Universität Hamburg
Arbeitsbereich Wissenschaftliches Rechnen

Masterarbeit
im Studiengang Informatik

Automatisches Lernen der Leistungscharakteristika von Paralleler Ein-/Ausgabe

---

Vorgelegt von: Eugen Betke
Matrikelnummer: 5799981
E-Mail-Adresse: eugen.betke@gmail.com
Erstgutachter: Prof. Dr. Thomas Ludwig
Zweitgutachter: Dr. Julian Kunkel
Betreuer: Dr. Julian Kunkel

27. Juli 2015

Inhaltsverzeichnis

Kurzfassung

Die Leistungsanalyse und -optimierung sind seit dem Beginn der elektronischen Datenverarbeitung notwendige Schritte in den Qualitätssicherungs- und Optimierungszyklen. Sie helfen eine qualitative und performante Software zu erstellen. Insbesondere im HPC-Bereich ist dieses Thema wegen der steigender Softwarekomplexität sehr aktuell.

Die Leistungsanalysewerkzeuge helfen den Prozess wesentlich zu vereinfachen und zu beschleunigen. Sie stellen die Vorgänge verständlich dar und liefern Hinweise auf mögliche Verbesserungen. Deren Weiterentwicklung und Entwicklung neuer Verfahren ist deshalb essentiell für diesen Bereich.

Das Ziel dieser Arbeit ist zu untersuchen, ob E/A-Operationen mit Hilfe von maschinellen Lernen automatisch den richten Cachetypen zugeornet werden können. Zu diesem Zweck werden Methoden entwickelt, die auf den CART-Entscheidungsbäumen und kMeans-Algorithmen basieren und untersucht.

Die erhofften Ergebnisse wurden auf diese Weise nicht erreicht. Deswegen werden zum Schluss die Ursachen indentifiziert und diskutiert.

Abstract

Since the beginning of electronic data processing the performance analysis and optimization are essential steps in the quality assurance and optimization. They help to develop qualitative and performant software. Especially in the HPC area, by reason of increasing complexity of software, is this a topical theme.

The performance analysis tools make this process easier and faster. They present the information about I/0 in a human understandable way and often provide hints for improvements. Therefore it is essential to develop software analysis tools to make the software better.

The goal of this thesis is to invetigate the possibility to characterize the I/O operations by means of mashine learning algorithms and map them to appropiate cache type. For this purpose two methods are introduced, that are based on the CART decision trees and the kMeans clustering algorithm.

The desired results couldn't be achieved in this way. Therefore problems will be identified and discussed at the end.

Kapitel 1

1 Einleitung

In den letzten Jahren nahm die Leistungsfähigkeit der Hardware von Computersystemen zu: es erfolgte ein exponentielles Kapazitätswachstum des Massenspeichers, begleitet von einer vergleichsweise langsamen Zunahme der Datenübertragungsgeschwindigkeiten, die Integrierung mehrerer Kerne in die Prozessoren und ein explosionsartiges Wachstum von Datenmengen.

Früher war die Erhöhung der Taktfrequenz die Hauptmethode die Prozessorleistung zu steigern. Nachdem aber die physikalischen Grenzen der Halbleiterkomponenten erreicht wurden und die Erhöhung der Taktfrequenz nicht mehr möglich war, ist man dazu übergegangen Prozessoren mit mehreren Kernen auszustatten. Der Trend hält bis heute an und heutzutage sind in den meisten modernen Rechner Multikern-Prozessoren verbaut.

Die Fortschritte in der Massenspeicherentwicklung der letzten Jahre haben vor allem zur höhere Datendichte geführt, was die Herstellung von Geräten mit hoher Kapazität ermöglicht hat. Das Kapazitätswachstum, wie im Moore'schen Gesetz bereits formuliert, hat sich etwa alle anderthalb bis zwei Jahre verdoppelt bei gleichzeitig sinkendem Preis pro Speichereinheit.

Laut dem Bericht der IDC-Studie [1] verdoppelt sich auch das weltweite Datenvolumen alle zwei Jahre. Im Jahr 2013 wurde das Gesamtdatenvolumen auf etwas 4.4 Zetabyte geschätzt. Im Jahr 2020 soll dieser Wert auf 44 Zetabyte steigen. Die Datenproduktion steigt mit der wachsender Anzahl an Geräten wie digitale Kameras und es gibt immer mehr datenbasierte Dienste, wie Online-Data-Clouds.

In der Wirtschaft gibt es zwei große Bereiche, die auf hohe E/A-Leistung angewiesen sind. Das sind OLAP (Online Analytical Procesing) und OLTP (Online Transaction Processing). Im OLAP-Bereich nutzt man relativ große Datenbestände für die Informationsgewinnung. Die gewonnen Informationen sollen den Führungskräften

und Managern bei den Entscheidungen helfen und deshalb liegt hier der Fokus auf kurzen Reaktionszeiten. Im OLAP-Bereich hingegen geht es um die Datenspeicherung. Das Ziel ist die Daten im konsistenten Zustand zu behalten und die Transaktionenzahl zu maximieren. Diese beiden Bereiche sind wahrscheinlich von dem Problem des Datenwachstums am schlimmsten betroffen.

Es gibt zahlreiche weitere Bereiche, die nicht weniger stark vom explosionsartigen Datenwachstum betroffen sind, z.B. die Verarbeitung der Messdaten in der Klimaforschung, Datensicherung im Unternehmen, interaktiven Knoten und auch private Computer.

1.1 Problematik

Die E/A-Leistung von HPC-Systemen wird hauptsächlich durch die Massenspeichergeschwindigkeit, die Kapazität der Datenübertragungskanäle und durch den Grad der Ausnutzung des Hardwarepotenzials beschränkt. Während die ersten zwei Parameter hardwarebedingt sind, ist der dritte Parameter softwareabhängig. Alle drei Parameter sind eng miteinander verknüpf und haben einen direkten Einfluss auf die E/A-Gesamtleistung des Systems.

Die relativ langsamen Massenspeicher und Datenübertragungskanäle, im Vergleich zu Rechenleistung, sind das Resultat der unausbalancierter Hardware-Entwicklung in den letzten Jahren. Bei der Verarbeitung von genügend großen Datenmengen kann es deshalb zu Engpässen kommen, die ein HPC-System ausbremsen können. Sie stellen deswegen ein Problem dar. Um dem entgegen zu wirken werden diverse Lösungen verwendet, wie Raid-Verbunde, parallele Dateisysteme, Caches und schnelle Hardware-Komponenten. In HPC-Systemen werden meistens mehrere solcher Lösungen gleichzeitig miteinander betrieben. Das beschleunigt zwar die E/A-Leistung des gesamten Systems, dadurch entsteht aber auch ein komplexes Zusammenspiel zwischen den Komponenten, das es schwierig macht die E/A-Vorgänge zu überblicken und zu verstehen.

Komplexe Systeme stellen auch besondere Ansprüche an die Software-Entwicklung, z. B. die Parallelisierung von Programmen und ein E/A-Zugriffsmanagement zum Erhalten der Datenkonsistenz. Die Software wird dadurch oft größer und kompli-

zierter und das E/A-Verhalten undurchschaubar. Eine unzureichende Analyse vom Programmverhalten führt dann oft zur ineffizienten Hardwarenutzung, suboptimaler Kommunikation zwischen Rechen- und Speichereinheit und einer unausbalancierten Lastverteilung. Je nach Komplexität kann die Verhaltensanalyse für einen Menschen zu einer unüberwindlichen Hürde werden. Ohne die genaue Vorstellung über die Vorgänge ist es kaum möglich das Programmverhalten vorherzusagen. Schwierig ist es auch die Auswirkung von einer Programmänderung auf die Leistungsänderung (quantitativ) zu erfassen und noch schwieriger ist es die problematischen Programmteile zu identifizieren.

Die Administratoren von grossen IT-Systemen müssen sich ebenfalls der Komplexität gegenüberstellen. Die Abstimmung der Hardware- und Softwarekomponenten aufeinander und die Optimierung des Systems auf die bevorstehende Nutzung kann schwierig sein. Die Nutzung kann dabei sehr unterschiedlich sein, z.B. kann man die Systeme auf kurze Reaktionszeiten oder großen Durchsatz optimiert. Die Optimierung auf kurze Reaktionszeiten kommt vor allem für die interaktiven Systeme in Frage, wo viele Benutzer durch viele gleichzeitige E/A-Zugriffe, z. B. durch Lesen von einer Vielzahl von kleinen Dateien oder durch Ausführen von E/A-lastige Prozessen, schnell die Grenzen der Hardware ausreizen. Das ist wichtig, weil höhere Reaktionszeiten von Benutzern in der Regel als besonders störend empfunden werden. Bei Computern, die E/A-lastige Hintergrunddienste ausführen, macht sich das Problem durch geringe Systemauslastung bemerkbar.

Die Ursache der E/A-Problematik kann in einem Satz zusammengefasst werden: die Prozessoren verbringen eine beträchtliche Zeit im Leerlauf, während sie auf die Daten warten.

1.2 Überblick über die Hardwarelösungen

Die Systemhersteller, Systembetreiber und die Wissenschaftler bemühen sich die Probleme mit verschiedenen Techniken in den Griff zu bekommen.

Viele Systeme, insbesondere Server und HPC-Systeme, sind mit Raid-Verbunden ausgestattet, wobei jeder Raid-Typ ein oder mehrere Ziele verfolgen kann. Die meisten Raid-Verbunde versuchen jedoch einen Kompromiss zwischen Leistung,

Datensicherheit und Kosten zu finden, wobei die Ziele gegeneinander gerichtet sind, was bedeutet, dass man das eine nicht verbessern kann ohne das andere zu verschlechtern. Die relativ hohen Anschaffungs-, Betriebs- und Wartungskosten machen die Raid-Systeme nur bis zur einer gewissen Größe wirtschaftlich. Außerdem erreichen die Raid-Verbunde eine höhere Leistung nur für Daten ab einer bestimmten Größe. Der Grund liegt in der Funktionsweise der Raid-Verbunde. Sie zerteilen die Daten in Blöcke und schreiben diese parallel auf mehrere Massenspeicher. Wenn die Daten nicht auf mehrere Blöcke zerteilen lassen, dann kann kein paralleler Schreibzugriff stattfinden, deshalb auch kein Leistungsgewinn. Für den Lesezugriff gilt die analoge Logik.

Verteilte Dateisystem fassen mehrere entfernte Massenspeicher zu einem Dateisystem zusammen. Für den Benutzer bleibt die verteilte Struktur des Dateisystems nicht sichtbar und er kann darauf wie auf lokales Dateisystem zugreifen. Somit erfüllen die verteilte Dateisysteme das Zugriffstransparenzkriterium. Die Kommunikation mit dem Dateisystem läuft über eine fest definierte Schnittstelle ab und macht die verteilten Dateisysteme betriebssystem- und hardwareunabhängig. Eine Besonderheit von vielen modernen verteilten Dateisystemen ist ihre Fähigkeit parallel auf die Daten zuzugreifen. Sie können dabei verschiedene Strategien verwenden. Ähnlich wie die Raid-Verbunde, können sie die Dateien in mehrere Blöcke zerteilen und parallel auf mehrere Massenspeicher verteilen. Bekannte Vertreter dieser Dateisystemfamilie im HPC-Bereich sind Lustre, GPFS.

Die Caches sind eine weitere weitverbreitete Methode die E/A-Leistung zu verbessern. Dabei werden die oft genutzte oder voraussichtlich bald benötigte Daten auf einem schnelleren Datenträger zwischengespeichert, z.B. ist in den meisten modernen Internet-Browsern das HTTP-Caching implementiert. Die Inhalte der Webseiten werden auf der Festplatte abgelegt, um beim wiederholten Zugriff bereits besuchte Seiten schnell aufzubauen und unnötigen Datenverkehr zu vermeiden. OLTP-Datenbankbetreiber (On-Line-Transaction Processing), bauen zusätzlich zu Festplattenverbunden SSD-Zwischenspeicher in Ihre Systeme ein, um die Transaktionenzahl zu erhöhen. OLAP-Lösungen setzen vermehrt auf die In-Memory Analyse um Zugriffszeiten auf E/A zu vermeiden. Das Dateisystem ZFS bietet die Möglichkeit Arbeitsspeicher als Cache zu nutzen.

Ein direkter Weg die E/A-Leistung zu verbessern ist die Verbesserung der Hardware

und hier können wir wahrscheinlich die meisten Entwicklungen sehen. Als Beispiel kann sich man die Entwicklung von Massenspeicher anschauen. Während der gesamten Entwicklung wurden die Drehzahlen der Festplatten erhöht, die Schreib- / Lesekopfgeschwindigkeiten verbessert, die Datendichte, die Cachegröße und die unterschiedliche Zugriffsstrategien und Optimierung der Schreib- /Lesekopf-Wege angewendet. Die letzten Generationen der Festplatten wurden mit Edelgas befüllt und haben Flüssigkeitsgleitlager. Einen wesentlichen Schritt hat man mit der Einführung der SSD gemacht.

Trotz allen Bemühungen verschärft sich das Problem zunehmend und stellt eine Herausforderung an die Forscher, eine wirtschaftliche und leistungsfähige Lösung zu entwickeln. Zahlreiche Unternehmen und Forschungseinrichtungen haben das Problem bereits angegangen und zahlreiche Projekte gestartet, die sehr unterschiedliche Wege gehen.

HP forscht an einem neuartigen Datenspeicher [2], der mit Hilfe von Memristoren realisiert werden und den flüchtigen, schnellen Arbeitsspeicher und den nichtflüchtigen langsamen Massenspeicher vereinheitlichen soll. Solch ein Speicher ermöglicht auch eine Computerarchitektur, in der die E/A-Problematik zumindest lokal wesentlich entschärft werden kann. Der erste Prototyp soll laut dem Zeitplan im Jahr 2017 erscheinen und im Jahr 2020 sollen die Rechner dem breiten Markt verfügbar sein.

1.3 Analyse und Optimierung von E/A-Operationen

Um die meist kostspielige Erweiterung oder den Ersatz von Hardware zu vermeiden oder hinauszuzögern bietet sich als Alternative die Optimierung der Software. Die Optimierung kann man auf verschiedenen Wegen erreichen.

Leistungssteigerung ist auch direkt bei der Programmierung möglich, z. B. kann man die Lokalität der Daten ausnutzen. Wenn man genau weiß wie die Daten im Speicher abgelegt werden, dann kann man seinen E/A-Befehl auf eine kleinere Anzahl von E/A-Zugriffen optimieren. Allerdings setzt dies ein tiefes Verständnis des Systems und sehr gute Kenntnisse in einer hardwarenahen Programmiersprache voraus. In der Praxis ist diese Methode nur auf einfache Fälle anwendbar. Außerdem

garantiert es nicht, dass die Optimierungen auf anderen Rechner genauso effizient laufen. Fertige Programme kann man nicht optimieren.

Bei der Analyse und Optimierung von E/A-Operationen muss man die Hardware mit in Betracht ziehen und die Besonderheiten beachten. Schließlich will man letztendlich die Hardware optimal nutzen. Eine andere interessante Forschungsrichtung ist die Optimierung der E/A-Aufrufe. In einem Zweig geht man davon aus, dass Systeme, die parametrisierten E/A-Aufrufe anbieten das Potenzial haben mit optimalen Parametern beschleunigt zu werden. Die Schwierigkeit dabei ist für jeden individuellen E/A-Aufruf die optimalen Parameter zu bestimmen und zu übergeben. Schafft man das, dann kann die Leistung in vielen Fällen zumindest theoretisch wesentlich verbessert werden. In einem anderen Zweig versucht man die E/A-Zugriffe so zuordnen, damit möglichst optimal auf die Hardware abgestimmt sind. Beides ist universell anwendbar, kann als Software implementiert werden und relativ kostengünstig dem breiten Publikum zur Verfügung gestellt werden.

Wie bereits angedeutet, kommen bei der Realisierung der E/A-Systeme verschiedene Hard- und Software-Technologien zum Einsatz, z.B. beim Zugriff auf die Festplatten spielt das Dateisystem, Bus, Blockgröße, diverse Caches usw. eine Rolle. Alle Komponenten stehen in einem komplexen Zusammenhang miteinander und führen zu Leistungseinbußen bei einer suboptimalen Konfiguration. Die Abstimmung der Komponenten aufeinander alleine erfordert meistens schon Expertenwissen, hinzu kommt noch Abstimmung der Komponenten auf die Nutzung, die in manchen Fällen variiert und unvorhersagbar sein kann. Eine universelle Methode oder sogar eine Software, die diese Aufgabe erleichtert, wäre hier sehr willkommen. Dazu fehlt allerdings noch die Grundlage, da Vorgänge auf der E/A-Ebene wurden bisher nicht ausreichend untersucht sind. Es fehlt Wissen über die Zugriffsmuster, die bei der komplexen Nutzung entstehen und deren Ursachen und Auswirkungen auf die Systemleistung. Dementsprechend fehlen auch Gegenmaßnahmen zur Vermeidung von unerwünschten Situationen.

1.4 Ziele und Vorgehensweise

Ziel der Arbeit ist die Analyse und Bewertung von Ein-/Ausgabe-Operationen mittels maschinellem Lernen vorzunehmen. Hierfür wird das SIOX-Framework

erweitert um das maschinelle Lernen zu ermöglichen und Bewertungs-Plugins mit verschiedenen Granularitäten zu erstellen. Im Detail gliedern sich die Teilziele des Projektes in:

- Workflow-Design und Umsetzung.
- Vorhersage von Durchsatz von E/A-Operationen.
- Bewertung der E/A-Operationen zur Laufzeit des Programms und Ausgabe einer Statistik nach der Terminierung. In der Bewertung wird den E/A-Operationen ein Typ zugeordnet (z.B. Zwischenspeicherzugriff, direkter Zugriff, ...), die Statistik zeigt die Häufigkeit an.
- Wissensgewinn durch Extraktion von Regeln aus dem verwendeten ML-Modell.

1.5 Aufbau der Arbeit

Das zweite Kapitel stellt einige verwandte Arbeiten aus dem Bereich der E/A-Zugriffsoptimierung vor. Im dritten Kapitel werden einige grundlegende theoretischen Aspekte behandelt, die zum Verständnis der Arbeit notwendig sind. Dort werden die relevanten Begriffe definiert, die Funktionsweise der aktuellen Technologie erläutert und die benutzten Werkzeuge vorgestellt. Das vierten Kapitel beschreibt das Design des Leistungsprädiktors und Ursache-Klassifikator und erklärt wie man mit diesen beiden Werkzeugen ein Parametrisierungstool zum setzen von optimalen Parametern bauen kann. Im fünften Kapitel wird die Umsetzung vorgestellt und die Ergebnisse ausgewertet. Zum Schluss gibt es noch eine Zusammenfassung und den Ausblick auf weitere Forschungsmöglichkeiten.

Kapitel 2

2 Verwandte Arbeiten und Hintergrund

Die Analysewerkzeuge haben, je nach Spezialisierung, in bestimmten Bereichen ihre Stärken und Schwächen. Deshalb sind einige für bestimmte Probleme besser geeignet als die anderen. Die Wahl des richtigen Analysewerkzeugs sollte man deshalb nicht unterschätzen.

Auf dem Markt befinden sich bereits mehrere ausgereiften, kommerziellen Leistungsanalysewerkzeuge, wie z.B. Darshan und Vampir. Es gibt aber auch Open-Source-Alternativen, wie das SIOX-Framework. Das Kapitel stellt kurz die drei Werkzeuge vor.

2.1 Darshan

Darshan [3] ist ein Analysewerkzeug zum Charakterisieren vom E/A-Verhalten auf HPC-Systemen. Es wurde entworfen, um mit einem minimalen Overhead ein genaues Bild vom Anwendungsverhalten und Eigenschaften, wie Zugriffsmuster in einer Datei, aufzunehmen.

Darshan nutzt unterschiedliche Wrapper zum Instrumentieren von Anwendungen. Sie erfassen die E/A-Operationen von allen von der Anwendung geöffneten Dateien und erzeugen zu jeder Datei eine bestimmte Menge von Daten. Statt alle Daten in einer Spurdatei zu speichern, wie die konventionelle Analysewerkzeuge es machen, berechnet Darshan bestimmte Statistiken, die er auch noch zusätzlich reduziert, komprimiert und zur einer kompakten zusammengefasst. Am Ende der Programmausführung werden sie in eine Log-Datei geschrieben. Die dort gespeicherten Statistiken beschreiben letztendlich das Verhalten von der gesamten

Anwendung. Dieses Prinzip macht es möglich den Overhead beträchtlich auf einen vernachlässigbaren Wert zu minimieren und erfordert nur eine begrenzte Menge von Arbeitsspeicher.

Zur Log-Datei-Analyse kann man auf die mitgelieferten Kommandozeilenwerkzeuge zurückgreifen. Eines davon heißt "darshan-job-summary" und wie der Name schon sagt, erzeugt es eine Zusammenfassung der Log-Datei. Das Beispiel [4] in der Abb. 2.1 zeigt so eine Zusammenfassung von einer Messung auf dem Mira IBM blue Gene/Q System von Argonne Leadershop Computing Facility (ALCF). Im Graph "I/O Operation Counts" kann man erkennen, dass das "MPI-IO collective buffering"-Optimierung aktiviert ist. In den Tabellen sieht man, dass die meisten Zugriffe 16MB gross sind, was der Puffergröße vom "collective write" entspricht. In den unteren Graphen kann an erkennen, dass die Schreiboperationen von mehreren Prozessen in unterschiedlichen Zeitintervallen geschrieben wurden.

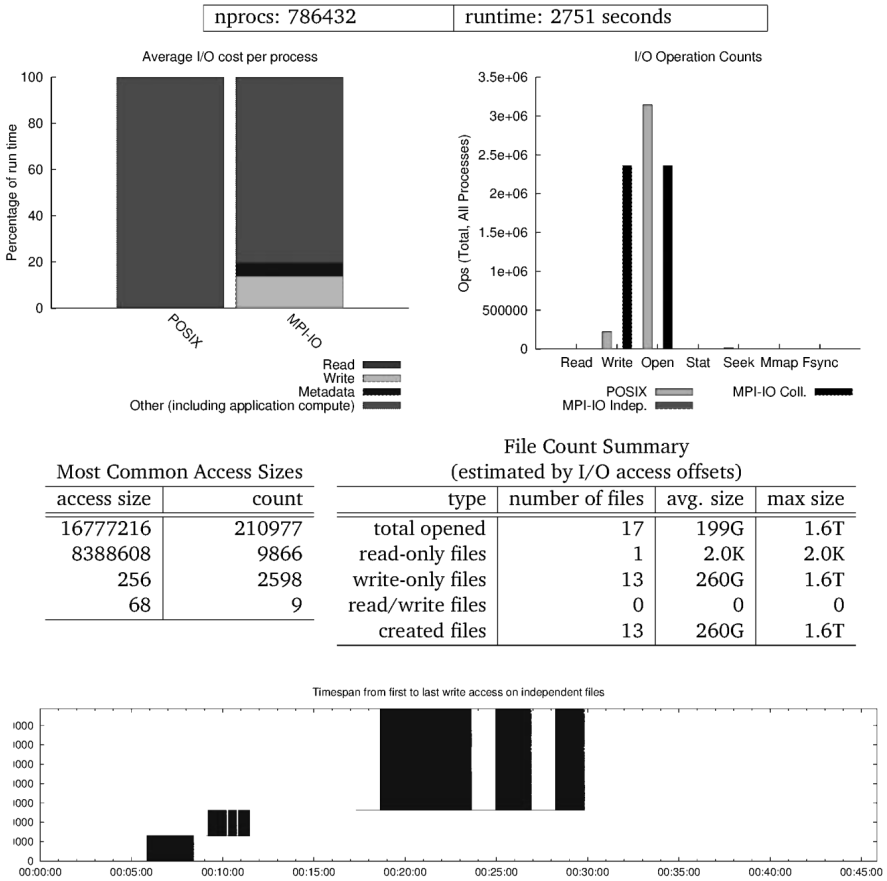
Die Instrumentierungswerkzeuge von Darshan beherrschen mehrere verschiedenen Schnittstellen. Eine vollständige Unterstützung haben die POSIX- und die MPI-IO-Schnittstelle. Die HDF5- und die PNetCDF-Schnittstelle werden zwar auch unterstützt, aber nur teilweise.

Darshan kann für breites Aufgabenspektrum eingesetzt werden, von Anwendungsoptimierung bis zur E/A-Verhaltensanalyse von gesamten HPC-Systemen. Das leichte und effiziente Design von Darshan macht ihn auch geeignet für die durchgehende Lastcharakterisierung in großen Systemen auch während des produktiven Betriebs.

2.2 Vampir

Vampir [5] ist ein grafisches Werkzeug zur Leistungsanalyse von parallelen Systemen. Es unterstützt die Offline-Analyse von parallelen (MPI, OpenMP, Multithreading) und hardwarebeschleunigten (CUDA und OpenCL) Anwendungen. Seine Analyseengine erlaubt eine skalierbare und effiziente Verarbeitung von großen Mengen von Leistungsdaten.

Vampir nutzt die Infrastruktur von Score-P [6] zum Instrumentieren von Anwendungen. Score-P speichert die Aktivitäten in einer Datei ab, die von Vampir analysiert



Most Common Access Sizes

access size	count
16777216	210977
8388608	9866
256	2598
68	9

File Count Summary
(estimated by I/O access offsets)

type	number of files	avg. size	max size
total opened	17	199G	1.6T
read-only files	1	2.0K	2.0K
write-only files	13	260G	1.6T
read/write files	0	0	0
created files	13	260G	1.6T

Abbildung 2.1: Ausschnitte aus "darshan-job-summary"-Ausgabe. Das Beispiel zeigt das E/A-Verhalten einer Turbulenzsimulation, die mehr als 3TB an Daten geschrieben hat und mit 786432 Prozessen ausgeführt wurde.

und in verschieden Sichten umgewandelt werden kann, z.B. können die Aktivitäten auf einer Zeitachse dargestellt werden oder zu einer der vielen Statistiken komprimiert werden. Einige Sichten verfügen über aufwendige Filter- und Zoom-Funktionen, mit den man sich leicht einen Überblick verschaffen und die Details betrachten kann.

Eine effektive Nutzung von Vampir erfordert ein tiefes Hintergrundwissen in der parallelen Programmierung. Das Programm ermöglicht die Aufzeichnung und Auswertung von POSIX-E/A-Zugriffen, aber wenig Informationen über deren Zustandekommen oder zur Bewertung der E/A. Der Einsatzbereich von Vampir wird durch die fehlende Unterstützung der Online-Analyse etwas eingeschränkt.

2.3 SIOX-Framework

SIOX [7] ist ein unter GPL veröffentlichte Open-Source-Framework, der seit dem Jahr 2011 im Rahmen eines Forschungsprojekts des BMBF an den Universitäten Hamburg, Dresden und Stuttgart entwickelt wurde. Es unterstützt neben der ereignisbasierte Erfassung und Analyse von E/A-Zugriffe auf Desktop- und HPC-Systemen auch die Möglichkeit die Ein/Ausgabe zu optimieren. Das Framework enthält Werkzeuge zur Instrumentierung der Schnittstellen, zur Auswertung von Spurdaten und diverse Analyse-Plugins.

Ähnlich wie Vampir, arbeitet SIOX mit Aktivitäten und Aktivitätensequenzen, die beim Ausführen der instrumentierten Anwendungen erfasst werden oder künstlich mit einem Plugin erzeugt werden können. Diese Aktivitäten sind Datenstrukturen, die relevanten Daten und Messungen über die E/A-Operation beinhalten. Sie können zur späteren Analyse in einer Datei gespeichert oder in Echtzeit verarbeitet werden.

Im Kern stellt SIOX eine Plugin-Infrastruktur bereit und erhält seine endgültige Funktionalität erst durch die Anbindung von Plugins und Modulen. Das macht SIOX leicht erweiterbar und fast unbegrenzt an die eigene Bedürfnisse anpassbar. So kann man z.B. eigene Module zur Instrumentierung von Schnittstellen anbinden oder die Aktivitäten künstlich mit einem eigenen Aktivitätsgenerator-Plugin erzeugen und die erzeugten Aktivitäten mit eigenen Analyse-Plugins verarbeiten.

SIOX kann nach der Terminierung der instrumentierten Anwendung statistische Information berechnen und in den s.g. Reports ausgeben. Komplexere Ausgaben und Auswertungen (z.B. Abbildungen, Datenmenge) können natürlich auch direkt durch die Plugins vorgenommen werden.

SIOX unterstützt sowohl Online- als auch Offline-Analyse. Zusätzlich erlaubt die Anbindung eines Aktivitätsgenerator-Plugins auch eine hypothetische Situation zu simulieren. Das Framework befindet sich noch im Entwicklungsstadium, hat noch keine feste Schnittstelle erhalten und verfügt bislang nur über wenige Plugins.

Die Nachfrage nach guten und für HPC-Systeme geeigneten Analysewerkzeugen wächst und führt hoffentlich zu weiteren neuen Projekten und Neuentwicklungen,

so wie das SIOX-Framework. Das SIOX-Framework besitzt ein grosses Entwicklungspotenzial und stellt kostenlose Alternative zu den kommerziellen Produkten bereit. Im Gegensatz zu proprietärer Software ist es hier möglich den Programmcode einzusehen und seinen eigenen Bedürfnissen anzupassen. Das ist auch eines der Hauptgründe für die Verwendung von dem Framework in dieser Arbeit.

Kapitel 3

3 Grundlagen

Einige der verwendeten Begriffe und Techniken gehören zu Spezialgebieten. Sie werden deshalb in diesen Abschnitt kurz behandelt. Als erstes wird das maschinelles Lernen allgemein vorgestellt und dann die Funktionsweise der Entscheidungsbäumen erläutert. Es wird die Cachehierarchie und deren Einfluss auf die E/A-Leistung erklärt. Schließlich werden die Struktur SIOX-Aktivitäten im Detail betrachtet und dann auf die Erzeugung von Aktivitätensequenzen eingegangen.

3.1 Maschinelles Lernen

Beim maschinellen Lernen geht es darum richtigen Datensatz zu finden, um ein richtiges Modell zu bauen, das ein Problem möglichst genaue löst.

Die Lernalgorithmen setzen in der Regeln eine geeignete Repräsentation der Welt voraus. Die betrachteten Objekte werden deshalb meistens als eine Reihe von Werten repräsentiert, den s.g. Features. Einfachheitshalber können sie auch als Messungen von bestimmen Objekteigenschaften gesehen werden, wie z.B. Länge, Anzahl, Geschwindigkeit. Der Erfolg hängt stark von der Wahl der richtigen Features ab.

Die Modelle sind die zentralen Konzepte im Bereich des maschinellen Lernen. Sie werden mit dem Lernalgorithmen trainiert, um bestimmte Probleme zu lösen. Es existiert bereits eine beträchtliche Anzahl von Modellen. Der Grund für die Vielfalt sind die zahlreichen unterschiedliche Probleme, die speziell behandelt werden müssen, wie z. B. Kategorisierung, Regression, Gruppierung, Assoziationanalyse. Bei den Modellen handelt es sich im Grunde genommen um die Abbildungen von den Features auf die Ausgabe.

Die Aufgabenstellung ist die abstrakte Beschreibung des Problems. Eine sehr oft vorkommende Aufgabenstellung ist beispielsweise die Kategorisierung von Objekten in zwei oder mehrere Kategorien. Viele existierende Aufgabenstellungen können auf die Abbildung von Datenpunkten auf Werte reduziert werden. Wenn das der Fall ist, dann können sie mit vielen Lernalgorithmen genutzt werden.

Die Lernalgorithmen können in drei Klassen unterteilt werden: unüberwachtes Lernen, überwachtes Lernen und bestärkendes Lernen. Beim überwachten Lernen stehen dem Algorithmus die Daten und die korrekte Ausgabe zur Verfügung. Beim unüberwachten Lernen stehen nur die Daten zur Verfügung und der Algorithmus muss die korrekte Ausgabe selber daraus schliessen. Beim bestärkendem Lernen interagiert der Algorithmus mit der Umgebung und lernt von dem Reaktionen auf seine Handlungen. Dieser Typ wird allerdings in dieser Arbeit nicht verwendet und wurde nur vollständigkeitshalber erwähnt.

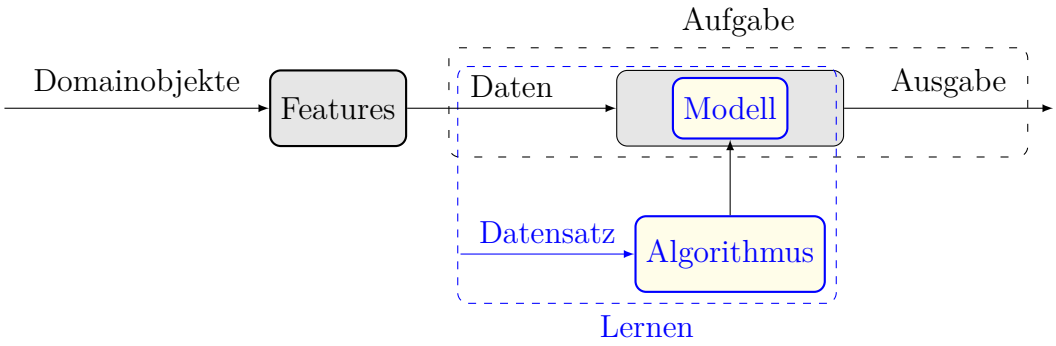


Abbildung 3.1: Grundkonzept des maschinellen Lernens.

Der Zusammenhang zwischen den Aufgaben, Modellen und Lernalgorithmen ist in der Abb. 3.1 dargestellt.

In der Trainingsphase trainiert ein Lernalgorithmus mit einem Trainingsdaten das Modell. (In der Abbildung blauen Farbe hervorgehoben.) In der Bewertungsphase werden die Features aus den Objekten extrahiert und an das Modell übergeben. Das Modell berechnet dann den Ausgabewert.

3.1.1 Begriffe und Definitionen

Die wichtigsten Begriffe aus den einleitenden Worten über das maschinelles Lernen werden in den folgenden Definitionen mit mathematischen Elementen erweitert. Einige Begriffe haben gegenseitige Abhängigkeiten und werden in den Definitionen benutzt bevor sie überhaupt definiert sind.

Definition 1 (Feature und Featuredomain)**.** Ein Feature f_i beschreibt eine Eigenschaft des untersuchten Objektes, z.B. kann es das Alter einer Person, die Hausnummer oder die Datenmenge sein. Eine Featuredomain $\mathscr{F}$ legt alle Werte fest, die ein Feature annehmen kann. Mathematisch gesehen handelt es sich um eine Abbildung von dem Instanzenraum auf die Featuredomain $f_i : \mathscr{X} \to \mathscr{F}_i$. Mehrere Features bilden ein Featurevektor x.

Definition 2 (Instanzraum)**.** Der Instanzenraum ist ein Kreuzprodukt aus den Featuredomainen $\mathscr{X} = \mathscr{F}_1 \times \cdots \times \mathscr{F}_d$, wobei die Anzahl der Features d die Dimension des Instanzenraumes festlegt.

Definition 3 (Label und Labeldomain)**.** Die Featurevektoren können mit einem Label behaftet sein. Die Bedeutung des Labels hängt von der Aufgabenstellung ab. Die Labeldomain $\mathscr{L}$ legt die gültigen Werte fest, die das Label annehmen kann. Es gilt der Zusammenhang $l : \mathscr{X} \to \mathscr{L}$.

Definition 4 (Modell und Ausgabedomain)**.** Ein Modell ist eine Abbildung vom Instanzenraum auf die Ausgabedomain $M : \mathscr{X} \to \mathscr{Y}$. Die Ausgabedomain und die Labeldomain sind nicht zwangsläufig gleich. Je nach Modelltyp und Lernerfolg können sie von einander abweichen.

Definition 5 (Datensatz)**.** Ein Datensatz ist eine echte Untermenge des Instanzenraumes $\mathscr{D} \subseteq \mathscr{X}$. Datensätze können aus belabelten oder auch nicht belabelten Features bestehen. Sie werden einerseits für das Modelltraining und andererseits zur Validierung des Modell eingesetzt.

3.1.2 Cluster-, Klassifikations- und Regressionsanalyse

Die Clusteranalyse ist eine Prozedur, die eine Menge von Featurevektoren in Gruppen bzw. in die s.g. Cluster eingeteilt. Dabei beinhalten die gleichen Cluster möglichst gleiche und unterschiedliche Cluster möglichst unterschiedliche

Featurevektoren. Die Ähnlichkeit zwischen den Featurevektoren bestimmt das Ähnlichkeitsmaß. Es bietet die Möglichkeit die Featurevektoren nummerisch zu vergleichen. Ein sehr oft verwendetes Ähnlichkeitsmaß ist die euklidische Distanz, die Entfernung zwischen zwei Punkten.

Die Regressionsanalyse stellt die eine mathematische Beziehung zwischen mehreren unabhängigen Variablen f und einer abhängiger Variable y, oder kurz $y = regr(f_1, f_2, \ldots, f_n) + e$, wobei $regr$ eine Funktion des Modell ist und e der Fehler. Die Regressionsanalyse kann eingesetzt werden, um fehlende Werte zu bisher nicht analysieren Variablen vorhersagen.

Die Klassifikationsanalyse ist ein mathematisch-statische Verfahren indem eine untersuchte Einheit, die aus mehreren unabhängigen Variablen f besteht, nach bestimmten Kriterien einer bestimmter Gruppe y zugeordnet wird, oder kurz $y = class(f_1, f_2, \ldots, f_n)$, wobei $class$ eine Entscheidungsfunktion des Modell ist. Im Gegensatz zur Regressionsanalyse und Clusteranalyse müssen hier die Klassen bekannt sein und unbekannte Klassen können nicht vorhergesagt werden.

3.1.3 Kreuzvalidierung

Die Kreuzvalidierung ist eine Auswertungsmethode der ML-Algorithmen. Im einfachen Fall wird damit der durchschnittlichen Fehler des Modells berechnet. Zur Berechnung vom Kreuzvalidierungsfehler wird zunächst ein Datensatz in k möglichst gleiche große Mengen aufgeteilt. Die $k-1$ Mengen werden zu einer Trainingsmenge zusammengefasst und die übriggebliebene Menge bildet die Testmenge. Als erstes wird das Modell mit der Trainingsmenge trainiert. Danach werden die Features aus dem Testmenge in das Modell eingegeben. Das trainierte Modell liefert nun die dazugehörigen vorhergesagten Labels l_p, die zusammen mit den echten Labels aus der Testmenge l_r und der Anzahl der Instanzen n zu einem Fehler e_j umgerechnet werden.

Bei der Regressionsanalyse berechnet sich der Fehler aus der Differenz der Labels.

$$e_j = \frac{\sum_{i=1}^{n} |l_{p_i} - l_{r_i}|}{n} \tag{3.1}$$

Bei der Klassifikationsanalyse werden die korrekt vorhergesagten Werte gezählt

und daraus der Mittelwert berechnet.

$$f(a,b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } a \neq b \end{cases} \tag{3.2}$$

$$e_j = \frac{\sum_{i=1}^{n} f(l_{p_i}, l_{r_i})}{n} \tag{3.3}$$

Nachdem der Schritt k-mal mit jeweils unterschiedlicher Testmenge ausgeführt wurde, werden die berechneten Fehler e_j gemittelt. Der Mittelwert e_{cv} ist dann das Ergebnis der Kreuzvalidierung.

$$e_{cv} = \frac{\sum_{j=1}^{k} e_j}{k} \tag{3.4}$$

3.1.4 Entscheidungsbäume

Einige Algorithmen aus der Entscheidungsbaum-Familie beherrschen sowohl die Klassifizierungs- als auch Regressionsanalyse. Sie nutzen zwar unterschiedliche Techniken, um ein Entscheidungsbaum zu erstellen, aber die Entscheidungsbäume funktionieren alle nach demselben Prinzip:

Die Entscheidungsbäume bestehen aus drei Knotentypen: einer Wurzel, den internen Knoten und den Blättern.

Jeder interner Knoten hat stets eine oder mehrere Verzweigungen, die als Brücke zu den Kindknoten dienen. Jedem seiner Zweige ist ein Wertebereich (oder eine Bedingung) zugeordnet. Ein interner Knoten besitzt auch stets einen Entscheidungsmerkmal a, der zusammen mit dem Wertebereich (oder Bedingung) darüber entscheidet welchen Zweig man als nächstes geht. Die Wurzel ist im Prinzip ein gewöhnlicher interner Knoten. Eine besondere Stellung erhält die Wurzel, weil sie erstens keine Elternknoten besitzt und zweites, dient sie als Ausgangspunkt beim Entscheidungsprozess. Die Blätter sind Terminierungsknoten. Sie beinhalten die Ausgabewerte.

Der Entscheidungsprozess benötigt der Baum ein Featurevektor als Eingabe und liefert stets einen Ausgabewert. An der Wurzel wird aus dem Featurevektor der Wert f_a entnommen und mit den Wertebereichen (oder der Bedingungen) der

Zweige verglichen. Liegt der Wert innerhalb des Wertebereichs eines Zweiges, so geht man über ihn zum Kindknoten über. Dieser Vorgang wiederholt sich rekursiv bis eine Blatt erreicht wird. Der Ausgabewert des Blattes wird zurückgegeben.

Binärbäume (Abb. 3.2) sind besondere Typen in der Entscheidungsbaum-Familie. Die interne Knoten in einem Binärbaum besitzen stets zwei Verzweigungen. Einfachheitshalber, reicht es hier ein Schwellenwert t zu definieren, um zu entscheiden welchen Zweig man folgt. Wenn die Bedingung $f_i < t$ gilt, dann geht man zum linken Knoten, andernfalls zum rechten.

Die Funktionsweise der Bäume ist relativ einfach und für den Menschen leicht verständlich. Man kann nachvollziehen wie bestimmte Entscheidungen zustande kommen und daraus Wissen ableiten. Das ist einer der Hauptvorteile der Entscheidungsbäume. Ein Beispiel soll die Funktionsweise verdeutlichen.

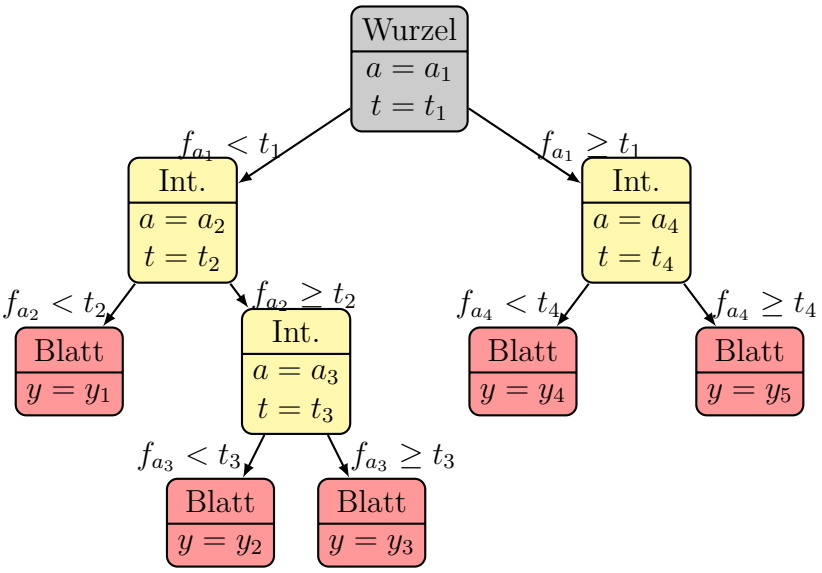


Abbildung 3.2: Schematische Darstellung eines binären Entscheidungsbaumes.

Beispiel 3.1.1. Ein Beispiel soll die Funktionsweise der Entscheidungsbäume illustrieren.

1. Gegeben sei ein Entscheidungsbaum M (Abb. 3.3).
2. Aufgabe: Berechne $M(x_1)$, mit dem Featurevektor $x_1 = (4, 5, 2)$.
3. Lösung:

a) An der Wurzel betrachten wir wegen $a = 3$ das dritte Feature $f_3 = 2$. Mit dem Schwellenwert $t = 4$ ist die Ungleichung auf dem linken Zweig $f_3 < 4$ erfüllt. Wir gehen den Zweig entlang zu den linken Knoten.

b) Angekommen auf dem linken Knoten, betrachten wir wegen $a = 2$ das zweite Feature $f_2 = 5$. Der Schwellenwert ist hier $t = 2$ Es gilt die Ungleichung auf dem rechten Zweig $f_2 > 2$, die zum rechnten Knoten führt. Wir gehen den Zweig entlang zu den rechten Knoten.

c) Ein Blatt wurde erreicht. Der Entscheidungsbaum gibt den Wert $y = B$.

d) Es gilt $M((4, 5, 2)) = B$.

Der Entscheidungsbaum im Beispiel kann die Featurevektoren in drei Kategorien einordnen. Für die Kategorisierung allerdings sind nur zwei Features relevant: f_2 und f_3. Das Feature f_1 kommt in dem Baum nicht vor und hat keine Auswirkung auf die Ausgabe. Wenn das passiert, dass ist das ein deutlicher Hinweis darauf, dass die Features möglicherweise nicht optimal gewählt sind.

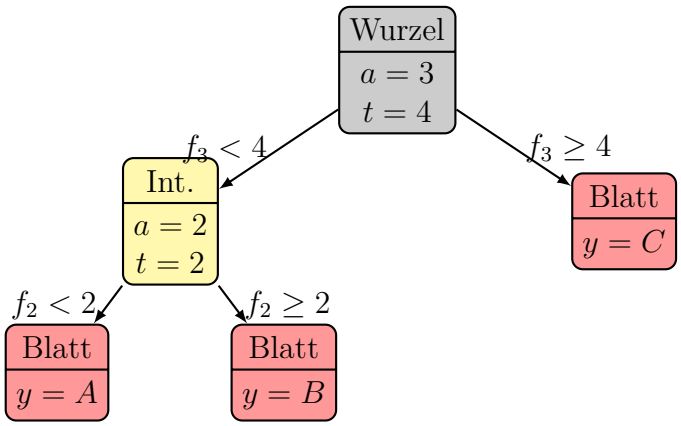


Abbildung 3.3: Beispiel.

In diesem einfachen Beispiel kann die Funktionsweise auch graphisch dargestellt werden. In der Abb. 3.4 werden die Features f_2 und f_3 in einem Koordinatensystem aufgespannt. Die Rechtecke zeigen wie die Kategorien zugeornet werden.

Aus dem Entscheidungsbaum kann man direkt die Regeln entnehmen, indem man den Pfad von dem Blattknoten zur Wurzel folgt. In diesem einfachen Fall sind es drei folgende Regeln:

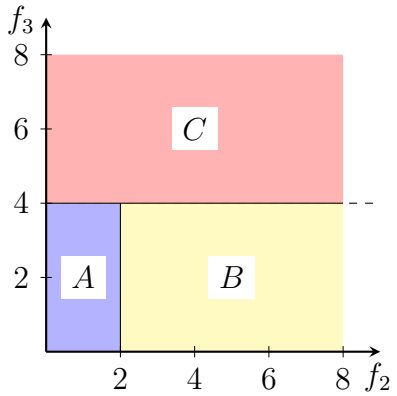


Abbildung 3.4: Aufgabenstellung.

$$f_3 < 4 \vee f_2 < 2 = A \tag{3.5}$$

$$f_3 < 4 \vee f_2 \geq 2 = B \tag{3.6}$$

$$f_3 \geq 4 = C \tag{3.7}$$

Die Entscheidungsbäume allgemein haben die Tendenz zum Übergeneralisieren. Das passiert, wenn der Baum sehr viele Spezialfälle lernt und in der Regel dadurch seine Vorhersagegenauigkeit von Labels zu unbekannten Featurevektoren verliert. Die Spezialfälle sind in den Entscheidungsbäumen im Form von Pfaden von dem Blattknoten zur Wurzel repräsentiert. Und sehr viele davon lassen den Baum sehr stark anwachsen. Die Binärbäume sind besonders davon betroffen, weil zur Integration von zwei Blättern mindestens ein weiterer interner Knoten benötigt wird.

Das Übergeneralisierungsproblem kann mit den s.g. Pruning-Verfahren gelöst werden. Beim Pruning wird der Baum einfach nach bestimmten Kriterien oder mit bestimmten Methoden beschnitten, um die Größe zu reduzieren.

Die Pruning-Verfahren werden noch zwischen Post- und Pre-Pruning unterschieden. Das Post-Pruning-Verfahren wird nach dem Training angewendet und das Pre-Pruning-Verfahren während des Trainings. Die wahrscheinlich einfachste und universelle Variante vom Post-Pruning ist die Baumtiefebegrenzung. Hierbei wird alles was länger ist als die festgesetzte Tiefe einfach abgeschnitten. Die internen

Knoten, die dadurch alle Kinder verlieren, werden zum Blättern umgewandelt und bekommen den Ausgabewert vom dominanten Kindknoten zugewiesen. Das ist der Kindknoten, das am meisten besucht wird, wenn man die Trainingsmenge anwendet. Die Pre-Pruning-Verfahren sind wesentlich komplizierter und spezifisch für den jeweiligen Lernalgorithmus.

3.2 E/A-Leistung

Um die E/A-Leistung besser zu verstehen und Missinterpretationen zu vermeiden muss man die grundlegende Funktionsweise der Hardware und Software verstehen und bei der Analyse berücksichtigen.

3.2.1 Leistungskennzahlen

Wegen den unterschiedlichen Einsatzzwecken lässt sich die Leistung der E/A-Systeme nicht durch eine einzige Kennzahl bemessen. Stattdessen ist die Leistungskennzahl abhängig vom Systemtypen und muss für jedes System individuell bestimmt werden. Die gängigsten Leistungskennzahlen sind:

Definition 6 (Datendurchsatz)**.** Der Datendurchsatz ist die übertragene Datenmenge pro Zeiteinheit.

Definition 7 (Latenz)**.** Ist die Zeit zwischen der Adressierung eines Speicherbereiches und der Bereitstellung der Daten.

Definition 8 (Auslastung)**.** Die Auslastung eines Systems ist der Quotient aus der tatsächlicher Nutzungszeit (inkl. Overhead) und der Gesamtlaufzeit des Systems.

3.2.2 Wahlfrei E/A-Operationen

Die Massenspeicher können nur eine bestimmte Anzahl der E/A-Operationen pro Sekunde $IOPS$ durchführen. Bei den Festplatten wird sie z. B. durch die Suchzeit t_{seek} und die Verzögerungszeit t_d bestimmt. t_{seek} die Zeit, die der Lesekopf im Schnitt benötigt, um die richtige Spur auf der Scheibe zu finden und t_d ist die Zeit

dei im Schnitt vergeht, bis der Lesekopf über die Daten fliegt. IOPS wäre in diesen Fall folgende:

$$IOPS = \frac{1}{t_d + t_{seek}} \tag{3.8}$$

Da bei jeder E/A-Operation die Datenmenge d_{size} sequentiell durchgeführt wird, lässt sich daraus $IOPS$ und d_{size} der Durchsatz bestimmen.

$$D = IOPS \cdot d_{size} \tag{3.9}$$

An den Formeln kann man sehen, dass der Datendurchsatz direkt durch t_{seek} beeinflusst wird. Wenn also t_{seek} sehr gross wird, dann müssen wir mit einem geringen Durchsatz rechnen. Die Situation ändert sich schlagartig mit der Einführung von Caches.

3.2.3 Caches

Caches sind schnelle Zwischenspeicher zum Beschleunigen von E/A-Zugriffen auf das darunterliegende langsamere Speichermedium. In den meisten Computern sind die Cacheschichten übereinander aufgebaut und bilden eine typische Cachehierarchie, wie in der Abb. 3.5 dargestellt. Sie unterscheiden sich hauptsächlich in der Größe und in der Geschwindigkeit (Bandbreite und Latenz), wobei die schnellen Caches meistens wesentlich kleiner sind als die langsamen. Der Grund dafür sind die Preise. Sie liegen für die schnellen Speichereinheiten wesentlich höher als für die langsamen. Die Abb. 3.6 zeigt die Spitze der Cachehierarchie von einem aktuellen Rechner. Man kann deutlich erkennen, dass die Bandbreite und Cachegrößen exponenziell verlaufen, wobei die Bandbreite exponenziell abnimmt während die Cachegröße exponenziell zunimmt.

Der Leistungsgewinn durch Einsatz von Caches entsteht nur dann, wenn die benötigten Daten im Cache liegen. Die Daten müssen also erst dahin kopiert werden. Das passiert ganz automatisch, ohne dass der Benutzer etwas davon merkt. Bei einem Lesezugriff prüft das System, ob die benötigten Daten im Cache liegen. Wenn das der Fall ist, dann handelt es ich um das ein Cache-Hit und die Daten

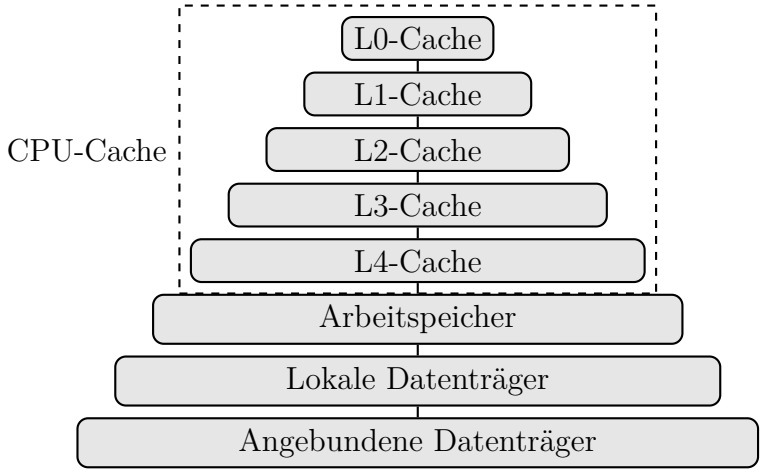


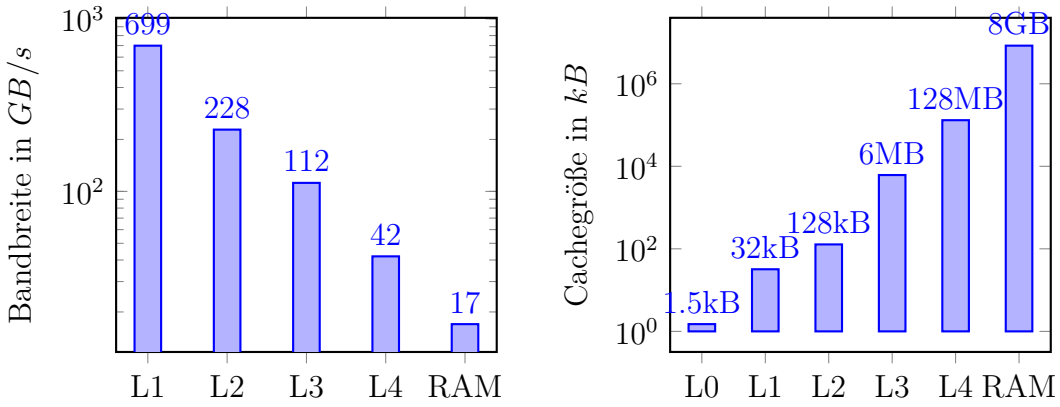
Abbildung 3.5: Cachehierarchie.

werden aus dem Cache gelesen, andernfalls, beim Cache-Miss, werden sie erst vom Speichermedium in den Cache eingelesen und dann aus dem Cache gelesen.

Der Einsatz von Caches kann zu einer scheinbaren Überschreitung des theoretischen maximalen Leistung der Komponenten führen, wenn die benötigten Daten bereits im Cache liegen. In diesem Fall ist der E/A-Zugriff auf den Massenspeichermedium überflüssig und wird übersprungen.

Das Cache-Systems verfolgt typischerweise das Ziel die voraussichtlich bald benötigten Daten im Cache zu halten und erreicht es u.a. durch die Ausnutzung von zeitliche und räumliche Lokalitäten. Bei zeitlicher Lokalität wird ein Speicherbereich in den Cache geladen und bleibt dort, wenn das Cache-System davon ausgeht, dass es bald wieder benutzt wird. Bei räumlicher Lokalität werden in den Cache auch die benachbarten Speicherbereiche geladen, wenn das Cache-System davon ausgeht, dass sie bald benötigt werden, z.B. wenn aus dem Code hervorgeht, dass ein Array sequentiell in einer Schleife verarbeitet wird.

Neben den traditionellen Cachehierarchie gibt es noch weitere Formen, die nach dem selben Prinzip funktionieren, sich aber nicht so einfach in die Cachehierarchie eingliedern lassen, z.B. SSD-Cachesysteme, Auslagerungsdatei, HTTP-Caching. Vom Prinzip her funktionieren sie ähnlich, wie hier beschriebene Caches.



(a) Gemessene Bandbreite

(b) Cachegröße

Abbildung 3.6: Beispiel: Spitze der Cachehierarchie in einem aktuellen Rechner mit dem "Intel Mobile Haswell (CrystalWell)" Prozessor und 2x4GB DDR3 1600MHz Arbeitsspeicher[8].

3.3 SIOX-Aktivitäten

Die SIOX-Aktivitäten sind Datenstrukturen, die Informationen über die E/A-Operationen beinhalten. Die aktuelle Version arbeitet bereits mit MPI-, POSIX-, HDF5- und NETCDF4-Schnittstellen. Die Aktivitäten dienen als eine Basis für die Leistungsanalyse und werden in diesem Abschnitt detailiert betrachtet.

Die Tab. 3.1 listet grob alle Bestandteile einer SIOX-Aktivität. Der Typ und der Name wurden direkt aus dem Quellcode übernommen und kurz beschrieben. Da in dieser Arbeit nur die POSIX-Schnittstelle betrachtet wird, werden nicht alle Bestandteile benötigt. Die nicht verwendeten Bestandteile wurde in der Tabelle ausgegraut.

Ganz besonders interessant sind die drei POSIX-Operationstypen: Eingabe-, Ausgabe- und Positionierungsoperationen, denn in den meisten Fällen bestimmen sie die E/A-Leistung. Um einmal zu zeigen wie diese Operationen in den SIOX-Aktivitäten behandelt werden schauen wir uns jeweils einen Vertreter von jedem Typ an, nämlich `fread`-, `fwrite`- und `fseek`-Operation (Code 3.1).

Typ	Name	Beschreibung
`ActivityID`	`aid`	Eindeutiger Bezeichner
`UniqueComponentActivityID`	`ucaid`	Typ der E/A-Operation
`Timestamp`	`time_start`	Startzeit in Nanosekunden
`Timestamp`	`time_stop`	Stopzeit in Nanosekunden
`vector<ActivityID>`	`parentArray`	Ausgangs-E/A-Operationen
`vector<RemoteCall>`	`remoteCallsArray`	-
`vector<Attribute>`	`attributeArray`	Parameter, Rückgabewert, usw.
`RemoteCallIdentifier*`	`remoteInvoker`	-
`ActivityError`	`errorValue`	-

Tabelle 3.1: Zusammensetzung der SIOX-Aktivitäten. (Die ausgegrauten Bestandteile sind nicht relevant für diese Arbeit.)

Listing 3.1: POSIX-Operationen

```
1 size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
2 size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
3 int fseek(FILE *stream, long offset, int whence);
```

Die **aid**, **time_start** und **time_stop** Elemente sind von Funktion her trivial. **ucaid** ist als eine Zahl kodierter Operationstyp und ist ebenfalls trivial. Spannend wird es bei **parentArray** und **attributeArray**.

Das Attribute **parentArray** verweist auf die Ausgangs-Aktivitäten, d. h. auf die Aktivitäten mit der der Zugriff auf eine bestimmte Datei begonnen hat. Beim POSIX ist es die **open**-Aktivität. Die Aktivitäten mit dem gleichen parentArray bilden eine Gruppe und, wenn zeitlich angeordnet, eine Aktivitätensequenz auf eine bestimmte Datei. Wenn eine Datei mehrmals geöffnet wurde und die Zugriffe über verschiedene FileHandler erfolgen, dann sind auch die parentArray's unterschiedlich. Hat eine Aktivität keine Ausgangsaktivität, wie z. B. open-Aktivität, dann ist parentArray leer.

Das Attribute attributeArray beinhaltet Informationen über den Systemaufruf. Das können die Funktionsparameter, der Rückgabewert, diverse Laufzeitinformationen usw. sein. Der Inhalt von attributeArray ist spezifisch für die E/A-Operation.

3.3.1 Erzeugung von Aktivitätensequenzen

Die Offline-Analyse wird in drei Schritten durchgeführt.

Attribute	Wert
POSIX/quantity/BytesToWrite	4
POSIX/descriptor/FilePointer	28317504
POSIX/data/MemoryAddress	4233230
POSIX/quantity/BytesWritten	4

Tabelle 3.2: Attribute und Beispielwerte einer write-Operation.

Attribute	Wert
POSIX/quantity/BytesToRead	100
POSIX/data/MemoryAddress	22192128
POSIX/descriptor/filehandle	0
POSIX/quantity/BytesRead	100

Tabelle 3.3: Attribute und Beispielwerte einer read-Operation.

Im ersten Schritt werden die Aktivitäten einer Anwendung aufgezeichnet und in einer Spurdatei gespeichert Abb. 3.7. Dazu wir das SIOX-Instrumentierungswerkzeug verwendet.

Abbildung 3.7: Erstellung einer Spurdatei.

Im zweiten Schritt werden die Aktivitäten ausgewertet. Dazu wird die erstellt Spurdatei mit siox-trace-reader eingelesen, der dann die Aktivitäten an den Analyse-Plugin weiterleitet. Das Plugin startet dann den Lernprozess. Die Ergebnisse im diesen Schritt sind zwei Entscheidungsbäume: das eine für die Leistungsvorhersage und das andere für die Klassifizierung.

Im dritten Schritt wird wieder eine Spurdatei (die gleiche oder eine andere) eingelesen und vom Plugin ausgewertet. Die Ergebnisse sind Vorhersagen der Leistungswerte, Kategoriezuordnung.

In diesem Kapitel haben wurden die Arbeit relevanten Begriffe, Techniken und Technologien behandelt. In den folgenden drei Kapiteln werden kombiniert, um Daten zu erzeugen und auszuwerten.

Attribute	Wert
POSIX/descriptor/filehandle	4
POSIX/file/position	0

Tabelle 3.4: Attribute und Beispielwerte einer read-Operation.

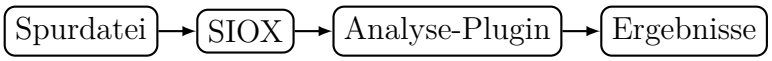


Abbildung 3.8: Vorhersage

Kapitel 4

4 Evaluation der ML-Bibliotheken

Die Evaluation der Bibliotheken für maschinelles Lernen (ML) ist unbemerkt zum Teil dieser Arbeit geworden. Es hat sich herausgestellt, dass sich die Bibliotheken in einigen wichtigen Details von einander unterscheiden und die Wahl einer passender Bibliothek nicht unterschätzt werden sollte. Es wurden die Bibliotheken OpenCV, Shark und Shogun ausgewählt und speziell die Entscheidungsbaum-Algorithmen evaluiert.

4.1 OpenCV

OpenCV ist eine unter BSD Lizenz veröffentlichte Bibliothek für Bildbearbeitung und maschinelles Sehen. Bei der Entwicklung hat man den Fokus auf effiziente Ausführung und Echtzeitverarbeitung gelegt. Die Bibliothek in optimierten C/C++ geschrieben und nutzt die Vorteile von Multikern-Prozessoren und kann mit OpenCL die Hardwarebeschleunigung nutzen. Neben den C/C++-Schnittstellen besitzt die Bibliothek auch Python- und Java-Schnittstellen.

Die Bibliothek eignet sich sehr gut für Einsteiger. Sie besitzt eine leicht verständliche und intuitive Schnittstelle, hat eine relativ grosse Community und eine bemerkenswert gute Dokumentation. Dies macht sie auch geeignet schnell und unkompliziert zu experimentieren.

Die Bibliothek ist in erster Linie auf die Bildbearbeitung ausgerichtet und ist für diese Arbeit, die eine spezialisierte Bibliothek mit einer erweiterter Funktionalität, nicht ganz optimal.

4.2 Shark

Shark ist eine unter GNU LGPL veröffentlichte Bibliothek für maschinelles Lernen. Sie ist eine objekt-orientierte, leicht erweiterbare und auf die Leistung ausgerichtete Bibliothek. Sie ist in C++ geschrieben und besitzt bereits in Beta-Version eine durchaus durchdachte Schnitte. Sie ist an die Nutzer gerichtet, die sich gut mit dem maschinellen Lernen und der Programmiersprache auskennen.

Die Bibliothek nutzt die Möglichkeiten der Programmiersprache, um aus dem Code eine exakt auf das Problem zugeschnittene Lösung zu bauen. Template-Klassen werden intensiv genutzt, um auch auf das Problem zugeschnittener, effizienter Code erzeugt. Sehr viele Objekte aus dem maschinellen Lernen sind als Template-Klassen implementiert und lassen sich noch zusätzlich mit Parametern konfigurieren. Weitere Gegenstände können implementiert und eingebunden werden, wenn was fehlen sollte.

Die Pruning-Funktion, die die Tiefe der Bäume begrenzt ist nicht vorhanden. Zur Verfügung steht nur eine auf Kreuzvalidierung basierte Pruning-Funktion. Die Funktion beschränkt leider nicht die Tiefe des Baumes, was in dieser Arbeit bei Erzeugung von kurzen Regeln sehr nützlich gewesen wäre.

Um mit der Bibliothek zu arbeiten sind sehr gute C++ erforderlich. Die Bibliothek eignet sich nur begrenzt zum ausprobieren und zum experimentieren. Lange Einarbeitungszeit und relativ hohe Komplexität erschweren das auch noch.

4.3 Shogun

Für diese Arbeit wurde die Shogun-Bibliothek verwendet. Die Bibliothek ist sehr umfangreich. Sie wird aktiv entwickelt und regelmässig erweitert und verbessert.

Es bietet eine Vielzahl von Algorithmen (Support Vektor Maschinen, Entscheidungsbäume, Neuronale Netze, ...), Eine weitere Besonderheit sind die vielen implementierten Schnittstellen (C, C++, Python, R, ...). Die Schnittstellen sind alle sehr ähnlich, wenn man eine kennt, dann kann man ohne grossen Aufwand auch die anderen benutzen. Das erlaubt z. B. schnelle Experimente in R und Python oder auch feste Integration in größere C++-Projekte.

- Die neue Features sich nicht gut dokumentiert. - Nicht vollständig implementiert, z. B. die Serialisierung.

Zudem haben die Regeln die maximale Länge d. Das Verfahren wurde nur zum Teil implementiert. Er erstellt zwar eine Menge von Regeln, reduziert sie aber nicht, weil der letzte Schritt im Algorithmus nicht klar ist. Hier ist es sinnvoll nachzuhacken

4.4 Zusammenfassung

Bibliothek	**Schnittstellen**	**Entscheidungsbäume**
OpenCV	C/C++, Python, Java	CART, Random Trees, Boosted Trees
Shark	C++	CART, Random Forest
Shogun	C/C++, Python, R, Matlab, ...	CART, C4.5, CHAID, ID3

Tabelle 4.1: Vegleich von Bibliotheken

Alle drei sind gute und umfangreiche Bibliotheken mit verschiedenen Vor- und Nachteilen.

Kapitel 5

5 Design

Ein Optimierung-/Wissenszyklus sorgt dafür, dass die E/A-Leistung der Software ständig verbessert wird. Die notwendigen Informationen dazu sollen der Cachetyp- und Leistungsprädiktor und der Ursachen-Klassifikator liefern, indem sie die Statistiken sammeln und Wissen aus den trainierten Modellen extrahieren.

5.1 Optimierungs-/Wissenszyklus

Der angestrebte Optimierungs- /Wissenszyklus (Abb. 5.1) kann grob in wenigen Schritte beschrieben werden. Als erstes werden die E/A-Operationen vom Instrumentierungswerkzeug abgefangen, in Aktivitäten umgewandelt und in einer Spurdatei gespeichert. Nach dem Terminieren der Anwendung beinhaltet die Spurdatei eine Datensammlung, die von einer Analysesoftware verarbeitet werden kann. Die Analysesoftware lernt von den Daten und erstellt davon ein Modell. Der Modellanalyser nutzt dieses Modell, um daraus das Wissen zu extrahieren und es der Anwendung zur Verfügung zu stellen. Der Kreis schliesst sich, wenn die Anwendung das gewonnene Wissen nutzt, um die E/A-Operationen zu optimieren.

5.2 Allgemeines Vorgehen

Die aufgenommene Aktivitätensequenz soll aus der instrumentierten Anwendung eine repräsentative Datenmenge beinhalten. Die Anwendung soll deshalb die E/A-Operationen mit möglichst vielen unterschiedlichen Parametern ausführen, so dass ein breites Spektrum an Lernmaterial vorliegt.

Zum Verarbeiten von gesammelten Daten werden die Entscheidungsbäume verwendet, da die zur Verfügung stehenden Lernalgorithmen keine Fortsetzung des

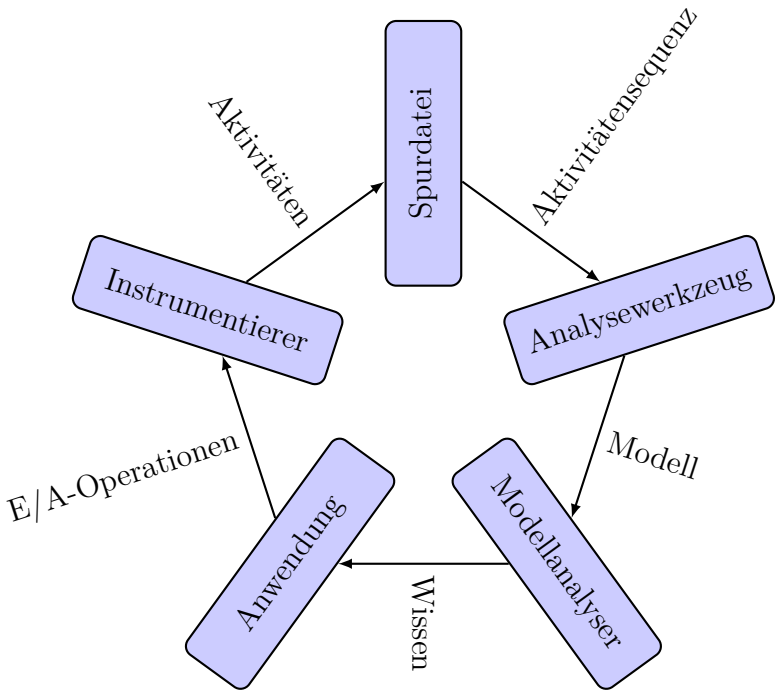


Abbildung 5.1: Optimierungs- / Wissenszyklus

Trainings unterstützen, muss hier auf die kontinuierliche Verbesserung des Modell verzichtet werden. Die Trainings- und Bewertungsphase müssen deswegen zwangsläufig von einander getrennt ausgeführt werden.

In der Trainingsphase wird mit einem Lernalgorithmus aus den gesammelten Daten ein Entscheidungsbaum erzeugt. Der Entscheidungsbaum hat zwei Funktionen, erstens wird er zum Vorhersagen der Labels benutzt, zweitens wird daraus das Wissen in Form von Regeln extrahiert. Da der Baum selber schon eine wichtige und für Menschen leicht verständliche Wissensquelle ist, wird diese visualisiert und mit zusätzlichen Informationen ergänzt.

Desweiteren werden aus dem Entscheidungsbaum Regeln gewonnen. Die Regeln sind vereinfachte Pfade von den Blattknoten zu der Wurzel. Sie sollen Hinweise darauf liefern, mit welchen Features man bestimmte Leistungswerte bzw. Klassen erreicht werden kann. Wenn diese Features Parameter repräsentieren, die vom Benutzer gesetzt werden können, dann ist das ein sehr nützliches Mittel für die Leistungsoptimierung, weil dann die richtigen Parameter direkt aus den Regeln abgelesen werden können.

In der Trainingsphase können drei verschiedene Methoden benutzt werden. Mit dem Leistungsprädiktor kann die Leistung vorhergesagt werden, mit dem Cachetypprädiktor können die Cachetypen vorhergesagt werden und der Ursache-Klassifikator soll in der Lage sein den Datensatz zu Clustern und die Cluster zu lernen.

5.3 Bereinigte und unbereinigte Datensätze

Es kann durchaus vorkommen, dass ein Datensatz gleiche Featurevektoren mit unterschiedlichen Labels beinhaltet. Das kann zu unerwünschten Effekten, wie Übergeneralisierung oder Verfälschung des Kreuzvalidierungsfehlers führen.

Vor dem Training werden deshalb, in einige Fällen, wo es notwendig ist, die Daten bereinigt, d.h. die gleichen Featurevektor werden zu einem einzigen Featurevektor zusammengefasst und bekommt als Label den Mittelwert. Für die korrekte Durchführung der Kreuzvalidierung ist dies notwendig, denn nur so können wir bestimmen wie gut das Modell mit den unbekannten Featurevektoren umgehen kann. Wir können ausserdem sehen wieviele verschiedene Featurevektoren den Knoten zugeordnet werden und daraus die Relevanz der Knoten ableiten.

Die Trainingsdaten müssen aber nicht unbedingt bereinigt werden, wenn Pruning aktiviert ist. Mit dem Pruning wird die Übergeneralisierung verhindert.

5.4 Leistungs- und Cachetypprädiktor

Die primäre Aufgabe der Prädiktoren ist es, die Daten auf Ihre Strukturen zu analysieren. Man kann von den strukturierten Daten ausgehen, wenn der Leistungsprädiktor zu unbekannten Featurevektoren die Leistungswerte ausreichend gut vorhersagen kann. Der umgekehrte Fall gilt natürlich nicht, d.h. falls der Prädiktor scheitert, folgt nicht, dass die Daten unstrukturiert sind.

Das sekundäre Ziel ist es, das Wissen aus den Prädiktoren in eine menschenverständliche Form zu bringen, so dass man es für die Anwendungsoptimierung nutzen könnte. Daraus sollte erkennbar sein welche Features zu besseren Ausgabewerten führen und welche Features die relevanten sind.

Das Funktionsprinzip der beiden Prädiktoren ist ähnlich. Beide nutzen für das Training einen belabelten Datensatz. Der einzige Unterschied ist, dass der Leistungsprädiktor ein Regressionsproblem löst und der Cachetypprädiktor ein Klassifikationsproblem.

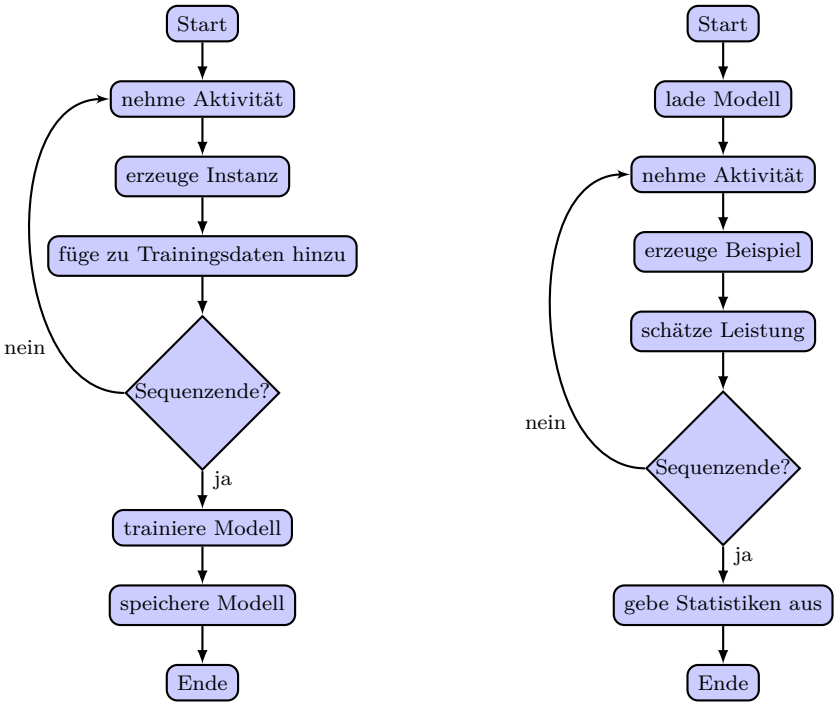
Der Leistungsrädiktor hat die Aufgabe die Informationen aus den Aktivitäten und die gemessene bzw. Leistungswerte zu nutzen, um Leistung vorherzusagen. Damit er diese Aufgabe erfüllen kann, wird ein Regressionslernalgorithmus benutzt, um ein Modell zu trainieren. Das Modell wird dann in der Lage sein, die bereits bekannte Werte und neue Werte vorherzusagen.

Leider unterstützt keines der Arbeit verwendeten Lernverfahren die Fortsetzung der Trainings, d.h. wir werden das Modell nicht kontinuierlich verbessern können, sondern müssen uns für eine andere Strategie entscheiden. Die naheliegendste Möglichkeit ist: zuerst die Trainingsdaten zu sammeln und dann damit das Modell zu trainieren. Die Funktionsweise des Plugins geht dann direkt aus dieser Überlegung aus zu einem zwei Phasenmodell. In der ersten Phase (Abb. 5.2(a)) erzeugt das Plugin aus der Aktivitätensequenz Trainingsdaten, trainiert damit das Modell und speichert es in einer Datei ab. In der zweiten Phase (Abb. 5.2(b)) erzeugt das Plugin aus der Aktivitätensequenz eine Testmenge und wendet sie an.

5.4.1 Kreuzvalidierung

Der Leistungsprädiktor erfordert auch eine besondere Implementierung der Kreuzvalidierung. Der Grund dafür ist, dass die konventionellen Kreuzvalidierung die Größenordnung der Leistungswerte nicht berücksichtigt. So würden die kleinere Leistungswerte beim Weiten einen kleineren Einfluss auf das Ergebnis haben als die größeren. Als Folge würde das Ergebnis die Abweichung von großen Werten zeigen. Um das zu umgehen, arbeitet diese Variante der Kreuzvalidierung mit relativen, statt absoluten Leistungswerten.

$$e_j = \frac{\sum_{i=1}^{n} \frac{|p_p - p_r|}{p_r}}{n} \tag{5.1}$$



(a) Training (b) Vorhersage

Abbildung 5.2: Arbeitsweise von Prädiktoren. In der Trainingsphase werden Daten gesammelt und das Modell trainiert. In der Bewertungsphase wird das Modell zur Vorhersage und Statistikerzeugung verwendet.

Aus den relativen Fehlern wird nochmal der Mittelwert gebildet. Das ist dann das Ergebnis der Kreuzvalidierung, der Fehler e_{cv}.

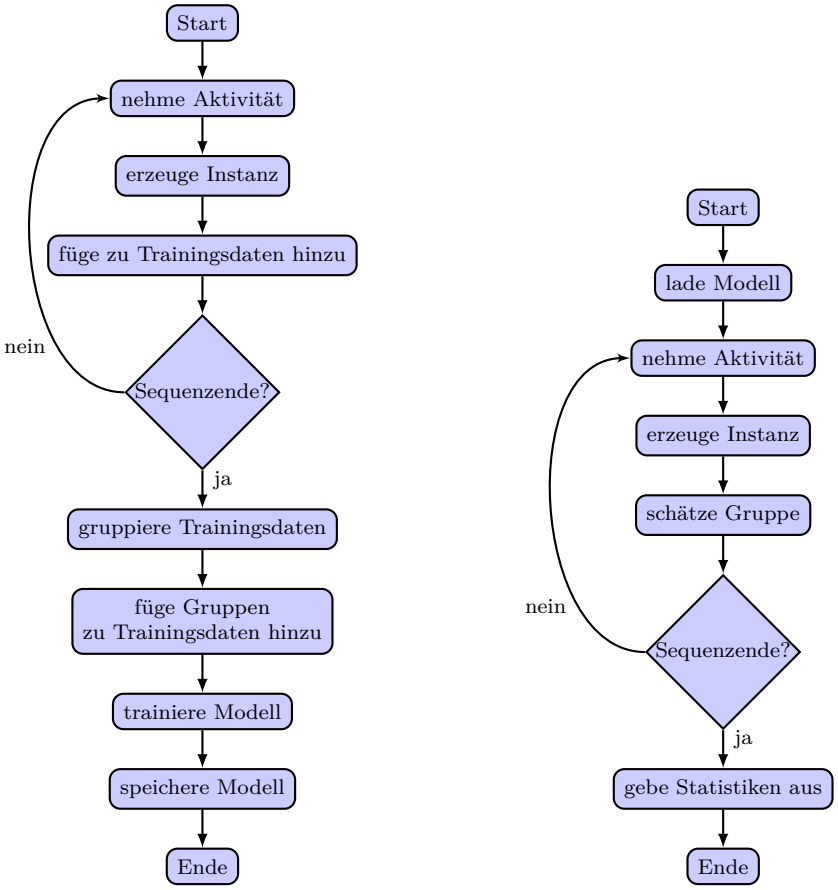
$$e_{cv} = \frac{\sum_{j=1}^{k} e_j}{k} \tag{5.2}$$

5.5 Ursachen-Klassifikator

Die Aufgabe des Ursachen-Klassifikators ist ähnliche E/A-Operation zu gruppieren und die Gruppen zu erlernen, um später die E/A-Operation klassifizieren zu können.

In ersten Phase (Abb. 5.3(a)) wird der Trainingsmenge mit einem Clusteralgorith-

mus in Gruppen eingeteilt und die erzeugten Gruppen werden den entsprechenden Vektoren zugeordnet. Die Gruppen bekommen natürlich keinen Cachetypen zugeordnet, weil diese Information durch Clustering nicht bestimmt werden kann. Daraus entsteht dann eine neue Trainingsmenge, die in der zweiten Schritt vom Klassifizierungsalgorithmus erlernt wird.



(a) Training (b) Vorhersage

Abbildung 5.3: Arbeitsweise vom Ursachen-Klassifikator. In der ersten Trainingsphase werden Daten gesammelt und das Modell trainiert. In der Bewertungsphase wird das Modell für die Gruppenzuordnung und die Statistikerzeugung verwendet.

Nach Ausführen des Klassifikators wird ein Entscheidungsbaum erzeugt und in einer Datei gespeichert.

Auch hier gilt, bei ausreichend guten Werten wird der Entscheidungsbaum in

eine Menge von Regeln umgewandelt. Die Wichtigkeit einer Regeln bestimmt die Mengen von Datensätzen, die diese Regeln benutzt haben. Das ist der Quotient aus der Anzahl der Datensätze und der Gesamtzahl der Datensätze.

Desweiteren bestimmt die Kreuzvalidierung wie gut der Baum mit den Daten umgehen kann und wie aussagekräftig die gewählten Features sind.

In der Bewertungsphase (Abb. 5.3(b)) wird der Entscheidungsbaum geladen und wird dazu benutzt, um aus den erzeugten Features die Gruppen vorherzusagen.

5.6 Ausgabe

Die Visualisierung der binären Entscheidungsbäumen ist etwas anders als im Grundlagenabschnitt dargestellt wurde. In den Abbildungen werden die Zweige nicht bezeichnet, es gilt aber stets: wenn $f(attr) < split$ dann gehe den linken Pfad entlang und wenn $f(attr) \geq split$ gehe den rechten Pfad entlang. Alle Knoten haben noch zusätzlich den Wert *amount* bekommen. Zum Illustrationszwecke zeigt die Abb. 5.4 einen kleinen Entscheidungsbaum. Bei den Kindknoten bedeutet er die Anzahl der während des Trainings zugeordneten Featurevektoren. Bei internen Knoten ist es die Summe der Werte aus den Kindknoten. Aus den *amount*-Wert kann sehen wie dominant der Knoten bzw. sein Label ist. Die internen Knoten habe noch zusätzlich ein Label bekommen. Das ist das Label des dominanten Kindes. Auf dieser Weise kann leichter welhe Entscheidung der Baum auf den niedrigeren Stufen treffen würde.

Das Design-Kapitel beschreibt das Verfahre nur sehr grob. Die nächsten zweit Kapiteln beschreiben die Detail. Um das Verfahren soll mit SIOX-Framework und einer geeigneter ML-Bibliothek realisiert werden.

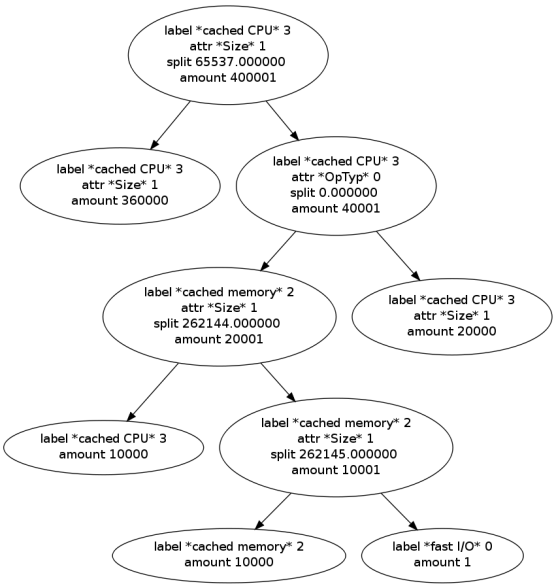


Abbildung 5.4: Beispiel eines Entscheidungsbaumes

Kapitel 6

6 Umsetzung

Die drei Methoden aus dem Design-Kapiteln werden nun als SIOX-Plugin implementiert. Das Kapitel geht auf die Details der Implementation ein.

6.0.1 CART- und kMeans-Algorithmen in der Shogun-Bibliothek

Der Entscheidungsbaum-Lernalgorithmus CART (Classification and Regression Tree) kann man entscheiden welche Features nominal und welche ordinal / bzw. kontinuierlich sind. Diese Möglichkeit wird allerdings nicht genutzt. Alle Features werden in kontinuierliche umgewandelt bevor sie an den Algorithmus übergeben werden. Der Algorithmus nutzt den Regressionsmodus bei der Leistungsvorhersage und den Kategorisierungsmodus bei der Bewertung der Leistung.

Ausserdem hat unterstützt die Implementierung die zwei Pruningsarten: Pruning mit Kreuzvalidierung (CV-Pruning) und Begrenzung der Baumtiefe. Das Pruning mit der Kreuzvalidierung benötigt die Anzahl von Teilmengen. Diese kann mit dem Parameter *cvFolds* bestimmt werden. Die maximale Baumtiefe kann durch den Parameter *treeDepth* bestimmt werden. Der Knoten bekommt das Label von seinem größten Kind, d. h. von dem Kind mit der größten Anzahl von zugeordneten Featurevektoren. Beide Pruningsarten können parallel verwendet werden.

Zum Initialisieren vom kMeans-Algorithmus sind zwei Parameter notwendig: die Anzahl der Klassen k und ein Ähnlichkeitsmaß d. Als Ähnlichkeitsmaß wurde wurde die euklidischen Distanz fest einprogrammiert. Die Anzahl der Klassen ist jedoch variabel geblieben. Sie kann bei manuellen Klassifizierung vom Benutzer mit dem Parameter *numberOfClasses* übergeben werden oder mit bei der automatischen Klassifizierung automatisch berechnet werden.

Index	Bezeichnung	Beschreibung
0	Index	Nummerierung
1	AbsTime	Zeit in Sekunden seit Beginn der ersten Aktivität
2	DeltaTime	Zeit in Sekunden zwischen Beginn der letzten und aktuellen Aktivität
3	fd	File descriptor
4	OpTyp	Typ der Operation (Lesen oder Schreiben)
5	DeltaOffset	Abstand in Bytes zwischen den Datenbereichen, auf die zugegriffen wurde
6	Size	Datengröße in Bytes
6	Duration	Dauer der Aktivität in Sekunden
8	Throughput	Durchsatz in MB/s
9	tr	Indikator für unlogischen Zugriff
10	Label	manuell berechnete Labels
11	Prediction	vom Modell vorhergesagter Wert

Tabelle 6.1: Spalten im DataStore-Objekt.

6.1 SIOX-Plugin

Das SIOX-Plugin verwendet Algorithmen aus der Shogun-Bibliothek verwendet. Der kMeans-Algorithmus wird verwendet, um die Daten in Kategorien einzuteilen und der CART-Algorithmus zum trainieren von Entscheidungsbäumen

6.1.1 Datenverwaltung

DataStore ist ein Datenverwaltungsobjekt. Seine Hauptaufgaben sind die Daten in Form einer Matrix in einer konsistenter Form zu behalten und den Umgang damit zu vereinfachen. Dazu gehören der Im- und Export der Daten in eine CSV-Datei, Erstellung von Features und Labels und Verwaltung von Metainformationen, wie Spaltenbezeichnung, Abbildung von Werten auf Namen.

Das DataStore-Objekt im Plugin wurde mit einer festen Anzahl von Spalten konfiguriert. Sie sind in der Tab. 6.1 aufgelistet.

Zum funktionieren benötigt er eine Aktivitätensequenz bzw. eine Datenbank und eine gültige Konfiguration.

6.1.2 Konfigurationsparameter

An das Plugin können eine Reihe von Konfigurationsparameter übergeben werden. Die verfügbaren Parameter sind:

- **mode** (string) Wählt zwischen Training und Bewertungsphase.
  - *training* - Trainingsphase
  - *prediction* - Bewertungsphase
- **method** (string) Problemtyp.
  - *performance* - Leistungsprädiktor
  - *categories* - Cachtyp-Prädiktor
  - *clustering* - Ursachen-Klassifikator
- **dataFile** (string) Pfad zur CSV-Daten mit Trainingsdaten. Dieser Parameter wird verwendet, wenn **mode** = *categories* und **method** = *training*.
- **numberOfClasses** (int) Anzahl der Kategorien, in die Daten geteilt werden sollen. Dieser Parameter wird verwendet, wenn **mode** = *categories* und **method** = *training*.
- **featsCols** (int[]) Indizien der Spalten zur Bildung von Featurevektoren. Die Indizien können der Tab. 6.1 entnommen werden.
- **labelCol** (int) Auswahl der Label-Spalte. Die Indizien können der Tab. 6.1 entnommen werden.
- **treeDepth** (int) Maximale Tiefe der Entscheidungsbäume.
- **cvFolds** (int) Anzahl der Menge für den Pruning-Funktion im CART-Algorithmus.
- **readTokens** (string[]) POSIX-Leseoperationen, die überwacht werden sollten.
- **writeTokens** (string[]) POSIX-Schreiboperationen, die überwacht werden sollen.
- **visualize** (bool) Aktiviert bzw. deaktiviert die Visualisierung der Entscheidungsbäume.

Beim Leistungsprädiktor und Kategorisierer

Das Plugin wandelt die Aktivitäten in Daten um und speichert sie in einer tabellenähnlicher Struktur. Mit den Parametern featsCols and labelCol kann man die Spalten auswählen, die man für ein Experiment für sinnvoll hält.

Das Plugin wird im nächsten Kapitel ausgewertet.

Kapitel 7

7 Auswertung

Für die Experimente werden verschiedene Datensätze vorgestellt und dann werden damit verschiedene Experimente durchgeführt. Währenddessen enthüllen sich diverse Problembereiche. Sie werden nach dem Experimenten näher untersucht. Anschliessend wird eine weitere Experimenten-Serie durchgeführt, die zeigt wie die Bedingungen aussehen müssen, um die Aufgabe erfolgreich zu lösen.

7.1 Daten

Die Experimente wurden auf einem Cluster mit 20 Knoten durchgeführt. 10 E/A-Knoten sind jeweils mit einem Intel Xeon E3-1278@3.4 GHz, 16 GByte RAM und einer Seagate Barracude 7200.12 ausgestattet. Die Knoten sind mit einem Gigabit Ethernet miteinander verbunden und die Leistung einer HDD ist ca. 100 MiB/s. Die E/A-Knoten laufen mit CentOS 6.5 und Lustre 2.5.

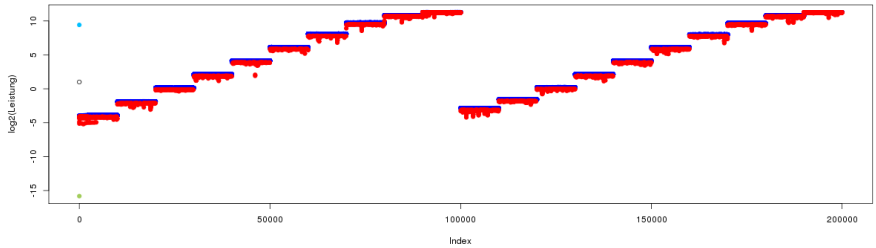
Die Datensätze wurden in zwei Schritten erstellt. Als erstes wurden E/A-lastigen Programme mit dem Tool “siox-inst” in Verbindung mit dem SIOX-Plugin “CSVReader” instrumentiert. Als Ergebnis entstanden jeweils eine Aktivitätensequenz und die dazugehörige CSV-Datei, in der jede Zeile Daten aus der entsprechender Aktivität enthält. Im zweiten Schritt wurden die CSV-Datei analysiert und jede Zeile hat ein Label mit dem Cachetypen erhalten.

Der Cachetyp wurde rechnerisch bestimmt. Dazu wurde zuerst der Overhead von dem Durchsatz abgezogen und dann echte Geschwindigkeit per Byte berechnet. Die Operationen mit den ähnlichen echten Geschwindigkeiten per Byte bilden gleiche Gruppen. Danach wurde das Wissen über die Cachehierarchie dazu benutzt, die Gruppen richtig zu bezeichnen, mit der Annahme, dass die Operationen mit niedrigsten Geschwindigkeiten per Bytes zu den obersten Cacheebenen gehören.

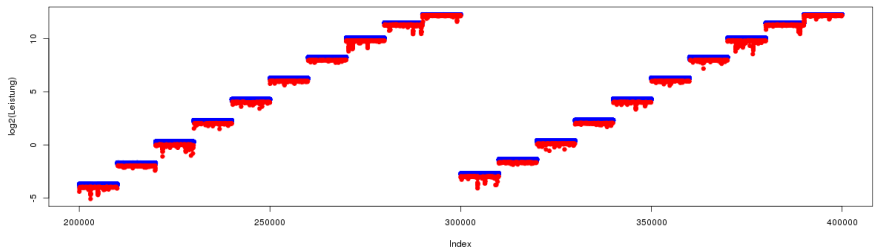
Während der Messungen wurden alle weiteren Prozesse auf das Minimum beschränkt, vor allem liefen keine weiteren E/A-intensiven Benutzerprozesse.

7.1.1 SCO-Datensatz

Der SCO-Datensatz wurde mit 20 verschiedenen Datengrößen aufgenommen. 10 davon sind Potenzen zu Basis 2 ($size_{i+1} = size_i \cdot 4$) und 10 sind keine ($size_{i+1} = size_i \cdot 4 + 1$), mit $size_0 = 1$ und $0 \leq i < 10$. Pro Datengröße wurden 10000 Messungen gemacht und der Wert gemittelt. Der Experiment wurde einmal für Leseoperationen und einmal für Schreiboperationen durchgeführt. Die Graphen in der Abb. 7.1 zeigen die Durchsatzwerte der Messreihe.



(a) Schreiboperationen.



(b) Leseoperationen.

Abbildung 7.1: Durchsatz von Operation mit verschiedenen Datengrößen. Die ersten 100000 Werte gehören zu den Datengrößen, die eine Potenz zu Basis 2 sind und die restlichen 100000 Werte gehören zu den, die es nicht sind.

Typ	Size
cached CPU	379472
cached Arbeitsspeicher	20527
schnelle E/A	1
unerwartet langsam	1

Tabelle 7.1: Verteilung der Daten für Lese- und Schreiboperationen.

7.1.2 DD-100- und DD-1024-Datensatz

Weitere Datensätze sind aus der Instrumentierung des Linuxtools **`dd`** entstanden. Zur Unterstützung wurde das s.g. Nulldevice (**`/dev/null`**) genommen. Das besondere an dieser Datei ist, dass die Zugriffe darauf keine echte E/A-Operationen sind. Es wird nur eine "leere" Funktion aufgerufen.

Für die Erzeugung der Datensätze der DD-100-Serie, wurde zunächst eine mit Nullen gefüllte Datei **`testfile`** mit der Größe $100bytes \cdot 100000 \approx 9.53MB$ erzeugt. Sie wurde dazu verwendet, um drei verschiedene Datensätze aufzunehmen. Im ersten Durchgang (Abb. 7.2) werden $100bytes$ grosse Blöcke 100000 mal gelesen und in das Nulldevice geschrieben.

```
1    siox-inst posix dd of=/dev/null if=testfile bs=100 count=100000
```

Im zweiten Durchgang (Abb. 7.2) werden $100bytes$ grosse Blöcke 100000 mal gelesen und in das Nulldevice geschrieben.

```
1    siox-inst posix dd if=/dev/zero of=testfile bs=100 count=1000000
```

Im dritten Durchgang (Abb. 7.4) wurde auch die Leerung von Cache verzichtet. $100bytes$ grosse Datenblöcke wurden 1000000 mal aus der Testdatei gelesen und in das Nulldevice geschrieben.

```
1    siox-inst posix dd of=/dev/null if=testfile bs=100 count=1000000
```

Die Datensätze der Serie DD-1024 wurden auf analoger Weise erzeugt. Sie sind in dargestellt in (Abb. 7.5), (Abb. 7.6) und (Abb. 7.7).

7.1.3 NCT-Datensatz

Der NCT-Datensatz wurde mit Hilfe von der NCT-Bibliothek [9] aufgenommen. Hierbei handelt es sich um Daten, die nicht kontinuierlich gelesen bzw. geschrieben wurden. In diesen Fall wurden $128kb$ gelesen, dann $256kb$ übersprungen.

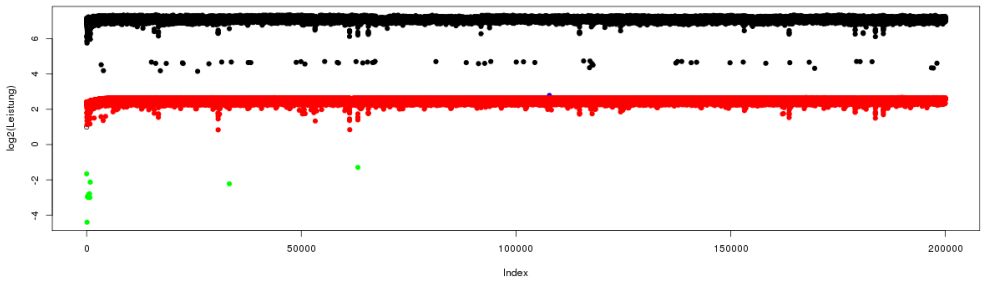


Abbildung 7.2: DD-R100

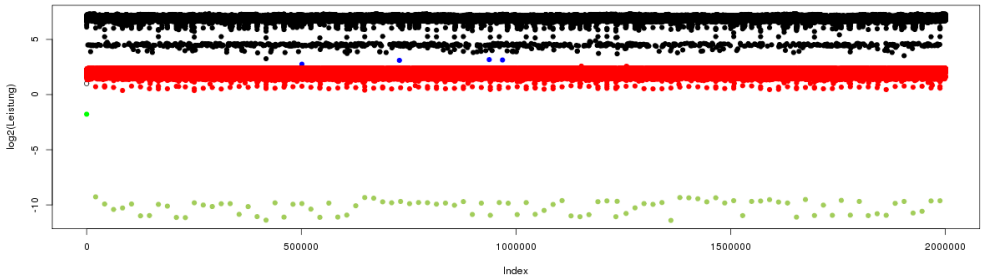


Abbildung 7.3: DD-W100

7.2 Leistungsvorhersage

Mit dem Leistungsprädiktor wurde ein Entscheidungsbaum trainiert. Als Features wurden benutzt die Datengröße, Dauer, Offset. Als Label wurde die Leistungswerte benutzt. Die Baumtiefe wurde auf 5 Ebenen beschränkt.

7.3 Kategorisierung von Daten

Das Ziel dieses Experimentes ist herauszufinden, wie ob man mit Hilfe von zur Verfügung stehenden Daten die Cachegruppen erlernen kann oder nicht. Gleich vorab, das Experiment wird scheitern. In diesem Abschnitt wird untersucht warum die gewählten Mitteln nicht geeignet sind, um die Aufgabe zu bewältigen.

Bei der Klassifizierung nach Kategorie wurden folgende Features benutze: Zeit zwischen, Operationsstart, Operationstyp, Datengröße, Dauer, Leistung. Es wurde

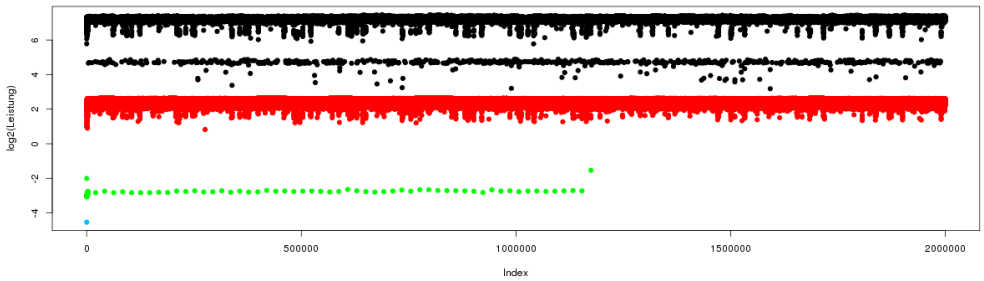


Abbildung 7.4: DD-R100C

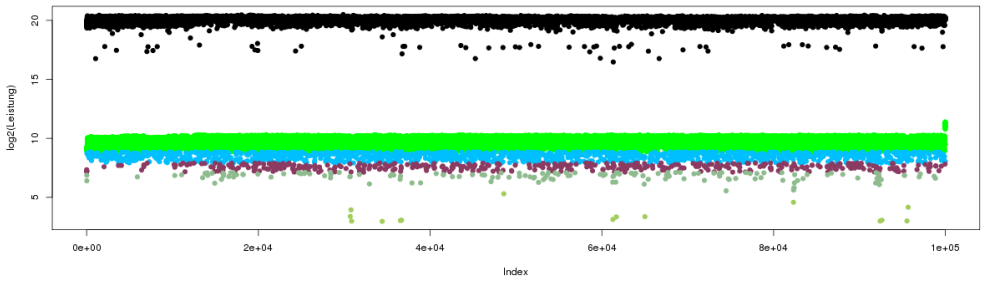


Abbildung 7.5: R1024K

ein Baum mit einer unbeschränkter Tiefe trainiert.

7.3.1 Analyse

Kategorisierung Das erste Problem ist, dass die Cachegruppen nicht klar trennbar sind. Das sieht man ab besten an der Abb. 7.12 . Der Clusteringalgorithmus fasst Features mit den kleinsten Abständen zu Gruppen zusammen. In der Abbildung kann man erkennen, dass verschiedene Cachetypen eine Gruppe bilden. Bei der Aufnahme von Daten hätten die Abstände so gewählt werden müssen, dass die Abstände zwischen den Datengrößen geringer sind als die Abstände zwischen den benachbarten roten und blauen Gruppen. Erst dann hätte der Clusteringalgorithmus eine Chance richtige Features die zu einem Cachetypen gehören in eine Gruppen zu bilden.

Baumkomplexität Aber auch in diesen Fall wäre es eine schwierige und unzu-

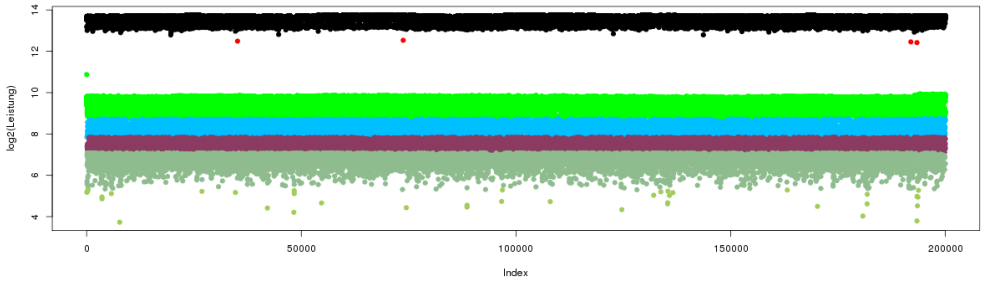


Abbildung 7.6: W1024K

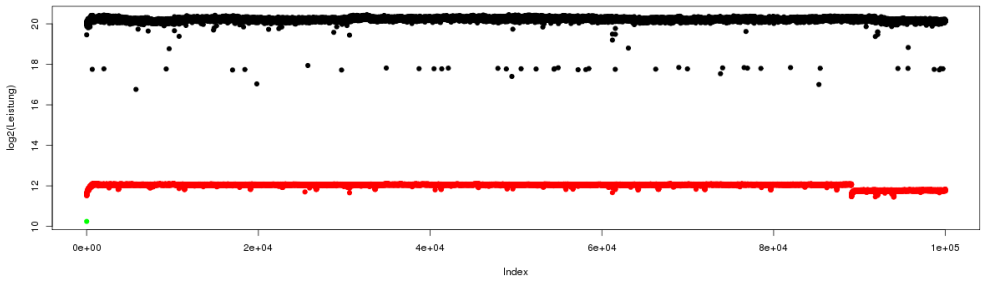


Abbildung 7.7: R1024KC

verlässige Methode, die nur durch einen sehr komplizierten Entscheidungsbaum zu realisieren wäre. Der Grund dafür liegt in dem Funktionsprinzip der Entscheidungsbäume. Die Trennung der Gruppen erfolgt nicht durch Funktionen, sondern stufenartig durch horizontale und vertikale Trennwände. Ideal wäre also, wenn sich alle Punkte die zu Cacheoperationen gehören auf einer Seite der Linie und die Punkte die zu Arbeitsspeicheroperationen gehören auf einer anderen Seite liegen würden. Wobei die Linie vertikal oder horizontal sein müsste. Andere Verfahren, die solche Fähigkeiten besitzen, wie z. B. die SupportVectorMachinen, wären aus dieser Perspektive besser geeignet.

Kategoriverknüpfung Betrachten wir aber nochmal in der Abbildung zwei Inseln. Auch wenn jede Insel sauber in zwei Gruppen getrennt werden könnte, hätten wir 4 Gruppen. Auf Insel 1 die Gruppen A und B und auf Insel 2 die Gruppen C und D. Wir wissen aber, dass es nur zwei Cachetypen gibt und müssten Paare bilden entweder AC BD oder AD BC. Hier wäre das Problem herauszufinden was die

Typ	R100	W100	R100C	R1024k	W1024k	R1024kC
verworfen	99999	999993	1000000	50000	99995	50000
cached CPU	13	4	-	-	-	-
cached Arbeitsspeicher	99987	999906	999903	-	-	49999
cached Massenspeicher	13	1	96	47990	77994	1
schnelle E/A	-	-	1	1412	6581	-
normale E/A	-	-	-	432	7665	-
langsame E/A	-	96	-	151	7723	-
unerwartet langsam	-	-	-	-	38	-
andere	-	-	-	15	4	-

Tabelle 7.2: Verteilung der Daten für Lese- und Schreiboperationen.

Typ	R128K-256K	W128K-256K	R128K-256KC
cached Arbeitsspeicher	3259	1285	25542
cached Massenspeicher	22278	23581	-
schnelle E/A	5	77	-
normale E/A	-	115	-
langsame E/A	-	272	-

Tabelle 7.3: Verteilung der Daten für Lese- und Schreiboperationen.

gleichen Gruppen sind und ob sie gleich sind.

Labelbezeichnung Die Gruppe kann zwar von dem Algorithmus gelernt werden, aber nicht richtig bezeichnet werden. Allein durch Gruppieren kann keine Informationen über die Cachekomponente herausbekommen. Man kann zwar über E/A-Zugriffsgeschwindigkeit gewisse Annahmen treffen, aber nicht mit einer 100%-ger Sicherheit sagen, dass die Annahme korrekt ist. SIOX-Aktivitäten verfügen momentan nicht über die Daten, die direkt ohne Umrechnung von kMeans-Clusteringalgorithmus in die richtigen Gruppen zugeordnet werden können. Das ist dennoch möglich, wenn der E/A-Pfad in seine Einzelteile zerlegt und in Features umgewandelt wird. In diesen Fall bilden die Cachetypen auf der Leistung/Byte Ebene eine zusammenhängende Gruppe, die vom Clusteringalgorithmus korrekt erkannt werden kann.

7.3.2 Schlussfolgerung

Die automatische Charakterisierung von Leistungswerten ist zumindest mit den hier gewählten Mitteln nicht sinnvoll. Die vier Gründe die momentan daran hindern weiter zu machen sind Unfähigkeit von kMeans-Algorithmus die aufgenommenen Daten richtig zu trennen.

Keine der Cachetypen bildet eine Gruppe, die vom Lernalgorithmus erkannt werden

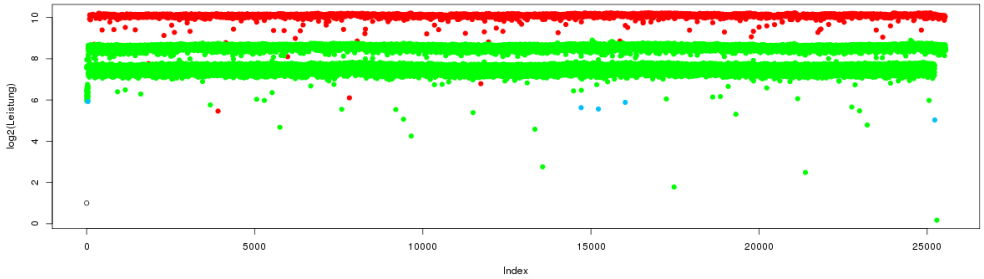


Abbildung 7.8: R128K-256K

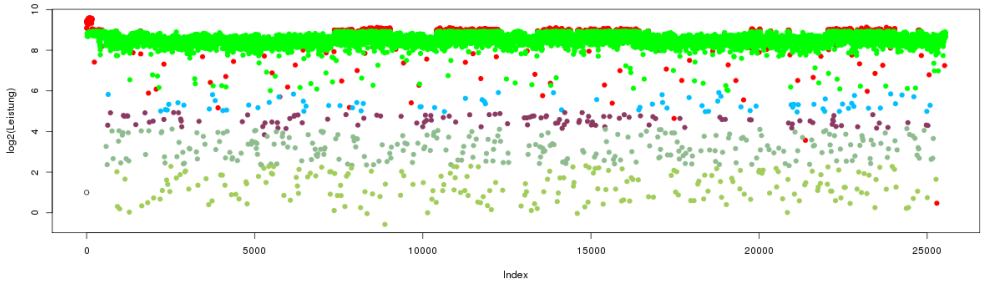


Abbildung 7.9: W128K-256K

kann. Jede Datengröße bildet eine Insel, die in zwei Gruppen unterteilt werden kann. Hauptsächlich ind CPU-Cached und Memory-Cached Gruppen. Der Clusteringalgorithmus sucht sich aber zusammen

Eine Vertiefung des Experiments würde in eine ganz anderen Forschungsbereich abgleiten, als in dieser Arbeit vorgesehen. Die Untersuchung des E/A-Pfades auf der Hardware- und im Betriebssystemebene.

7.4 Bedingungen für die automatische Charakterisierung

In diesem Unterabschnitt zeigt eine Serie der Experimente, wie die Daten strukturiert sein müssen und unter welchen Bedingungen eine automatische Charakterisierung von Leistungsdaten mit Hilfe von maschinellen Lernen funktionieren könnte.

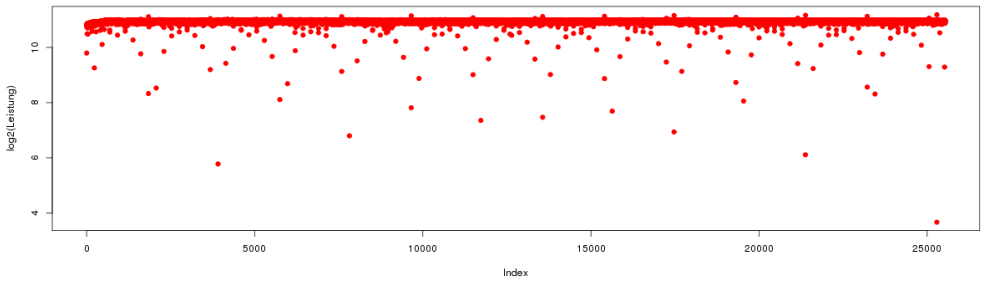


Abbildung 7.10: R128K-256KC

Die Ergebnisse sollen auch das Ziel etwas verdeutlichen.

Unter Annahmen, dass (a) die Cache-Typen vorhanden sind und (b) durch Umrechnung die separierbare Gruppen hätten. Könnte man unter diesen Voraussetzungen einfache Regeln erzeugen? In dem Unterabschnitt werden zwei weitere Experimente durchgeführt auf jeweils einem Datensatz aus der DD-100-, DD-1024-Serie. Bei allen Experimenten wurden die Entscheidungsbäume mit aktivierten CV-Pruning und ohne einer Begrenzung der Baumtiefe erzeugt.

7.4.1 R100C

Um die Frage zu beantworten führen wir einfach die Experimenten an den Datensätzen der 100-Serie und 1024-Serie.

Mit dem aktivierten CV-Pruning und unbegrenzten Baum Tiere erzeugt CART eine relativ kleinen Entscheidungsbaum Abb. 7.13.

Der Baum hat nur vier Blätter, die auch unterschiedliche Labels haben. Daraus lassen sich sofort vier Regeln erzeugen.

$$\text{cached memory} = \mathit{Duration} < 0.000054 \tag{7.1}$$

$$\text{cached storage} = 0.000803 > \mathit{Duration} >= 0.000054 \tag{7.2}$$

$$\text{fast I/O} = \mathit{Duration} >= 0.00083 \tag{7.3}$$

Der Vergleich von Originaldaten mit dem Entscheidungsbaum zu 99.8% reproduzieren. Der Entscheidungsbaum beschreibt das Leider kann der Duration-Parameter vom Nutzer nicht direkt kontrollieren, somit lassen damit zwar eine Statistik erstellen, aber der Nutzer kann mit diesen Wissen die Anwendung nicht optimieren. Zum Charakterisieren der Leistung ist die Dauer das wichtigste und das einzig benötigte Feature. Das Ergebnis zeigt nur das wir intuitiv vermuten würden. Das zeigt aber auch, dass die Zusammensetzung der Dauer analysiert und in seine Einzelteile gespalten werden muss, damit die Entscheidungsbaumalgorithmen besser lernen können.

7.4.2 W1024k

Im zweiten Experiment werden die Daten aus der DD1024-Serie analysiert.

$$\text{cached memory} = Duration < 0.000054 \tag{7.4}$$

$$\text{cached storage} = 0.000803 > Duration >= 0.000054 \tag{7.5}$$

$$\text{fast I/O} = Duration >= 0.00083 \tag{7.6}$$

$$\text{slow I/O} = Duration >= 0.00083 \tag{7.7}$$

$$\text{normal I/O} = Duration >= 0.00083 \tag{7.8}$$

Die Erkenntnisse aus den Experimenten haben gezeigt, dass die gewählten Mitteln aus dem Bereich des maschinellen Lernens nicht ausreichen, um das Problem zu lösen. Es wurde aber auch klarer was verbessert werden muss, damit es dennoch geht. Auf jeden kann die Arbeit erstmal nicht fortgesetzt werden und wir kommen mal zum Schlusswort.

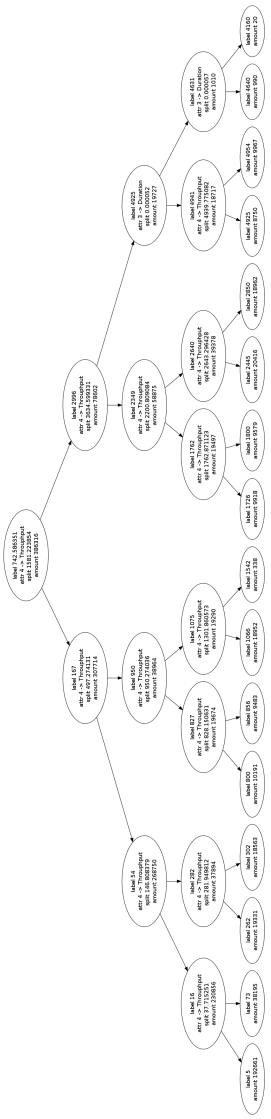


Abbildung 7.11: Entscheidungsbaum vom SCO-Datensatz. Der Kreuzvalidierungsfehler ist ca. 55.6%

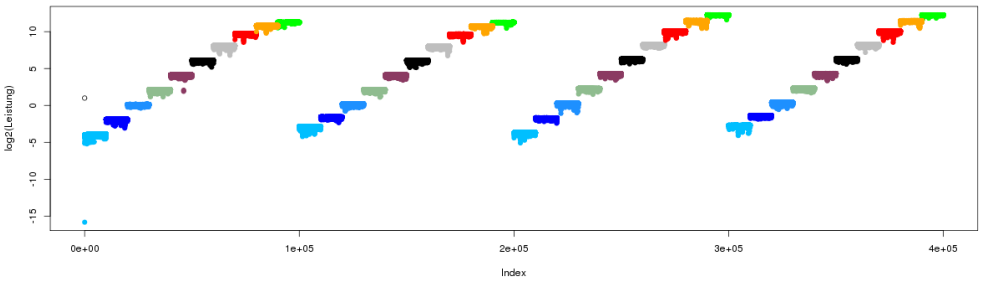


Abbildung 7.12: Kategorisierung des SCO-Datensatzes.

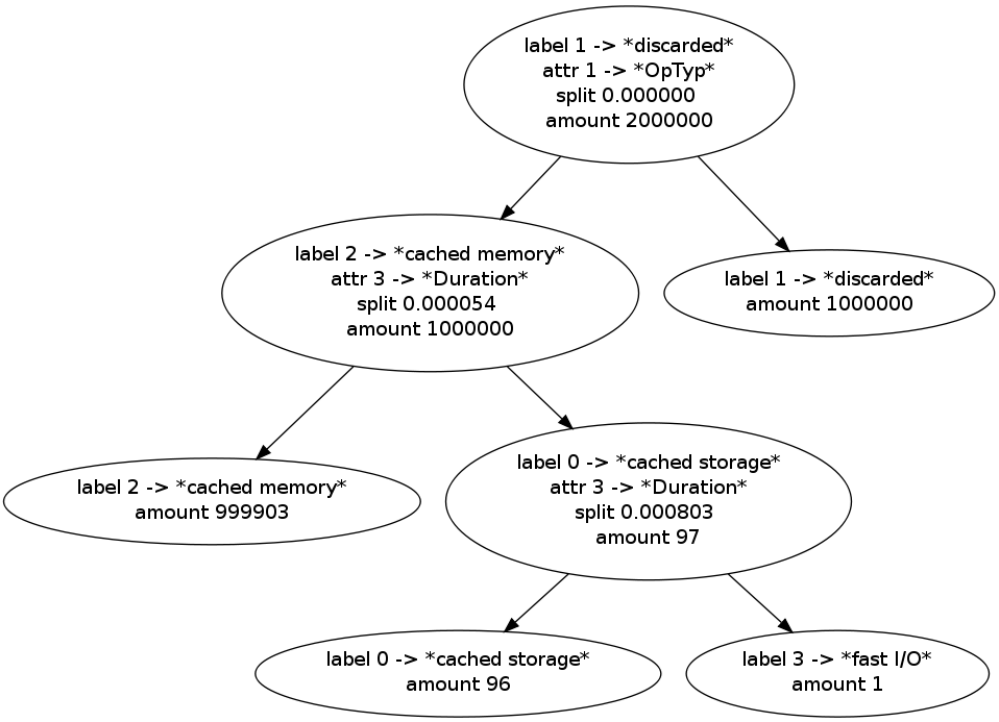


Abbildung 7.13: DD-R-100K-C-Entscheidungsbaum

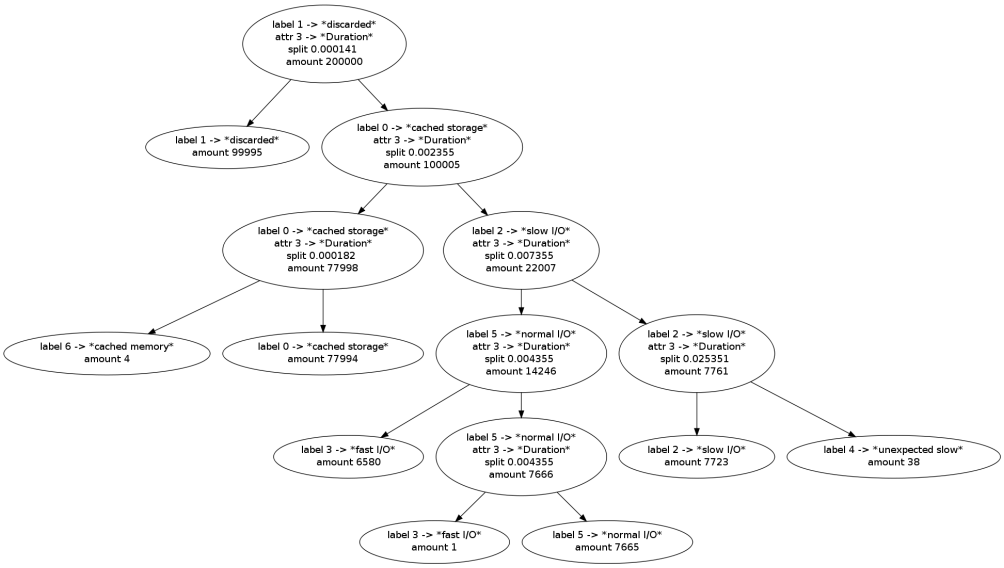


Abbildung 7.14: DD-R-100K-C-Entscheidungsbaum wurde mit aktivierten CV-Pruning und ohne einer Begrenzung der Baumtiefe erzeugt.

Kapitel 8

8 Zusammenfassung und Ausblick

Dieses Kapitel fasst die Arbeit zusammen und gibt ein Ausblick auf die möglichen Forschungsrichtungen.

8.1 Zusammenfassung

Die Leistungsanalyse und -optimierung im HPC-Bereich sind zwei wichtig Schritte in den Qualitätssicherungs- und Optimierungszyklen. Sie helfen Software zu entwickeln, die Hardware besser ausnutzt und E/A-Leistung steigert. Ohne Unterstützung von geeigneten Werkzeugen können diese beiden Aufgaben zur einer schwierigen Herausforderung für die Softwareentwickler werden. Die Entwicklung von Werkzeugen, die die Vorgänge, die bei der Ausführung von Software stattfinden, analysieren und für die Entwickler geeigneter Form präsentieren, ist deswegen notwendig.

Es gibt bereits eine Reihe von solchen Analysewerkzeugen. Sie spezialisieren sich in der Regel entweder auf der Offline- oder auf der Online-Analyse. Bei der Offline-Analyse werden zuerst die Daten gesammelt und dann ausgewertet. So funktioniert z.B. Vampir, der kostenpflichtiger Marktführer, mit sehr großen Funktionsumfang und Visualisierungsmöglichkeiten. Bei der Online-Analyse findet die Auswertung der Daten während der Programmausführung statt. So funktioniert z.B. Darshan, ebenfalls eine kostenpflichtige Software, dessen Stärke in Effizienz liegt. Sie macht ihn geeignet zur Lastcharakterisierung während des produktiven Betriebs. Das SIOX-Framework ist eine freie Software, die beide Analysearten beherrscht. Er befindet sich aber noch in Entwicklung und besitzt momentan einen bescheidenden Funktionsumfang. Das Funktionsumfang kann aber fast uneingeschränkt erweitert werden.

Im Rahmen dieser Arbeit wurde das SIOX-Plugin “KnowledgeExtractor” programmiert, der mit Hilfe von maschinellen Lernen die Leistung und Cachetypen bestimmen sollte. Die festgesetzten Ziele wurden aber aus folgende Gründen nicht erreicht:

- Die Datenpunkte liegen sehr ungünstig für die Entscheidungsbäume. Das führt zu sehr komplexen Bäumen.
- Die benachbarten Datenpunkte bestehen aus mehrere Cachetypen.
- Die Labels können nicht ohne weiteres bestimmt werden.

Die Ziele werden erst unter optimalen Bedingungen erreicht, die leider im normalen Betrieb nicht realistisch sind:

- Die Datenpunkte, die zum gleichen Cachetypen gehören, müssen einen zusammenhängende Cluster bilden.
- Features müssen mit Cachetypen belabelt sein.

Vermutlich kann die Bedingungen erfüllen, wenn man den Overhead aus den Durchsatzwerten entfernt und die echte Geschwidigkeit / Byte ausrechnet.

8.2 Ausblick

Die automatische Charakterisierung der E/A-Leistung mittels dem maschinellen Lernen ist erstmal an ihre Grenzen gestossen, zumindest mit den hier gewählten Algorithmen. Es wurde aber auch klar was als nächstes gemacht werden muss, damit die Arbeit fortgesetzt werden kann. Das ist:

- Zerlegung des Durchsatzes in seine Bestandteile.
- Suche nach weiteren Parametern, die von dem Benutzer gesetzt werden können.
- Entwicklung des robusten (wahrscheinlich Low-Level-) Verfahrens zur Bestimmung des Cachetypen.

Sobald diese drei Punkte erfüllt sind, kann das Verfahren unter optimalen Bedingungen fortgesetzt werden.

Literaturverzeichnis

[1] IDC, "The digital universe of opportunities: Rich data and the increasing value of the internet of things." http://germany.emc.com/leadership/digital-universe/2014iview/executive-summary.htm, 2014.

[2] Hewlett-Packard, "The machine: The future of technology," 2014.

[3] "Darshan hpc i/o characterization tool." http://www.mcs.anl.gov/research/projects/darshan/, 2015.

[4] P. Carns, "Darshan," in *High Performance Parallel I/O*, Computational Science Series, pp. 309–315, Chapman & Hall/CRC, 2015.

[5] "Vampir." http://www.paratools.com/Vampir, 2015.

[6] "Score-p." http://www.vi-hps.org/projects/score-p/, 2015.

[7] N. H. A. A. H. M. X. W. A. C. T. B. J. L. R. M. J. W. Julian Kunkel, Michaela Zimmer, "The siox architecture – coupling automatic monitoring and optimization of parallel i/o." 2014.

[8] SiSoftware, "Benchmarks : Intel mobile haswell (crystalwell): Memory subsystem." http://www.sisoftware.co.uk/?d=qa&f=mem_hsw, 2015.

[9] E. D. Zickler, "Optimization of non-contiguous MPI-I/O Operations." Online http://wr.informatik.uni-hamburg.de/_media/research:theses:enno_david_zickler_optimization_of_non_contiguous_mpi_i_o_operations.pdf, 01 2015.

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.

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

Hamburg, den 26.01.2015