Vorhersage von E/A-Leistung im Hochleistungsrechnen unter der Verwendung von neuronalen Netzen

— Bachelorarbeit —

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Vorgelegt von:	Jan Fabian Schmid
E-Mail-Adresse:	2schmid@informatik.uni-hamburg.de
Matrikelnummer:	6440383
Studiengang:	Computing in Science - SP. Physik
Erstgutachter:	Dr. Julian Kunkel
Zweitgutachter:	Prof. Dr. Thomas Ludwig
Betreuer:	Dr. Julian Kunkel

Hamburg, den 17.12.2015

Abstract

Inhaltsverzeichnis

1. Einleitung

Im folgenden wird zunächst kurz dargelegt mit welcher Problemstellung sich diese Thesis befasst, welches Ziel verfolgt wird, und wie dieser Text im Weiteren aufgebaut sein wird.

1.1. Motivation

Hochleistungsrechnen ist in der Wissenschaft ein Thema mit zunehmender Signifikanz, viele komplexere Fragestellungen, insbesondere in den Naturwissenschaften und der Informatik, können z.B. nur in einer effizienten Weise durch Analyse einer Simulation eines Modells gelöst werden. Der extensive Rechenaufwand solcher Simulationen erfordert, dass Wissenschaftler nicht am eigenen Computer rechnen, sondern hierfür die Dienste eines Hochleistungszentrum in Anspruch nehmen. Die Entwicklung der Computer-Hardware in den vergangen Jahrzehnten drängte die Hochleistungszentren dazu für den gewünschten Leistungsgewinn in stark parallelisierte Systeme zu investieren. Sodass, statt einzelner sehr schneller Prozessoren heutzutage viele Tausend Prozessoren vernetzt arbeiten. Diese horizontale Leistungssteigerung am Hochleistungsrechner umgeht die technischen Flaschenhälse, welche die Leistung eines einzelnen Prozessors beschränkt, die zur Verfügung stehende Leistung wird dadurch allerdings schwieriger nutzbar. Einerseits liegt dies am großen technische Aufwand, der zur Vernetzung der Recheneinheiten notwendig ist, andererseits liegt es an der komplexen Programmierung der Software, welche die Parallelität des Rechners berücksichtigt. Insbesondere ist es auch bei der Ein-/Ausgabe (E/A) von erforderlichen Dateien und Ergebnissen des Programms wichtig, dass sie parallel durchgeführt wird. Um die Wissenschaftler beim Programmieren zu unterstützen, gibt es hilfreiche Tools zur Fehlerdiagnostik, Leistungsanalyse, Visualisierung des Programms und der Ergebnisse, sowie zum Parallelisieren von Programmcode. Wünschenswert ist es dabei, wenn diese Tools die Optimierungen möglichst selbstständig durchführen können, sodass der Wissenschaftler sich auf die Funktionalität seines Programms konzentrieren kann, statt sich mit Leistungsoptimierung abzulenken.

1.2. Problemstellung

Im Bereich der Leistungsanalyse stellt sich unter anderem die Problematik der effizienten Ausnutzung der verschiedenen Puffer-Speicher (Caches), wie Arbeitsspeicher, und die direkt auf dem Prozessor liegenden Caches. Autonome Tools zur Optimierung der E/A können versuchen die Dauer von Dateizugriffen vorherzusagen **(warum?)**. Diese Arbeit versucht dies mit dem Hilfsmittel neuronaler Netze zu erreichen.

1.3. Ziele der Thesis

Main goal:
A neural network that is reliable in predicting performance of HPC-IO with sufficient quality

Subgoals:
Knowing which data can be provided by SIOX
Identify interesting (data mining) features, that can be derived from the available data
Having an understanding of what kind of neural network is suitable for the task
A measure for adequate quality of an IO-performance predictor
Benchmark of different predictors using different approaches
Implementation of a predictor module for SIOX for online evaluation of application performances

1.4. Strukturierung

Nachdem in diesem Kapitel die Metaebene der Thesis behandelt wird, soll das zweite Kapitel alle nötigen Hintergrundinformationen zum Verstehen der Thematik und der hier angewandten Ansätze liefern. Das dritte Kapitel beschäftigt sich mit bereits vorhandenen Ansätzen und Arbeiten zur Problemstellung. Im vierten Kapitel werden die Konzepte, sowie die Implementierung, der verschiedenen Modelle beschrieben, die als potenzielle Lösungen des Problems entwickelt und getestet wurden. Die Evaluierung der verschiedenen Lösungsansätze wird daraufhin im fünften Kapitel vorgenommen. Im sechsten Kapitel wird untersucht, welche Lösungsansätze für welche Anwendungsfälle am besten geeignet sind, um dann mit diesen Erkenntnissen im siebten Kapitel ein Fazit der Arbeit zu ziehen.

2. Hintergrund

2.1. Ein-/Ausgabe

Wie werden Dateien abgelegt, wie greift man darauf zu. Wie funktioniert Caching? Read ahead, Write ahead? **Was soll hier so hin?**

Als Ein-/Ausgabe (E/A) bezeichnet man jedweden Austausch von Informationen eines Informationssystems mit der Außenwelt (Wiki). Durch Eingaben erhält der Rechner auszuführende Befehle, die Programme und Funktionen mit denen er etwas ausführen soll, sowie die Daten, die verarbeitet werden sollen. Eine Ausgabe des Rechners gibt dem Nutzer Informationen zum inneren Zustand des Systems und er erhält die Ergebnisse seiner Eingaben. Im Kontext dieser Arbeit handelt es sich bei Ein-/Ausgaben um Dateien, die vom Computersystem von einem Datenträger eingelesen und wieder darauf geschrieben werden. Die in den Testsystemen verwendeten Datenträger sind Festplattenlaufwerke, bei diesen werden Informationen durch magnetische Polarisierung von Speicherzellen auf Magnetscheiben gespeichert und durch Abtastung dieser Magnetisierung mit einem Lesekopf ausgelesen. (**Bild einer Festplatte**) Festplatten sind in Datenblöcke (auch Sektoren) unterteilt, diese bilden die kleinste Einheit, die von dem Medium gelesen bzw. darauf geschrieben werden kann. Durch eine eindeutige Adressierung dieser Sektoren kann der Schreib-/Lesekopf durch Aus- und Einfahren, sowie einer Drehung der Magnetscheibe, direkt auf den gewünschten Datenblock zugreifen. Aufgrund des vergleichsweise geringen Durchsatzes, und insbesondere wegen der großen Latenz bei der Durchführung von Festplattenaufrufen, sind zwischen Festplatte und den tatsächlichen Recheneinheiten im Prozessor mehrere Schichten von Speichern zwischengeschaltet. Von der Festplatte gelesene Daten befinden sich zunächst auf dem Arbeitsspeicher und werden dann in die auf dem Prozessor liegenden Pufferspeicher (Caches) geladen. In den verschiedenen Cache-Ebenen geschieht vergleichbares, die Ebenen gehen von kleinen, sehr schnellen zu größeren, langsameren Speichern über, üblich sind hier zwei oder drei Level. Die Zugriffszeit auf eine Datei ist durch diese Struktur stark davon abhängig in welcher Speicherebene es einen Treffer zu den gesuchten Speicheradressen gibt. Wenn die Datei bereits vollständig im Level 1 Cache liegt, ist sie schon nach wenigen Prozessorzyklen geladen, wenn sie aus dem Arbeitsspeicher geholt werden muss, braucht es mehrere hundert Prozessorzyklen. Das lesen von der Festplatte dauert hingegen einige Millionen Zyklen (**Link? duartes.org**). Im wesentlichen kann also unterschieden werden, ob eine Datei "gecached", sich also im Arbeitsspeicher oder Prozessor-Cache befindet, oder noch von der Fesplatte geladen werden muss. Verschiedene Cache-Strategien erlauben eine noch effizientere Nutzung der Zwischenspeicher

2.2. Hochleistungsrechnen

Wofür HPC? c Was zeichnet HPC aus? c Ein-/Ausgabe im HPC c Besondere herausforderungen

Man spricht von Hochleistungsrechnen, wenn der Rechenaufwand eines Programms außerhalb dessen liegt, was ein einzelner Desktop-Computer in vertretbarer Zeit bearbeiten kann. Die im Hochleistungsrechnen verwendeten Computer werden als Superrechner bezeichnet (wiki Superrechner), hierbei handelt es sich heutzutage üblicherweise um Rechnerverbünde (englisch: Cluster) in denen große Anzahlen Prozessoren und Speichermedien zusammengeschaltet werden. Notwendig wird Hochleistungsrechnen in der Forschung für die Simulation von numerischen Modellen aus verschiedensten Bereichen, beispielsweise zur Mehrkörpersimulation in der Astronomie, für Strömungssimulationen oder zur Berechnung von Klimaprognosen. Wichtige Themen im Hochleistungsrechnen sind die Ausnutzung der zur Verfügung stehenden Leistung, das Erkennen und Beheben von Fehlern des genutzten Programmcodes, die Bereitstellung der Rechen- und Speicherkapazitäten, sowie die Energieeffizienz von Hard- und Software. Um den Superrechner gut ausnutzen zu können ist eine effiziente Ein-/Ausgabe bei vielen Anwendungen von großer Wichtigkeit, da die Menge der anfallenden Datenmengen wesentlich stärker ansteigt, als die Geschwindigkeit der Verbindungen zwischen den verschiedenen Speichermedien und -orten. So steigt beispielsweise die Rechenleistung und die Speicherkapazität beim dem Superrechner des Deutschen Klimarechenzentrum (DKRZ) gegenüber dem Vorgänger wesentlich stärker an, als der mögliche Datendurchsatz (**Verweis**). Die in 2.1 beschriebene Ein-/Ausgabe erweitert sich im Rechnerverbund zur parallelen E/A, dies bedeutet einerseits, dass eine Datei von mehreren Prozessen zeitgleich gelesen und bearbeitet werden kann, und andererseits, dass eine Datei über mehrere Festplatten verteilt sein kann. Diese Parallelität hat einen wesentlichen Einfluss auf die Aufgabe der E/A-Leistungsvorhersage, denn statt nur den Aufwand der Arbeitsschritte auf einer einzelnen Festplatte abzuschätzen, müssen hier die verstrickten Zusammenhänge zwischen Netzwerken von Festplatten und Rechnern, den jeweiligen Auslastungen der Komponenten, sowie Priorisierungen bestimmter Aufgaben und Instanzen.
Dies woanders hin? Die Erfassung dieser Informationen wäre sehr aufwendig, sodass dies zur Zeit nicht möglich ist. Die Genauigkeit einer Vorhersage von E/A-Leistung eines parallelen Systems sollte daher systematisch ungenauer sein, als die zu einem seriellen System.

2.3. Ein-Ausgabe Optimierung mit SIOX

Ziele Vorgehen Wofür diese Arbeit?

[KZB15] The SIOX framework [13,14] aims to become a holistic approach covering the full cycle of monitoring, analysis, machine learning of the adequate settings and their automatic enactment.

2.4. Maschinelles Lernen

Wozu c Verschiedene Klassen, un -/ supervised c kmeans regresson trees neuronale netze -> verweis Bagging

Maschinelles Lernen gehört in die Bereiche künstliche Intelligenz und automatisierte Wissensgenerierung. Verfahren dieser Disziplin versuchen durch intelligentes lernen von Mustern Vorhersagen und Entscheidungen, wie die Zuordnung zu einer bestimmten Klasse, zu treffen. Intelligent bedeutet hierbei (beispielweise), dass vorgegebene Informationen nicht schlicht auswendig gelernt und wiedergegeben werden, sondern von diesen Informationen abstrahiert wird, um Gesetzmäßigkeiten innerhalb der Trainingsdaten zu erkennen. Zu unterscheiden sind Verfahren des überwachten und des unüberwachten Lernens. Während beim überwachten Lernen die idealen Ausgaben zu den Eingabedaten vorgegeben werden, wird beim unüberwachten Lernen kein bestimmtes Ergebnis erwartet, stattdessen muss der Algorithmus versuchen den Informationen inhärente Abhängigkeiten und Zusammenhänge zu erkennen. Die Trainingsdaten sind die Informationen, die dem Algorithmus bekannt sind, von diesen versucht er zu lernen. Falls ein Testdatensatz vorliegt, kann damit anschließend die berechnete Ausgabe zu bisher unbekannten Daten bewertet werden. Sowohl Trainingsdaten, sowie Testdaten enthalten gemessene Werte zu den Attributen, von denen gelernt werden soll (Eingabewerte). Ein Attribut ist eine messbare Größe des Systems, das untersucht wird. Dies könnte beispielsweise bei einem Datensatz über Blumen die Farbe der Blütenblätter sein. Ein Datenpunkt beschreibt eine Instanz des untersuchten Systems (z.B. ein exemplar der Blume), dazu enthält er einen gemessenen Wert zu jedem Attribut. Die Datenpunkte der Trainingsdaten beim überwachten Lernen, enthalen auch die Lösungen"(die gesuchten Ausgabewerte) zu den Attributen, dessen Werte der Algorithmus vorhersagen soll. Diese Informationen werden dann beim Test vorenthalten, sodass durch den Vergleich zwischen tatsächlicher und vorhergesagter Lösung Rückschlüsse auf die Qualität der Vorhersagen gezogen werden können. Zur Anwendung eines Verfahrens des maschinellen Lernens müssen zunächst Kriterien bzw. Leistungsmetriken eingeführt werden, anhand derer die Qualität der Vorhersagen und Entscheidungen gemessen werden können. Einerseits zum Vergleich verschiedener Ansätze, aber insbesondere für den Lernprozess des Algorithmus selbst, damit dieser sozusagen aus seinen Fehlern und Erfolgen lernen kann. Einfache Metriken wären beispielsweise für einen Klassifizierungsalgorithmus der Anteil falsch zugeordneter Datenpunkte. Oft is es notwendig die zur Verfügung stehenden Daten aufzubereiten, bevor ein maschineller Lernalgorithmus effizient und korrekt Informationen aus diesen ableiten kann. So weist Alpaydin [Alp10] (Seite 13-15) auf fehlerhafte Daten hin, die durch zufälligen Messfehler oder eine systematisch inkorrekte Messung entstanden sind. Zudem schreibt er, dass Teile der Daten überflüssig sein können, da sie redundant sind oder keine relevanten Informationen enthalten. Problematisch sind auch widersprüchliche (engl. inconsistent) Datenpunkte, hierbei stellen mehrere Datenpunkte dem maschinellen Entscheider die selben Eingabewerte zur Verfügung, sodass dieser sie nicht unterscheiden kann, sie haben jedoch unterschiedliche Ausgabewerte [Alp10] (Seite 14). Mit all diesen Problemen muss, unter Beachtung der Eigenschaften des vorliegenden Datensatzes, bei der Aufbereiung der Daten sinnvoll umgegangen werden. So können Ausreißer bei den

Daten aussortiert werden, da diese eventuell durch eine Fehlmessung entstanden sind. Widersprüchliche Datenpunkte können zusammengefasst werden, indem sie zusammen einen neuen Datenpunkt mit eindeutigen Ausgabewerten bilden (dies könnten die Mittelwerte sein).
Unüberwachtes Lernen findet bei der Clusteranalyse statt, Kantardzic beschreibt Clusteranalyse folgendermaßen: Cluster analysis is the formal study of methods and algorithms for natural grouping, or clustering, of objects according to measured or perceived intrinsic characteristics or similarities."[Kan11] (Seite 250). Ein einfaches und anschauliches Beispiel sind Punkte im zweidimesionnalen Raum, die hinsichtlich ihrer Position gruppiert werden. (**BILDER SELBER MACHEN HIERZU**) Ein einfacher Clustering-Algorithmus ist der k-Means-Algorithmus. Bei diesem muss zunächst die Anzahl Cluster k festgelegt werden. Die k Mittelwerte der Cluster mit zufälligen Werten initialisiert. Dann wird jeder Datenpunkt dem Cluster zugeordnet, dessen Mittelwert seinem am dichtesten ist. Danach werden die Mittelwerte der Cluster anhand der Ihnen zugeordneten Punkte mit einer beliebigen Metrik berechnet und gesetzt. Das Neuzuordnen der Punkte und Berechnen der Mittelwerte wird nun solange wiederholt, bis sich keine Änderung der Zuordnung mehr ergibt. Als Endergebnis eines Cluster-Algorithmus erhält man eine Gruppierung der Datenpunkte in Mengen mit möglichst niedriger Varianz innerhalb der Menge und möglichst großer Varianz zwischen den Mengen.

2.5. Künstliche Neuronale Netze

Data Mining kurz allg. zu lösende Problemklasse lineare serperabilität, -> mächtigkeit von NN RNN Vor- und Nachteile

Bei künstlichen neuronalen Netzen, im Folgenden oft nur neuronale Netze genannt, handelt es sich um eine Methode aus dem Bereich des maschinellen Lernens zum Approximieren einer unbekannten Funktion, die der Relation zwischen zwei Datenmengen zugrunde liegt. Die Methode ist inspiriert von biologischen neuronalen Netzen, wie sie im Gehirn vorkommen. Dafür verwenden sie einen statistischen Ansatz, ihre Lösung für das Problem wird zunächst zufällig im Lösungsraum angelegt und dann mit Hilfe eines Gradientenverfahrens optimiert.
Rojas [Roj96] vergleicht neuronale Netze mit einer Black Box, also einem System mit beobachtbarer Ein- und Ausgabe, aber unbekannter innerer Verarbeitung der Informationen. Zur Verwendung eines neuronalen Netzes gibt man dem Netz eine Menge von Eingabevektoren $E \in \mathbb{R}^n$ mit jeweils zugehörigem Ausgabevektor $A \in \mathbb{R}^m$ vor und dieses versucht eine passende Funktion $F : \mathbb{R}^n \to \mathbb{R}^m$ zu finden. Dementsprechend handelt es sich hierbei um überwachtes Lernen, da eine gewünschte ideale Ausgabe vorgegeben wird.
Ein feedforward-Netz neuronales Netz besteht aus einer Eingabeschicht, einer beliebigen Anzahl verborgener Schichten und einer Ausgabeschicht (siehe 2.1), wobei die Verbindungen aus jeder Schicht jeweils nur in die nächsthöhere Schicht gehen. Die Eingabeschicht besteht meiner vorherigen Definition entsprechend aus n Stellen, an denen die Werte

eines Eingabevektors stehen. Jeder Eingabewert wird dann an jedes Neuron in der ersten verborgenen Schicht weitergegeben und dort verrechnet. Die Ergebnisse der Neuronen der verborgenen Schicht werden dann an die nächste Schicht gegeben und so weiter, bis die Ergebnisse der Ausgabeschicht als Ausgabevektor interpretiert werden.
Rekurrente Netze haben die gleiche Struktur, doch es können auch Verbindungen zu zurückliegenden Schichten vorkommen, dadurch ist die Berechnung mehr deterministisch bestimmt. Es müssen Berechnungs- und Determinierungsregeln festgelegt werden. Ein

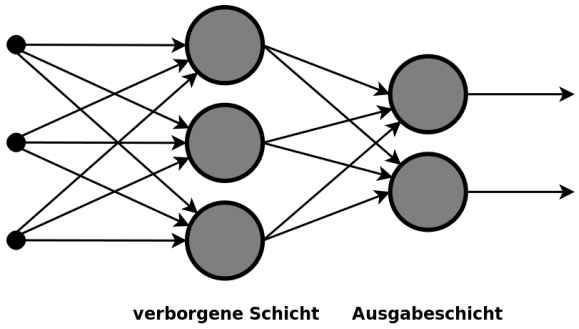

Abbildung 2.1.: Zweischichtiges Netz (wiki)

jedes Neuron rechnet die hineinkommenden Eingabewerte mit einer zur Übertragungskante zugehörigen Gewichtung mit einer Übertragungsfunktion zusammen, die Anzahl der Eingaben ist hierbei unbegrenzt (ünlimited fan-in property"[Roj96]). Die Übertragungsfunktion kann hierbei schlicht die Summe aller gewichteten Eingaben sein. Der errechnete Wert wird als Netzeingabe an die Aktivierungsfunktion gegeben.

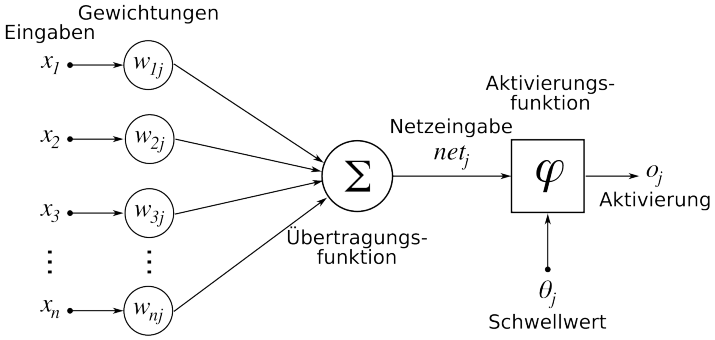

Abbildung 2.2.: Schema eines künstlichen Neurons (wiki)

Die Aktivierungsfunktion berechnet eventuell mit einem Schwellwert die Aktivierung des Neurons, welche dann an alle verbundenen Neuronen der nächsten Schicht gegeben wird. Die Aktivierungsfunktion kann beispielsweise eine simple Stufenfunktion sein, die

allen Netzeingaben kleiner des Schwellwertes eine Null und allen Eingaben größer gleich des Schwellwertes eine Eins zuweist.
Ein Neuron mit der gewichteten Summe aller Eingaben als Übertragungsfunktion und einer Stufenfunktion mit Schwellwert als Aktivierungsfunktion wird von Rojas Perzeptron bezeichnet [Roj96] (Seite 60). Ein feedforward-Netzwerk aus Perzeptrons, in dem jedes Neuron mit allen Neuronen der folgenden Schicht verbunden ist, wird als multilayer perceptron (kurz MLP) bezeichnet. Ein neuronales Netz lernt die Abbildung zwischen den vorgegebenen Paaren von Eingabe- und Ausgabedaten durch Anpassung der Gewichte an den Kanten, nachdem es mit zufällig initialisierten Kantengewichten gestartet ist. Diese Anpassung geschieht beim MLP durch Fehlerrückführung (engl. backpropagation). Dabei wird der mittlere quadratische Fehler der berechneten Ausgabe gegenüber der vorgegebenen Ausgabe ermittelt und daraufhin unter Rücksichtnahme auf eine Lernrate mit Hilfe des Gradientenverfahren minimiert. Vor der Anwendung eines künstlichen neuronalen Netzes für ein Problem stellt sich die Frage, ob dieses für das Problem geeignet ist. Dazu gibt es einige interessante mathematische Beweise. Ein einzelnes Perzeptron kann alle linear separierbaren logischen Funktionen exakt approximieren [Roj96] (Seite 62-63). Wobei lineare Separierbarkeit definiert ist als: Zwei Mengen von Pukten A und B .. (auf deutsch?)

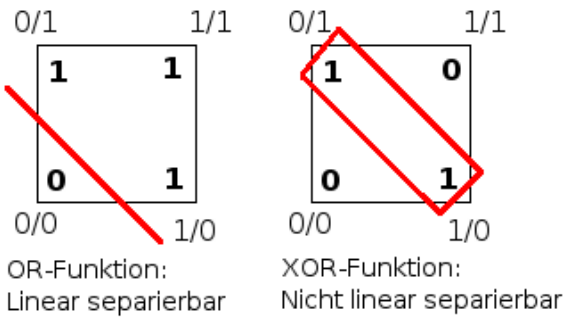

Abbildung 2.3.: Lineare Separierbarkeit von Funktionen (wiki)

Diese Beschränkung gilt allerdings nicht für ein Netzwerk von Neuronen. Bereits ein zweilagiges Netzwerk aus noch simpleren McCulloch-Pitts-Zellen kann jede beliebige logische Funktion berechnen [Roj96] (Seite 37). Für MLPs hat Cybenko [Cyb89] bewiesen, dass sie belieblige kontinuierliche Funktionen auf einer kompakten Teilmenge des euklidischen Raums $\mathbb{R}^n$ approximieren kann. Der entsprechende Satz hierzu ist das üniversal approximation theorem".

Zusammenfassung: *2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.*

3. Verwandte Arbeiten

Um weniger Umschreiben zu müssen führe ich zunächst zwei Kategorien von Lösungsansätzen ein. Danach erwähne ich kurz einige Arbeiten, dessen Themen sich mit dem dieser Arbeit überschneiden, und danach gehe ich noch etwas detailierter auf Veröffentlichungen mit hoher Ähnlichkeit und Relevanz ein. Mini-Gliederung was jetzt kommt. Einleitung ist OPTIONAL!!!! In Sektion X steht, in Y steht.

3.1. Leistungsvorhersage von Ein-/Ausgabe

3.2. Leistungsvorhersage mit Maschinellem Lernen

Artificial neural networks for modelling and control of non-linear systems `https://books.google.de/books?hl=de&lr=&id=tmTTBwAAQBAJ&oi=fnd&pg=PR9&dq=%22neural+networks%22+storage&ots=KyYA14xY1w&sig=yAQG0zn41xAHccDNaUWYd3L_ywI`

3.3. Neuronale Netzen im Hochleistungsrechnen

Viele Projekte verwenden Neuronale Netze für die Modellierung vom Gehirn.... http://world-comp.org/p2012/PDP3003.pdf

Machine Learning for Machines: Data-Driven Performance Tuning at Runtime Using Sparse Coding SJ Tarsa - 2015 - dash.harvard.edu ... 83 5.2 Workload Models and Data Collection . . . . . 85 5.3 Predicting Storage Performance with CART . . . . . 87 5.3.1 Applying CART to Multi-Tenant Scenarios . . . . . 87 5.3.2 Variation-Tolerant CART Using Workload Labels . . . .

Predicting disk drive failure at a central processing facility using an evolving disk drive failure prediction algorithm

ABER sie werden nur unzureichend eingesetzt für I/O und deswegen braucht es die Arbeit.

10-20 andere literatur

3.4. Kategorisierung (woanders hin?)

Das Problem, die Zugriffszeiten auf Festplatten vorherzusagen, kann im wesentlichen durch zwei verschiedene Ansätze gelöst werden.
Zum einen kann man versuchen ein Modell des Festplattensystems zu erstellen, indem

Hardwaredetails, wie die Rotationsgeschwindigkeit der Platte, die Reaktionszeit und Geschwindkeit des Lesekopfs, sowie das Zusammenspiel der Komponenten, bekannt sind oder entsprechende Parameter werden durch gezielte Untersuchung approximiert. (**stimmt das? Referenz?**) Mit diesem möglichst exakten Modell können dann Zugriffe simuliert und so deren Aufwand vorhergesagt werden.
Der zweite Ansatz ist auch der dieser Arbeit, es wird von dem eigentlichen Festplattensystem abstrahiert und stattdessen ein mathematisches Modell gesucht, das aus gemessenen Leistungswerten von Festplattenzugriffen die Werte von neuen Zugriffen ableitet. Ich unterscheide daher im Folgenden zwischen dem Modellierungssansatz (in der Fachliteratur etwa als analytic device modeling und simulation modeling bezeichnet) , bei dem versucht wird das Festplattensystem nachzubilden und dem Interpolationsansatz, bei dem ein numerisches Modell entwickelt wird. Da hier ohne Wissen über die inneren Zustände des Systems modelliert wird, wird dieser Ansatz auch black-box modeling bezeichnet. (Quelle? Fourier assisted ml)

3.5. Modellierungssansatz gegenüber Regressionsansatz

Die Nachteile von Modellen mit Modellierungssansatz liegen insbesondere daran, dass sie aufwendig zu konfigurieren sind İn fact, one of us (Oldfield) spent several months configuring DiskSim to model an existing device"[CMW+13] (S.1) und naturgemäß schnell veraltern, da sie jeweils an spezielle Hardware angepasst sind. Der Vorteil dagegen ist, dass sie bei korrekter Konfiguration sehr präzise sind, wie z.B. hier gezeigt [RW94].
Der Interpolationsansatz ist in der Anwendung einfacher und flexibler, da es sich automatisch an das System anpasst. Dafür erwartet man aufgrund der fehlenden analytischen Einsicht ins System eine etwas schlechtere Präzision.
Für die Anwendung im HPC-Bereich spielt der analytische Ansatz eine untergeordnete Rolle, da hier unterschiedliche Festplattensysteme zusammenarbeiten und stark mit der Netzwerkarchitektur verstrickt sind, sodass eine entsprechende Analyse des Systems zu aufwendig wird. Furthermore, [...] building an accurate model or simulator using white box method cannot be a genereal solution in serving a variety of very different workloads"[ZLZ+10] (S.2, Zeile 20-24).

3.6. xxx

Im HPC Bereich ist die Leistungsanalyse generell ein wichtiger Punkt, so wird beispielsweise das Scheduling-Algorithmen vom Dateisystem simuliert [LFC+], hier wird DiskSim [BSSG08] zur Vorhersage der Festplattenzugriffszeiten genutzt, dabei nutzt DiskSim einen Modellierungssansatz (analytical simulation). Eine weitere Arbeit, in der ein Modellierungsansatz genutzt wird stammt von Lebrecht et al. [LDK09].
Bei Arbeiten, in denen ein Interpolationsansatz genutzt wird, werden verschiedene Data-Mining bzw. stochastischen Methoden angewandt, beispielsweise eine Kombination aus regression trees und support vector mashines [DLZC12], bagging classification und re-

gression trees [ZLZ+10]. Verschiedene statistische Methoden werden von Kelly et al. untersucht [KCGK04].

3.7. Fourier-Assisted Machine Learning

Adam Crume et al. [CMW+13] gehen davon aus, dass der entscheidende Faktor bei der Vorhersage von Zugriffszeiten in der Erkennung von periodischen Mustern liegt. Diese Annahme ist für eine einzelne Festplatte gerechtfertigt, da man die Zugriffstzeit grob in zwei Teile aufteilen kann, zum einen die Bewegung des Lesekopfs auf die richtige Spur und die Bewegung des Kopfes entlang der Spur zum richtigen Punkt, auch wenn diese beiden Bewegungen in der Realität überlappen. Jede Festplattenspur hat entsprechend des Radius eine andere Periodendauer.
Durch eine Fourier Analyse finden sie die Hauptfrequenzen heraus und können diese dann nutzen, um mit einem neuronalen Netz Vorhersagen zu treffen. Eine Schwierigkeit in dieser Arbeit ist unter anderem, dass durch die große Anzahl Perioden nur ein kleiner Ausschnitt der Festplatte in seiner Gesamtheit, also alle möglichen Paare von Ausgangs- und Endspuren, untersucht werden kann. Die Priorität solcher Frequenzen für die exakte Leistungsvorhersage, ist im komplexen HPC-E/A-System zunächst einmal nicht zu erwarten, müsste allerdings untersucht werden.

3.8. Predicting Performance of Non-Contiguos I/O with Machine Learning

Direkt aus dem Bereich des Hochleistungsrechnens stammt die Arbeit von Kunkel et. al [KZB15]. Hier wird versucht mit Hilfe von decision trees die performantesten Parameter für nicht zusammenhängende Zugriffe auf Dateien durch ROMIO, einer Implementierung von MPI-2 I/O, zu finden. Dabei sagen sie mit den decision trees die Performance für die verschiedenen Parameter auf den Daten voraus, sodass sie anhand dieser Vorhersagen die besten Parameter finden. (**habe ich das richtig verstanden**?) Mit dieser Methode haben sie es geschafft zufriedenstellende Ergebnisse zu erzielen. Die Simplizität von decision trees, beispielsweise können diese nur Entscheidungen als eine Aneinanderreihung von linearen Separationen des Werteraums treffen, lässt vermuten, dass mit Hilfe von komplexeren Data Mining - Methoden, wie neuronalen Netzen noch bessere Ergebnisse erzielt werden könnten. Hier findet sich ein potenzielles Anwendungsgebiet der Ergebnisse dieser Bachelorarbeit.

Zusammenfassung: *2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.*

4. Gestaltung der Analyse

test

4.1. Modell der Ein-/Ausgabe

Zeit zum Bewegen des Lesekopfes + Zeit zum rotieren + lese-/schreibzeit + Aufrufprozedur mit festem Beitrag; t = tNetwork + tHDD +tMem
mit tHDD(deltaOffset, sizeInBlocks) = tBus(size) + f(deltaOffset)
track-to-track-seek-time, rotational time Welche Abhängigkeiten gibt es, oder könnte es geben? Z.B. Umso größer die Size, desto länger die Lese-/Schreibzeit, umso größer Offset, desto größer die Seekzeit

4.2. Leistungs- und Ausreißervorhersage

Die Beiden Dinge, die ich versuche vorherzusagen. Warum? Unterschiede. **Macht es Sinn zu versuchen Ausreißer vorherzusagen?**

4.3. Attribut Selektion und Generierung

Korrelationen untersuchen, neues Attribute generieren (wesentlich für die verschiedenen Modelle)

4.4. Modellklassen

Zeitreihenanalyse, Aggregierung, Hybride

4.5. Leistungsmetriken

Welche Leistungsmerkmale werden warum untersucht?

Zusammenfassung: *2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.*

5. Implementierung

test

5.1. Untersuchte Modelle

jeweils Beschreiben, wie die Daten hierfür vorbereitet werden mussten, was die Idee davon ist (warum macht es Sinn, dies zu testen), (wie soll die Performance hier untersucht werden?, Design?)

5.2. Bagging zur Leistungssteigerung

Konzept und Anwendung

5.3. Anwendung? und Parameter

Wie wird mit Ausreißern umgegangen, wie werden sie definiert, Cross-Validation, Anzahl berechneter Netze pro Modell, threshold, Größe der Netze, lern- und fehler-funktionen,

5.4. Verwendete Programmiersprache und Bibliotheken

5.5. Testsystem, Mistral

Speziell die Hardware von Mistral, von hier kommen die Benchmarks Zur Evaluation verschieben: Testsystem

Zusammenfassung: *2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.*

6. Evaluierung

test

6.1. Leistungsvorhersage

Zu jedem Modell schreiben, welche Attribute und Parameter sich als geeignet herausgestellt haben. Ergebnisse anhand von Graphen veranschaulichen.
Ergebnisse für verschiedene Fälle vergleichen, seq,rnd,mistral vs pc?
Jeweils schreiben, ob bagging etwas bringt.

6.1.1. Aggregationsmodelle

6.1.2. Zeitreihenmodelle

6.1.3. Hybridemodelle

6.2. Ausreißervorhersage

6.2.1. Aggregationsmodelle

6.2.2. Zeitreihenmodelle

6.2.3. Hybridemodelle

Zusammenfassung: *2-5 Sätze, BLA In diesem Kapitel hab ich gesehen BLA und jetzt sehen wir Z. Wie hängen die Sections dieses Kapitels zusammen und warum brachte es was das zu lesen.*

7. Fazit

Welche Modelle sind für welchen Fall erfolgsversprechend? Eignen sich Neuronale Netze zum Vorhersagen von E/A-Leistung im HPC?
Wo könnten die Ergebnisse eingesetzt werden?
Was müsste als nächstes getan werden?

Literaturverzeichnis

[Alp10] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[BSSG08] John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. The disksim simulation environment version 4.0 reference manual, 2008.

[CMW+13] Adam Crume, Carlos Maltzahn, Lee Ward, Thomas Kroeger, Matthew Curry, and Ron Oldfield. Fourier-assisted machine learning of hard disk drive access time models. In *Proceedings of the 8th Parallel Data Storage Workshop*, PDSW '13, pages 45–51, New York, NY, USA, 2013. ACM.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[DLZC12] Chengjun Dai, Guiquan Liu, Lei Zhang, and Enhong Chen. Storage device performance prediction with hybrid regression models. In *Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, PDCAT '12, pages 556–559, Washington, DC, USA, 2012. IEEE Computer Society.

[Kan11] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[KCGK04] Terence Kelly, Ira Cohen, Moises Goldszmidt, and Kimberly Keeton. Inducing models of black-box storage arrays. Technical report, 2004.

[KZB15] Julian Kunkel, Michaela Zimmer, and Eugen Betke. Using Machine Learning to Predict the Performance of Non-Contiguous I/O, 07 2015.

[LDK09] AS Lebrecht, NJ Dingle, and WJ Knottenbelt. A performance model of zoned disk drives with i/o request reordering. pages 97–106. IEEE COMPUTER SOC, 2009.

[LFC+] Yonggang Liu, Renato Figueiredo, Dulcardo Clavijo, Yiqi Xu, and Ming Zhao. Towards simulation of parallel file system scheduling algorithms with.

[Roj96] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[RW94] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27:17–28, 1994.

[ZLZ+10] Lei Zhang, Guiquan Liu, Xuechen Zhang, Song Jiang, and Enhong Chen. Storage device performance prediction with selective bagging classification and regression tree. In *Network and Parallel Computing, IFIP International Conference, NPC 2010, Zhengzhou, China, September 13-15, 2010. Proceedings*, pages 121–133, 2010.

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Anhänge

A. Anhangskapitel

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen, als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internetquellen – benutzt habe, die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Optional: Ich bin mit der Einstellung der Bachelor-Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik einverstanden.

Hamburg, den 01.01.2012 .....