HTL - Perg

Höhere Abteilung für Informatik

Diplomarbeit

XBOX Cluster

Projektteam: Florian DANIEL

Projektbetreuer: Dipl-Ing. Dr. Michael Buchberger

Bearbeitungszeitraum: 01.10.2014 – 30.04.2015

Eidestaatliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Ort, Datum	Florian DANIEL

Danksagung

Ich möchte mich zuerst bei Herrn Dipl.-Ing. Dr. Michael Buchberger bedanken, da ohne Ihn die Fertigstellung und Realisierung dieser Diplomarbeit nie möglich gewesen wäre. Mein weiterer Dank gilt der HTL Perg, welche die Infrastruktur dieser Arbeit zur Verfügung gestellt hat. Sämtlichen Schülern, allen voran Lukas Bindreiter, welche mich ebenso unterstützt haben, gilt ein ganz besonderer Dank. Weiters ist ein herzliches Dankeschön für meine Eltern angebracht, die mich nach ihren besten Möglichkeiten unterstützt haben.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Kurzfassung

Das Ziel dieser Arbeit liegt in der Errichtung einer Cluster-Infrastruktur. Der Cluster wird durch den Zusammenschluss von Spielkonsolen realisiert. Die verwendete Konsole, die XBOX 360, wurde von Microsoft entwickelt und im Zuge der Arbeiten dahingehend modifiziert, dass es nun möglich ist, Programme auf der XBOX zur Ausführung zu bringen, welche von Microsoft nicht kontrolliert und zertifiziert werden müssen. Jede eingesetzte XBOX wurde baulich verändert und kommuniziert über das Netzwerk mit dem sogenannten XBOX-Master, welcher den Cluster und somit jede XBOX steuert und kontrolliert. Die Kontrolle wird durch die Implementierung einer Webseite realisiert, welche bestimmte Daten einer XBOX abfragen kann und diese in einer Datenbank persistiert. Zudem wird auf die Verwendung von Drittanbieter Software gesetzt, welche eine detaillierte Überwachung des Clusters ermöglicht. Um Berechnungen und Tasks, welche dem Cluster übergeben werden, gleichmäßig zu verteilen wird die Software „Simple Linux Utility for Resource Management“ (SLURM) eingesetzt.

1.2 Abstract

The goal of this thesis is the construction of a cluster infrastructure. Gaming consoles are used to realise the cluster. The used console, the XBOX 360, was developed by Microsoft and has been modified to support the execution of programms, which are not certified and controlled by Microsoft. Every single XBOX's Hardware has been adapted und is communicating with the so called XBOX-Master, which controls the Cluster and therefore every XBOX. For monitoring purpose a website has been implemented, gathering information from a XBOX and saving this information in a database. Furthermore third-party software is used to supervise the cluster in detail. To distribute Tasks and Calculations equaly the software „Simple Linux Utility For Resource Managment“ (SLURM) has been installed.

1.3 Motivation

Die Realisierung eines High Performance Clusters, welcher exklusiv innerhalb der Infrastruktur der HTL Perg erreichbar ist, ermöglicht eine wesentliche Erweiterung der möglichen Themengebiete, welche in den technischen Fächern gelehrt werden können. Nicht nur kann mit Schüler und Schülerinnen der Cluster und dessen Aufbau studiert werden, es wird auch möglich, Programme zu entwickeln, welche parallele Berechnungen auf verteilten Systemen unterstützten.

1.4 Gliederung

Die Einführung, Kapitel 1, enthält eine Kurzfassung der Arbeit in deutscher und englischer Fassung, sowie die Motivation und die Gliederung der Arbeit. Vergleichbare Arbeiten werden im Kapitel 2 „Stand der Wissenschaft“ behandelt. Im Kapitel 3 „Hardware“ wird die verwendete XBOX 360, sowie weitere verwendete Hardware beschrieben. Zudem wird erläutert, welche Schritte notwendig sind, um die XBOX 360 zu hacken. Protokolle, Software und entsprechende Konfigurationen um einen Cluster-Verbund funktionsfähig einzurichten, werden im Kapitel 4 „Low-Level Software“ erläutert. Die verwendeten Betriebssysteme und deren Installation werden im Kapitel 5 „Betriebssysteme“ beschrieben. Weiters findet sich dort eine Anleitung zum Erzeugen eines Linux-Kernels. Die Erläuterung der implementierten Webseite zur Überwachung des Clusters findet sich im Kapitel 6 „Überwachung des Clusters“. Open Source Programme, welche ebenfalls zur Überwachung des Clusters eingesetzt werden können, werden im Kapitel 7 „Managementsoftware“ beschrieben. Kapitel 8 „Weiterführende Arbeiten“ beschäftigt sich mit möglichen Erweiterungen der Arbeit. Im letzten Kapitel 9 „Fazit“ gibt der Autor einen Überblick über seine Erfahrungen und Eindrücke.

Kapitel 2

Stand der Wissenschaft

In den folgenden Abschnitten finden sich Beschreibungen für ähnliche Infrastrukturen. Die Entwicklung dieser Systeme sind vergleichbar mit der des XBOX Clusters und dienen so als Referenzbeispiele.

2.1 AppleTV Cluster

Im April 2011 haben die drei Professoren der Ludwig-Maximilians-Universität (LMU) Karl Führlinger, Christof Klausecker und Dieter Kranzmüller einen Cluster mithilfe von AppleTVs errichtet. Sie berichten in ihrer Arbeit[25] über die bedeutenden Fortschritte von High Performance Clusters (HPC). Sie schreiben „supercomputers have sustained a remarkable growth in performance that even out-performed the predictions of Moore's law“ und gehen im weiteren Verlauf auf die Notwendigkeit energieeffizienter Supercomputer ein. Da gerade im mobilen Bereich Energiemanagement eine bedeutende Rolle spielt verweisen sie darauf, dass früher Server den Grundstein für Supercomputer gelegt haben und diese Rolle in der Zukunft möglicherweise, gewöhnliche Elektronikprodukte des täglichen Gebrauchs einnehmen werden, welche die notwendigen Technologien für HPC bilden. Sie untersuchen in ihrer Arbeit mehrere aktuelle Geräte auf ihre Leistungsfähigkeit und gehen mit der Entwicklung eines HPC mithilfe von AppleTVs der 2. Generation sogar einen Schritt weiter und beweisen, dass solche Geräte durchaus ihre Berechtigung in der Verwendung von Supercomputer finden.

2.2 Sony PS3 Cluster

Dr. Frank Mueller, Professor an der North Carolina State University, veröffentlichte im Jahr 2007 eine technische Realisierung[31] eines funktionsfähigen HPC, basierend auf der Playstation3 von Sony. Ähnlich des in dieser Arbeit beschriebenen XBOX Cluster wird für die Umsetzung auf ein, auf Linux basierendes Betriebssystem, Fedora Core 5, gesetzt. Ebenfalls wurde von ihm eine Version des Message Passing Interface (MPI) implementiert, welches das Durchführen von schnellen Berechnungen auf dem HPC ermöglicht. In einem, bereits 2007 veröffentlichten Artikel[50], auf der Homepage der Universität, berichtet der Professor von seinem Ziel, einen effizienten, billigen Cluster für seine Universität zu errichten.

2.3 Unmodified XBOX Cluster

Bereits die erste Version der XBOX wurde von den Entwicklern B.J. Guillot, B. Chapman und J.-F. Pâris dazu verwendet einen Cluster zu errichten. Ziel der Arbeit war es mithilfe des XBOX Clusters Wiederherstellungsmethoden für große Computer Cluster analysieren und verbessern zu können. Der Cluster, bestehend aus insgesamt 4 Knoten, wurde durch die Verwendung eines Linux Betriebssystem realisiert. Das Starten eines Linux Betriebssystems wurde durch das Ausnützen eines Softwarefehlers notwendig und benötigte keine zusätzliche Hardware.

Kapitel 3

Hardware

Im folgenden Kapitel wird die, für die Realisierung benötigte, Hardware erläutert. Es sollen die einzelnen Komponenten, sowie deren Vorteile beschrieben werden.

3.1 Cluster

Ein Cluster, im Deutschen Rechnerverbund genannt, ist ein Zusammenschluss von mehreren Hardwarekomponenten, beispielsweise gewöhnlichen PCs. Zwischen den einzelnen Recheneinheiten in einem Cluster, den sogenannten Knoten (eng. Nodes), wird eine Netzwerkverbindung eingerichtet, welche die Kommunikation untereinander sicherstellt. Das Kernstück eines jeden Rechnerverbundes bildet ein PC, oftmals „Master“ genannt. Dieser ist das Gehirn des Clusters, verwaltet somit die Kommunikation mit den einzelnen Knoten und fungiert als Steuereinheit des Verbundes.

3.1.1 Clusterarten

Je nach Cluster wird zwischen verschieden Typen unterschieden. Zur Verbesserung der Ausfallsicherheit von Systemen und Diensten und somit zur Förderung der Erreichbarkeit werden Hochverfügbarkeitscluster verwendet. Sie duplizieren kritische Dienste auf jeden Knoten in einem Verbund. Sollte durch einen Fehler ein Knoten ausfallen, kann der Dienst noch immer genutzt werden, da er auf einen anderen Knoten weiterläuft.
Für den Umgang von hohen Lasten auf einem System werden Load-Balancing Cluster eingesetzt. Deren Aufgabe ist es, Zugriffe auf ein einziges System auf die einzelnen Knoten aufzuteilen.
Der High Performance Computing Cluster (HPC-Cluster) wird dann eingesetzt, wenn es darum geht Berechnungen möglichst effizient zu erledigen, wodurch auch der XBOX-Cluster als HPC-Cluster zu sehen ist.
Eine weiteres Konzept, welches bei Clustern zum Einsatz kommt, ist das Shared-Memory Konzept. Dabei teilen sich sämtliche Knoten eines Clusters den physikalischen Arbeitsspeicher. Dies ermöglicht es große Mengen an Arbeitsspeicher für die Knoten verfügbar zu machen. Davon können sämtliche Clusterarten profitieren. Beispielsweise werden durch das Shared-Memory Konzept Berechnungen eines HPC-Clusters schneller

erledigt. Zudem ist es möglich große Datenbestände im Arbeitsspeicher zu halten. Dies ist zum Beispiel für sogenannte „In-Memory“ Datenbanken entscheidend, da diese vollständig in den Arbeitsspeicher geladen werden können.
Zudem wird zwischen homogenen und heterogenen Clustern unterschieden. Als heterogen wird ein Cluster bezeichnet, wenn sowohl Big-Endian und Little-Endian zum Einsatz kommen. Big- und Little-Endian sind zwei unterschiedliche Arten der Speicherorganisation. Von Big-Endian spricht man, wenn das höchstwertige Byte zuerst gespeichert wird. Umgekehrt wird bei Little-Endian das kleinstwertige Byte zuerst gespeichert.
Aufgrund der verwendeten Power-PC Prozessoren in der XBOX, welche die Big-Endian Speicherorganisation verwenden, und der Verwendung einer Little-Endian Architektur für den XBOX-Master ist der XBOX-Cluster ein heterogener Cluster.

3.2 Master-PC

Der Master-PC ist das Kernstück des Clusters und fungiert als Schnittstelle zum Anwender. Als Betriebssystem kann eine beliebige Linux Distribution zum Einsatz kommen. Auf ihm werden alle notwendigen Dateien abgelegt, welche jeder Knoten benötigt. Der Master-PC wird über eine Switch mit allen Knoten des Clusters verbunden.

3.2.1 Der XBOX-Master

Der Shuttle XS35GS V3[21] ist ein kleiner Computer, produziert von der Firma Shuttle Inc und dient dem XBOX-Cluster als Master-PC. Der XS35GS V3 zeichnet sich durch seine Kompaktheit und seiner Ausstattung aus. Mit einer Abmessung von 252(L) x 38.5(B) x 162(H) mm benötigt er nur sehr wenig Platz. Ausgestattet mit einem Intel Atom D2700 Dual Core Prozessor, 4 Gigabyte Arbeitsspeicher und einer dedizierten Grafikkarte, hergestellt von AMD, genügt er perfekt den Anforderungen und bietet ein sehr gutes Preis/Leistung Verhältnis.

Abbildung 3.1: Der XS35GS V3 von Shuttle Inc.

3.3 XBOX360

Den Grundstein des Clusters bildet die XBOX360[54]. Sie wurde von Microsoft in Europa Ende 2005 veröffentlicht und war der Nachfolger der XBOX. Die neueste Version wurde in diesem Jahr in Umlauf gebracht und wird als XBOX One bezeichnet. Im Laufe der Jahre wurden 3 verschiedene Versionen der XBOX 360 veröffentlicht. Die XBOX 360 Core, welche als erste auf den Markt kam. Um 5 Jahre nach der Vorstellung der ersten XBOX noch konkurrenzfähig zu bleiben, erschien 2010 die XBOX 360 Slim[34], welche im Cluster Verwendung findet. Zahlreiche technische Komponenten wurden aufgerüstet und das Design zeitgemäßer gestaltet. Als 3. Version erschien die XBOX 360 E im Jahr 2013. Sie gleicht optisch bereits sehr der XBOX One und zeichnet sich im Vergleich zur XBOX 360 Slim durch der erneut verbesserte Hardware aus. Mit theoretischen 77 GFLOPS[53] des Prozessors und 240 GFLOPS der GPU[52] ermöglicht es die Hardware komplexe Berechnungen in kurzer Zeit zu erreichen. Der Autor Andrew „bunnie“ Huang hat in seinem Buch „Hacking the XBOX, An Introduction to Reverse Engineering“[8] jede Komponente der XBOX untersucht und mit dem Aufbau des PCs und anderer Konsolen verglichen.

3.3.1 Verwendungszweck von Spielekonsolen

Nicht nur Microsoft hat eine Reihe von Spielekonsolen auf den Markt gebracht. Sony kann mit der PlayStation und Nitendo mit der Wii ein vergleichbares Produkt anbieten. Ziel dieser Spielekonsolen ist es, den Käufern eine möglichst gute Spielerfahrung bieten zu können. Zahlreiche Unternehmen entwickeln hierzu immer neuere und bessere Spiele für die jeweiligen Geräte.
Um sicherzustellen, dass ausschließlich von den Herstellern genehmigte Spiele auf deren Konsolen laufen können und auch nur ihre Steuerungssoftware zum Einsatz kommt, wird der Quellcode jeder Software signiert[5]. Hierbei handelt es sich um ein Verfahren, welches sicherstellt, dass dieser Code geprüft und berechtigt ist, auf der Konsole ausgeführt zu werden. Dies dient zudem als Schutz vor schädlicher Software, welche ohne Signatur zur Ausführung kommen könnte. Hier unterscheiden sich Spielekonsolen entscheidend von gewöhnlichen PCs. Auf einem normalen Windows oder Linux System kann jederzeit selbst oder fremd programmierter Code zur Ausführung kommen, sollte dies vom User gewünscht sein.
Aus dieser Tatsache ergibt sich das Problem, dass die XBOX nur signierten Code ausführt. Umgehen lässt sich dies durch das Hacken der XBOX. Dies wird im Kapitel „Der Reset Glitch Hack“ beschrieben.

3.3.2 NAND-Flashspeicher

Der NAND-Flashspeicher[38] ist ein entscheidender Bestandteil der XBOX 360 Slim. Er zeichnet sich durch einen niedrigen Preis je Megabyte, hohen Schreib- und Lesegeschwindigkeiten bei großen Datenmengen, sowie durch die einfache und somit billige Verwendung des Speichers. Allerdings ist für die korrekte Ansteuerung des NAND-Flashspeichers ein hoher Softwareaufwand erforderlich. Zudem ist die Verwendung des Speichers auf etwa 1 Million Schreib-Lösch Zyklen limitiert.

Abbildung 3.2: Die XBOX360

Im NAND-Flash wird der Bootloader der XBOX 360 gespeichert. Deswegen wird dieser Speicher ausgelesen und verändert, um den Xell[43] Bootloader starten zu können. Dieser ermöglicht später das Ausführen von unsigniertem Code, wie beispielsweise Emulatoren, Programme und Betriebssysteme.

3.4 Glitcher

Als „Glitcher“ wird im Allgemeinen ein Chip bezeichnet, welcher nach erfolgreicher Anbringung am Mainboard, kleine Impulse an den Prozessor sendet und ihm dadurch glaubhaft macht, dass der Code im NAND-Flash korrekt gehasht[27] und signiert ist. Dies ermöglicht in weiterer Folge die Ausführung des XeLL Bootloaders, welcher weitere Schritte durchführt. Entdecker der Schwachstelle ist der französische Entwickler GliGli, welcher bei der Veröffentlichungen des Hacks eine genauere Erklärung präsentierte.[17] Die Verwendung des „Reset Glitch Hacks“ fand Bevorzugung, da andere verfügbare Hacks, entweder nicht dieselbe Funktionalität bereitstellen, oder nur auf einigen wenigen Versionen der XBOX 360 funktionieren.

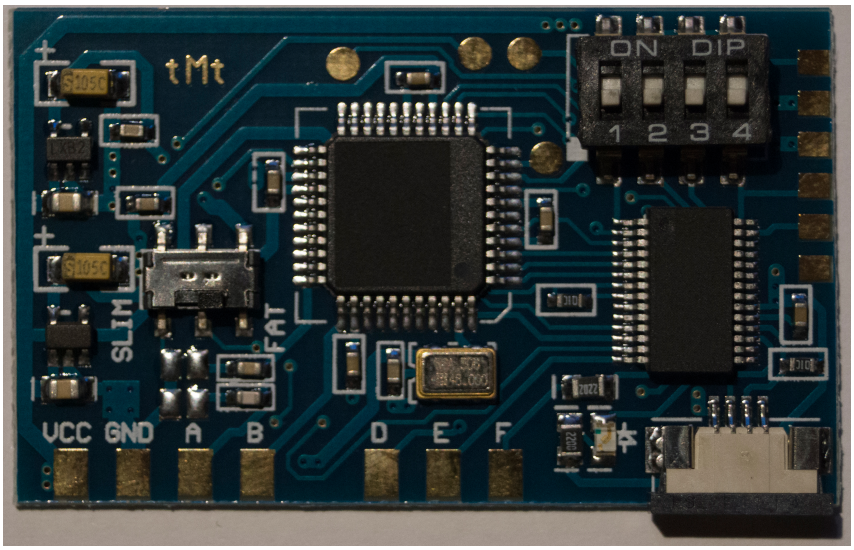


Abbildung 3.3: Der Glitcher

3.5 Der Reset Glitch Hack

Ermöglicht wird das Hacken der XBOX durch eine Schwachstelle im vorhanden HANA Chip. Dieser hat ein konfigurierbares Register welches es ermöglicht die CPU und GPU mit alternativen Taktraten zu versorgen. Das angesprochene Register wird über einen frei zugänglichen Bus angesteuert, welcher einen Lötpunkt am Mainboard hat. Folglich kann genau an dieser Stelle angesetzt werden, um in das System der XBOX zu gelangen. Der exakte Vorgang des Glitchen wird im Forum „Gulli.com“[32] beschrieben. Hier eine vereinfachte Version:

- Zuerst wird ein Befehl an den Hauptprozessor gesendet, welcher die Taktrate der CPU verringert.
- Anschließend wird ein Zähler gestartet, welcher nachdem er einen bestimmten Wert erreicht hat, einen 20ns Impuls an den Prozessor schickt und welcher einer Aufforderung gleicht, sich zurückzusetzen. Dieser Impuls ist allerdings zu schwach und es kommt nicht zur vollständigen Rücksetzung des Prozessors.
- Nach etwas Wartezeit wird erneut ein Befehl an den Prozessor gesendet und die CPU somit wieder auf ihre Ausgangstaktrate gesetzt.
- Mit etwas Glück wird der Bootvorgang nun fortgesetzt und es kommt erstmals zur Ausführung von nicht signierten Code.

So ist es nun möglich weiteren, unsignierten Code auszuführen.
Um den Reset Glitch Hack erfolgreich durchzuführen bedarf es guter Hardware, sowie entsprechender Softwarekenntnisse. Weiters ist darauf hinzuweisen, dass es durch die Veränderungen jederzeit zu schweren oder irreparablen Fehlern kommen kann!
Zuallererst wird der bereits im vorgehenden Kapitel erwähnte Chip am Mainboard angebracht. Dies erfordert gute Lötkenntnisse und am besten spezielles Lötwerkzeug.

Nach der erfolgreichen Unterbringung des Chips am Mainboard wird dieser beschrieben. Dazu verbindet man sich mittels entsprechender Kabel und deren Anlötung auf dem Mainboard mit dem Speicher. Durch die anschließende Verwendung eines Tools, wie „360gcprog“ , wird der Chip mit einer neuen Firmware bespielt. Das Programm quittiert dessen erfolgreiche Installation mit „The Board has been successfully flashed and verfied“. Abschließend muss der Chip noch fertig mit der XBOX verbunden werden.

Im nächsten Schritt wird der NAND-Flashspeicher ausgelesen. Dies kann auf mehrere verschiedene Arten passieren, beispielsweise durch die Verwendung eines „USB NAND Spi Flasher“[16]. Für das Auslesen des Speichers kann das Tool nandpro von Tiros verwendet werden. Dieses Konsolenprogramm ermöglicht die relativ einfache Sicherung der im Speicher vorhandenen Daten. Dieses Backup kann bei Bedarf wieder auf den Speicher zurückgespielt werden. Um nun den Xell Reloaded Bootloader installieren zu können wird erneut nandpro verwendet. Nandpro wird dazu verwendet ein Image des Speichers zu erzeugen, welches den Xell reloaded Bootloader enthält. Diese Image wird abschließend im NAND gespeichert. Zuguterletzt kann die XBOX gestartet werden und, sollte der gesamte Vorgang erfolgreich funktioniert haben, wird nach einiger Zeit der neue Bootloader auf dem Bildschirm erscheinen.

Die detaillierte Anleitung wurde in deutscher Sprache[32] sowie auch in Englisch[37] bereits umfassend beschrieben und dokumentiert.

Kapitel 4

Low-Level Software

Im Kapitel Low-Level Software wird auf Protokolle und Software eingegangen, welche für die Realisierung des Clusters notwendig sind.

4.1 Trivial File Transfer Protocol

Das Trivial File Transfer Protocol, auch bezeichnet als TFTP, dient zur einfachen Übertragung von Daten über das Netzwerk. Es verzichtet, im Gegensatz zum File Transfer Protocol auf viele Funktionen, wie etwa der Rechtevergabe mittels chmod. In der offiziellen Dokumentation[42] aus dem Jahre 1992 kann darüber genauer nachgelesen werden.
Das TFTP wird eingesetzt, um notwendige Betriebssystemdateien auf die XBOX übertragen zu können. Dafür wird ein TFTP-Server installiert und konfiguriert.

4.2 Kommunikation

Die Kommunikation der XBOXen untereinander und mit dem XBOX-Master erfolgt durch die Einrichtungen eines Domain Name System(DNS) und durch die Verwendung des Dynamic Host Configuration Protocol(DHCP). Für den Austausch von Dateien im Netzwerk wird ein Network File System(NFS) am XBOX-Master verwendet, welches die notwendigen Dateien zur Verfügung stellt. Diese können, dank des Trivial File Transfer Protocol (TFTP), einfach auf eine beliebige XBOX übertragen werden. Für die Verwendung dieser Protokolle und Systeme wird ein einfach zu konfigurierender DNS und DHCP Server verwendet, welcher zusätzlich die Integration eines TFTP Server ermöglicht.

4.3 Konfiguration der Komponenten

Wie bereits zuvor erwähnt, wird für die Realisierung der notwendigen Bestandteile zu großen Teilen der Dienst dnsmasq verwendet, welcher für Linux und auch Windows zur Verfügung steht. Aufgrund der Verwendung von Ubuntu finden sämtliche Konfigurationen in der dnsmasq-dpkg Datei statt. Unter anderen Umgebungen geschieht

dies in der Regel unter „/etc/config“”. Hierzu wird für jede im Netzwerk existierende XBOX ein Eintrag angegeben welcher nun genauer beschrieben wird.

Listing 4.1: Auschnitt aus der dnsmasq-dpkg Konfigurationsdatei

```
1         enable-tftp
2         tftp-root=/srv/tftp
3
4         #XBOX01
5         dhcp-boot=net:xbox01,kboot_xbox01.conf,192.168.1.1,192.168.1.1
6         dhcp-host=00:25:AE:EC:C2:A6,xbox01,192.168.1.101,infinite,net:
             xbox01
```

Ausschnitt 4.1 zeigt einen Auschnitt der dnsmasq-dpkg Datei, wobei dieser exakt die notwendigen Zeilen für die Konfigurationen einer XBOX enthält. Die Identifizierung einer XBOX erfolgt mittels MAC-Adresse[26]. So bekommt jede XBOX, je nach ihrer MAC-Adresse, eine eigene IP-Adresse[9] zugeordnet, welche benötigt wird, um im Netzwerk mit der XBOX zu kommunizieren.
Der Ausschnitt der Konfigurationsdatei, welcher im Ausschnitt 4.1 zu sehen ist, führt in Zeile 4 einen Kommentar, welcher Aufschluss darüber gibt, für welche XBOX die folgende Konfiguration gültig ist. Der in Zeile 6 zu findende Eintrag „dhcp-host“ definiert für welche XBOX dieser Eintrag gültig ist. Im Beispiel ist dies die XBOX mit der MAC-Adresse „00:25:AE:EC:C2:A6“. Genau diese XBOX erhält nun, wie im 2. beziehungsweise 3.Eintrag zu sehen ist, den Hostnamen „xbox01“ und die IP Adresse „192.168.1.101“. Der 4.Eintrag „infinite“ ermöglicht die stattgefundene Zuteilung der IP-Adresse auf unbestimmte Zeit zu garantieren. Der 5. und damit letzte Eintrag in Zeile 6 registriert den gesamten Eintrag im Dienst dnsmasq. Dies geschieht mit dem Schlüsselwort „net“ und der Zuweisung „:xbox01“. Nun ist die Registrierung unter dem Namen „xbox01“ abgeschlossen. Zeile 5 beginnt mit der Zuordnung für welche XBOX diese Zeile gültig ist, hierfür ist der letzte Eintrag in Zeile 22 notwendig. Mit „dhcp-boot=net:xbox01“ wird klar definiert, dass dieser Eintrag für jene XBOX gültig ist, welcher mit Hilfe des Schlüsselwortes „net“ der Eintrag „xbox01“ zugordnet worden ist. Daraufhin folgt die Bekanntgabe des Boot-Files. Diese Datei wird während des Startens von der XBOX bezogen und enthält Information über die zu ladenden Dateien. Eine genauere Erklärung ist unter „Das Boot-File“ zu finden. In der Abbildung 4.1 wird die Datei mit dem Namen „kboot_xbox01.conf“ angegeben. Der vorletzte und der letzte Eintrag geben nun noch die Auskunft über den TFTP Server, von dem die Datei bezogen werden kann. Hierbei muss mindestens einmal die IP-Adresse des Servers angegeben werden, optional kann als zweites der Name des Servers angegeben werden. In der zu sehenden Konfiguration wird zweimal die IP-Adresse „192.168.1.1“ verwendet. Dies lässt erkennen, dass der TFTP Server auf dem XBOX-Master PC läuft.

4.4 Der TFTP Server

Für die Konfiguration des TFTP Servers wurden in der dnsmasq-dpkg Datei 2 Einträge hinzugefügt, welche im Ausschnitt 4.1 zu sehen sind. Der in Zeile 1 stehende Befehl

„enable-tftp“ aktiviert in der Software den TFTP-Server, wobei in weiterer Folge der in Zeile 2 stehende Befehl angibt, wo genau sich das Root-Verzeichnis des TFTP-Servers befindet. Das Root-Verzeichnis ist jenes Verzeichnis, von dem alle weiteren Ordner und Dateien über den TFTP-Server freigegeben werden. Das heißt, dass jede Datei, welche im Root-Verzeichnis oder in einem Unterordner davon gespeichert ist, über TFTP bezogen werden kann. In den nun angegeben Root-Verzeichnis „/srv/tftp“ finden sich alle erforderlichen Dateien für den Startvorgang einer XBOX. Das wohl wichtigste Verzeichnis ist „gdl“, welches das Betriebssystem beherbergt. Ebenso sind die Konfigurationsdateien jeder XBOX vorhanden.

4.5 Kboot

Kboot[2] ist ein, auf Kexec[4] basierender Bootloader. Kexec ist ein Programm welches normalerweise dazu verwendet wird von einem Kernel in einen anderen Kernel zu laden. Ziel des Kboot Bootloaders ist es, den ansonsten sehr simplen Bootvorgang eines Linuxsystems zu erweitern. Bei der Verwendung von Kboot wird zuerst, wie bei der Verwendung ohne Kboot, ein Bootloader geladen. Dieser lädt den Kernel sowie das Initramfs. Doch anstatt nun weiter fortzufahren, greift Kboot ein. Die sogenannte „Kboot-Shell“ aktualisiert Konfigurationsdateien und bietet dem User anschließend eine Möglichkeit der Interaktion mit Kboot. Entweder aktiv durch den User, oder passiv durch das verstreichen eines Timers, wird mithilfe des Tools Kexec der Kernel neu geladen.

4.6 Das Boot-File

Wie unter dem Kapitel „Konfiguration der Komponenten“ bereits erwähnt, ist das Boot-File unerlässlich um eine XBOX erfolgreich zu starten. Das jeweilige Boot-File wird über den TFTP-Server bezogen und vom XeLL-Reloaded[43] verarbeitet. In dieser Datei werden nun die Boot-Parameter definiert, welche die XBOX beim Starten verwendet. In der Tabelle 4.1 wird das Bootfile der XBOX01 genauer analysiert. Hierzu wurde der Eintrag in die entsprechenden Parameter unterteilt und in einer Tabelle übersichtlich angeordnet

4.7 XeLL-Reloaded

XeLL-Reloaded[43] ist ein Xenon Linux Bootloader basierend auf Xell, welcher von tmbinc(Felix Domke) entwickelt wurde. Die Software wird von der Free60 Community[13] entwickelt, welche sich zudem für die Entwicklung weiterer notwendiger Software für die Realisierung des XBOX-Cluster verantwortlich zeichnet.
Wie bereits erwähnt, wird dieser Bootloader dazu verwendet, ein alternatives Betriebssystem auf der XBOX zu starten. Beim Bootvorgang der XBOX wird der Bootloader, welcher bereits im NAND-Flash abgelegt ist, aufgerufen. Anschließend wird eine EL(Executable and Linkable) Datei von einem beliebigen Speicherort geladen.

gentoo-diskless=/gdl/zImage.xenon	Datei zum Starten des Bootvorgangs
initrd=/gdl/initramfs	Angabe des initramfs, welches für das Laden der wichtigsten Treiber zuständig ist
ramdisk=16383	Spezifizierung der ramdisk Größe
root=nfs4:192.168.1.1:/srv/tftp/gdl:udp	Angabe des Root-Verzeichnisses und wie diese Dateien geladen werden sollen. Dabei wird dieser Ordner mit nfs4 gemountet
init=/boot/stateless.sh	Angabe des zu startenden Init-Script. Dieses wird direkt nach dem Laden des Kernels sowie des INITRAMFS ausgeführt.
xenon_net.macaddr=00:25:AE:EC:C2:A6	Angabe der MAC-Adresse der XBOX. Dieser Parameter wurde nachträglich hinzugefügt und wird im Kapitel „HTL Perg Kernel“ nochmals erläutert

Tabelle 4.1: Auflistung der wichtigsten Bootparameter

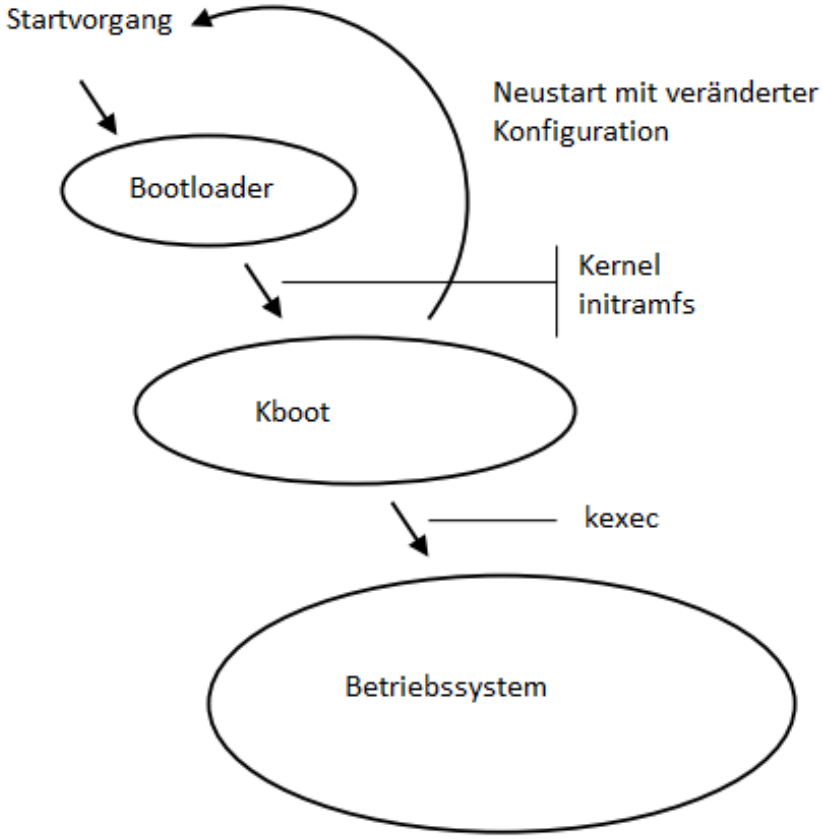


Abbildung 4.1: Boot Sequenz mit der Verwendung von Kboot

XeLL-Reloaded wurde entwickelt um Linux-Betriebssysteme zu starten, kann allerdings auch dazu verwendet werden, andere ELF Dateien zu laden, um beispielsweise LibXenon (siehe LibXenon und Toolchain) basierende Anwendungen zur Ausführung zu bringen.

In der aktuellen Version ist Xell-Reloaded in 2 „Stages“ unterteilt. Stage 1 initialisiert den Großteil der Hardware und ruft den 2. Teil des Bootloaders auf. Dieser basiert auf LibXenon, lädt alle benötigten Treiber und erledigt weitere notwendige Aufgaben.

4.8 HTL Perg XeLL

Der Bootloader wird kontinuierlich weiterentwickelt und verbessert. Jedoch kam es in der zuletzt veröffentlichten Version zu einem schwerwiegenden Fehler bei der Implementierung des Netzwerkboots. Dabei scheitert das Programm den Zugriff auf das Boot-File ordnungsgemäß durchzuführen.Um diesen Fehler zu beheben wurde in der Datei „file.c“ die Methode „tftp_loop()“ modifiziert.

```
1 // bevor Fehlerbehebung
2 wait_and_cleanup_line();
3 printf("Trying␣TFTP␣%s:%s...\r",boot_server_name(),boot_file_name
     ());
4 /* Assume that bootfile delivered via DHCP is an ELF */
5 boot_tftp(boot_server_name(),boot_file_name(),TYPE_ELF);
```

```
1 // nach Fehlerbehebung
2 wait_and_cleanup_line();
3 printf("Trying␣TFTP␣%s:%s...\r",boot_server_name(),boot_file_name
     ());
4 if(strstr(boot_file_name(), ".conf")!=NULL)
5  boot_tftp(boot_server_name(),boot_file_name(),TYPE_KBOOT);
6  // Assume, when file ends with .conf, that it is a KBOOT-File
7 else
8  boot_tftp(boot_server_name(),boot_file_name(),TYPE_ELF);
9  // Assume that bootfile delivered via DHCP is an ELF
```

Wie im Vergleich zu erkennen ist, wurde in der ursprünglichen Version kein Unterschied zwischen einem Executable Link File (ELF) und einer Konfigurationsdatei getroffen. Durch den angefügten Sourcecode in der verbesserten Version wurde diese Überprüfung erneut eingeführt und ein erfolgreiches Laden der KBOOT-Datei garantiert.

4.9 LibXenon und Toolchain

Für die Entwicklung von XBOX kompatiblen Programmen muss eine Toolchain[51] eingerichtet werden. Dazu wurde von der Free60 Community eine genaue Anleitung für das Erzeugen und Installieren der Xenon Toolchain zur Verfügung gestellt. Als wesentliche Bestandteile gelten der GCC[12], ein Compiler für viele Programmiersprachen wie C und C++. Dem Xell-Reloaded Bootloader liegt die Bibliothek LibXenon[14][44] zugrunde. Sie wird von der Free60 Community entwickelt und betreut. Zudem ist sie eine notwendige Bibliothek für die Xenon Toolchain.

4.9.1 Installieren der Toolchain

Die Xenon-Toolchain ist notwendig, da es notwendigen Code in PowerPc-Binaries übersetzt. Diese PowerPc-Binaries sind notwendig, da diese von der Xenon-Hardware, also der XBOX ausgeführt werden können. Sie kann gemeinsam mit LibXenon vom Git-Repository bezogen werden. Dieses steht unter „git://free60.git.sourceforge.net/gitroot/free60/free60“ für das Klonen zur Verfügung. Um die Xenon-Toolchain zu übersetzen und zu installieren wird in das Verzeichnis der Toolchain gewechselt wie in Zeile 1 des Ausschnitts 4.2 zu sehen ist. Mit dem Befehl in Zeile 2 wird die Xenon-Toolchain übersetzt und installiert. Zum Abschluss wird man dazu aufgefordert 2 weitere Zeilen zu der Datei ./bashrc hinzuzufügen. Diese 2 Zeilen sind im Ausschnitt 4.3 zu sehen.

Listing 4.2: Übersetzen der Xenon-Toolchain

```
1 cd free60/toolchain
2 ./build-xenon-toolchain toolchain
```

Listing 4.3: Notwendige Zeilen in der Datei ./bashrc

```
1 export DEVKITXENON="/usr/local/xenon"
2 export PATH="$PATH:$DEVKITXENON/bin:$$DEVKITXENON/usr/bin"
```

Um die Installation zu testen, kann der Befehl „xenon-gcc“ ausgeführt werden. Sollte dieser die Meldung „xenon-gcc: no input files“ zurück geben, ist die Installtion der Xenon-Toolchain erfolgreich abgeschlossen.
Die Installation von LibXenon beginnt erneut im Verzeichnis der Xenon-Toolchain. Im Ausschnitt 4.4 in Zeile 1 findet zuerst die Installation von LibXenon statt. Durch den Befehl in Zeile 2 kann die Installation getestet werden.

Listing 4.4: Installation von LibXenon

```
1 ./build-xenon-toolchain libxenon
2 ./build-xenon-toolchain cube
```

Für eine genauere Erklärung der Installation der Xenon-Toolchain und von LibXenon gibt es online weitere Dokumentation[47].

4.9.2 Secure Shell

Secure Shell (SSH) ist ein Protokoll, welches es ermöglicht eine verschlüsselte Netzwerkverbindung aufzubauen. SSH existiert seit 1995 und wurde 1996 in der Version 2 veröffentlicht. Da diese allerdings vom Entwickler als kommerzielle Software veröffentlicht wurde, entstand schnelle eine neue Version mit dem Namen „OpenSSH“. Dies ist eine Weiterentwicklung von ssh1, deren Quellcode öffentlich ist. Die wesentliche Eigenschaften sind in der Dokumentation[6] der Webseite „ubuntuuseres.de“ wie folgt nachzulesen:

- Authentifizierung der Gegenstelle, kein Ansprechen falscher Ziele
- Verschlüsselung der Datenübertragung, kein Mithören durch Unbefugte
- Datenintegrität, keine Manipulation der übertragenen Daten

Diese Eigenschaften gewährleisten eine abhörsichere Kommunikation zwischen zwei Endgeräten.

Passwortfreie Authentifizierung

Um es Programmen und Diensten zu ermöglichen Befehle über SSH, ohne die Eingabe eines Passworts durch den Benutzer, an andere Endgeräte zu schicken kann die passwortfreie Authentifizierung (eng. password less login) verwendet werden. Hierzu werden zuerst öffentliche Schlüssel generiert wie im Ausschnitt 4.5 in Zeile 1 zu sehen ist. Nach der Eingabe des Befehls wird der Benutzer aufgefordert, einen Speicherort für die Schlüssel sowie ein Passwort anzugeben. Wichtig hierbei ist, das Passwort leer zu lassen. Anschließend ist es notwendig die zwei erzeugten Dateien auf all jene Hosts zu übertragen, welche ohne die Eingabe eines Passworts erreichbar sein sollen. Hierzu kann der Befehl in Zeile 3 verwendet werden, wobei der erste Parameter der Speicherort jener „.pub“ Datei sein muss, welche zuvor erzeugt wurde. Dieser Befehl überträgt nun die Schlüssel zum angegeben Host und ermöglicht den zukünftigen Verbindungsaufbau ohne die Eingabe eines Passworts.

Listing 4.5: Einrichten der passwortfreien Authentifizierung

```
1 ssh-keygen -t rsa
2
3 ssh-copy-id -i ~/speicherort/schlüssel.pub benutzername@hostname
```

4.10 Programmierung von parallelen Algorithmen

Bei der Programmierung von parallelen Algorithmen wird das Prinzip des „Teilen und Herrschen“ angewendet, da ein zentraler Prozess eine Aufgabe zerteilt und zur weiteren Berechnung an einen anderen Prozess abgibt. Die Resultate der einzelnen Berechnungen werden an den zentralen Prozess, den herrschenden, zurückgeliefert, welcher diese zusammenfügt und ein Ergebnis erhält. Auch bei High Perfomance Clustern kommt dieses Prinzip zum Einsatz. Dabei wird der herrschende Prozess zumeist am „Master“ ausgeführt. Jene Prozesse, welche Berechnungen durchführen, werden auf die Knoten des Clusters, den „Slaves“, aufgeteilt. Um dieses Prinzip anwenden zu können muss es möglich sein, Daten zwischen Prozessen, welche auf unterschiedlichen Knoten ausgeführt werden, auszutauschen. Hierzu wird das sogenannte „Message passing“ oder „Nachrichtenaustausch“ verwendet.

4.10.1 Nachrichtenaustausch

Um es parallelen Algorithmen zu ermöglichen Daten auszutauschen kommt das „Message passing“ zum Einsatz. Dabei sendet ein Prozess Daten zu einem anderen Prozess, wobei es die Aufgabe des empfangenden Prozesses ist, die Daten entgegenzunehmen und entsprechend zu verarbeiten. Ob dies tatsächlich der Fall ist, kann vom sendenden Prozess nicht überprüft werden. Es unterscheidet sich grundlegend von Methoden und Funktionen da diese über eine Namen oder anonym aufgerufen werden. Beim „Message passing“ kann, ohne zu wissen, ob ein Prozess die Daten empfangen wird, jederzeit das Senden von Daten erfolgen. Ebenso ist es möglich, dass

ein Prozess auf die Übertragung von Daten wartet, ohne zu wissen, woher und ob diese Daten jemals gesendet werden. In der Praxis übernimmt diese Funktionalitäten eine Sofware, wie Open Message Passing Interface.

4.10.2 Installation von Open Message Passing Interface

Um Open Message Passing Interface (Open MPI) für ein Linux System zu kompilieren, ist Java sowie ein C Compiler, beispielsweise die GNU Compiler Collection (GCC), notwendig. Auf der offiziellen Seite von Open MPI[36] finden sich verschiedene Versionen der Software, wobei zu empfehlen ist, jene Version zu verwenden, welche unter „Current stable release“ angeführt ist. Dieses Archiv muss zuerst entpackt werden. Anschließend kann diese Version übersetzt werden. Ausschnitt 4.6 zeigt die verwendeten Befehle.

Listing 4.6: Übersetzen von Open MPI

```
1 gunzip -c openmpi-1.8.4.tar.gz | tar xf -
2 cd openmpi-1.8.4
3 ./configure --enable-java --prefix=/usr/local --enable-heterogeneous
4 make all install
```

Um Open MPI fähige Programme zu entwickeln, ist es notwendig, die Java Bibliothek „mpi.jar“, welche sich im Verzeichnis „/usr/local/lib“ befindet, in das Projekt einzubinden. Diese Bibliothek wird durch den Parameter „–enable-java“, welcher im Ausschnitt 4.6 in Zeile 3 zu sehen ist, erzeugt. Da, wie im Kapitel „Cluster“ erklärt, der XBOX-Cluster heterogen ist, ist es zudem notwendig, den Parameter „–enable-heterogeneous“ anzugeben, um eine entsprechende Kompatibilität zu erreichen.

4.10.3 Ausführung auf mehreren Knoten

Das Ausführen eines Programms, welches Open MPI verwendet, erfordert den Einsatz von SSH, welches im Kapitel Secure Shell beschrieben wird. Zu beachten ist, dass jeder Knoten ohne Eingabe einer Passphrase erreichbar sein muss. Wie dies erreicht werden kann wird im Kapitel Secure Shell, unter Passwort-freie Authentifizierung, erläutert. Open MPI ermöglicht es, über eine Datei festzulegen, welche Knoten an einer Berechnung beteiligt sein sollen. Die Datei, welche am XBOX-Master eingesetzt und auch im Kapitel Performancevergleich benötigt wird, ist in Abbildung 4.2 zu sehen. Dabei kann mittels des Parameters „slots“ angegeben werden, wie viele Prozesse auf einem Knoten ausgeführt werden sollen. Anstelle der Hostnamen kann auch die IP-Adresse angegeben werden. Das Ausführen eines Open MPI fähigen Java-Programms gelingt mit dem im Ausschnitt 4.7 gezeigten Befehl.

4.10.4 Implementierung eines Beispiels

Ein sehr einfaches Beispiel, welches für einen Cluster entwickelt werden kann, ist die Multiplikation zweier Zahlen. Das folgende Beispiel wurde in Java implementiert, allerdings existieren die verwendeten Open MPI Methoden auch in anderen Programmiersprachen. Für das Entwickeln eines entsprechendes Algorithmus sind die

```
mpi@xbox-master:~$ cat mpihosts
#xbox-master slots=1
xbox02 slots=1
xbox03 slots=1
xbox04 slots=1
xbox05 slots=1
#xbox06 slots=1
#xbox07 slots=1
#xbox08 slots=1
#xbox10 slots=1
#xbox11 slots=1
#xbox12 slots=1
#xbox13 slots=1
#xbox14 slots=1
#xbox15 slots=1
#xbox16 slots=1
#xbox18 slots=1
#xbox19 slots=1
#xbox20 slots=1
```

Abbildung 4.2: Verwendetes Hostfile am XBOX-Master

Listing 4.7: Ausführen eines Open MPI fähigen Programms

```
1 mpirun --hostfile <Pfad zum Hostfile> [-n <Anzahl der Prozesse auf jedem
      Knoten>] java <Klassenname>
```

zwei Methoden Scatter und Gather notwendig. Scatter ist für die Verteilung der Aufgaben auf die Knoten zuständig. Gather hingegen empfängt die Ergebnisse der Knoten.
Als ersten Schritt der Entwicklung ist es notwendig, angegebene Argumente von der Kommandozeile zu lesen, sowie zu bestimmen, welche Prozessnummer der jeweilige Knoten besitzt und wie viele Prozesse beteiligt sind. Diese Schritte werden im Ausschnitt 4.8 nochmals gezeigt.

Listing 4.8: Initialisierung von MPI und Bestimmung wichtiger Werte

```
1 // MPI mit den Kommandozeilenargumenten initialisieren
2 MPI.Init(args);
3
4 // Der wievielte Prozess ist der Knoten?
5 int prozessnummer = MPI.COMM_WORLD.getRank();
6
7 // Wieviele Prozesse gibt es insgesamt?
8 int prozessanzahl = MPI.COMM_WORLD.getSize();
```

Im zweiten Schritt ist es wichtig, sich für eine Prozessnummer zu entscheiden. Empfehlenswert ist der Prozess mit der Nummer 0, welcher immer existiert. Dieser Prozess dient ab sofort als zentraler, herrschender Prozess und ist für die Verteilung der Berechnungen verantwortlich. Auf dem Knoten, auf dem der Prozess mit der Prozessnummer 0 läuft, wird nun ein Buffer initialisiert, welcher für jeden Prozess 2 Zufallszahlen speichert. Dieser Vorgang ist im Ausschnitt 4.9 gezeigt.Die

Implementierung der Multiplikation ist ab Zeile 12 zu sehen.

Listing 4.9: Generierung der Zufallszahlen im Prozess mit der Nummer 0

```
9  // Ein Prozess besitzt einen IntBuffer, bei allen anderen ist er null
10 IntBuffer senderbuffer = null;
11 // Der Prozess mit der Nummer 0 erzeugt jetzt zufällige Zahlen für alle
       anderen Prozesse
12 if(prozessnummer == 0) {
13         Random r = new Random();
14         senderbuffer = MPI.newIntBuffer(prozessanzahl * 2);
15         for(int i = 0; i < prozessanzahl; i++) {
16                 int n1 = r.nextInt(1000);
17                 int n2 = r.nextInt(1000);
18                 senderbuffer.put(n1); senderbuffer.put(n2);
19                 System.out.printf("Task for Rank %d: What is %d*%d?\n",
                       i, n1, n2);
20         }
21 }
```

Anschließend legt jeder Prozess einen Buffer mit der Länge 2 an, da jeder Prozess in weiterer Folge zwei Zahlen für die Multiplikation erhält, dies ist im Ausschnitt 4.10 zu sehen.
Nun wird mittels des Befehls Scatter die Verteilung der Aufgaben an alle Prozesse begonnen. Die Parameter der Methode werden in der Tabelle 4.2 erklärt. Zu beachten ist, dass die drei ersten Einträge nur für den Sender, also dem Prozess mit der Prozessnummer 0, von Bedeutung sind. Die zweiten drei Einträge hingegen sind nur für die Empfänger, also alle übrigen Prozesse, relevant.

Listing 4.10: Verteilung der Aufgaben auf alle Prozesse

```
22 //Anlegen des Empfangsbuffers für zwei Zahlen
23 IntBuffer empfangsbuffer = MPI.newIntBuffer(2);
24
25 //Aufruf des Verteilungsbefehls
26 MPI.COMM_WORLD.scatter(senderbuffer, 2, MPI.INT, empfangsbuffer, 2, MPI.
       INT, 0);
```

Nachdem dieser Befehl von jedem Prozess abgearbeitet wurde, enthält jeder Empfangsbuffer zwei der zuvor erzeugten Zufallszahlen. Im nächsten Schritt wird das Ergebnis der Multiplikation berechnet und in einen Ergebnisbuffer geschrieben, wie im Codeausschnitt 4.11 zu sehen ist. Der Ergebnisbuffer wird in Zeile 27 mit einer Länge von 1 erzeugt. In Zeile 28 findet die Multiplikation der im Empfangsbuffer stehenden Zahlen statt.
Um die Ergebnisse nun verwalten zu können, wird erneut ein Buffer angelegt. Wie auch bei der Initialisierung des Senderbuffers wird auch hier der Buffer nur für den Prozess

Buffer senderbuffer	Buffer, welcher alle Werte des Senders beinhaltet. Diese Werte werden auf alle Prozesse aufgeteilt
int sendzaehler	Angabe, wie viele Elemente des Sender-Buffers an einen Prozess gesendet werden
MPI.Datatype datentyp	Angabe, welcher Datentyp wird an die Prozesse gesendet
Buffer empfangsbuffer	Bestimmt, in welchen Buffer die empfangen Werte gespeichert werden
int empfangszaehler	Angabe, wie viele Elemente in einem Empfangsbuffer gespeichert werden
MPI.Datatype datentyp	Angabe, welcher Datentyp vom Sender erwartet wird
int prozessnummer	Festlegen, welcher Prozess der Sender ist

Tabelle 4.2: Beschreibung der Parameter des Scatter-Befehls

Listing 4.11: Berechnung der Multiplikation

```
27 IntBuffer ergebnisbuffer = MPI.newIntBuffer(1);
28 ergebnisbuffer.put(empfangsbuffer.get(0) * empfangsbuffer.get(1));
```

mit der Nummer 0 angelegt. Wie im Codeausschnitt 4.12 in Zeile 31 zu sehen entspricht die Länge des Ergebnisempfangsbuffers der Anzahl der laufenden Prozesse, da von jedem Prozess ein Ergebnis erwartet wird. Das Empfangen der Ergebnisse wird vom Gather Befehl verwaltet. Dessen Parameter werden in der Tabelle 4.3 erklärt. Der Gather Aufruf, welcher in der Beispielimplementierung verwendet wird, ist im Teilausschnitt 4.12 in Zeile 33 zu sehen.

Buffer ergebnisbuffer	Buffer, welcher das Ergebnis beinhaltet. Diese Werte werden an den Sender übertragen
int laenge	Längeangabe des Ergebnisbuffers
MPI.Datatype datentyp	Angabe, welcher Datentyp übertragen wird
Buffer ergebnisempfangsbuffer	Buffer, in dem alle Einzelergebnisse gespeichert werden
int ergebenislaenge	Angabe, aus wie vielen Elementen ein Ergebnis besteht
MPI.Datatype datentyp	Angabe, welcher Datentyp vom Ergebnis erwartet wird
int prozessnummer	Festlegen, welcher Prozess die Ergebnisse sammelt

Tabelle 4.3: Beschreibung der Parameter des Gather-Befehls

Listing 4.12: Empfangen der Ergebnisse

```
29 IntBuffer ergebnisempfangsbuffer = null;
30 if(prozessnummer == 0)
31         ergebnisempfangsbuffer = MPI.newIntBuffer(prozessanzahl);
32
33 MPI.COMM_WORLD.gather(ergebnisbuffer, 1, MPI.INT, ergebnisempfangsbuffer
       , 1, MPI.INT, 0);
```

Listing 4.13: Ausgabe der Ergebnisse

```
34 if(rank == 0) {
35         for(int i = 0; i < size; i++) {
36                 System.out.printf("Result from rank %d: %d\n", i,
                       ergebnisempfangsbuffer.get(i));
37         }
38 }
39
40 // Programm beenden
41 MPI.Finalize();
```

Die erhalten Ergebnisse werden nun noch vom Prozess mit der Nummer 0, also dem Senderprozess auf der Konsole, ausgegeben. Dabei wird der Ergebnisempfangsbuffer durchlaufen, wie im Ausschnitt 4.13 in Zeile 36 zu sehen. Das Beenden des Programms erfolgt in Zeile 41 des Programmausschnitts.

4.10.5 Performancevergleich

Für den Performancevergleich wurde ein Algorithmus für die Berechnung von Primzahlen entwickelt. Dieser errechnet jede einzelne Primzahl von 0 bis zu einem beliebigen Wert. Abbildung 4.3 zeigt die Berechnung der Primzahlen, bis 32.000.000, auf 8 Knoten. Im Vergleich dazu wurde, wie in Abbildung 4.4 abgebildet, dieselbe Berechnung mit 16 Knoten durchgeführt. Eine fast 50-prozentige Reduzierung der Berechnungsdauer zeigt den Vorteil eines High Performance Clusters mit möglichst vielen Knoten. Ein genauer Vergleich ist in Tabelle 4.4 zu sehen. In Spalte 1 ist die Anzahl jener Zahlen zu sehen, die bis zu dem angegebenen Wert auf Primzahlen untersucht wurden. Spalte 2-6 zeigen die Anzahl der verwendeten Knoten sowie die verstrichene Zeit, welche notwendig war um die Zahlen zu untersuchen, in Sekunden an.

```
mpi@xbox-master:~$ mpirun --hostfile mpihosts primesCalc
Using 8 tasks to scan 32000000 numbers
Done. Largest prime is 31999939 Total primes 1973815
Wallclock time elapsed: 19.74 seconds
```

Abbildung 4.3: Berechnung der Primzahlen von 1 - 32.000.000 auf 8 Knoten mit je einem Prozess

```
mpi@xbox-master:~$ mpirun --hostfile mpihosts primesCalc
Using 16 tasks to scan 32000000 numbers
Done. Largest prime is 31999939 Total primes 1973815
Wallclock time elapsed: 9.91 seconds
```

Abbildung 4.4: Berechnung der Primzahlen von 1 - 32.000.000 auf 16 Knoten mit je einem Prozessen

	Anzahl der verwendeten Knoten				
Anzahl der untersuchten Zahlen	1	2	4	8	16
4*10^6	8.29	4.14	2.11	1.08	0.64
8*10^6	21.97	10.99	5.53	2.82	1.42
16*10^6	58.68	29.32	14.68	7.38	3.74
32*10^6	157.41	78.76	39.41	19.74	9.91

Tabelle 4.4: Performancevergleich bei der Berechnung von Primzahlen. Werte entsprechen der verstrichenen Zeit in Sekunden.

4.10.6 Das Mandelbrot Fraktal

Das Mandelbrot Fraktal ist in seiner Berechnung ein sehr aufwendiges Fraktal, weshalb es sich ideal als Demonstrationsbeispiel für einen High Performance Cluster eignet. Ein Fraktal ist „ein Objekt, das bei fortgesetzter Vergrößerung immer neue, selbstähnliche

Details zeigt“[48]. Basierend auf der rekursiven Formel $z_{n+1} = z_n^2 + z_0$ ist jeder Punkt z_0 Teil der Mandelbrotmenge für den die Folge $z_0, z_1, z_2, z_3, ...$ in einem Radius von 2 vom Ursprung bleibt. Divergiert die Folge hingegen, so liegt der Punkt nicht in der Mandelbrotmenge und, wie in Abbildung 4.5 zu sehen ist, werden all jene Punkte, welche nicht Teil der Mandelbrotmenge sind, in Farbe dargestellt. Dabei wird bestimmt, ab wann die Folge eines Punktes den Radius 2 übersteigt. Dieser Wert wird als Divergenzgeschwindigkeit bezeichnet und kann in einer Farbskala umgesetzt werden [11]. Es gibt zahllose Beschreibungen des Fraktals, welche von sehr einfachen anschaulichen Beispielen bis hin zu rein mathematischen Erklärungen gehen[24] [55] [7]. In Abbildung 4.5 ist eine visuelle Darstellung des Mandelbrot Fraktals zu sehen. Zu diesem Zeitpunkt sind kaum Berechnungen notwendig.

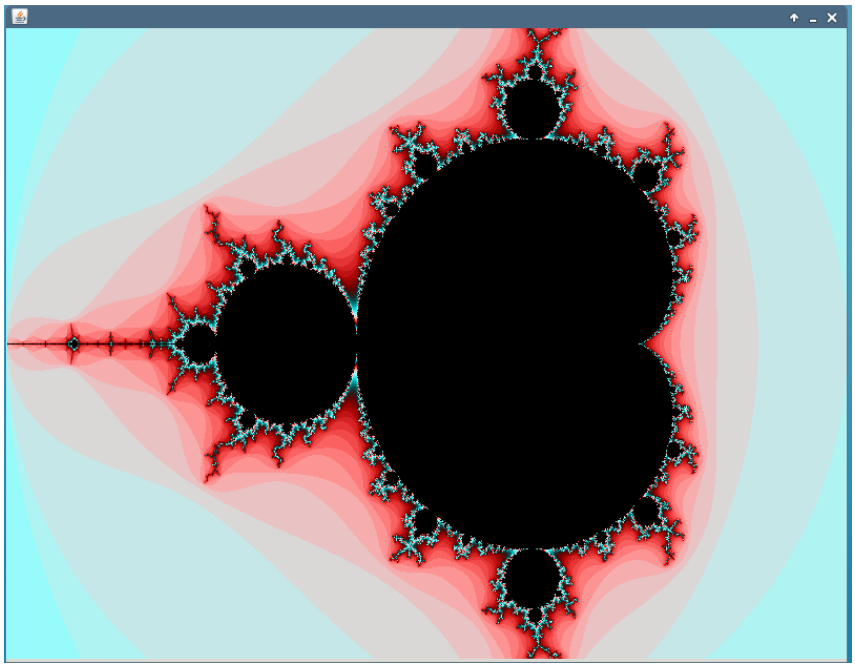


Abbildung 4.5: Das Mandelbrot Fraktal

Beginnt der Benutzer Ausschnitte zu vergrößern, steigen die erforderlichen Berechnungen und somit die CPU Auslastung. Dies ist in Abbildung 4.6 für die XBOX04 bis XBOX07 zu sehen. Ebenso steigt die CPU Auslastung der anderen Knoten, wie der aggregierte Graph in Abbildung 4.7 zeigt. Abbildung 4.8 zeigt die Visualisierung des Mandelbrot Fraktals, nachdem der User einen Ausschnitt vergrößert hat.

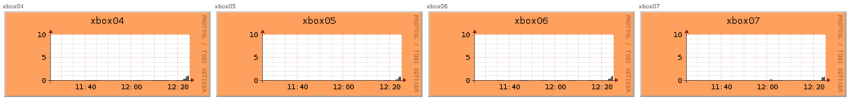


Abbildung 4.6: CPU Auslastung bei ersten Vergrößerungen

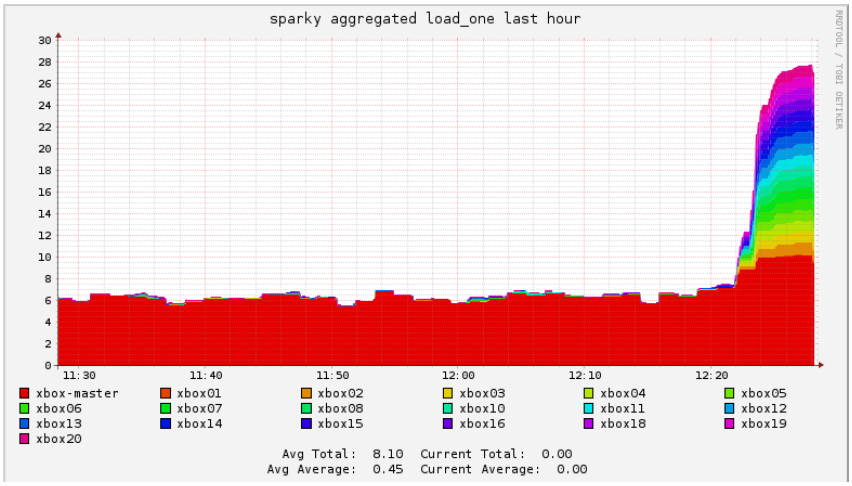


Abbildung 4.7: CPU Auslastung des gesamten Clusters

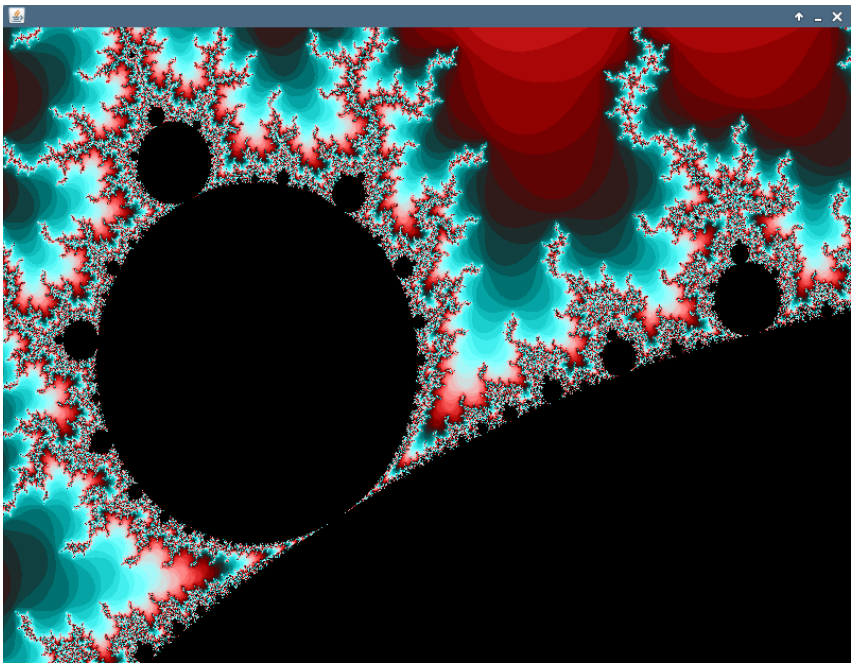


Abbildung 4.8: Vergrößerung durch den User

Kapitel 5

Betriebssysteme

5.1 Gentoo

Gentoo ist ein auf dem Linux Kernel[23] basierendes Betriebssystem. Daniel Robbins gründete Gentoo am 4.Oktober 1999. Bis zum Jahr 2004 wurde die Entwicklung auch von diesem profitorientiert geleitet. Mitte 2004 zog sich der Gründer aus dem Unternehmen zurück. Aus Gentoo Technologies Inc. wurde die Gentoo Foundation und damit einen nicht gewinnorientierte Organisation. Das Betriebssystem zeichnet sich durch seine Offenheit für Bastler und Entwickler aus, wodurch es auch letzten Endes für den Cluster verwendet wurde. Der gesamte Quellcode ist offen und frei verfügbar, zudem finden sich große Mengen an Dokumentationen und Erfahrungsberichte im Internet welche helfen, sich besser zurecht zu finden, denn für die Installation von Gentoo erfordert es Fachwissen im Bereich der Linux Welt, sodass es für den Standarduser als ungeeignet einzustufen ist.

5.1.1 Installieren von Gentoo

Die Installation von Gentoo erfordert eine Reihe von Tätigkeiten. Im ersten Schritt wird mit Hilfe einer Gentoo Installations CD, welche als CD-Abbildung auf der Gentoo Homepage[19] zu finden ist, das System hochgefahren. Die hier erklärten Schritte sind die wichtigsten und sind zwingend notwendig um eine erfolgreiche Installation des Gentoo Systems zu garantieren. Allerdings bietet das Handbuch[19] der Gentoo Community eine detailliertere Anleitung und ist für unerfahrene Anwender klar zu bevorzugen.
Für das erfolgreiche Starten des Systems muss bei der Eingabeaufforderung der im Ausschnitt 5.1 in Zeile 1 stehende Befehl eingegeben werden. Der in Zeile 2 zu sehende Befehl kann verwendet werden um notwendige Kernelmodule[20] zu laden. Zu diesem Zeitpunkt ist das Gentoo System vollständig hochgefahren und bereit für die weitere Konfiguration. Zunächst sollte sichergestellt werden, dass eine Internetverbindung vorhanden ist, dazu kann der in Zeile 3 zu sehende Befehl verwendet werden. Selbstverständlich benötigt ein Betriebssystem auch ein Dateisystem welches im nächsten Schritt konfiguriert wird. Hierzu kann das Systemtool „fdisk“ verwendet werden. Für die Formatierung der angelegten Partitionen und für die Aktivierung des SWAP-Speichers[49] werden die im Ausschnitt 5.2 gelisteten Kommandos benötigt: Die

Listing 5.1: Erste Befehle für das Aufsetzen von Gentoo

```
1 gentoo dopcmia
2 modprobe
3 ifconfig
```

Befehle in Zeile 2 und 5 dienen zur Erzeugung eines Dateisystems mit dem entsprechenden Typ. Zeile 8 und 11 dienen zur Erzeugung eines SWAP Speichers. Dies ist ein Speicherbereich welcher vom Betriebssystem für eine Erweiterung des Arbeitsspeicher reserviert wird. Der Befehl in Zeile 13 bindet die zuvor erstellte Partition ein. Die Zeilen 14 und 15 dienen zur Einbindung der Bootpartition „sda1“ in das angeführte Verzeichnis.

Listing 5.2: Erzeugen des Dateisystems

```
1  //Dateisystem EXT2
2  mkfs.ext2
3
4  //Dateisystem EXT4
5  mkfs.ext4
6
7  //SWAP Speicher erzeugen
8  mkswap
9
10 //Aktivieren des SWAP
11 swapon
12
13 mount /dev/sda3 /mnt/gentoo
14 mkdir /mnt/gentoo/boot
15 mount /dev/sda1 /mnt/gentoo/boot
```

Im nächsten Schritt wird der sogenannte Stage 3 Tarball verwendet. Dieser definiert die eigentliche Version des Gentoo Systems. Das Beziehen des Tarballs zeigt der Befehl in Zeile 2 im Ausschnitt 5.3. Entpackt wird er mit dem Befehl in Zeile 5.

Listing 5.3: Beziehen des Tarball

```
1 //Beziehen des Tarballs
2 links http://gentoo.org/main/en/mirrors.xml
3
4 //Entpacken des Tarballs
5 tar xvjpf stage3-*.tar.bz2
```

Nun wird die Datei „make.conf“ nach individuellen Bedürfnissen und nach der zugrundeliegenden Hardware angepasst. Hierzu finden Sie eine detaillierte Anleitung im bereits erwähnten Handbuch.
Anschließend kann das Basissystem von Gentoo konfiguriert werden, genannt der „Portage-Tree“. Er bietet eine Sammlung von Programmen und Paketen an, welche

mittels dem Befehl „emerge“ installiert werden können. Der „Portage-Tree“ wird mit den im Ausschnitt 5.4 zu sehenden Befehlen konfiguriert welche eine Serverauswahl aufrufen. In dieser Auswahl können jene Server bestimmt werden, von denen die Spiegelung des Portage Trees vorgenommen werden soll.

Listing 5.4: Konfigurieren der Portage-Tree Server

```
1 mirrorselect -i -o >> /mnt/gentoo/etc/portage/make.conf
2 mirrorselect -i -r -o >> /mnt/gentoo/etc/portage/make.conf
```

Um die Internetverbindung im neuen System aufrecht zu erhalten müssen die DNS Informationen aus der Live Umgebung kopiert werden. Dies ist im Ausschnitt 5.5 in Zeile 1 zu sehen. Das Einbinden der benötigten Dateisysteme erfolgt in Zeile 3 bis 5. Die Befehle in Zeile 7,8 und 9 werden ausgeführt um die neue Umgebung zu betreten.

Listing 5.5: Befehle vor dem Betreten der neuen Umgebung

```
1 cp -L /etc/resolv.conf /mnt/gentoo/etc/
2
3 mount -t proc none /mnt/gentoo/proc
4 mount --rbind /sys /mnt/gentoo/sys
5 mount --rbind /dev /mnt/gentoo/dev
6
7 chroot /mnt/gentoo /bin/bash
8 source /etc/profile
9 export PS1='(chroot) \$PS1'
```

Sobald die neue Umgebung erfolgreich betreten wurde ist es an der Zeit den zuvor konfigurierten Portage-Tree zu installieren und auf den neusten Stand zu bringen, wie im Ausschnitt 5.6 in Zeile 2 und 3 zu sehen ist. Hierzu wird in Zeile 1 ein eigenes Verzeichnis angelegt in welches der Portage-Tree gespeichert wird.

Listing 5.6: Aktualisierung des Portage-Trees

```
1 mkdir /usr/portage
2 emerge-webrsync
3 emerge --sync
```

Eine der wichtigsten Variablen im gesamten System ist die „USE“ Variable. Sie definiert, welche Komponenten zusätzlich installiert werden sollen. Beispielsweise kann mit „“-gnome“ eine graphische Oberfläche installiert werden. Eine genaue Auflistung der möglichen Optionen findet sich erneut im Handbuch. Gespeichert wird die „USE“ Variable in der Datei „make.conf“ und kann beispielsweise mit dem Texteditor „nano“ bearbeitet werden, wie der Befehl in Zeile 1 des Ausschnitts 5.1 zeigt. Die Befehle der Zeile 4,8 und 9 dienen zum Setzen der Zeitzone. Diese Konfiguration ist nicht zwingend notwendig.

Listing 5.7: Bearbeiten der „USE“-Variable und setzen der Zeitzone

```
nano -w /etc/portage/make.conf

//Anzeige der moeglichen Zeitzonen
ls /usr/share/zoneinfo

//Beispiel: Europe/Brussels
//Setzen der Zeitzone
cp /usr/share/zoneinfo/Europe/Brussels /etc/localtime
echo "Europe/Brussels" > /etc/timezone
```

Abbildung 5.1: Bearbeiten der „USE“-Variable und setzen der Zeitzone

Im Anschluss folgt das Erzeugen des notwendigen Linux Kernel. Eine Beschreibung hierzu finden sie im Kapitel Kernelkompilierung.
Sobald der Kernel ohne Fehler kompiliert wurde können benötigte Kernelmodule, wie in Ausschnitt 5.8 gezeigt wird, festgelegt werden.

Listing 5.8: Hinzufügen von Kernelmodulen

```
//Auflistung aller verfuegbaren Kernelmodulen
find /lib/modules/<kernel version>/ -type f -iname '*.o' -or -iname '*.
   ko' | less

//Module hinzufuegen
nano -w /etc/conf.d/modules
```

Der nächste Schritt dient zur Konfiguration der sogenannten fstab Datei. Diese wird während des Starten des Betriebssystems gelesen und enthält relevante Informationen über die einzubindenden Festplatten beziehungsweise Partitionen. In Ausschnitt 5.9 sehen Sie jene fstab Datei welche in einer XBOX zum Einsatz kommt.

Listing 5.9: Einträge in der fstab Datei einer XBOX

```
 # <fs> <mountpoint>    <type>          <opts>          <dump/pass>
192.168.1.1:/srv/tftp/gdl       /               nfs     rw,soft,relatime
   ,bg,intr,udp,timeo=11,retrans=3 0 0
192.168.1.1:/srv/tftp/home      /mnt/home nfs   _netdev,rw,soft,nolock,
   bg,intr,udp,actimeo=11   0 0

proc            /proc   proc    defaults        0 0
shm             /dev/shm        tmpfs   nodev,nosuid,noexec     0 0
tmpfs           /tmp    tmpfs   defaults,noatime,nosuid,nodev,noexec,
   mode=1777,size=30M 0 0
/dev/sda2       /mnt/usb        ext3    defaults,noatime
/dev/sda1       swap    swap    defaults        0 0
```

Weiterführend sind nun einige kleinere Einstellungen vorzunehmen. Diese werden in

Ausschnitt 5.2 gezeigt und mit einem Kommentar, gekennzeichnet durch die Zeichenfolge „\\“, kurz erklärt.

Listing 5.10: Kleine Abschlussarbeiten

```
1  //Hostname setzen
2  nano -w /etc/conf.d/hostname
3
4  //DHCP aktivieren
5  nano -w /etc/conf.d/net
6  config_eth0="dhcp"
7
8  //Netzwerk beim Systemstart aktivieren
9  cd /etc/init.d
10 ln -s net.lo net.eth0
11 rc-update add net.eth0 default
12
13 //root Passwort setzen
14 passwd
15
16 //Systemeinstellungen konfigurieren
17 nano -w /etc/rc.conf
18
19 //Keymap aendern
20 nano -w /etc/conf.d/keymaps
```

Abbildung 5.2: Kleine Abschlussarbeiten

Abschließend muss das System nochmals neu geladen werden, gefolgt vom Neustart des Betriebssytems welches zudem den Abschluss der Arbeiten bildet.Der Ausschnitt 5.11 zeigt die hierzu notwendigen Befehle. Sollte das System auf einem normalen PC aufgesetzt werden, ist es zudem erforderlich einen Bootloader zu installieren. Dies wird im Gentoo Handbuch detailliert beschrieben.

Listing 5.11: Neuladen des Systems

```
1  //System neu laden
2  env-update && source /etc/profile
3
4  //Neustart des Systems
5  exit
6  cd
7  umount -l /mnt/gentoo/dev{/shm,/pts,}
8  umount -l /mnt/gentoo{/boot,/proc,}
9  reboot
```

Durch die getätigten Arbeiten wurde das Gentoo System vollständig aufgesetzt sowie konfiguriert und kann als Produktivumgebung verwendet werden.

5.1.2 Kernelkompilierung

Das Kompilieren des Kernel stellt eine der wichtigsten Arbeiten während des Aufsetzen eines Gentoo Systems dar. Zudem ist es auch eine der zeitaufwändigsten. Die Erzeugung des Kernels kann grundlegend auf zwei verschiedene Arten erledigt werden: Automatisch oder manuell.
Für die automatische Generierung des Kernels kann das Tool „genkernel“[46] verwendet werden, welches in Ausschnitt 5.12 in Zeile 2 installiert wird. Nach der Installation kann der Kernel, falls notwendig durch die Parametrisierung des genkernel-Aufrufs angepasst und anschließend erzeugt werden. In Zeile 5 wird ein Kernel erzeugt, welcher durch den angeführten Parameter eine möglichst große Unterstützung für unterschiedliche Hardware bereitstellt.

Listing 5.12: Erzeugen eines Kernel mit genkernel

```
1 //Installieren von genkernel
2 emerge genkernel
3
4 //Erzeugen eines generischen Kernel
5 genkernel all
```

Der nun erzeugte Kernel ist durch die Nichtparametrisierung sehr groß und überladen, da er viele Optionen anbietet welche eventuell für seinen Einsatzzweck nicht benötigt werden. Deswegen besteht die Möglichkeit den Kernel manuell zu erzeugen. Hierzu werden anstatt des genkernel Tools die sogenannten „gentoo-sources“ bezogen. Dies zeigt Ausschnitt 5.13. In Zeile 2 werden die Gentoo-sources bezogen. Mit dem Befehl in Zeile 4 wird in das Verzeichnis der gentoo-sources gewechselt. Abschließend wird mit dem Befehl in Zeile 8 eine graphische Oberfläche aufgerufen, mit der es möglich ist den Kernel beliebig anzupassen und zu erweitern.

Listing 5.13: Manuelles erzeugen eines Kernel

```
1 //Installieren der gentoo-sources
2 emerge gentoo-sources
3
4 //Wechseln in das Verzeichnis der gentoo-sources
5 cd /usr/src/linux
6
7 //Starten der graphischen Benutzeroberflaeche
8 make menuconfig
```

HTL Perg Kernel

Aufgrund eines schwerwiegendes Fehlers war es notwendig, Änderungen am Kernel vorzunehmen. Dieser Fehler, welcher es unmöglich machte, mehrere XBOXen gleichzeitig zu starten soll nun genauer beschrieben werden.
Während des Startvorgangs erhält eine XBOX 2 Mal eine neue MAC Adresse, wobei sie das erste Mal vom Bootloader vergeben wird. Sie ist korrekt und wird am Bildschirm

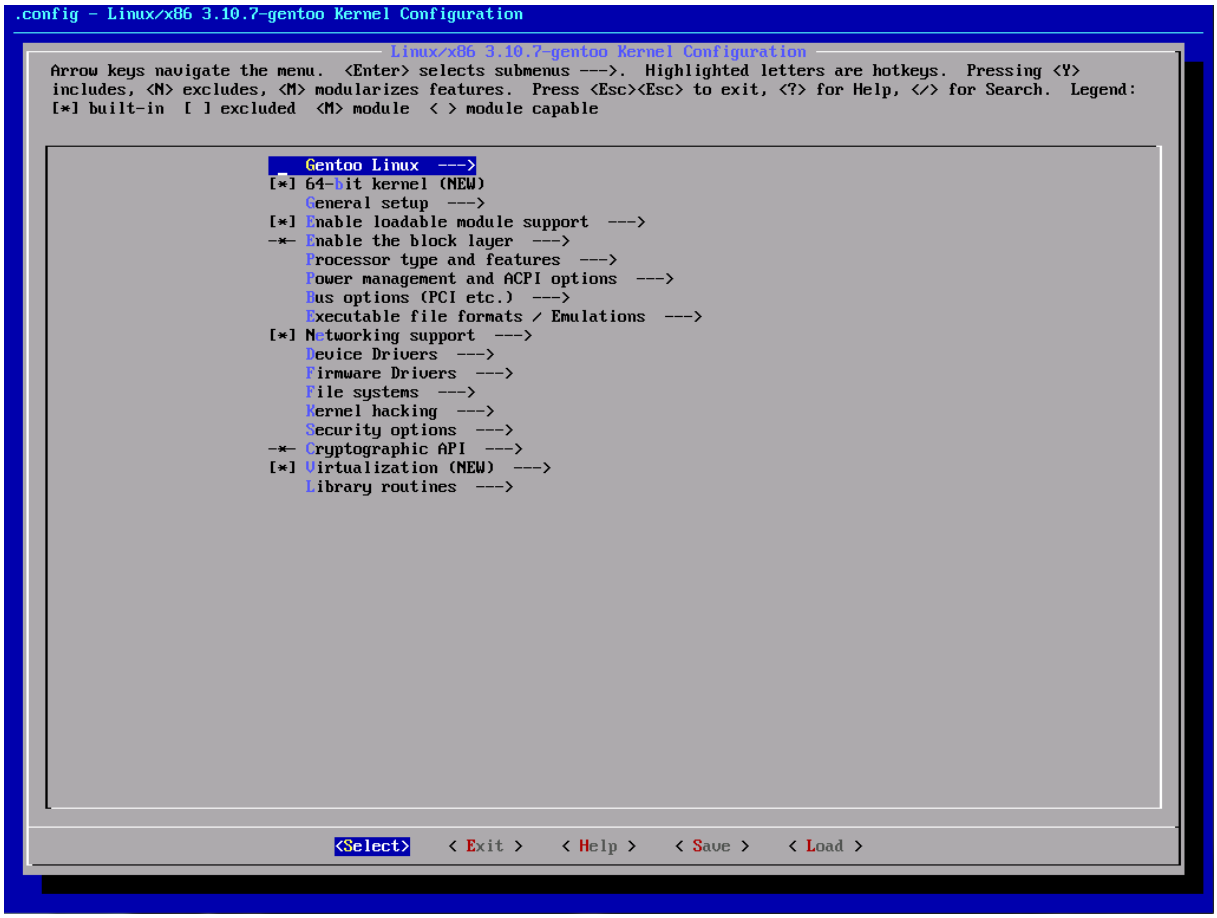


Abbildung 5.3: Benutzeroberfläche für das manuelle Erzeugen des Kernels

kurz angezeigt. Nachdem jedoch der Bootloader das Betriebssystem geladen hat, setzt dieses die MAC Adresse neu. Dafür zuständig ist der Netzwerktreiber, welcher in der konkreten Situation ein Standardtreiber ist und dadurch keinen Zugriff auf die tatsächliche MAC Adresse der Netzwerkkarte hat. Das Entwicklungsteam der Free60 Community hat sich daraufhin entschlossen nicht einen neuen Netzwerktreiber zu implementieren, sondern immer die selbe MAC Adresse statisch zu vergeben, wie im Ausschnitt 5.14 zu sehen ist. Dies hatte im Cluster zur Folge, dass sobald mehr als eine

Listing 5.14: Vergabe der statischen MAC Adresse im Netzwerktreiber

```
1 random_ether_addr(dev->dev_addr);
2 memcpy(dev->dev_addr, "\x00\x78\x65\x6E\6F\6E", 6);
```

XBOX hochgefahren wurde, die XBOX nicht mehr eindeutig identifizierbar war, da die MAC Adressen identisch waren. Ein weiteres Problem ergab sich zudem dadurch, dass der DHCP Server die IP Adresse aufgrund der MAC Adresse vergibt, sodass dies ebenfalls nicht mehr möglich war. Die Vergabe einer dynamischen MAC Adresse wurde in der Datei des Netzwerktreibers „xenon_net.c“ nachträglich implementiert. Dazu wurde zu Beginn ein neuer Kernel Bootparameter definiert, wie im Ausschnitt 5.15 zu

sehen ist. Die in Zeile 1 zu sehende Variable wird durch den letzten Eintrag in der Tabelle 4.1 initialisiert und wird im weiteren Ablauf verwendet.

Listing 5.15: Erzeugen eines neuen Bootparameters mit dem Namen „macaddr“ in Zeile 2. Zeile 3 fügt die Beschreibung für den Parameter hinzu.

```
1 static char* macaddr;
2 module_param(macaddr, charp, 0000);
3 MODULE_PARM_DESC(macaddr, "MAC-Address for Xenon Fast Ethernet Interface
    ");
```

Listing 5.16: Vergabe der Mac Adresse aufgrund des Bootparameters „macaddr“ in der Datei „xenon_net.c“.

```
1  if ((macaddr!=NULL) && (strlen(macaddr)==17)) {
2          printk("Received Kernel 'macaddr' parameter...\n");
3          printk("Parameter 'macaddr' is: %s Length is: %lu\n",macaddr,
               strlen(macaddr));
4          mac_address = string_to_mac(macaddr);
5    printk("Converted mac address is: %s\n",mac_address);
6          if (mac_address != NULL)
7                  memcpy(dev->dev_addr, mac_address, 6);
8    else
9                  printk("ERROR: Could not set MAC-Address!\n\n");
10 }
11 else {
12   printk("Not Received Kernel 'macaddr' parameter, setting default MAC-
       Address...\n");
13   memcpy(dev->dev_addr, "\x00\x01\x30\x44\x55\x66", 6);   /* same as
       xell */
14 }
```

Im Ausschnitt 5.16 wird nun mit der Vergabe der MAC Adresse begonnen. Zuallererst wird überprüft ob die angegebene Zeichenkette tatsächlich den Anforderungen einer MAC Adresse entspricht. In Zeile 4 wird im nächsten Schritt die Zeichenkette in Hexadezimaldarstellung umgewandelt, da dies der standardisierten Form entspricht. Sollte bis zu diesem Zeitpunkt kein Fehler aufgetreten sein wird mit dem Befehl „memcpy(void *destination, const void *source, size_t n)“ die MAC Adresse im System gesetzt. Im Falle eines Fehlers wird in Zeile 13 eine statische MAC Adresse vergeben.

5.1.3 INITRAMFS

Das sogenannte INITRAMFS[40] ist ein wesentliches Glied in der Kette, für einen erfolgreichen Start eines Gentoo Systems. Es wird in der Praxis als cpio-Datei ausgeliefert und wird mit mehreren Algorithmen komprimiert. Bestehend aus einem vollständigen Set von Ordnern, welches dem eines normalen Dateisystems gleicht, ist die einzige Aufgabe des INITRAMFS das Mounten des Root Verzeichnis. Durch die Verwendung eines INITRAMFS kann jedoch die Startzeit eines Systems signifikant

Listing 5.17: Generische Erzeugung eines INITRAMFS

```
1 mount /boot
2 dracut
```

beeinträchtigt werden. Zum Zeitpunkt des Starts lädt der Bootloader den Kernel sowie das INITRAMFS in den Arbeitsspeicher. Sogleich prüft der Kernel ob ein INITRAMFS vorhanden ist und bindet dieses, falls vorhanden, im Dateisystem ein. Heutzutage wird ein INITRAMFS meistens dazu verwendet, benötigte Kernel Module[20] zu laden.

DRACUT

Für die Erzeugung des verwendeten INITRAMFS wurde das Tool DRACUT[45] verwendet. Diese Software ermöglicht das Generieren eines INITRAMFS auf Basis eines laufenden Systems. Eine detailierte Beschreibung der Software wurde von Harald Hoyer in der Revision 3.0 2013 veröffentlicht[18]. Für das Erzeugen eines INITRAMFS speziell für Gentoo stellt erneut die Gentoo Community eine sehr gute Anleitung bereit. Die einfachste Form, wie sich eine entsprechendes INITRAMFS erzeugen lässt wird im Ausschnitt 5.17 gezeigt. Durch die Möglichkeit weitere Argumente an das Tool zu übergeben, lassen sich die im INITRAMFS enthalten Module beliebig modifizieren.

5.2 Ubuntu

Ubuntu[28] ist ein Betriebssystem welches, wie Gentoo, auf dem Linux Kernel basierend arbeitet. Es wird momentan von der Firma Canonical Ltd. entwickelt und betreut. Der Unterschied zwischen Ubuntu und Gentoo ist jener, dass Ubuntu um ein vielfaches benutzerfreundlicher ist als Gentoo. Durch eine moderne und einfache graphische Oberfläche und einer Fülle an Programmen kann es mittlerweile als kostenlose Alternative zu Microsofts Windows[29] angesehen werden. Zum Einsatz kommt Ubuntu auf dem Shuttle-PC welcher als XBOX-Master PC für den Cluster dient.

5.2.1 Network File System Mount (3)

Der „Network File System Mount„ (NFS Mount) findet auf dem XBOX-Master PC statt und dient der Bereitstellung der benötigten Systemdateien für die XBOX. Die Konfiguration wird hierbei in der Datei „exports“ vorgenommen. Die im Ausschnitt 5.4 angegeben Verzeichnisse werden von jeder XBOX während des Startvorgangs eingebunden. Dies geschieht in der „fstab“ Datei, welche sich im Verzeichnis „etc“ befindet. Ausschnitt 5.19 zeigt einen Auschnitt dieser Datei.

5.2.2 Another Union Filesystem

Another Union Filesystem(AUFS)[33] ist ein Dateisystem welches es vereinfacht gesagt ermöglicht, mehreren Clients den Lesezugriff auf Dateien zu gewähren. Sollte ein Client jedoch Änderungen an einer Datei vornehmen wollen, wird diese Änderung nicht direkt auf die gelesene Datei geschrieben sondern es wird die zu ändernde Datei kopiert, die

Listing 5.18: Angabe der freizugebenden Verzeichnisse am XBOX-Master PC

```
1 /srv/tftp/gdl 192.168.1.0/255.255.255.0(rw,sync,no_all_squash,
     no_root_squash,no_subtree_check)
2 /srv/tftp/gdlusb 192.168.1.0/255.255.255.0(rw,async,no_all_squash,
     no_root_squash,no_subtree_check)
3 /srv/tftp/home 192.168.1.0/255.255.255.0(rw,async,no_all_squash,
     no_root_squash,no_subtree_check)
```

Abbildung 5.4: Angabe der freizugebenden Verzeichnisse am XBOX-Master PC

Listing 5.19: Einbinden der vom NFS Mount freigegeben Verzeichnisse auf der XBOX

```
1 192.168.1.1:/srv/tftp/gdl    /         nfs     rw,soft,relatime,bg,intr,
     udp,timeo=11,retrans=3 0 0
2 192.168.1.1:/srv/tftp/home   /mnt/home nfs     _netdev,rw,soft,nolock,bg
     ,intr,udp,actimeo=11   0 0
```

Änderung in dieser Datei gespeichert und anschließend wird sie auf einen darüber liegenden Schreibbereich, welcher für jeden Client exklusiv zur Verfügung steht, gespeichert. Diese Methode wird „file based Copy on Write (COW) method“[33] genannt.

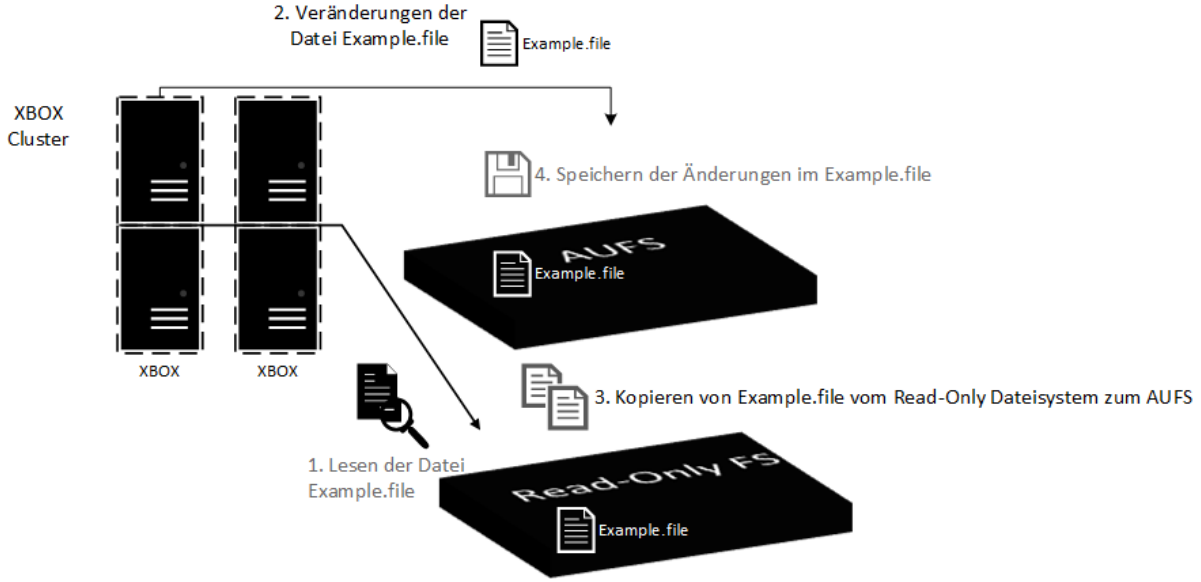


Abbildung 5.5: Beschreibung der Funktionsweise von AUFS

Diskless-Client

Das im Kapitel „Another Union Filesystem“ beschriebene Dateisystem wird für die Realisierung von sogenannten „Diskless-Clients“ verwendet. Ein „Diskless-Client“, also ein PC ohne Festplatte, kommt dann zum Einsatz, wenn der Client selber keinen

Speicher benötigt, oder aus Platz- bzw Preisgründen keinen hat. Die XBOX360 verfügt standardmäßig über Festplattenspeicher, dieser musste jedoch im Zuge der Modifikationen ausgebaut werden, um Platz für den Glitcher zu schaffen. Für die Realisierung dieser Architektur wurde auf ein Script aufgebaut, welches beim Starten des Gentoo Systems, anstatt des standardmäßigen Initialisierungsscript, ausgeführt wird. Die Methode „aunionfs()“ stellt die Funktionalität für das Einbinden von unbegrenzt

Listing 5.20: Das Initialisierungsscript „stateless.sh“

```
1  #!/bin/bash
2
3  MODPROBE=/sbin/modprobe
4  IFCONFIG=/sbin/ifconfig
5  MYHOST=$(/bin/hostname) # By default, we'll use the DHCP assigned
       hostname
6
7  NODE_NAME="Node"
8
9  ahostname(){
10         if [ -z ${MYHOST} ]; then
11                 echo "DHCP didn't tell me my name. Generating my own
                       hostname..."
12         MYHOST="${NODE_NAME}`$IFCONFIG eth0 | awk '/inet addr/ {print $2
               }' | tr -t . ' ' | awk '{print $4}'`"
13         else
14                 echo DHCP told me my hostname is ${MYHOST}...
15                 echo "Setting domainname to DHCP's settings"
16         fi
17
18         echo "STATELESS: Setting Hostname to $MYHOST"
19         echo "HOSTNAME=\"$MYHOST\"" > /etc/conf.d/hostname
20 }
21
22 aunionfs() {
23                 echo "STATELESS: Loading module unionfs ..."
24                 while [ "$1" != "" ]; do
25                         echo "STATELESS: Mounting tmpfs $1 ..."
26                         mount -n -t tmpfs -o defaults none /mnt/unionfs/
                               $1
27                         echo "STATELESS: Mounting $1 unionfs ..."
28                         mount -t aufs -o br=/mnt/unionfs/$1=rw:/$1=ro
                               none /$1
29                         shift
30                 done
31                 echo "Mounting HOME-NFS-Directories...";
32                 mount -t nfs4 192.168.1.1:/srv/tftp/home /mnt/home -o rw
                       ,soft,nolock,bg,intr,udp,actimeo=11
33 }
34
35 aunionfs var etc lib tmp
36 ahostname
37 exec /sbin/init
```

vielen AUFS zur Verfügung. Das Einbinden eines einzigen AUFS funktioniert aufgrund von zwei einfachen Befehlen die im Ausschnitt 5.21 gezeigt werden.

Listing 5.21: Einbinden eines AUFS

```
1 mount -n -t tmpfs -o defaults none /mnt/unionfs/$1
2 mount -t aufs -o br=/mnt/unionfs/$1=rw:/$1=ro none /$1
```

„$1“ steht hierbei für ein beliebiges Verzeichnis, welches als AUFS eingebunden werden soll. Der Befehl in Zeile 1 bindet zunächst das Verzeichnis als temporäres Dateisystem ein. In Zeile 2 wird darüber das AUFS gelegt, welches nun die die Funktionalität der zuvor erklärten COW Methode realisiert.
Das in Abbildung 5.20 gezeigte Script bindet, wie in Zeile 35 zu sehen ist, die Verzeichnisse „var“, „etc“, „lib“ und „tmp“ als AUFS ein. Der „mount“ Befehl in Zeile 32 bindet zusätzlich ein „Home“ Verzeichnis ein, welches von allen laufenden XBOXen gemeinsam genutzt wird. In „Home“ hat jede XBOX zusätzlich normale Schreib- und Leserechte.

Kapitel 6

Überwachung des Clusters

Würde man jede XBOX einzeln beobachten, wäre es notwendig für jede XBOX einen Bildschirm mit Tastatur zur Verfügung zu stellen. Diese Lösung ist nicht zufriedenstellend, trotzdem ist es notwendig einen Überblick über den Zustand jeder XBOX zu behalten. Aus diesem Grund wurde eine Software entwickelt, welche es ermöglicht eine XBOX zu registrieren und ihre Daten auf einer Weboberfläche darstellen zu können. Zudem können auch Befehle an eine XBOX gesendet werden.

6.1 Verwendete Technologien

Für die Realisierung der Überwachungssoftware kommen eine Vielzahl an Technologien zum Einsatz. Manche sind notwendig um die Kommunikation mit einer XBOX zu realisieren andere wiederum dienen lediglich der optischen Gestaltung der Software.

6.1.1 PHP: Hypertext Preprocessor

PHP: Hypertext Preprocessor (PHP)[1] ist eine Open-Source Skriptsprache welche entwickelt wurde um die Funktionalität von Web Anwendungen zu erweitern. Sie zeichnet sich dadurch aus, dass der Code bereits auf dem Server zur Ausführung kommt. Im Gegensatz dazu steht beispielsweise Javascript (JS), erklärt im Kapitel „Javascript“, welches am Client ausgeführt wird.

Smarty

Smarty [30] ist eine Template-Engine für PHP. Diese Engine dient zur Trennung der Applikations-Logik und der Anzeige. Hierzu wird ein sogenanntes Template entworfen, welches mithilfe von Smarty designiert wird. Dies sind Dateien mit der Endung „.tpl“. Nicht zwangsläufig wird die Applikations-Logik aus dem Template entfernt. Zur Anzeige notwendige Logik wird weiterhin im Template bestehen bleiben, beispielsweise kann dies ein beliebiger Javascript Code sein.

PHP Secure Communications Library

Die PHP Secure Communications Library (phpseclib)[3] dient zur Kommunikation über SSH in PHP. Phpseclib kann ab der PHP Version 4 verwendet werden und unterstützt

SSH 1 sowie auch SSH 2.

6.1.2 Javascript

Javascript (JS)[15] ist ähnlich wie PHP eine Skriptsprache, allerdings unterscheidet sie sich wesentlich dadurch, dass Javascript Code meist auf dem Rechner des Benutzers ausgeführt wird. JS wird von allen modernen Browsern unterstützt. Wie auch PHP lässt sich JS durch Bibliotheken erweitern, welche es ermöglichen Funktionalitäten leichter zu implementieren.

jQuery

jQuery[22] ist eine JS Bibliothek welches es ermöglicht viele Anforderungen leichter realisieren zu können. jQuery ist eine der größten und bekanntesten Bibiliotheken, unterstützt Mobilgeräte und auch das Entwerfen von graphischen Oberflächen.

jQuery lässt sich durch das Verwenden von Plug-Ins beliebig ergänzen. Das in der Software verwendete Plug-In Growl dient zur Darstellung von wichtigen Informationen. Diese können durch Growl aufbereitet werden und jederzeit angezeigt werden.

6.1.3 Apache Web Server

Der Apache Web Server wurde erstmals im April 1995 veröffentlicht und ist ein HTTP (Web) Server. Zuvor existierte keine einheitliche Version des Web Servers, da der Erfinder Rob McCool die Software ab 1994 nicht mehr weiterentwickelte. So entstanden vorübergehend viele kleine Verbesserungen, welche 1995 durch ein Team von Netzwerkadministratoren zusammengetragen wurden. Ziel des Apache HTTP Server Projekts[10], hinter welchem die Apache Software Foundation steht, ist eine stabile und freie Version des HTTP Web Servers anbieten zu können.

6.1.4 MySQL

MySQL ist eine Open-Source Datenbank, welche sowohl für Linux als auch für Windows zur Verfügung steht. Für die Entwicklung und Wartung zeichnet sich die Oracle Corperation verantwortlich.

6.2 Aufbau der Software

Die folgenden Kapitel beschreiben die einzelnen Komponenten der Software. In Abbilung 6.1 ist der Aufbau der Software zu erkennen.

6.2.1 Beschaffung der Daten

Für das Beschaffen der Daten von einer XBOX wurde die Technologie SSH verwendet. In Kombination mit der PHP Bibliothek phpseclib ist es möglich sich zu einer beliebigen XBOX zu verbinden, Befehle abzusetzen und den Rückgabewert weiterzuverarbeiten. Die gesamte Funktionalität welche benötigt wird, um die Daten von einer XBOX zu

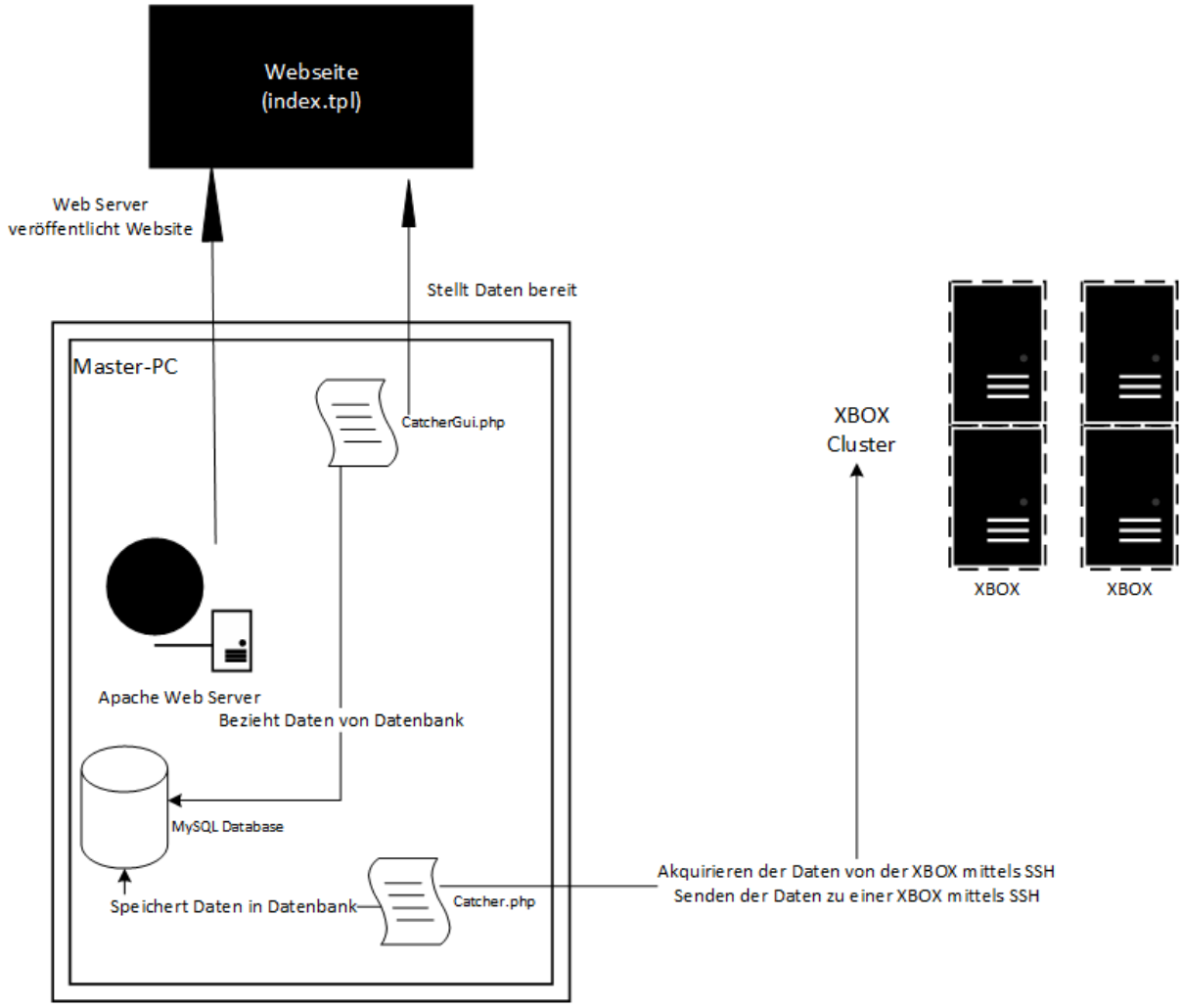


Abbildung 6.1: Systemarchitektur der Überwachungssoftware

beschaffen, wurde in die PHP Klasse Common.php ausgelagert. Für die notwendigen Verbindungsdaten, welche für Datenbank und XBOX benötigt werden, liest die Methode parseINI() die Informationen aus der catcher.ini Datei aus.
Um sich zu einer XBOX zu verbinden wird die Methode createConnection($ipAddress) aufgrufen, welche sich mittels SSH zu einer XBOX verbindet, wie Auschnitt 6.1 zeigt. Das Objekt „con“ vom Typ Net_SSH2 welches von der Methode zurückgegeben wird, wird in einer Liste von Verbindungsobjekten gespeichert. Mit dem gespeicherten Objekt können zu einem späteren Zeitpunkt Daten von einer XBOX angefordert werden. Der Programmcode 6.2 zeigt beispielsweise das Abfragen der MAC-Adresse. Der in Zeile 4 zu sehende Regex Ausdruck dient zur Überprüfung, ob die erhaltene Zeichenkette dem Format einer MAC-Adresse entspricht. Da die Daten persistiert werden müssen bietet die Klasse Common.php auch die Möglichkeit eine Verbindung zu einer MySql Datenbank herzustellen. Hierzu verwendet die Methode connectToDb() die Verbindungsdaten welche aus der catcher.ini Datei ausgelesen werden. Der Code 6.3 zeigt in Zeile 4 das Verbinden zu einer MySql Datenbank und in Zeile 6 das Überprüfen

Listing 6.1: Verbindungsaufbau zu einer XBOX

```
1 public function createConnection($ipAdress)
2 {
3         $con = new Net_SSH2($ipAdress);
4         if (!$con->login($this->ssh_username, $this->ssh_password)) {
5                 return null;
6         }else{
7                 return $con;
8         }
9 }
```

Listing 6.2: Abfragen der MAC Adresse

```
1  public function getMac($con)
2 {
3         $result = $con->exec('netstat -ei');
4         if(preg_match('/((?:[0-9a-f]{2}[:-]){5}[0-9a-f]{2})/i', $result,
              $match)){
5                 return $match[0];
6         }
7 }
```

der Verbindung. Die Verbindung wird in einer globalen Variable gespeichert.

Listing 6.3: Verbinden zur MySql Datenbank

```
1 function connectToDB()
2 {
3         // Create connection
4         $this->db = new mysqli($this->serverurl, $this->db_username,
              $this->db_password, $this->databasename);
5         // Check connection
6         if ($this->db->connect_error) {
7                 die("Connection failed: " . $this->db->connect_error);
8         }
9         mysqli_autocommit($this->db, FALSE);
10 }
```

6.2.2 Graphische Oberfläche

Die graphische Oberfläche wird durch die Template Engine Smarty erzeugt, welche im Kapitel Smarty erklärt wird. Weitere Funktionalitäten und Verbesserungen der Anzeige werden durch die Verwendung von Javascript, respektive JQuery, sowie Cascade Style Sheets realisiert. In Abbildung 6.2 ist die Anzeige einer XBOX zu sehen. Mittels der Schaltfläche „show Services“ werden alle momentan laufenden Dienste auf der XBOX angezeigt. „show SSH-Live“ ermöglicht das Senden von SSH Befehlen an die XBOX.

Das Löschen der XBOX aus der Datenbank ist durch die Schaltfälche „delete XBOX“ möglich. Der in der Abbildung 6.2 zu sehende rote Kreis gibt Auskunft darüber, ob die XBOX momentan erreichbar ist. Der Kreis wird in Grün angezeigt sobald erkannt wird, dass die XBOX erreicht wird. Die Anzeige der laufenden Prozesse ist ebenfalls in Abbildung 6.2 zu sehen. Die angezeigte Tabelle listet jeden aktuell laufenden Prozess mit dessen Auswirkungen auf die CPU und den Arbeitsspeicher auf.

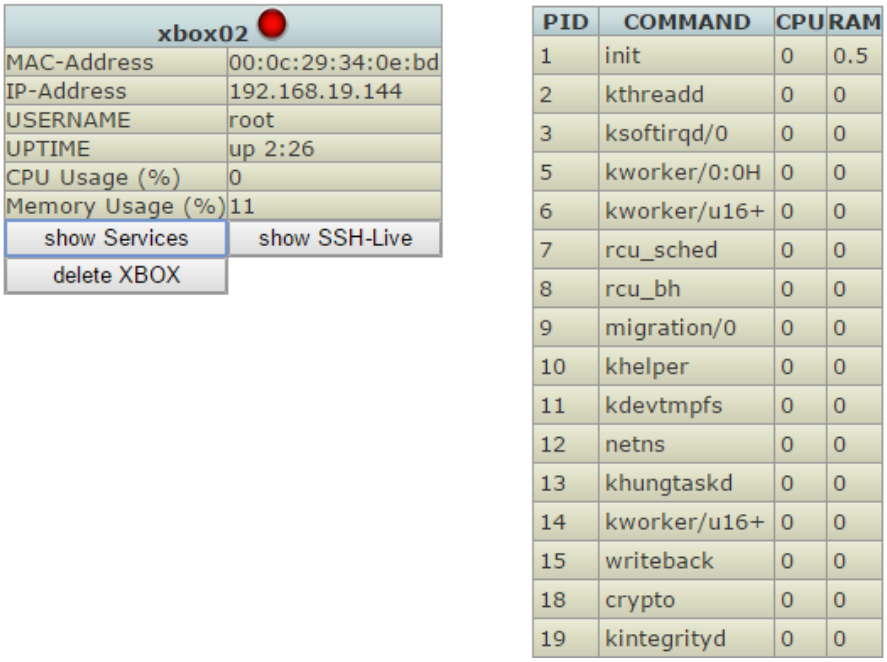
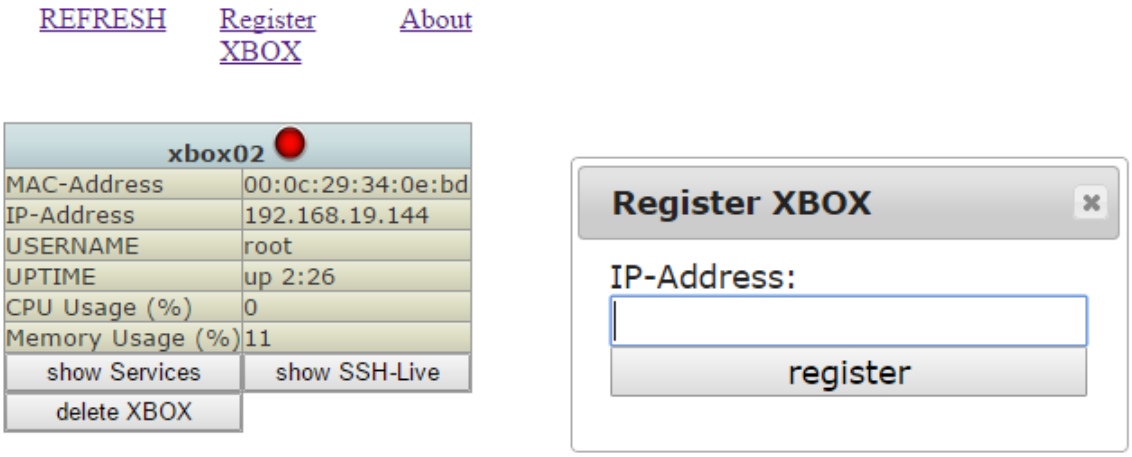


Abbildung 6.2: Anzeige einer XBOX und der dazugehörigen Prozesse

Um eine neue XBOX registrieren zu können, ist im oberen Bereich der Webseite eine einfache Navigation angebracht, die in Abbildung 6.3 zu sehen ist. Durch das Aufrufen von „Register XBOX“ wird ein Formular angezeigt, welches mithilfe von JQuery dargestellt wird. Wie in Abbildung 6.3 zu sehen ist, reicht es die IP-Adresse der XBOX anzugeben. Durch die Schaltfläche „register“ wird die XBOX registriert und in der Oberfläche angezeigt.



(a) Navigation (b) Registrierung

Abbildung 6.3: Registrierung einer neuen XBOX

Fehlerbehandlung

Sämtliche Fehler, welcher durch Usereingaben, Netzwerkprobleme oder Programmfehlern auftreten werden durch die JQuery Erweiterung Growl angezeigt. Die in Abbildung 6.4 gezeigte Fehlermeldung wird im rechten oberen Bereich angezeigt. In diesem konkreten Fall lässt sich dank der Fehlermeldung darauf schließen, dass der User versucht hat eine neue XBOX zu registrieren. Allerdings wurde entweder die IP Adresse nicht richtig angegeben, oder die Verbindungsdaten in der Konfigurationsdatei sind fehlerhaft. Um erfahrenen Nutzern eine bessere Rückmeldung zu geben, wird zusätzlich zu der einfachen Fehlermeldung, welche in der 2.Zeile zu sehen ist, noch der gesamte Stacktrace der Fehlermeldung angezeigt.

6.2.3 Laufende Aktualisierung der Daten

Für die Aktualisierung der Daten wird ein PHP Script zur Verfügung gestellt, welches in periodischen Zeitabständen die vorhanden Daten in der Datenbank aktualisiert. Hierzu werden aus der Datenbank die IP-Adresse jeder XBOX ausgelesen und mittels der Methoden, welche die Common Klasse bereitstellt, aktualisiert. Dies ist in Codeausschnitt 6.4 zu sehen.

6.2.4 Beziehen der Daten aus der Datenbank

Um die Daten von der Datenbank beziehen zu können wird in der Datei catcherGui.php eine Verbindung zur Datenbank hergestellt. Hierzu wird erneut ein Objekt der Klasse Common verwendet. Anschließend werden die Daten aus der Datenbank gelesen und in ein Array gespeichert. Dieses wird von Smarty verwendet, um die Oberfläche zu erzeugen. Smarty ruft die Seite index.tpl auf, welche die Oberfläche beinhaltet. Die Datei index.tpl ist ein HTML Dokument, in dem der Quellcode für eine XBOX steht. Smarty erstellt vor der Anzeige für jede XBOX welche im Array enthalten ist, denselben Quellcode, wobei spezifische Daten der XBOX eingesetzt werden. Der Codeausschnitt,

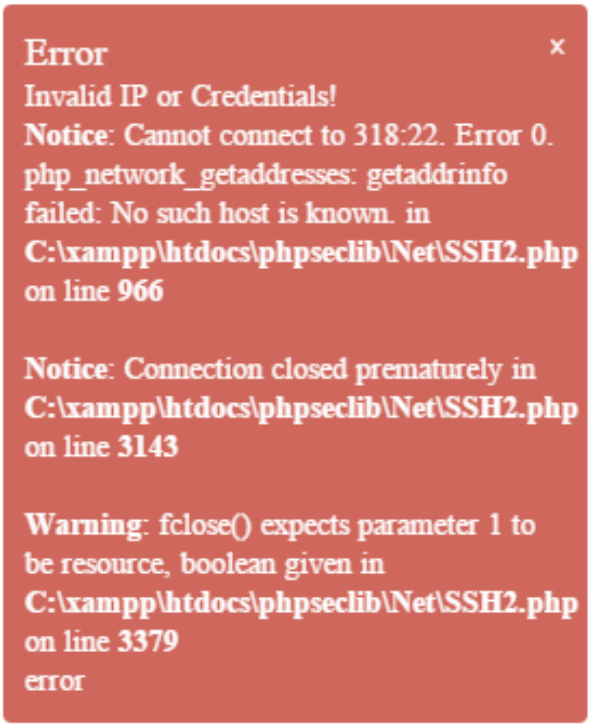


Abbildung 6.4: Fehlermeldung

Listing 6.4: Aktualisieren der Daten einer XBOX

```
1 function updateXBOX($connection)
2 {
3         global $commonObject;
4         $mac = $commonObject->getMac($connection);
5         $commonObject->clearServiceTable($mac);
6         $commonObject->insertServices($c->getServices($connection),$mac)
              ;
7         $commonObject->update_UP_CPU_MEM($connection);
8
9 }
```

welcher für jede XBOX erneut generiert wird ist im Codeausschnitt 6.5 zu sehen. In Zeile 1 wird auf das Array, welches vom PHP Script übergeben wird, zugegriffen. Der Zugriff auf konkrete Daten erfolgt mittels einer speziellen Syntax, welche zum Beispiel in Zeile 5 zu sehen ist. Dort wird der Hostname einer XBOX aus dem Array ausgelesen und in den Tabellenkopf geschrieben.

Listing 6.5: Quellcodeauschnitt von index.tpl

```
1  {section name=xboxData loop=$xboxData step=-1}
2  <div class="xboxContent" id="{$xboxData[xboxData].MAC}">
3  <div class="xboxDiv">
4  <table class="xboxTable">
5          <tr><th colspan="2">{$xboxData[xboxData].HOSTNAME}</th></tr>
6          <tr><td>MAC-Address</td><td>{$xboxData[xboxData].MAC}</td></tr>
7          <tr><td>IP-Address</td><td>{$xboxData[xboxData].IP_ADDRESS}</td
               ></tr>
8          <tr><td>USERNAME</td><td>{$xboxData[xboxData].username}</td></tr
               >
9          <tr><td>UPTIME</td><td>{$xboxData[xboxData].UPTIME}</td></tr>
10         <tr><td>CPU Usage (%)</td><td>{$xboxData[xboxData].cpu_total}</
               td></tr>
11         <tr><td>Memory Usage (%)</td><td>{$xboxData[xboxData].mem_total
               }</td></tr>
12         <tr><td> <input type="submit" value="show Services" onclick="
               showService('{$xboxData[xboxData].MAC}')" /> </td>
13                         <td> <input type="submit" value="show SSH-Live"
                               onclick="showSSHLive('{$xboxData[xboxData].
                               MAC}')"/></td>
14         </tr>
15         <tr><td> <input type="submit" value="delete XBOX" onclick="
               deleteXBOX('{$xboxData[xboxData].MAC}')"/> </td>
16         </tr>
17 </table>
18 </div>
19
20         <div class="furtherContent">
21                 <!--Services-->
22                 <!--SSH Live-->
23
24         </div>
25 </div>
26 {/section}
```

Kapitel 7

Managementsoftware

Eine genauere Überwachung des Clusters lässt sich durch die Verwendung von Open-Source Software erreichen. Diese zeichnen sich durch hohe Funktionalität und übersichtliche Oberflächen aus.

7.1 Ganglia

Ganglia[35] ist eine Überwachungssoftware, welche speziell für HPCs entwickelt wurde. Ganglia besteht aus drei Komponenten, welche es gemeinsam ermöglichen Änderungen im Cluster-Verbund zu erkennen, Daten innerhalb des Clusters zu übertragen und die erkannten Änderungen und gesammelten Daten auf einer Weboberfläche darzustellen. Mittels zwei Diensten, dem Ganglia Monitoring Daemon (gmond) und dem Ganglia Meta Daemon (gmetad), werden Änderungen im Cluster-Verbund erkannt und alle notwendigen Daten des Clusters gesammelt. Für die Anzeige der gesammelten Informationen ist das Ganglia PHP-Web-Front-End zuständig. Es wird beispielsweise die CPU Auslastung einzelner Knoten, oder des gesamten Clusters visualisiert. Diese Visualisierungen sind bereits auschnittsweise im Kapitel „Das Mandelbrot Fraktal“ zu sehen. Im oberen Bereich findet sich eine Navigationsleiste, welche in 7.1 zu sehen ist. Im Bereich „Main“ wird eine Übersicht über den gesamten Cluster und über jeden einzelnen Knoten angezeigt. Zusätzlich gibt es die Möglichkeit die Anzeige auf einen einzelnen Knoten zu begrenzen.

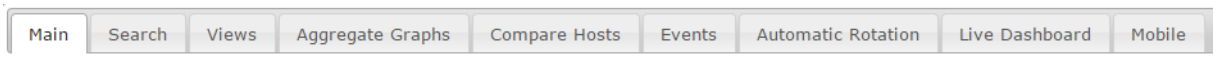


Abbildung 7.1: Navigationsleiste des Ganglia PHP-Web-Front-End

Um verschiedene Werte anzeigen zu können, gibt es eine Auswahl an Metriken welche im unteren Bereich der „Main“ Seite ausgewählt werden kann. Ein Ausschnitt dieser Metriken ist in Abbildung 7.2 zu sehen. Die gewählte Metrik wird nach der Auswahl für jeden Knoten angezeigt.

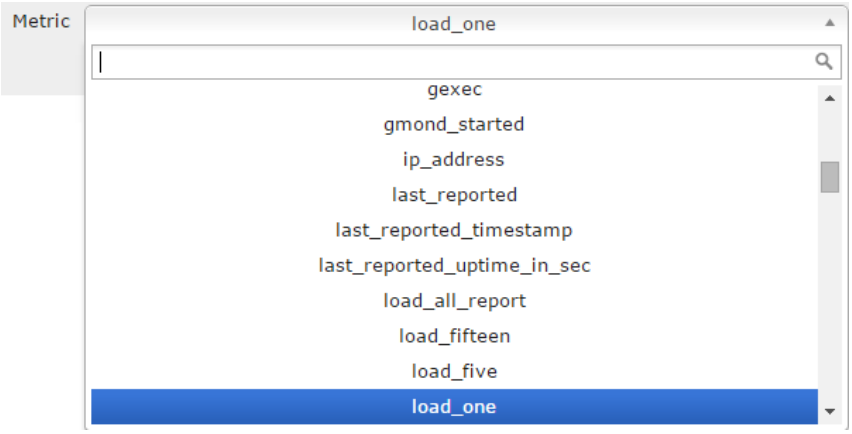


Abbildung 7.2: Auswahl einer Metrik

Um einen schnellen Überblick über einen Knoten zu erlangen bietet der „Search“ Bereich die Möglichkeit nach einem Knoten oder einer Metrik zu suchen. Anschließend erfolgt eine detaillierte Anzeige des ausgewählten Knoten.

Der Bereich „View“ ermöglicht das Erstellen eigener Anzeigen um sich einen besseren Überblick über den Cluster verschaffen zu können. In Abbildung 7.3 ist die View mit dem Namen „SPARKY“ zu sehen. Sie zeigt die Metrik „load_one“ für jeden Knoten an. Der Zeitrahmen kann individuell eingestellt werden.

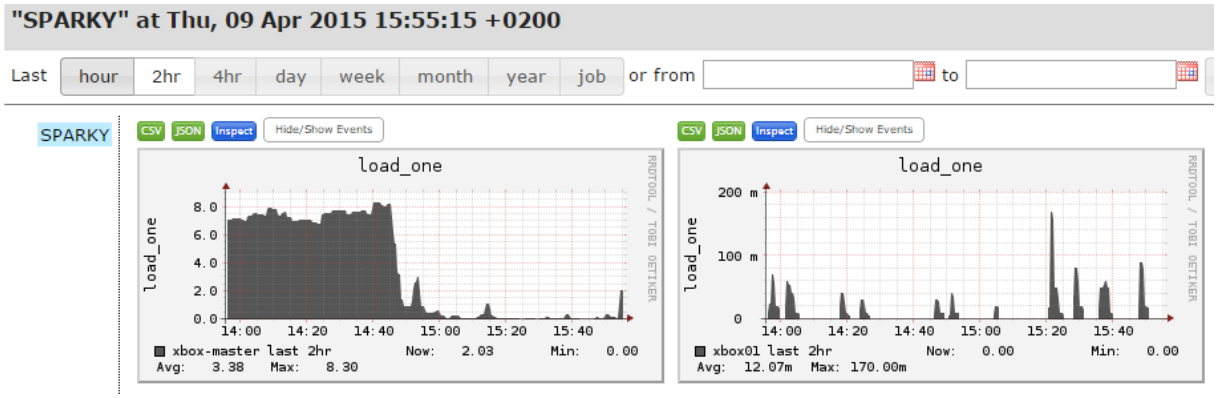


Abbildung 7.3: Anzeige der View „SPARKY“

Dank des Bereichs „Aggregate Graphs“ ist es möglich aggregierte Graphen zu erstellen. In Abbildung 7.4 wird ein Graph erstellt, um den freien Arbeitsspeichers der XBOX01, XBOX02 und XBOX03 aggregiert darzustellen. Ganglia erstellt hierzu Graphen für verschiedene Zeitperioden. Beginnend mit einer Übersicht über die letzte Stunde bis hin zu einem Graphen für das gesamte letzte Jahr. Ein Ausschnitt dieser Graphen ist in Abbildung 7.5 zu sehen. Mithilfe von Regulären Ausdrücken ist es möglich beliebige Knoten oder verschiedene Metriken einzustellen.

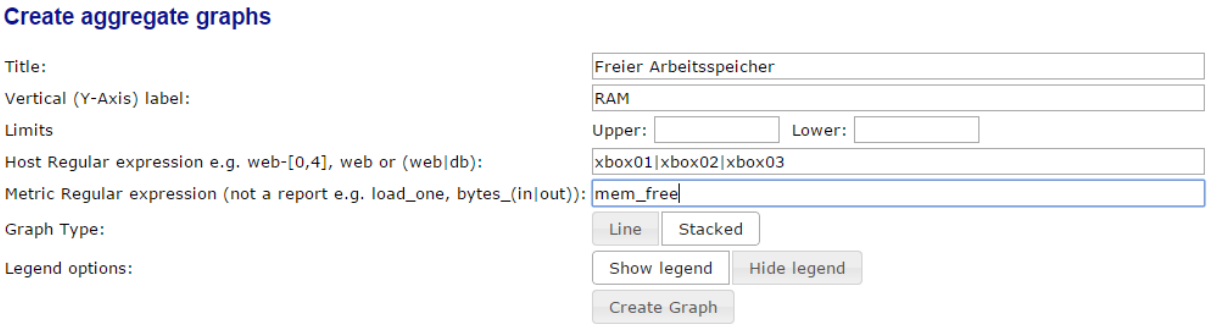


Abbildung 7.4: Erstellen eines Vergleichs des freien Arbeitsspeichers

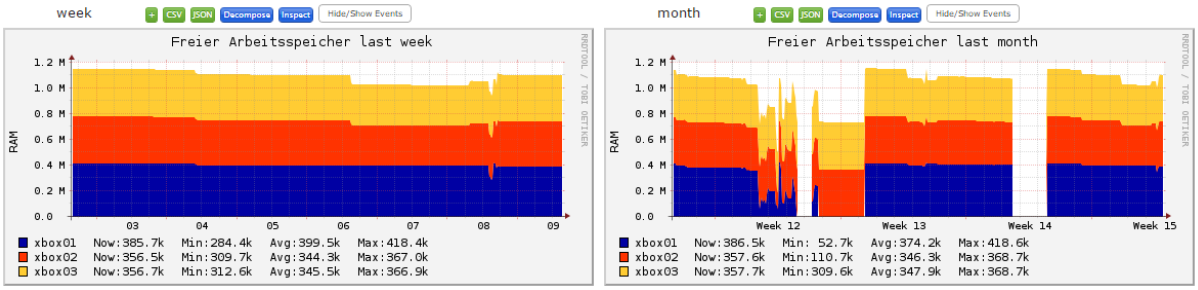


Abbildung 7.5: Ausschnitt der erstellten aggregierten Graphen

Für den praktischen Gebrauch ist es oft nützlich mehrere Knoten direkt miteinander Vergleichen zu können. Hierfür bietet der Bereich „Compare Hosts“ die entsprechenden Möglichkeiten. In Abbildung 7.6 werden die XBOX01, XBOX02 und die XBOX07 miteinander verglichen. Dies wird erneut mittels regulären Ausdrücken gesteuert. Zusätzlich kann über der Eingabe der Hosts ein Zeitrahmen definiert werden. Entweder durch einen der vorgegebenen Perioden oder durch die manuelle Eingabe eines Start- und Endzeitpunktes. Anschließend werden sämtlich verfügbare Metriken angezeigt, wie ausschnittsweise in Abbildung 7.7 gezeigt wird.

Abbildung 7.6: Vergleich der XBOX01, XBOX02 und XBOX07

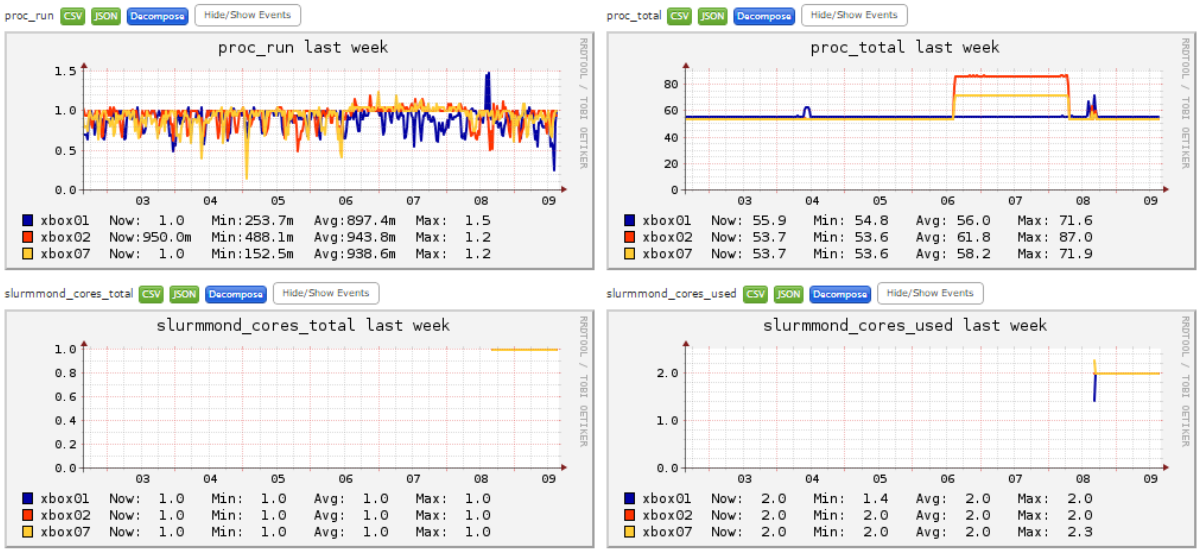


Abbildung 7.7: Ausschnitt der angezeigten Metriken für die XBOX01, XBOX02 und XBOX07

Mithilfe des „Event“ Bereichs ist es bei Auftreten eines bestimmten Events möglich eine vertikale Linie über einen Graphen zu legen. Dadurch können nach einen bestimmten Ereignis Änderungen erkannt zugeordnet und verglichen werden.

Da die Anzeige sämtlicher Knoten auf einmal sehr unübersichtlich und oftmals gar nicht notwendig ist, kann im Bereich „Automatic Rotation“ in gewissen Zeitabständen automatisch zwischen den einzelnen Knoten gewechselt werden. Dazu genügt es eine View anzugeben, zwischen deren Knoten anschließend gewechselt werden.

Um die Darstellung einer View in Echtzeit und möglichst übersichtlich zu ermöglichen gibt es den Bereich „Live Dashboard“. In Abbildung 7.8 ist ein Ausschnitt des Dashboards der View „SPARKY“ zu sehen.

Abbildung 7.8: Ausschnitt aus dem Live Dashboard der View „SPARKY“

Weitere Funktion

Ganglia ermöglicht es die meisten Graphen weiter zu analysieren oder zusätzliche Funktionen auszuführen. In Abbildung 7.9 sind im oberen Bereich 6 Schaltflächen zu sehen. Die Schaltfläche „+“ ermöglicht das Hinzufügen des Graphen zu einer beliebigen View. Die Schaltflächen „CSV“ und „JSON“ exportieren die angezeigten Daten im jeweiligen Format. „Decompose“ wird lediglich bei aggregierten Graphen angezeigt und zerteilt den aggregierten Graphen in Graphen für jeden Knoten. Um einen Graphen genauer zu analysieren kann die Schaltfläche „Inspect“ verwendet werden. Wie in Abbildung 7.10 zu sehen ist ermöglicht es diese Funktion genaue Zeitpunkte zu betrachten. Dazu kann ein Bereich des Graphen markiert werden, welcher anschließend vergrößert wird. Für das Ein- beziehungsweise Ausblenden von aufgetretenen Events dient die Schaltfläche „Hide/Show Events“.

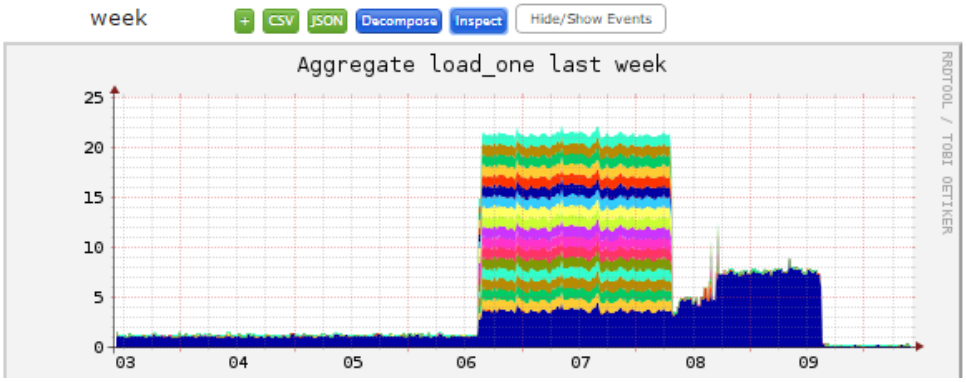


Abbildung 7.9: Anzeige der Schaltflächen oberhalb eines Graphen

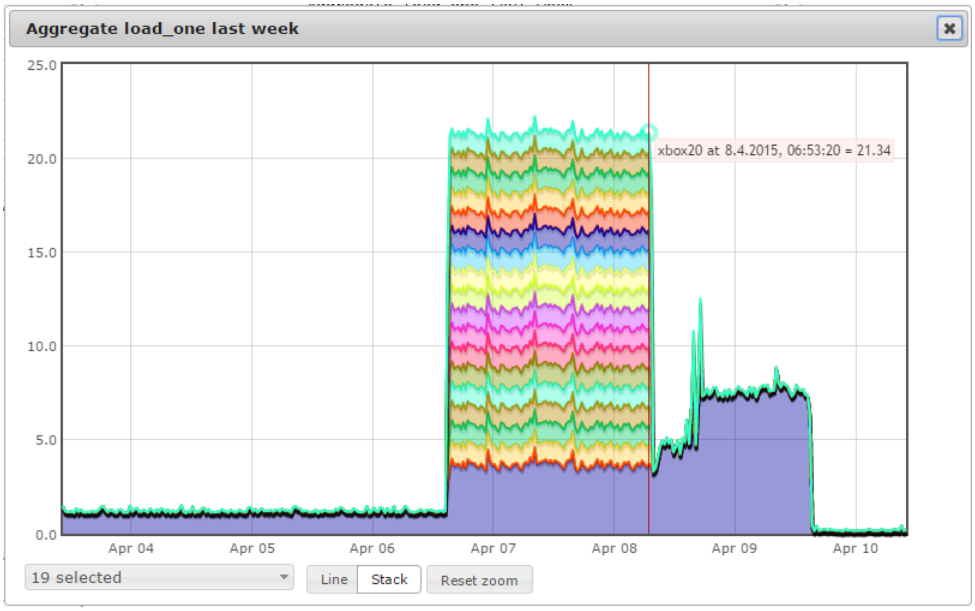


Abbildung 7.10: Inspektion eines aggregierten Graphen

7.2 Simple Linux Utility for Resource Management

Simple Linux Utility for Resource Management (SLURM) ist ein quelloffener Ressourcenmanager für HPCs. Dabei übernimmt SLURM drei wesentliche Aufgaben.

1. SLURM reserviert, optional exklusiv, Ressourcen (einzelne Knoten des Clusters) um es Anwendern zu ermöglichen Berechnungen durchzuführen.

2. SLURM bietet Funktionen zum Starten, Ausführen und Überwachen von Prozessen auf Knoten.

3. SLURM verwaltet auszuführende Berechnungen in einer Warteschlange und führt die Berechnungen zu möglichst idealen Zeitpunkten aus. Sollte eine Berechnung nur sehr wenig Ressourcen des Clusters beanspruchen, kann SLURM mehrere Berechnungen gleichzeitig starten, oder einzelne Berechnungen in der Warteschlange vorreihen.

7.2.1 Funktionsweise von SLURM

In Abbildung 7.11 sehen Sie den Aufbau von SLURM. Der Kontrolldienst „slurmctld“ läuft auf dem XBOX-Master. Die Kommunikation zu den einzelnen Knoten findet über den Dienst „slurmd“ statt, welcher sich auf jedem Knoten befindet. Die aufgelisteten Benutzerkommandos sind ein Teil der unterstützten Befehle von SLURM, welche entweder verwendet werden um generelle Tätigkeiten auszuführen („scontrol“, „sinfo“, „squeue“, „scancel“) oder um konkrete Befehle an die Knoten zu senden („sacct“, „srun“). Eine detailierte Beschreibung der Befehle ist in der Onlinedokumentation zu finden.[41]

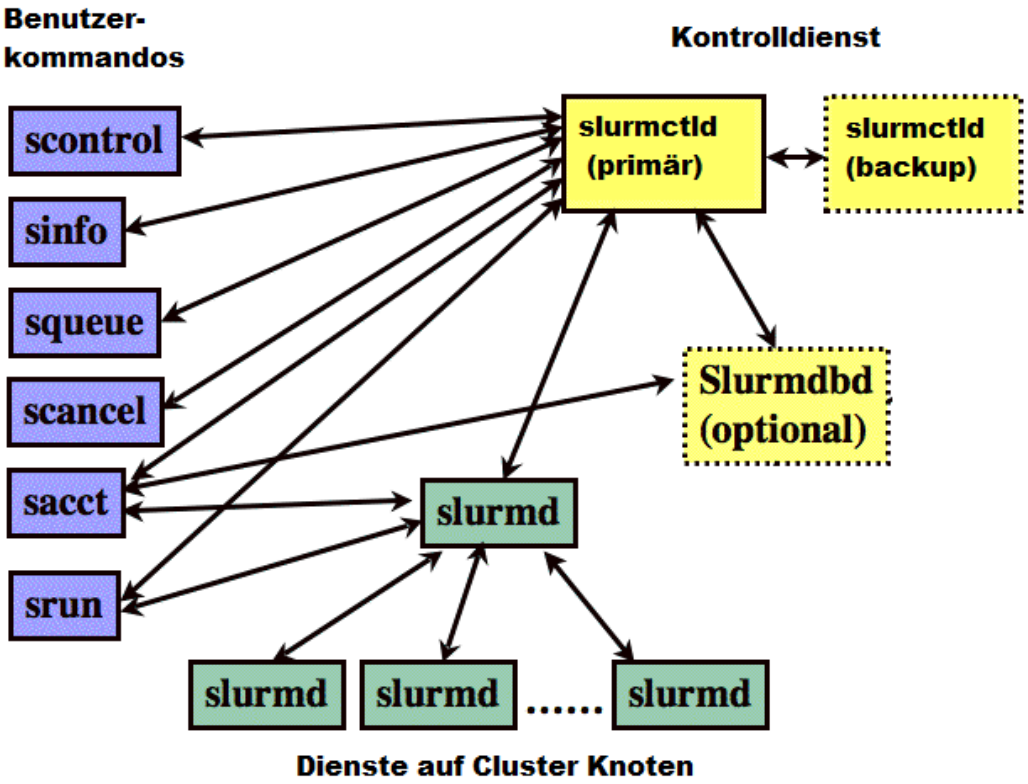


Abbildung 7.11: Funktionsweise von SLURM

7.2.2 Konfiguration von SLURM

Die Konfiguration von SLURM findet in der Date „slurm.conf“ statt, welche in Ausschnitt 7.12 zu sehen ist. Zeile 1 bis 5 zeigt die Konfiguration des Kontrolldienstes „slurmctld“, welcher in Abbildung 7.11 zu sehen ist. Optional kann ein „BackupController“ angegeben werden, sollte die „ControlMachine“ ausfallen. Zeile 10 und 11 definieren jene Knoten, welche von SLURM verwendet werden können. Hier wird der Master-PC sowie die Knoten xbox01 bis xbox20 angelegt. In Zeile 12 ist die Konfiguration einer einzelnen „Partitionen“ zu sehen. Diese besteht aus einer Sammlung von Knoten, wobei ein Knoten in mehreren Partitionen eingetragen werden kann. Bei der Weitergabe von Tasks in die Warteschlange entscheidet SLURM eigenständig welcher Task welcher Partition zugeordnet wird, vorausgesetzt es existieren mehrere Partitionen. Die Konfigurationsdatei lässt sich optional online mithilfe eines Tools konfigurieren[**?**], welches eine genaue Erklärung der einzelnen Parameter bereitstellt.

7.2.3 Verwendung von SLURM

Um Tasks an SLURM zu übergeben gibt es mehrere Möglichkeiten. Der Befehl „sbatch“ ermöglicht es ein Batch-Script, wie es im Codebeispiel 7.2 zu sehen ist, zu übergeben.

Listing 7.1: Konfiguration von SLURM in der Datei „slurm.conf“

```
1  ClusterName=sparky
2  ControlMachine=xbox-master
3  ControlAddr=192.168.1.1
4  #BackupController=
5  #BackupAddr=
6
7  [...]
8
9  # COMPUTE NODES
10 NodeName=xbox-master CPUs=4
11 NodeName=xbox[01-20] CPUs=1 Sockets=1 CoresPerSocket=1 ThreadsPerCore=3
12 PartitionName=sparky Nodes=xbox[01-20],xbox-master Default=YES MaxTime=
       INFINITE State=UP
```

Abbildung 7.12: Konfiguration von SLURM in der Datei „slurm.conf“

Beginnend mit Zeile 2 bis Zeile 5 werden Parameter mithilfe von „sbatch“ gesetzt. Eine Dokumentation des „sbatch“-Befehls ist in der Onlinedokumentation zu finden.[**?**] Zeile 2 definiert die Datei, in der das Ergebnis geschrieben werden soll. Zeile 3 legt fest, dass 4 Tasks verwendet werden sollen und in Zeile 5 wird festgelegt, dass die Ressourcen für maximal eine Minute reserviert bleiben sollen. Der Aufruf des auszuführendes Tasks ist in Zeile 7 zu sehen. Es wird mittels des Befehls „mpirun“ jenes Programm aufgerufen, welches bereits im Kapitel „Perfomancevergleich“ verwendet wurde, um Primzahlen zu berechnen. Sobald der Task abgearbeitet wurde wird auf jedem Knoten die Datei mit dem in Zeile 2 des Codebeispiels 7.2 angegeben Namen erzeugt. Die Ausgabedatei für das gezeigte Batchscript ist in Abbildung 7.3 zu sehen.

Listing 7.2: Mögliches Batch-Script zur Übergabe an SLURM

```
1 #!/bin/sh
2 #SBATCH --output=primesOutput.txt
3 #
4 #SBATCH --ntasks=4
5 #SBATCH --time=01:00
6
7 mpirun  mpiprimes
```

Listing 7.3: Ausgabedatei des Codebeispiels 7.2.

```
1 Using 4 tasks to scan 4000000 numbers
2 Done. Largest prime is 3999971 Total primes 283146
3 Wallclock time elapsed: 5.41 seconds
```

Eine weitere Möglichkeit stellt der Befehl „salloc“ dar. Ein Aufruf welcher dem in Beispiel 7.2 entspricht ist in 7.4 in Zeile 1 zu sehen. Durch die Angabe von „-N 4“ wird

festgelegt, dass der Task auf 4 Knoten laufen soll. Ab Zeile 4 ist die Ausgabe des Befehls zu sehen. Im Unterschied zum Befehl „sbatch“ erfolgt die Ausgabe direkt in der Konsole.

Listing 7.4: Aufruf und Ausgabe des „salloc“-Befehls

```
1 salloc -N 4 mpirun mpiprimes
2
3 #OUTPUT
4 salloc: Granted job allocation 2329
5 Using 4 tasks to scan 4000000 numbers
6 Done. Largest prime is 3999971 Total primes 283146
7 Wallclock time elapsed: 2.73 seconds
8 salloc: Relinquishing job allocation 2329
```

Einer vom Aufruf fast identer Befehl ist der „srun“-Befehl. Der Unterschied besteht darin, dass „srun“ dafür sorgt, dass der Task parallel gleichzeitig auf jedem Knoten ausgeführt wird. Das bereits zuvor verwendete Beispiel des MPI-Aufrufs wird nun vierfach auf vier Knoten ausgeführt, wie die Ausgabe 7.5 zeigt.

Listing 7.5: Aufruf und Ausgabe des „srun“-Befehls

```
1  srun -N 4 mpirun mpiprimes
2
3  #OUTPUT
4  Using 4 tasks to scan 4000000 numbers
5  Using 4 tasks to scan 4000000 numbers
6  Using 4 tasks to scan 4000000 numbers
7  Using 4 tasks to scan 4000000 numbers
8  Done. Largest prime is 3999971 Total primes 283146
9  Wallclock time elapsed: 10.86 seconds
10 Done. Largest prime is 3999971 Total primes 283146
11 Wallclock time elapsed: 10.82 seconds
12 Done. Largest prime is 3999971 Total primes 283146
13 Wallclock time elapsed: 10.78 seconds
14 Done. Largest prime is 3999971 Total primes 283146
15 Wallclock time elapsed: 10.71 seconds
```

7.2.4 Überwachung der SLURM Warteschlange

Für die Überwachung von SLURM wird das Tool „slurmmon“ verwendet. Es generiert eine Weboberfläche, welche eine Übersicht über wartende, laufende und bereits beendete Tasks bietet. Ein Auschnitt davon ist in Abbildung 7.14 zu sehen. Slurmmon bezieht die angezeigten Daten nicht direkt von SLURM, sondern über das bereits beschriebene Überwachungstool Ganglia. Daher finden sich im Tool Ganglia auch die entsprechenden Metriken wie in Abbildung 7.15 zu sehen ist. Um „slurmmon“ verwenden zu können ist es notwendig, in der Konfigurationsdatei anzugeben, unter welcher URL Ganglia erreichbar ist, wie in Zeile 4 der Datei 7.13 zu sehen ist.

Listing 7.6: Konfigurationsdatei „slurmmon.conf“

```
1 {
2         "web_root": "/var/www/html/slurmmon",
3   "probejob_partitions": ["sparky"],
4   "ploturl_gmetaurl": "http://xbox-router:8080/ganglia",
5   "ploturl_cluster": "sparky",
6   "ploturl_host": "xbox-master"
7 }
```

Abbildung 7.13: Konfigurationsdatei „slurmmon.conf“

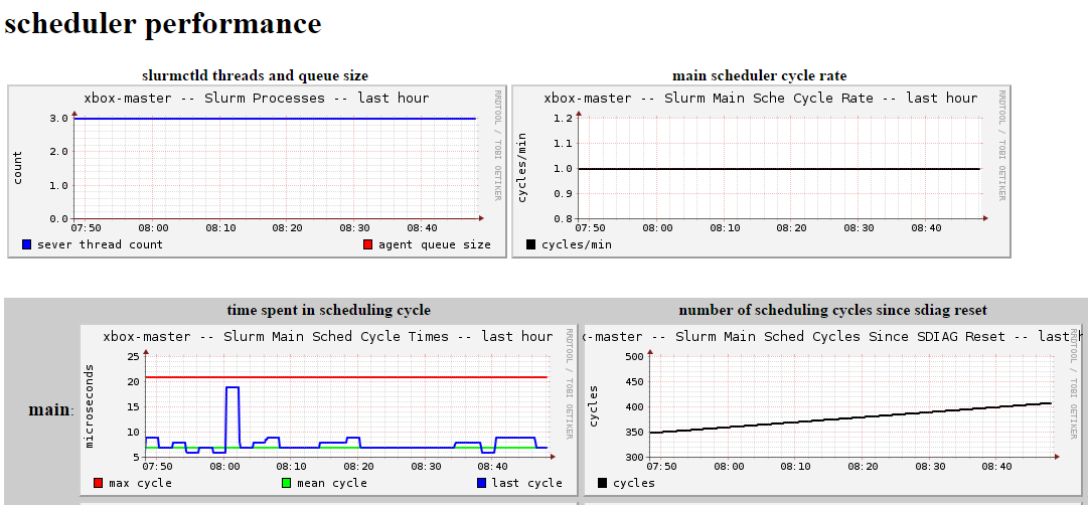


Abbildung 7.14: Ausschnitt aus der „slurmmon“ Weboberfläche

Slurmmon verwendet für das Beziehen der Daten zwei Dienste. Der „slurmmond“ Dienst läuft auf dem Master-PC. Seine Konfiguration bezieht er aus der „slurmmon.conf“ Datei welche in 7.13 zu sehen ist. Zudem läuft auf jedem Knoten der Dienst „slurmmond-computenode“. Dieser wird eingesetzt, um weitere Daten von den Knoten zu beziehen und dient zur Feststellung, wann und mit welchen Ressourcen ein Task tatsächlich auf einem Knoten ausgeführt wird.
Das Starten der zwei Dienste ist in Abbildung 7.7 zu sehen. Durch die Aufrufe in Zeile 4 und 9 wird sichergestellt, dass der Dienst in Zukunft automatisch gestartet wird.

Abbildung 7.15: Ansicht aller „slurmmon“ Metriken im Tool Ganglia

Listing 7.7: Starten der „slurmmond“ Dienste

```
1 //Master-PC
2 service slurmmond start
3
4 chkconfig slurmmond on
5
6 //Knoten
7 service slurmmond-computenode start
8
9 chkconfig slurmmond-computenode on
```

Kapitel 8

Ausblick

Der Verwendung des Clusters sind keine Grenzen gesetzt. Es können eine Vielzahl an Programmen entwickelt werden, welche speziell für die Berechnung auf einem HPC ausgelegt sind. Die Überwachungssoftware, welche in Form einer Webseite realisiert war, bietet Platz für Erweiterungen und Verbesserungen.
Da der Cluster theoretisch mehr Leistung aufbringen kann, als es momentan der Fall ist, würde es eine enorme Verbesserung darstellen Berechnungen auf der GPU ausführen zu können. In der derzeitigen Konfiguration finden sämtliche Berechnungen auf der CPU statt.
Weiteres Potential steckt in der Entschlüsselung von Passwörtern. Durch die Installation einer Software, wie Jack the Riper, welche es ermöglicht, Passwörter zu entschlüsseln, könnten vergleichsweise gute Ergebnisse erzielt werden. Diese Ergebnisse würden durch die Verwendung einer Software zur Erkennung von semantischen Strukturen erneut verbessern. Simon Scheuchenpflug Msc. hat hierzu in seiner Master Arbeit[39] mit dem Titel „Effiziente Passwortsuche durch Analyse von semantischen Strukturen“ ausreichende Beweise geliefert und eine entsprechende Software entwickelt, welche am XBOX-Master lauffähig wäre.

Kapitel 9

Fazit

Nur sehr selten konnte ich für eine Tätigkeit, welche für meine Ausbildung eine zentrale Rolle spielt, so viel Begeisterung aufbringen. Das Entwickeln des Clusters war mit einem hohen Zeitaufwand verbunden, was das Bestreben, eine bestmögliche Arbeit abzuliefern, noch verstärkte. Die Vielfalt an technischen Herausforderungen hat mein Verständnis in vielen Bereichen erweitert und durch die intensive Betreuung war es mir zudem möglich in gänzlich neue Bereiche der Technik Einblicke zu gewinnen.
Trotz der für mich langwierigen und oft anstrengenden Arbeiten bin ich froh, diesen Weg gewählt zu haben. Eine gänzlich neue Erfahrung stellte das Verschriftlichen der getanen Arbeit dar. Zu Beginn schrieb ich mit einem gewissen Widerwillen, der sich jedoch, je länger ich mich mit der Schreibarbeit beschäftigte, in den Ehrgeiz verwandelte die Arbeit bestmöglich zu bewältigen.
Rückblickend betrachtend, würde ich solch eine Herausforderung ohne zu zögern erneut in Angriff nehmen, da das angeeignete Wissen weit über das des Schulalltags hinaus geht.

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[1] Mehdi Achour. What is php? http://php.net/manual/en/intro-whatis.php, 2015.

[2] Werner Almesberger. kboota boot loader based on kexec. https://www.kernel.org/doc/ols/2006/ols2006v1-pages-27-38.pdf, Juli 2006.

[3] Hans-Jrgen petrichPetrich terrafrost Andrea "bantuFischer, monnerat. phpseclib: An introduction. http://phpseclib.sourceforge.net/, 2015.

[4] Axper. Kexec. https://wiki.archlinux.org/index.php/kexec, September 2014.

[5] Lea Beilcke. Sag mir wer du bist oder: wozu code signing? https://www.globalsign.com/de-de/blog/wozu-code-signing/, Juni 2014.

[6] Benno-007. Ssh. http://wiki.ubuntuusers.de/SSH, Januar 2015.

[7] Boehm. Mandelbrot-menge. https://de.wikipedia.org/wiki/Mandelbrot-Menge, M 2015.

[8] Andres "bunnie"Huang. *Hacking the Xbox - An Introduction to Reverse Engineering*. no starch press, 2013. http://bunniefoo.com/nostarch/HackingTheXbox_Free.pdf.

[9] Ele. Ipv4 - internet protocol version 4. http://www.elektronik-kompendium.de/sites/net/0811271.htm.

[10] The Apache Software Foundation. Apache http server project. https://httpd.apache.org/ABOUT$_APACHE.html$, 2015.

[11] FractSurf. Die mandelbrotmenge. http://www.fractsurf.de/index2.html.

[12] Inc. Free Software Foundation. Gcc, the gnu compiler collection. https://gcc.gnu.org/, Februar 2015.

[13] Free60. Free60. http://free60.org/wiki/Main_Page, Februar 2015.

[14] Free60. Libxenon forum. http://libxenon.org/, Februar 2015.

[15] fscholz. Javascript. https://developer.mozilla.org/de/docs/Web/JavaScript, Januar 2015.

[16] funnybox2006. Nand spi flasher(usb variante) schaltplan, teileliste und informationen. http://board.gulli.com/thread/1487306-nand-spi-flasher-usb-variante-schaltplan-teileliste-und-informationen/, Dezember 2009.

[17] GliGli. The reset glitch hack: a new exploit on xbox 360. http://www.logic-sunrise.com/news-341321-the-reset-glitch-hack-a-new-exploit-on-xbox-360-en.html, April 2011.

[18] Harald Hoyer. dracut. https://www.kernel.org/pub/linux/utils/boot/dracut/dracut.html, Oktober 2013.

[19] Gentoo Foundation Inc. Gentoo handbook. http://wiki.gentoo.org/wiki/Handbook Main_Page, Januar 2015.

[20] Gentoo Foundation Inc. Kernel modules. http://wiki.gentoo.org/wiki/Kernel_Modules, Januar 2015.

[21] Shuttle Inc. Shuttle inc. http://global.shuttle.com/main/productsDetail?productId=1587, Februar 2015.

[22] The jQuery Foundation. jquery. https://jquery.com/, 2015.

[23] Justin. What is the linux kernel and what does it do? http://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/, 2010.

[24] Kapslog.de. Apfelmchen fr schler. http://www.kapslog.de/wissen/wp-content/uploads/2010/01/Mandelbrotmenge.pdf, M 2012.

[25] Christof Klausecker; Dieter Kranzlmuller Karl Furlinger. The appletv-cluster: Towards energy efficient parallel computing on consumer electronic devices. http://wwwmnmteam.informatik.uni-muenchen.de/common/projects/ATV2CLUSTER/atvcluster.pdf, April 2011.

[26] Elektronik Kompendium. Mac-adresse (ethernet). http://www.elektronik-kompendium.de/sites/net/1406201.htm.

[27] Pinpoint Labs. What is a hash value? http://pinpointlabs.com/2010/12/what-is-a-hash-value/, Dezember 2010.

[28] Canonical Ltd. Ubuntu. http://www.ubuntu.com/, Februar 2015.

[29] Microsoft. Windows. http://windows.microsoft.com/en-us/windows/home, Februar 2015.

[30] Uwe Tews Monte Ohrt. What is smarty?
http://www.smarty.net/docs/en/what.is.smarty.tpl, 2015.

[31] Dr. Frank Mueller. Sony ps3 cluster (ibm cell be).
http://moss.csc.ncsu.edu/ mueller/cluster/ps3/, Februar 2009.

[32] nightcode. The reset glitch hack - ein umfangreiches tutorial v1.1.0 *08.12.2011*.
http://board.gulli.com/thread/1683082-im-aufbau-the-reset-glitch-hack-ein-umfangreiches-tutorial-v1-0-25-11-2011-/?p=14270422, November
2011.

[33] Junjiro R. Okajima. Aufs3 – advanced multi layered unification filesystem version 3.x.
http://aufs.sourceforge.net/, Januar 2015.

[34] Player. E3: Microsoft stellt komplett neue xbox 360 und kinect-steuerung vor.
http://www.player.de/2010/06/15/e3-microsoft-stellt-komplett-neue-xbox-360-kinect-steuerung-vor-alle-details/, Juni
2010.

[35] The Ganglia Project. Ganglia. http://ganglia.info/, M 2015.

[36] The Open MPI Project. Open mpi: Version 18.4.
http://www.open-mpi.org/software/ompi/v1.8/, Dezember 2014.

[37] Razkar. [en] reset glitch hack tutorial. http://www.logic-sunrise.com/dossiers-et-tutoriaux-341328-en-reset-glitch-hack-tutorial.html, April
2011.

[38] Margaret Rouse. Nand-flashspeicher.
http://www.searchstorage.de/definition/NAND-Flashspeicher, April 2013.

[39] Simon Elias Scheuchenpflug. Effiziente passwortsuche durch analyse von semantischen strukturen. Master's thesis, FH Hagenberg, Mai 2014.

[40] Beyond Linux From Scratch. About initramfs.
http://www.linuxfromscratch.org/blfs/view/svn/postlfs/initramfs.html, Februar 2015.

[41] LLNL (Lawrence Livermore National Security). Slurm@llnl.
https://computing.llnl.gov/linux/slurm/documentation.html, Oktober 2002.

[42] K. Sollins. The tftp protocol (revision 2). https://www.ietf.org/rfc/rfc1350.txt, Juli
1992.

[43] Swizzy. Xell. https://github.com/Free60Project/xell, August 2013.

[44] Swizzy. Libxenon source. https://github.com/Free60Project/libxenon, Juli 2014.

[45] Harald Talk. Dracut. https://dracut.wiki.kernel.org/index.php/Main_Page, Dezember
2014.

[46] Thomas Seiler Joshua Saddler Sebastian Pipping Josournier Tim Yamin, Jimi Ayodele. genkernel. http://wiki.gentoo.org/wiki/Genkernel, Februar 2015.

[47] tuxuser. Setup your pc for libxenon programming. http://www.xboxhacks.de/C14CEE33-089C-4094-B2A4-4433E832DEA7/FinalDownload/DownloadId-4F839A6B20CCC7555602F2315D365B95/C14CEE33-089C-4094-B2A4-4433E832DEA7/index.php?page=Attachment&attachmentID=8829&h=ebb61fbf69fe5956d2399425dc5b13f37fff8ed4, 2011.

[48] Stefan Nehrkorn Uwe Rehfeld. Aspekte der chaostheorie und der fraktalen geometrie. http://www.humboldtgesellschaft.de/inhalt.php?name=chaos, Mai 1995.

[49] Ole Vanhoefer. Swap: Der auslagerungsspeicher. http://www.fibel.org/linux/lfo-0.6.0/node255.html, Oktober 2010.

[50] Jennifer Weston. Nc state engineer creates first academic playstation 3 computing cluster. http://moss.csc.ncsu.edu/ mueller/cluster/ps3/coe.html, Februar 2007.

[51] Whhenyuan. Toolchains. http://elinux.org/Toolchains, November 2014.

[52] Wiki. Grafikprozessor. https://de.wikipedia.org/wiki/Grafikprozessor, Oktober 2014.

[53] Wikipedia. Floating point operations per second. https://de.wikipedia.org/wiki/Floating$_{P}oint_{O}perations_{P}er_{S}econd$, $Jaenner$2015.

[54] Wikipedia. Xbox 360. https://de.wikipedia.org/wiki/Xbox$_{3}$60, $Jaenner$2015.

[55] Jens Zumbrl. Mandelbrot-fraktal. http://www.math.tu-dresden.de/ jzumbr/fractals/fracmand.html, 2014.