ermöglicht, beliebige DAGs von Map- und Reduce-Schritten zu spezifizieren. TEZ verspricht eine deutliche Leistungssteigerung im Vergleich zu älteren Hadoop-Versionen, da es auf YARN aufsetzt.

Eine weitere Schwierigkeit bei der Umsetzung ist einerseits das teilweise Fehlen von Java-Interfaces in den benötigten Tools des Workflows, andererseits die Inkompatibilität der Kommandozeilen-Tools bei der Verwendung als Mapper/Reducer. Split-Vorgänge sowie Key-Value-Paare werden bei diesen Kommandozeilen-Tools teilweise anders behandelt als in MapReduce. Somit ist es erforderlich Java-Wrapper für die Kommandozeilen-Tools entsprechender Map- und Reduce-Schritte bereitzustellen, da Reengineering der Tools in Java, wie bei ADAM umgesetzt, im Rahmen der Bachelorarbeit ausgeschlossen ist. Weiterhin ist es für diese Tools notwendig, eine Brücke zum lokalen Dateisystem zu schaffen, während Hadoop standardmäßig auf HDFS zugreift.

Hadoop Streaming, das ebenfalls Teil von Hadoop ist, benötigt keine Java-Interfaces und stellt eine Verbindung zum lokalen Dateisystem bereit. Es lässt eine direkte Zuweisung von Kommandozeilen-Tools als Mapper und Reducer zu und ermöglicht weiterhin das Streaming von Dateien zu den Tools, die auf den Workern installiert sind. Crossbow verwendet das Hadoop Streaming Interface zum Aufruf von Bowtie als Mapper und SOAPsnp als Reducer. Der von Crossbow umgesetzte Workflow erfasst aber nur die sekundäre Phase vom NGS. Trotzdem lassen sich hier Parallelen zum zu implementierenden Workflow erkennen.

Die von den Sequenzierungsgeräten stammenden FASTQ-Dateien werden zum parallelen Alignment erst geeignet aufgeteilt, um dann von unterschiedlichen Map-Tasks parallel verarbeitet werden zu können. Das Alignment der Splits findet auf den jeweiligen Map-Tasks mit Hilfe der drei unterschiedlichen Tools Bowtie, SHRiMP [19] und PerM [20] statt. Dazu liegt auf allen Maschinen eine Kopie des benötigten Referenzgenoms vor. Die anschließende Sor-

tierung nach Key-Value-Paaren erfolgt automatisch durch den Shuffle-Prozess von Hadoop zwischen Map- und Reduce-Phase. Im Anschluss werden diese alignierten sowie sortierten Sequenzierungsteile (SAM-Dateien) in Partitionen bezüglich des Referenzgenoms aufgeteilt und mittels SAMtools weiterverarbeitet und zusammengeführt. Die Ausgabe der Ergebnisse erfolgt über die selbe Verbindung zum lokalen Dateisystem wie die Eingabe. Teil der Bachelorarbeit ist es also ebenfalls, bei der Implementierung des NGS-Workflows herauszufinden, ob TEZ oder Hadoop Streaming besser für die Umsetzung, d. h. geringere Schwierigkeiten bei der Anpassung sowie bessere Performance, geeignet ist. Zur Messung der Skalierbarkeit wird die benötigte Gesamtzeit zur Ausführung des Scientific Workflows ins Verhältnis zur Anzahl der Worker der Ausführungsarchitektur gesetzt. Weiterhin wird der Zeitaufwand einzelner Aufgaben des Scientific Workflows betrachtet. Die Arbeitsschritte, die im Rahmen der Bachelorarbeit vorgesehen sind, können folgendermaßen zusammengefasst werden:

- Einrichtung der Ausführungsumgebung Hadoop sowie Einbindung der Erweiterung TEZ
- Integration der Kommandozeilen-Tools der jeweiligen Map-/Reduce-Schritte in Hadoop Streaming sowie zusätzliche Implementation von Wrapperklassen dieser Tools für TEZ
- Dokumentation der Vorgänge bei der Implementation
- Entwicklung, Ausführung und Auswertung geeigneter Benchmarks

4 Verwandte Arbeiten

Bestehende Ansätze, die MapReduce und NGS zusammenbringen, gibt es bereits einige: Zum Beispiel Crossbow [12], CloudBurst [16], das Genome Analysis Toolkit (GATK) [17] und Adam [18]. Diese Ansätze ermöglichen zum Teil eine Berechnung via Cloud Computing, d. h. mittels virtueller, über das Internet erreichbarer Server.

Crossbow ist eine skalierbare Software-Pipeline zur Sequenzanalyse ganzer Genome. Es kombiniert Bowtie [13], ein Tool zum Alignment von kurzen Reads, mit SOAPsnp [14], einem Tool zur Erkennung von Übereinstimmungen und Abweichungen unterschiedlicher Genome. Diese Pipeline von Tools läuft in der Cloud (z. B. unter Verwendung von Elastic MapReduce [15], eines mietbaren Amazon Web Services), auf einem lokalen Hadoop-Cluster unter Verwendung mehrerer Computer oder auf einem einzelnen Computer unter Verwendung mehrerer CPUs, falls möglich. Abbildung 4 verdeutlicht die Architektur von Crossbow.

CloudBurst ist ein paralleler Read-Alignment-Algorithmus für NGS-Daten. Zur Anwendung kommt dabei das Tool RMAP, das speziell in Hadoop und MapReduce integriert wurde, um eine Verteilung und damit verbundene Beschleunigung zu ermöglichen.

Das GATK verwendet wie Crossbow und CloudBurst zwar MapReduce. Als Programmiermodell, jedoch wird bei der Ausführungsumgebung nicht auf Hadoop gesetzt, sondern ein eigenes Java-Framework verwendet. Das GATK deckt alle Bereiche des Genom-Analyse-Problems ab und kann mit einer Reihe verschiedener Daten unterschiedlicher Sequenzierungstechnologien umgehen.

Ein weiterer Ansatz, der jedoch nicht auf Hadoop, sondern auf Spark basiert, ist ADAM, ein Satz neuer Dateiformate, APIs und Implementationen von Workflow-Schritten, bei dessen Entwicklung die Skalierbarkeit auf große Datenmengen im Vordergrund steht. Nachteil dabei ist die fehlende Flexibilität sowie Kompatibilität zu etablierten Datenformaten.
Somit kommen die beiden Arbeiten von Crossbow und CloudBurst der Aufgabenstellung am nächsten.

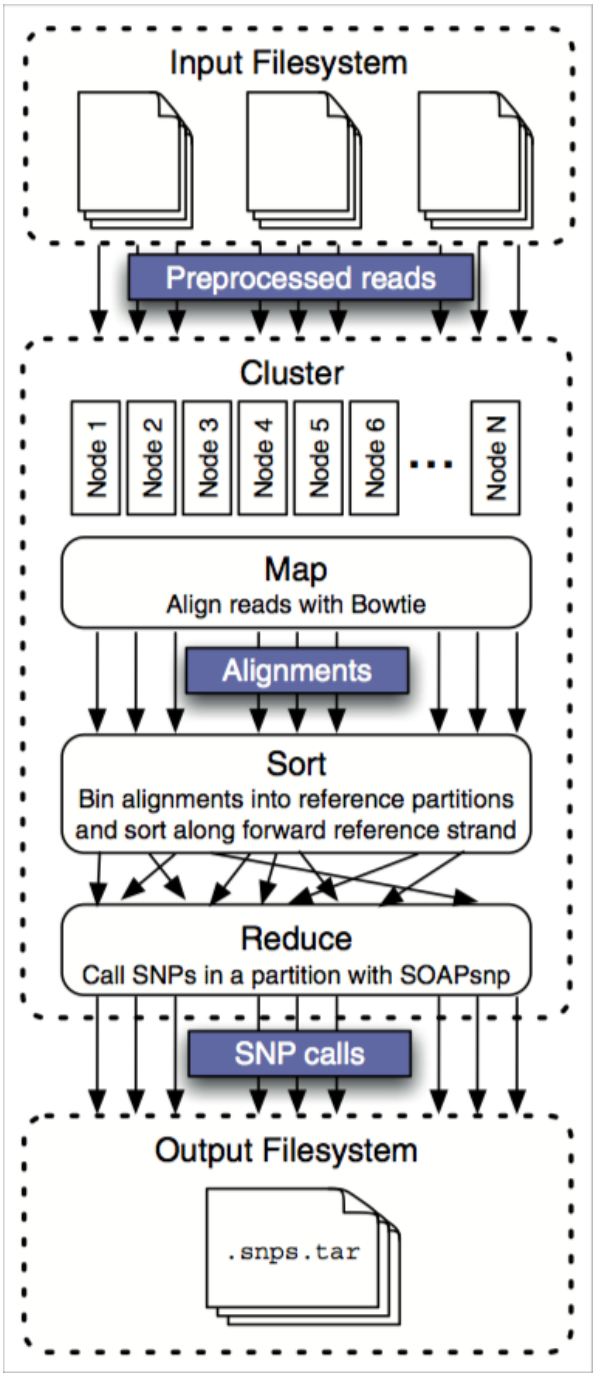

Abbildung 4: Crossbow-Workflow. Vorher kopierte Reads werden in den Cluster geladen, dekomprimiert und mit einer großen Anzahl von Bowtie-Instanzen parallel aligniert. Anschließend sortiert und verteilt Hadoop bezüglich primärer und sekundärer Schlüssel. Sortierte Alignments werden nach Partitionen des Referenzgenoms zusammengefasst und an parallele Instanzen von SOAPsnp geschickt. Das Ergebnis sind Erkennungsdaten von Einzelnukleotid-Polymorphismen (SNP-Calls). Bild entnommen aus [12].

Literatur

[1] *Apache Hadoop.* URL: http://hadoop.apache.org/

[2] Tom White. *Hadoop: The Definitive Guide*, 2009. URL: http://www.dhafley.com/books/OReilly.Hadoop.The.Definitive.Guide.Jun.2009.pdf

[3] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sharad Agarwal, Hites Shah, Sanja Radia, Benjamin Reed und Eric Baldeschwieler. *Apache Hadoop YARN: Yet Another Resource Negotiator*, 2013. URL: http://www.cs.cmu.edu/~garth/15719/papers/yarn.pdf

[4] *Apache Tez.* URL: http://tez.incubator.apache.org/

[5] Gabe Rudy. *A Hitchhiker's Guide to Next-Generation Sequencing.* URL: http://www.goldenhelix.com/pdfs/whitepapers/Hitchhikers-Guide-to-NGS.pdf

[6] *NextSeq 500 and HiSeq X Ten: NT Lowering Cost/Mbp.* URL: http://blog.genohub.com/nextseq-500-and-hiseq-x-ten-services-coming-soon-to-genohub-com/

[7] Elizabeth Pennisi. *Will Computers Crash Genomics?* URL: http://www.stanford.edu/class/cs262/papers/WillComputersCrashGenomics.pdf

[8] Lincoln D Stein. *The case for cloud computing in genome informatics.* URL: http://genomebiology.com/2010/11/5/207

[9] Quan Zou, Xu-Bin Li,Wen-Rui Jiang, Zi-Yu Lin, Gui-Lin Li und Ke Chen. *Survey of MapReduce frame operation in bioinformatics.* Dezember 2012. URL: http://bib.oxfordjournals.org/content/early/2013/02/07/bib.bbs088.full.pdf

[10] Heng Li und Nils Homer. *A survey of sequence alignment algorithms for next-generation sequencing.* April 2010. URL: http://bib.oxfordjournals.org/content/11/5/473.full.pdf

[11] Marc Bux und Ulf Leser. *Parallelization in Scientific Workflow Management Systems.* 2013. URL: http://www2.informatik.hu-berlin.de/buxmarcn/publications/bux_par_swfms_corr_2013.pdf

[12] Ben Langmead, Michael C. Schatz, Jimmy Lin, Mihai Pop und Steven L. Salzberg. *Searching for SNPs with cloud computing.* November 2009. URL: http://genomebiology.com/content/pdf/gb-2009-10-11-r134.pdf

[13] *Bowtie.* URL: http://bowtie-bio.sourceforge.net/

[14] *SOAPsnp.* URL: http://soap.genomics.org.cn/soapsnp.html

[15] *Elastic MapReduce.* URL: http://aws.amazon.com/elasticmapreduce/

[16] Michael C. Schatz. *CloudBurst: highly sensitive read mapping with MapReduce.* April 2009. URL: http://bioinformatics.oxfordjournals.org/content/25/11/1363.full.pdf

[17] *Genome Analysis Toolkit.* URL: http://www.broadinstitute.org/gatk/

[18] Matt Massie, Frank Nothaft, Christopher Hartl, Christos Kozanitis, André Schumacher, Anthony D. Joseph und David A. Patterson. *ADAM: Genomics Formats and Processing Patterns for Cloud Scale Computing.* Dezember 2013. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-207.pdf

[19] *SHRiMP.* URL: http://compbio.cs.toronto.edu/shrimp/

[20] *PerM.* URL: https://code.google.com/p/perm/