TECHNISCHE UNIVERSITÄT DRESDEN

FAKULTÄT INFORMATIK
INSTITUT FÜR TECHNISCHE INFORMATIK
PROFESSUR FÜR RECHNERARCHITEKTUR
PROF. DR. WOLFGANG E. NAGEL

Proseminar “Rechnerarchitektur”

Veritas - ein Machine Learning-Framework für die Performance-Coverage-Analyse von Proxy-Applications

Philipp Matthes
(Mat.-Nr.: 4250691)

Hochschullehrer: Prof. Dr. Wolfgang E. Nagel
Betreuer: Ronny Tschüter, Christian Herold, Matthias Weber

Dresden, 7. Juni 2018

Inhaltsverzeichnis

1 Einleitung

Für die Planung von Hochleistungsrechnersystemen sollten die Eigenschaften der später auf diesen Systemen auszuführenden Anwendungen bereits beim Entwurf der Hardwarearchitektur berücksichtigt werden. Dafür können Benchmark-Kernel zusammengestellt werden, welche das Verhalten der Anwendungen repräsentieren - so genannte Proxy-Applications. Bei der Auswahl der Kernel muss jedoch auf eine vollständige Abdeckung der Anwendungseigenschaften geachtet werden. Machine Learning-Verfahren können den Nutzer unterstützen, eine geeignete Auswahl zu treffen.

Problemstellung Ein großes Problem bei der Verwendung von Proxy-Applications ist die Verträglichkeit der jeweiligen Proxy-Application mit den Anforderungen des zu erstellenden Systems, denn sie basieren ausschließlich auf der Intuition des Programmierers. Die Korrelation des Benchmarks mit den Anforderungen des Systems lässt sich über die Performance-Coverage (deutsch: Leistungsabdeckung) bestimmen. Das Team um Tanzima Z. Islam, Jayaraman J. Thiagarajan, Abhinav Bhatele, Martin Schulz und Todd Gamblin vom Center for Applied Scientific Computing am Lawrence Livermore National Laboratory in California entwickelte ein neuartiges Machine Learning Framework für diese Problemstellung. Veritas, wie es die Forscher nennen, soll die Leistungsabdeckung eines Benchmarks in Beziehung zum Hardwaresystem analysieren und Probleme in der Leistungsabdeckung finden. Im Rahmen ihrer Arbeit hat das wissenschaftliche Team des Lawrence Livermore National Laboratory Tests an den Benchmarks STREAM und DGEMM sowie an den Applikationen OpenMC und CMTnek mit deren Proxy-Applications durchgeführt.

Relevanz In einer Gesellschaft mit exponentiellem technologischem Anforderungswachstum und einer IT-Industrie, die sich immer mehr in Richtung Exascale bewegt (Abbildung 1), ist effektives Hardware-Software-Co-Design einer der wichtigsten Schwerpunkte in der Konstruktion von großen und kleinen Rechensystemen, um deren Rechenleistung und Energieeffizienz zu optimieren.
Die beste Möglichkeit, Hardwaresysteme auf die leistungstechnische Kompatibilität zum Anwendungsspektrum zu kontrollieren, ist die Prüfung mit der Anwendung selbst. In vielen Fällen jedoch lässt sich dies nicht praktikabel realisieren, da die Anwendungen entweder zu komplex und zu groß sind oder sich aufgrund anderer Probleme vor allem im Konzeptionsstadium nicht einsetzen lassen. Deshalb existiert heute eine Vielzahl an ähnlichen Stellvertreterapplikationen (Proxy-Applications), welche in Kombination genutzt werden können, um eine aussagekräftige Charakteristik über das Anwendungssystem zu erhalten.

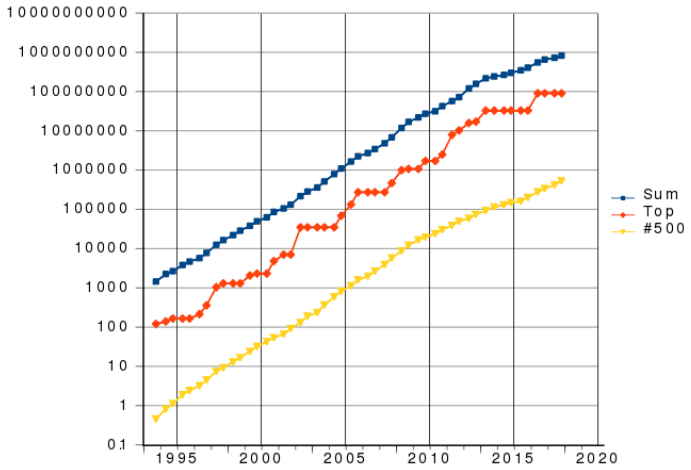


Abbildung 1: Die exponentielle Zunahme der Rechenleistung von Supercomputern zeigt sich besonders gut in der logarithmischen Darstellung. Supercomputer, die im Bereich von 10^{18} FLOPS rechnen, arbeiten im quantifizierenden Bereich der Exascale. Quelle: Von AI.Graphic - Eigenes Werk, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=33540287

Lösungsstrategie Die Kombination von Proxy-Applications stellt sich jedoch als schwierig heraus, denn sie hat vielen verschiedenen Heuristiken zu folgen. Machine Learning hat sich als Ansatz solcher komplexer Probleme vor allem in den letzten Jahren als praktikabel etabliert und verdrängt sukzessiv statisch programmierte Algorithmen in vielen Bereichen der Industrie. Insbesondere durch den Anstieg der Rechnerleistung und die Ermöglichung von hochparallelen Berechnungen auf Grafikkarten durch spezielle Schnittstellen wie CUDA führten dazu, dass das Machine Learning als Idee aus dem Jahr 1959 plötzlich für viele Anwendungen umsetzbar wurde. Vor allem künstliche neuronale Netze haben sich als gute Lösung für komplexe Probleme der Bildanalyse und der Vorhersage von Aktienkursen, aber auch für die Übersetzung von Sprachen etabliert. Auch wenn Machine Learning eines der mächtigsten Gebiete der modernen Informatik darstellt, soll sich diese Arbeit auch mit bekannten Problemen des Machine Learnings beschäftigen und sich auf mögliche Lösungen beziehen.

Ansatz Um Proxy-Applications nach ihrer Charakteristik auswählen zu können, hat das Team des Lawrence Livermore National Laboratory Strategien wie lineare Korrelation oder Hauptkomponentenanalyse (Abbildung 2) untersucht.

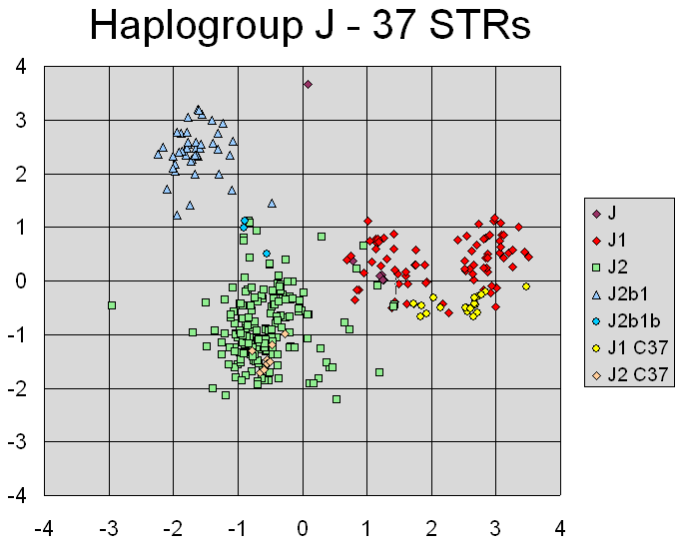


Abbildung 2: Machine Learning-Techniken wie Hauptkomponentenanalyse (Principal Component Analysis - PCA) und t-Distributed Stochastic Neighbor Embedding (t-SNE) sind mächtige Werkzeuge, um Zusammenhänge in multidimensionalen Datensätzen zu finden. Zu sehen: ein Datensatz eines speziellen Segments (Y-STR) von 354 Y-Chromosomen. PCA ist in der Lage, aus diesem Datensatz die verschiedenen genetischen Linien der Individuen zu bestimmen. Quelle: By User:Jheald - Own work. PCA calculated in R, plotted using Excel, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3619555

Auf der Grundlage dieser Algorithmen führen die Forscher vom Lawrence Livermore National Laboratory neue Machine Learning Techniken ein, um Proxy-Applications zu klassifizieren. Sie greifen Ideen aus der Machine Learning Theorie auf, um Kostenfunktionen zu definieren und entwerfen neue Metriken, um die Eignung der jeweiligen Proxy-Application festzustellen. Schließlich wird dieser Ansatz in Veritas, einem Machine Learning Framework implementiert, welches die Applikationen in ein internes Modell überführt, um schließlich deren Eignung zu quantifizieren. Genutzt werden hierbei verschiedene Kernel, welche in den Proxy-Applications integriert sind. Das Netzwerk fokussiert sich dabei hauptsächlich auf die Analyse einzelner Knotenpunkte (Single-Nodes), die Forscher schlagen jedoch vor, dass das Netzwerk auch zur Analyse anderer Charakteristiken bei der Kommunikation zwischen den Nodes eingesetzt werden könnte.

2 Proxy-Applications

Proxy-Applications sind gegenüber dem Deployment von Anwendungssoftware auf dem Zielsystem in mehreren Bereichen überlegen, denn sie sind flexibler, einfacher zu handhaben und benötigen weniger Expertise. Gleichzeitig wird die Repräsentativität der Tests nur gering beeinträchtigt, da alle wichtigen leistungsrelevanten Charakteristiken der Anwendungssoftware simuliert werden können.

2.1 Etymologie

In der Informatik wird unter Proxy meist ein System verstanden, welches Anfragen eines Systems auf ein anderes Ziel umleiten kann. Als Proxy-Applications jedoch sind im Kontext dieser Arbeit Stellvertreterapplikationen (von englisch proxy - Stellvertreter) gemeint, mit welchen stellvertretend, statt der letztendlich für ein System bestimmten Applikationen, Performance-Analysen durchgeführt werden können.

2.2 Genutzte Benchmarks und Proxy-Applications

In diesem Abschnitt sollen Anwendungen vorgestellt werden, welche später vom Veritas-Framework eingebunden werden können. Da es sich bei Veritas um ein Framework handelt, welches mit Anwendungen ausgestattet werden muss, um zu funktionieren, und um die Funktionsweise von Proxy-Applications zu verstehen, ist es wichtig, sich mehrere solcher Anwendungen und ihre Stellvertreter genauer anzusehen.

STREAM Benchmark Der STREAM Benchmark von John McCalpin wurde an der University of Delaware entwickelt. Als Angestellter bei SGI, IBM, AMD und im Rahmen der Analyse von Hauptsystemkomponenten des Texas Advanced Computing Center an der University of Texas in Austin flossen McCalpins Erfahrungen im Bereich erweiterter und neuartiger Computer-Architekturen ein. [McC95]
Als unabhängiges Projekt ist STREAM vor allem auch durch dessen breite Verfügbarkeit auf DOS, Windows95/98/NT, Linux und Power Macintosh Systemen gut als Benchmark geeignet. Mit dem STREAM Benchmark lässt sich die effektive Speicherbandbreite von einfachen vektorbasierten Systemen messen (Simple Vector Kernels). Der STREAM Benchmark ist dabei so gestaltet, dass die benötigten Rechenoperationen nicht zum Engpass werden, sodass die Speicherperformance des Systems möglichst störfrei getestet werden kann. Hierbei können Probleme in der Speicherarchitektur aufgedeckt werden, welche die Leistung des Systems drastisch verringern. Auf seiner Website zeigt McCalpin ein anschauliches Beispiel für die Relevanz seines Benchmarks und die Wichtigkeit der Speicherperformance: (Sinngemäß übersetzt aus dem Englischen) "[...] verschiedene aktuelle high-end Maschinen laufen auf einfachen arithmetischen Kernels für Operanden, welche sich außerhalb des Caches befinden, nur bei 4-5% ihrer eigentlichen Spitzenleistungen. Das heißt, sie verbringen 95-96% ihrer Zeit im Leerlauf, wartend auf die fehlenden Speicherblöcke."[McC07]

DGEMM Benchmark Der DGEMM Benchmark ist ein einfacher Benchmark, welcher Matrixmultiplikationen auf mehreren Threads durchführt. DGEMM steht dabei für double-precision matrix-matrix multiplication. An diesem Benchmark lässt sich gut zeigen, dass die Ausführung solcher Software im Unterschied zur Anwendungssoftware sehr einfach sein kann. Das Download-Verzeichnis des Crossroads/N9 DGEMM Benchmark des National Energy Research Scientific Computing Center des U.S. Department of Energy zum Beispiel besteht lediglich aus 3 Dateien: dem Makefile, dem Benchmark-File und einer Readme. Um den Benchmark zu starten, muss lediglich folgendes in die Kommandozeile eingegeben werden:

```
$ make
$ export OMP_NUM_THREADS=12
$ ./mt-dgemm
```

Der DGEMM Benchmark lässt sich auch so konfigurieren, dass er mit Matrizen einer bestimmten Größe rechnet. Mit dem folgenden Kommando lässt sich das Programm auf 4096x4096-Matrizen ausführen:

```
$ ./mt-dgemm 4096
```

Auf einem MacBook Pro Retina 2015 wird mit Standardeinstellungen Folgendes ausgegeben:

```
Matrix size defaulted to 256
Alpha =    1.000000
Beta  =    1.000000
Allocating Matrices...
Allocation complete, populating with values...
Performing multiplication...
Calculating matrix check...
============================================================
Final Sum is:         256.033333
 -> Solution check PASSED successfully.
Memory for Matrices:  1.500000 MB
Multiply time:        2.233222 seconds
FLOPs computed:       1010565120.000000
GFLOP/s rate:         0.452514 GF/s
============================================================
```

Genau wie im STREAM Benchmark lässt sich hiermit die effektive Floating-Point-Performance des Systems messen. Eine Besonderheit des DGEMM Benchmark liegt in seiner Ausführung. Bei Systemen, die aus einzelnen Knoten (Nodes) zusammengesetzt sind, wird der Benchmark einzeln auf jedem Node ausgeführt.

XSBench Benchmark und OpenMC Der XSBench Benchmark ist eine am Center for Exascale Simulation of Advanced Reactors (CESAR) am Argonne National Laboratory unter John Tramm, Ron Rahaman und Amanda Lund entwickelte Stellvertreterapplikation für die Simulation von Neutronen in Atomreaktoren über die Monte-Carlo-Methode mit OpenMC [TSIS] [Tra18]. Die Autoren des Artikels

A Machine Learning Framework for Performance Coverage Analysis of Proxy-Applications

stellen an dieser Stelle die folgenden zwei Fragen:

- Hat die Stellvertreterapplikation `XSBench` dieselbe Performance-Charakteristik wie `OpenMC`?
- Welche Unterschiede gibt es?

Weitere Benchmarks Die Benchmarks STREAM und DGEMM gehören nach der Definition der Alliance for Application Performance at Extreme Scale (APEX) zu den so genannten Micro-Benchmarks, denn sie prüfen das System auf einer elementaren Ebene, wie z.B. einem Aufruf an den Kernel. Im Folgenden eine Auswahl einiger Benchmarks aus einer Liste der APEX [APE18]:

- `IOR` für das Testen der Performance von parallelen Dateisystemen über verschiedene Schnittstellen und Zugriffsmuster
- `Mdtest` als Metadaten-Benchmark für weitere Operationen an Dateien und Verzeichnissen

Neben OpenMC, XSBench, STREAM und DGEMM nutzen die Forscher auch noch folgende Anwendungen:

- `RSBench` als möglicherweise effizientere Abwandlung des XSBench-Algorithmus.
- `CMTnek` zur Simulation von unter hohem Druck stehenden Partikeln, sowie die dazugehörige Stellvertreterapplikation CMTbone. (Weiter erläutert in Punkt 5)

3 Machine Learning-Methoden

Dieser Abschnitt stellt verschiedene Techniken vor, welche von den Forschern eingesetzt werden, um das Framework Veritas zu realisieren. Dazu werden sowohl mathematische und theoretische Grundlagen, Algorithmen, als auch Probleme bei der Realisierung beleuchtet.

3.1 Merkmalsselektion über Sparse Coding

Ein wichtiger Teil des Machine Learning ist die Vorverarbeitung der Daten. Aurélien Géron, der Autor von

Praxiseinstieg Machine Learning mit Scikit-Learn & Tensorflow [Gé18]

beleuchtet in seinem Buch anschaulich, warum das Problem der Vorbereitung der Daten von großer Relevanz ist:

> "Komplexe Modelle wie Deep-Learning-Netze können subtile Muster in den Daten erkennen. Wenn aber der Trainingsdatensatz verrauscht oder zu klein ist (wodurch die Stichprobe verrauscht ist), entdeckt das Modell Muster im Rauschen selbst. Diese Muster lassen sich natürlich nicht auf neue Daten übertragen. Nehmen wir beispielsweise an, Sie stellen Ihrem Modell für die Zufriedenheit [der Personen in einem Land] viele weitere Merkmale zur Verfügung, darunter wenig informative wie den Namen des Lands. Ein komplexes Modell

> könnte dann herausfinden, dass alle Länder mit einer Zufriedenheit über 7 ein w im Namen haben: [...]. Wie können Sie sicher sein, dass diese W-Regel sich auf Rwanda oder Zimbabwe anwenden lässt? [...] [D]as Modell ist nicht in der Lage zu entscheiden, ob ein Muster echt oder durch das Rauschen in den Daten bedingt ist."

Es existieren viele Möglichkeiten, Daten für das Machine Learning vorzubereiten. Gängige Methoden sind zum Beispiel:

- `Data Augmentation` zur Erweiterung des Datensatzes. In manchen Fällen kann hier sogar ein kreatives neuronales Netz neue Daten erzeugen.
- `Scaling` zur Beschränkung eines Datensatzes auf einen bestimmten Wertebereich. Viele Machine Learning Algorithmen reagieren empfindlich auf sehr große und sehr kleine Werte.

Zur Vorbereitung von Daten ist es häufig sinnvoll, diese auf wichtige Merkmale zu reduzieren. Das folgende Beispiel erläutert dieses Problem:

> Stellen Sie sich einen Datensatz vor, der verschiedene Merkmale von Häusern und Wohnungen innerhalb eines Bezirkes enthält. Ihre Aufgabe ist es, einen Machine Learning Algorithmus zu entwickeln, der aus diesen Merkmalen den Preis des Hauses oder der Wohnung vorhersagt. Hierbei gibt es nun Merkmale, die direkt mit dem Preis zusammenhängen, zum Beispiel die Anzahl an Schlafzimmern. Es gibt auch Merkmale, die indirekt mit dem Preis zusammen hängen, wie die Kriminalitätsrate. Merkmale wie das Gefälle des Geländes, auf dem das Haus steht, haben jedoch wahrscheinlich kaum einen Zusammenhang mit dem Preis des Hauses oder der Wohnung. Deshalb können Sie Merkmale wie das Gefälle des Geländes einsparen, um die Komplexität des Datensatzes zu verringern und seine Varianz dabei hinsichtlich des Zielmerkmals hoch zu halten. Infolgedessen lässt sich Ihr Machine Learning Algorithmus schneller und effizienter trainieren.

Nicht immer lassen sich, wie im gezeigten Beispiel, Datensätze direkt reduzieren. Für diesen Zweck können Algorithmen wie PCA genutzt werden, welche den hochdimensionalen Datensatz auf einen niederdimensionalen Raum projizieren können, indem sie diejenige Hyperebene finden, die den Datenpunkten am nächsten liegt. Hiermit lassen sich auch Zusammenhänge zwischen komplexen Datensätzen grafisch visualisieren, indem der hochdimensionale Datensatz auf einen drei- oder zweidimensionalen Raum projiziert wird. Für die Auswahl der Merkmale und die Reduzierung dieser auf einen niederdimensionalen Unterraum nutzen die Forscher zusätzlich das so genannte Sparse Learning. Der Vorteil des Sparse Coding besteht darin, dass sich dieses Verfahren bei hochdimensionalen Datensätzen besonders robust gegenüber rauschbehafteten Daten verhält. Das Sparse Coding kommt aus dem Bereich der Autoencoder. Autoencoder sind neuronale Netzwerke, die hochdimensionale Daten effizient in niederdimensionale Daten überführen können, indem sie lernen, wichtige Merkmale der Ausgangsdaten zu behalten und unwichtige Merkmale zu rationalisieren. Dazu werden die Daten zunächst mit einem Encoder kodiert und dann aus der effizienten Repräsentation im Decoder rekonstruiert. Problematisch hierbei ist, dass der Autoencoder seine Eingaben nicht direkt kopieren darf, sondern Merkmale aus den Daten extrahieren soll, welche er dann zur Kompression verwendet. Um sicher zu gehen, dass der Autoencoder seine Eingaben nicht kopiert, wird die Anzahl der aktiven Neuronen im Autoencoder reduziert. Diese Technik nennt man Sparse Coding (sparse engl. - spärlich). Auf diese Weise lässt sich sichergehen,

dass die wenigen noch aktiven Neuronen auch effizient Merkmale kodieren. Géron beschreibt dies sehr anschaulich:

> "Dadurch repräsentiert am Ende jedes Neuron ein wichtiges Merkmal (wenn Sie nur ein paar Wörter pro Monat sprechen dürften, würden Sie diese auch gut wählen, sodass sich das Zuhören lohnt)."

3.2 Klassifizierung und Bewertung

Im Einsatzbereich des Veritas Framework, dem High Performance Computing, kann es viele verschiedene Faktoren geben, welche die Performance des Systems beeinflussen können. Deshalb ist es selbst nach der Komprimierung des Merkmalssatzes schwierig, diesen zu untersuchen und wissenschaftliche Beweiskraft zu extrahieren. Infolgedessen nutzen die Forscher die Evidenztheorie von Dempster und Shafer, um die Merkmale aus den unterschiedlichen Quellen unter Berücksichtigung der Verlässlichkeit dieser Quellen zu einer Gesamtaussage zu kombinieren, welche einfach zu verstehen ist und eine möglichst hohe Beweiskraft trägt.
Um die Metriken sinnvoll und intuitiv bewerten zu können, werden diese in semantische Gruppen statisch und über Substring Matching direkt nach ihren Betriebsmitteln eingeordnet. Mit diesen sortierten Ressourcen lassen sich diese nun nach ihrer Signifikanz aufschlüsseln, indem häufig genutzte Ressourcen nach der Anwendung vorhergesagt werden und die Kompatibilität von Stellvertreterapplikation und der Applikation selbst geprüft wird. Auf diese Weise lassen sich Flaschenhälse voraussagen und geeignete Stellvertreterapplikationen finden. Als zuverlässiger Indikator der Performance von Großrechensystemen wird hierzu der parallele Effizienzverlust (Gleichung 1) kalkuliert, denn speziell bei Anwendungen, die viele Kerne nutzen, bildet dieser Indikator aufgrund des direkten Zusammenhanges zwischen paralleler Effizienz und Geschwindigkeit den Geschwindigkeitsverlust gut ab.

$$efficiency\_loss(p) = 1 - \frac{T(1)}{T(p) \times p} \tag{1}$$

wobei $p =$ Anzahl von Tasks und
$T(p) =$ Ausführungszeit bei Nutzung von p Tasks

Um eine Konstellation von Merkmalsbewertungen zu finden, welche die Zieleigenschaft (efficiency_loss) am besten darstellt, lassen sich bestimmte Algorithmen nutzen. Die Korrelation der Daten über den R^2-Wert kann nicht sicher als Indikator für eine gute Vorhersage genutzt werden, da Korrelation nicht zwangsweise einen kausalen Zusammenhang beinhaltet. Methoden wie das bereits erläuterte PCA oder auch CCA (Canonical Component Analysis) sind zwar gut für rauschbehaftete Daten geeignet, erstellen jedoch neue Merkmale durch die Kombination der Ursprungsmerkmale, was die Aussagekraft der Daten verringert. Die Forscher verwenden stattdessen sparse representations, welche besonders gut bei rauschbehafteten Daten funktionieren und auch Anwendung in der Bildbearbeitung, der Signalverarbeitung und medizinischen Anwendungen in der Bildverarbeitung finden. Die Resistenz des genutzten Verfahrens gegenüber Rauschen ist sehr wichtig, da durch die Komplexität von Hochleistungsrechnern mehrere Störeinflüsse gegeben sind. Diese zu reduzieren ist vor dem von den Forschern gewählten Machine Learning

Ansatz gut über sparse representations realisierbar und besonders wichtig, da die genutzten Algorithmen sonst möglicherweise Muster im Rauschen erkennen. Die Errechnung der Repräsentationen stellt sich als NP-schweres Problem heraus, benötigt also unter Umständen sehr viel Zeit. Deshalb ist es hier effizienter, mit Näherungsalgorithmen zu arbeiten. Das Forscherteam nutzt Orthogonal Matching Pursuit (OMP). OMP beherbergt jedoch den Nachteil, dass Metriken mit denselben Mustern zusammengefasst werden. Um dies zu verhindern, hat das Forscherteam den OMP-Algorithmus zu einem Ensemble-OMP-Algorithmus weiterentwickelt, indem dieser durch stochastische Faktoren erweitert wird. Mit den Repräsentationen der Merkmale über sparse representation sollen die richtigen Merkmale gefunden werden. Um dies zu realisieren, entwickelten die Forscher einen neuen Algorithmus, Resource Significance Measure (RSM). Der RSM-Indikator für eine Ressource wird dabei direkt aus den Sparse Representations gewonnen, indem die Wichtigkeit jeder Metrik aufgeschlüsselt wird. Um die Ergebnisse besser interpretieren zu können, wird der RSM-Indikator zwischen null und eins normalisiert.

3.3 Extraktion des Abdeckungsindikators

Mit diesem Messinstrument lassen sich letztendlich die Zusammenhänge zwischen Hardware und Performance modellieren und mit der jeweiligen Proxy-Application vergleichen. Hierfür werden für beide Merkmalsräume über das bereits erwähnte Verfahren PCA niederdimensionale Räume konstruiert, und deren Abstand über eine geodätische Linie (Abbildung 3) gemessen.

Abbildung 3: Zur Abstandsmessung zwischen zwei Punkten auf der Erde zieht man eine so genannte Geodäte oder geodätische Linie zwischen den zwei Punkten. Sie entspricht dem kürzesten Weg im Betrachtungsraum. Auch für die Abstandsmessung der erzeugten PCA-Räume lässt sich eine solche Geodäte konstruieren. So genannte Grassmann-Mannigfaltigkeiten werden als Berechnungsmodell verwendet. Quelle: Von Orion 8 - Eigenes Werk, based on File:Globe Atlantic.svg., Gemeinfrei, https://commons.wikimedia.org/w/index.php?curid=29671836

Diesen Abstand nennen die Forscher resource subspace dissimilarity measure (RSDM). Je kleiner dieser Indikator, desto größer ist die Kompatibilität von Proxy-Application und der abzubildenden Anwendung auf dem System. Die dazu verwendete Hauptkomponentenzerlegung (PCA) lässt sich leicht konfigurieren, sodass beide hochdimensionale Merkmalsräume auf einen n-dimensionalen Raum projiziert werden. Hierbei wird das n so gewählt, dass die erzeugten Räume nicht zu niederdimensional, jedoch auch nicht zu hochdimensional sind, um eine vollständige Orthogonalität der Räume zu verhindern und eine gute Aussagekräftigkeit des Indikators RSDM zu gewährleisten. Das eigentliche Ziel des Modells ist die zuverlässige Vorhersage von geeigneten Proxy-Applications. Zu diesem Zweck werden die gewonnenen Merkmale zu einer Gesamtaussage über die Performance-Coverage kombiniert. Hierzu nehmen die Forscher den gewonnenen RSDM-Messwert, normalisieren ihn (zwischen 0 und 1) und extrahieren den Coverage-Wert (Gleichung 2) für eine gegebene Ressource.

$$coverage(r) = 1 - RSDM(r) \tag{2}$$

wobei $r =$ Gegebene Ressource

Der Indikator coverage lässt sich nun einfach, zum Verständnis auch als Prozentzahl, interpretieren:

- Passen die PCA-Räume nicht zusammen, ist der RSDM-Wert hoch, und die Coverage gering.
- Sind die PCA-Räume sehr ähnlich, ist der RSDM-Wert gering, und die Coverage hoch.

4 Umsetzung im Veritas-Framework

Zur Transkription der diskutierten Algorithmen und Verfahrensweisen implementieren die Forscher das so genannte Veritas-Framework, welches dem Nutzer geeignete Proxy-Applications zu seiner Anwendung vorschlägt. Um dies unter Berücksichtigung der Minimierung von Fehleranfälligkeit und Maximierung der Skalierbarkeit werden die verantwortlichen Funktionalitäten im Veritas-Framework in Module untergliedert (Abbildung 4).

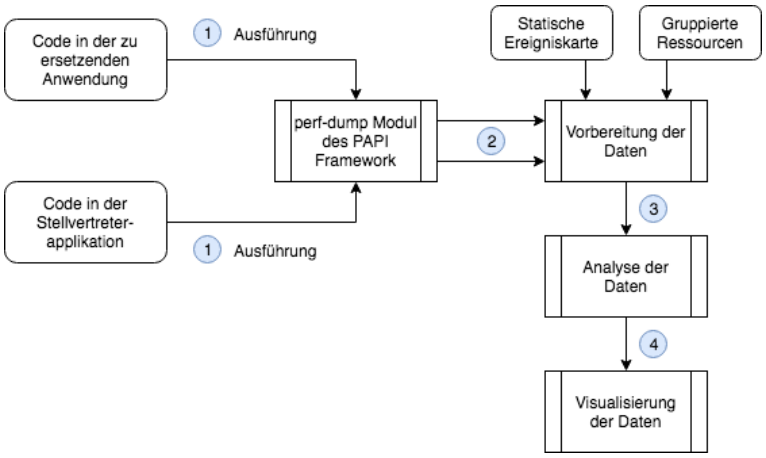


Abbildung 4: Diese Grafik zeigt den Informationsfluss im Veritas-Netzwerk. Quelle: Eigenes Werk, basierend auf der Grafik im Dokument Veritas [ITB+16]

Beginnend mit Schritt 1 (Abbildung 4) wird sowohl der effektive Code in der zu ersetzenden Anwendung (Ursprungsanwendung) als auch der wirksame Code in einer Proxy-Application ausgeführt. Bei der Ausführung werden spezifische Performance-Messwerte wie Speicherzugriffsfehler durch das Modul perf-dump aufgenommen und abgelegt.
Die Vorbereitung der durch perf-dump erzeugten Daten stellt sich als Problem heraus, denn die Daten und die funktionalen Gruppen sind zum Teil systemspezifisch. Das in `Python` programmierte Vorbereitungsmodul löst dieses Problem. Am Intel Blue Gene/Q (BG/Q) HPC zum Beispiel gibt es das Ereignis `PEVT_L1P_BAS_*_STALL_*`. Wie im Punkt 3.2 erläutert, sollen diese Ereignisse jedoch in semantische Gruppen gruppiert werden. Um dies zu gewährleisten, wird eine Statische Ereigniskarte erstellt, welche Ereignisse des spezifischen Systems der entsprechenden Gruppe zuordnet. Zur Gruppierung werden folgende Heuristiken verwendet:

1. Daten, die sich direkt auf Zugriffe (`access`) und Treffer (`hit`) berufen, werden direkt ihrer Ressource zugeordnet.

2. Daten, welche sich hingegen auf Zugriffsfehlschäge (`miss`) berufen, werden der zuliefernden Architektur zugeordnet, da die verfehlten Daten aus ebendieser geholt werden müssen.

3. Daten, die sich auf Zugriffsfehlschläge des Transaction Lookaside Buffers (TLB) berufen, werden gesondert behandelt, da die daraus resultierende Latenz spezifisch vom System abhängt.

4. Daten, welche sich auf verzögerte Ressourcen berufen (`stall`), werden genau wie Zugriffsfehlschläge auf die verantwortliche darüber liegende Architektur abgebildet.

In der folgenden Tabelle sind Beispiele für die Zuordnung nach einer Statischen Ereigniskarte zu sehen.

Heuristik	perf-dump Output	Bedeutung	Zugeordnete Ressource
1	`PAPI_L1_DCA`	L1-Cache Zugriff	`L1`
2	`PAPI_L1_DCM`	L1-Cache Miss	`L2`
3	`PAPI_TLB_DM`	TLB-Buffer Miss	`TLB`
4	`PEVT_L1P_BAS_*_STALL_*`	L1-Prefetcher Stall	`L2`

Zusätzlich führt das Vorbereitungs-Modul noch eine Durchschnittsberechnung der gruppierten Metriken durch. Die vorbereiteten Daten werden nun an das in Matlab implementierte Analyse-Modul in Schritt 3 (Abbildung 4) weitergegeben, welches den in Punkt 3.2 erklärten Indikator RSM für jede gruppierte Ressource berechnet.
Die gewonnen Erkenntnisse werden dem Nutzer nun in Schritt 4 (Abbildung 4) über das Visualisierungsmodul präsentiert. Das Visualisierungsmodul bietet dem Nutzer verschiedene Dialoge:

- Eine übersichtliche Darstellung der allgemeinen Abdeckung der einzelnen Ressourcen
- Eine detaillierte Aufschlüsselung der Abdeckung nach Auslastung
- Eine Auflistung der Ressourcen, die nach dem Glauben des Netzwerks relevant sind

Im Folgenden ein Beispiel für die visuelle Darstellung der gewonnenen Erkenntnisse des Modells:

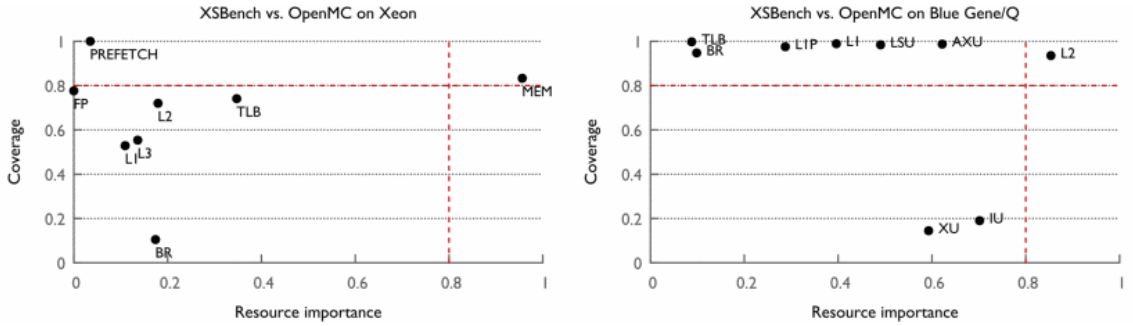


Abbildung 5: Der Nutzer kann aus der Darstellung mehrere Informationen entnehmen. In der Abbildung sieht man die Ressourcen aufgeschlüsselt nach ihrem RSM-Wert (Resource importance) in Relation zur Abdeckung der Proxy-Application (Coverage) auf zwei unterschiedlichen Systemen (Xeon links und BG/Q rechts). Im Diagramm markiert ist die Grenze von 80%, ab der nach den Forschern mit einer guten Abbildung der Wirklichkeit zu rechnen ist. Quelle: Figure-6 im Dokument Veritas [ITB+16]

Über dieses Diagramm lässt sich die Eignungsvoraussage für die Proxy-Application direkt ablesen. Nach der Interpretation des linken Diagramms aus Abbildung 5 lässt sich vermuten, dass die Leistung des Xeon-Systems offenbar stark von der Ressourcen-Gruppe MEM abhängt und der XSBench diese Ressource gut abdeckt. Ressourcen wie BR (Branch unit), L1 und L3 haben einen geringeren Coverage-Wert, scheinen allerdings weniger relevant zu sein. Im direkten Vergleich mit dem Diagramm rechts sagt das Framework offensichtlich eine bessere akkumulierte Coverage des XSBench Benchmarks voraus, fast alle Ressourcen erreichen einen Coverage-Wert über 80%. Bei den Ressourcen XU (Execution Unit)

und IU (Instruction Unit) jedoch zeigt sich eine schlechte Abdeckung bei hoher Signifikanz. Bei der Verwendung von XSBench auf dem BG/Q sollte also außerdem auf eine Proxy-Application zurückgegriffen werden, welche diese Ressourcen gut abdeckt. Die Visualisierung lässt sich auch verwenden, um kritische Ressourcen zu identifizieren. Während für die Performance der OpenMC Software auf dem Xeon-System die Ressourcen-Gruppe MEM kritisch zu sein scheint, hängt die Performance der Software auf dem BG/Q-System stark von der Ressourcen-Gruppe L2 ab.

5 Validierung des Frameworks in der Praxis

Die Validität des Veritas Frameworks wird von den Forschern an den in Punkt 2 vorgestellten Kombinationen aus Anwendung und Stellvertreter getestet.

Ergebnisse bei der Validierung mittels STREAM Der STREAM-Benchmark besteht aus vier verschiedenen Operationen (Kernels) auf Zahlen, die aus Vektoren geholt werden:

- `ADD` zur Addition
- `SCALE` zur Multiplikation mit einem konstanten Faktor
- `COPY` zur Zuweisung
- `TRIAD` zur Kombination von `ADD`, `SCALE` und `COPY`

Nach der Ausführung des Modells auf diesem Benchmark schlüsseln die Forscher die Kernels nach der verwendeten Ressourcen-Gruppe auf:

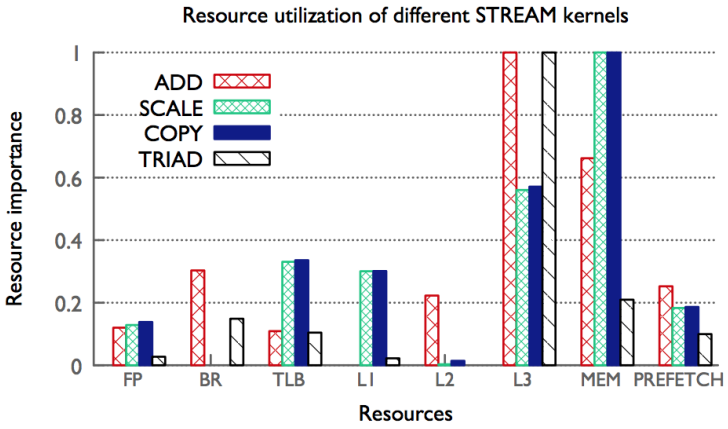


Abbildung 6: Aus dieser Abbildung lassen sich mehrere Informationen entnehmen. Die Ressourcen L3 und MEM scheinen die nach RSM wichtigsten Ressourcen zu sein. Die Unterschiede zwischen den Operationen können stellenweise jedoch groß sein. Quelle: Figure-3 im Dokument Veritas [ITB+16]

Diagramm 6 zeigt, dass die Operationen ADD und TRIAD, sowie COPY und SCALE aufgrund ihres paarweise ähnlichen Aufbaus auch eine ähnliche Auswirkung auf den RSM-Indikator haben. Deshalb testeten die Forscher die einzelnen Methoden gegeneinander und ermittelten den nach Ressourcen aufgeschlüsselten Coverage-Indikator:

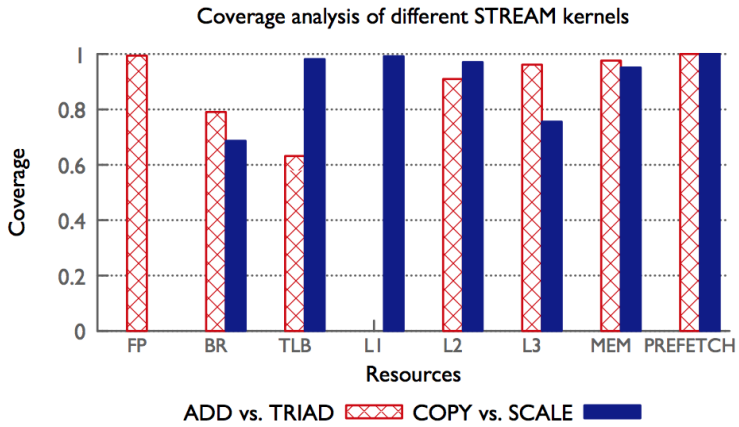


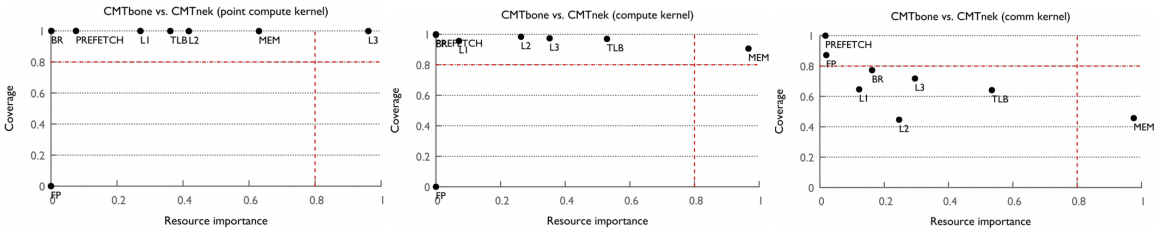
Abbildung 7: Da sich die Operationen paarweise in ihrem Aufbau ähneln, ist vice versa auch ein geringer RSDM-Wert zu erwarten. Genau dies zeigt die Aufschlüsselung nach dem aus der Berechnung des RSDM-Wert ermittelten Coverage-Indikator. Das Netzwerk scheint erfolgreich die internen Zusammenhänge der kohärenten Operationen zu erkennen, ohne den eigentlichen Aufbau dieser zu kennen. Deutlich wird dies durch einen akkumuliert hohen Coverage-Wert jeweils zwischen ADD/TRIAD und COPY/SCALE. Quelle: Figure-4 im Dokument Veritas [ITB+16]

Um dieses Ergebnis zu bestätigen, führten die Forscher weitere Tests an den Benchmarks DGEMM, XSBench (Abbildung 5) und RSBench aus, welche alle vielversprechende Ergebnisse lieferten [ITB+16]. Als gutes Veranschaulichungsbeispiel soll die Validierung mittels CMTbone und CMTnek dienen.

Veranschaulichungsbeispiel CMTbone und CMTnek Die Stellvertreterapplikation CMTbone soll das Verhalten der Applikation CMTnek auf dem Anwendersystem simulieren. Dazu sind in CMTbone mehrere Kernel implementiert:

- `point_compute_kernel` zur Simulation von Integralberechnungen über Vektormultiplikationen
- `compute_kernel` zur Simulation von Partikelbewegungen über Matrixmultiplikationen als Kernbestandteil von CMTnek, dem deshalb große Relevanz zukommt
- `comm_kernel` zur Simulation von Oberflächenflüssen über speicherintensive Prozesse der CMTnek Anwendung

Bei der Performance-Coverage-Analyse dieser Kernel sind die Forscher auf ein Problem von CMTnek gestoßen, visualisiert durch folgende Diagramme:



(a) `point_compute_kernel` (b) `compute_kernel` (c) `comm_kernel`

Abbildung 8: Während die beiden Kernel `point_compute_kernel` und `compute_kernel` einen akkumulierten Coverage-Wert nahe 1 besitzen, scheint `comm_kernel` aus CMTbone nur bedingt das Verhalten von CMTnek abzubilden. Besonders Ressourcen, die zur Speicherarchitektur zugeordnet werden können, sind nicht ausreichend abgedeckt, wohingegen die Proxy-Application CMTbone genau an dieser Stelle durch eine geringere Rechenauslastung und höhere Speicherauslastung profitieren müsste. Quelle: Figure-8, Figure-9, Figure-10 im Dokument Veritas [ITB+16]

Die festgestellte Diskrepanz im Kernel `comm_kernel` von CMTbone hat dazu geführt, dass sich dieser nun in der Weiterentwicklung befindet, um die Performance-Coverage zu verbessern.

6 Fazit

Das Prinzip von Proxy-Applications ist die Emulation des Verhaltens von Anwendungen, welche meist zu komplex sind, um ohne größeren Aufwand auf einem System ausgeführt werden zu können. Mit einer guten Auswahl von Proxy-Applications lassen sich Informationen über die Performance und eventuelle Engpässe insbesondere von Hochleistungsrechnern ermitteln. Um Stellvertreterapplikation und Endanwendung zu vergleichen, ist meist eine Kenntnis über den detaillierten Aufbau beider Applikationen notwendig. Das von den Forschern implementierte Framework Veritas kann diesen Prozess vereinfachen und beschleunigen, indem es Methoden und Algorithmen implementiert, welche die Kovarianz zwischen Hardwareressourcen und Software nutzen, um die Eignung der jeweiligen Proxy-Application in Relation zur Endanwendung über die einfach zu verstehenden Indikatoren RSM und Coverage zu visualisieren. Die Forscher haben gezeigt, dass das Framework Veritas durch den Aufbau der Proxy-Application STREAM bedingte Zusammenhänge zwischen Operationen erkennen und visualisieren kann. Darüber hinaus zeigten die Forscher an ihrem entwickelten Framework, dass sich mit dessen Hilfe auch Dissonanzen, wie die zwischen CMTnek und CMTbone, zwischen Proxy-Application und Endanwendung aufdecken lassen.

Das entwickelte Machine-Learning Framework Veritas [ITB+16] wurde 2016 auf der International Conference for High Performance Computing, Networking, Storage and Analysis in Salt Lake City vorgestellt. Um einen unabhängigen Eindruck von der Eignung des Frameworks zu erhalten, sollte dieses ursprünglich im Zusammenhang mit dem Proseminar auf den HPC-Systemen der Technischen Universität Dresden getestet werden. Leider war es auch nach wiederholter Anfrage und trotz ausreichender Zeitplanung nicht möglich, den Quellcode des Forschungsprojektes zu beziehen. Auch wenn die Strukturen und die Verfahrensweisen im Veritas Framework evident zu sein scheinen, sollte es auch Teil des wissenschaftlichen Diskurses sein, die Untersuchungen der Forscher auf Grundlage von eigenständigen Experimenten zu bestätigen oder in Frage stellen zu können. Die Standhaftigkeit des Frameworks kann daher leider im Rahmen dieser Arbeit über die im Veritas-Dokument dargelegten Tests hinausgehend nicht weiter bestätigt werden.

Literatur

[APE18] APEX. *Alliance for Application Performance at Extreme Scale (APEX) Benchmarks (URL: http://www.nersc.gov/research-and-development/apex/apex-benchmarks/, Zugriff am 20. April 2018)*. apr 2018

[Gé18] GÉRON, Aurélien: *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow : Konzepte, Tools und Techniken für intelligente Systeme*. Heidelberg : O'Reilly, 2018. – ISBN 978–3–96009–061–8

[ITB+16] ISLAM, Tanzima Z. ; THIAGARAJAN, Jayaraman J. ; BHATELE, Abhinav ; SCHULZ, Martin ; GAMBLIN, Todd: A Machine Learning Framework for Performance Coverage Analysis of Proxy Applications. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Piscataway, NJ, USA : IEEE Press, 2016 (SC '16). – ISBN 978–1–4673–8815–3, S. 46:1–46:12

[Mar15] MARSLAND, Stephen: *Machine Learning : an algorithmic perspective*. Boca Raton, FL : CRC Press, 2015. – ISBN 978–1466583283

[McC95] MCCALPIN, John D.: Memory Bandwidth and Machine Balance in Current High Performance Computers. In: *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter* (1995), Dezember, S. 19–25

[McC07] MCCALPIN, John D.: STREAM: Sustainable Memory Bandwidth in High Performance Computers / University of Virginia. 1991-2007. – Forschungsbericht. A continually updated technical report. http://www.cs.virginia.edu/stream/

[Tra18] TRAMM, John: The XSBench Mini-App - A Discussion of Theory / Center for Exascale Simulation of Advanced Reactors (CESAR), Argonne National Laboratory. 2018. – Forschungsbericht. XSBench is a mini-app representing a key computational kernel of the Monte Carlo neutronics application OpenMC.

[TSIS] TRAMM, John R. ; SIEGEL, Andrew R. ; ISLAM, Tanzima ; SCHULZ, Martin: XSBench - The Development and Verification of a Performance Abstraction for Monte Carlo Reactor Analysis. In: *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*. Kyoto,

Danksagung

Mein besonderer Dank gilt meinen Betreuern Ronny Tschüter, Christian Herold und Matthias Weber für die Möglichkeit, in meinem besonderen Interessenspektrum des Machine Learning Erfahrungswerte zu sammeln und neues Wissen im Bereich High Performance Computing zu erhalten. Ich danke meiner Fakultät für die Möglichkeit, dieses Projekt durchzuführen und in dessen Rahmen meine analytischen Fähigkeiten weiterzubilden, besonders im Umgang mit englischer Fachliteratur und der Präsentation dieses Themas.

Erklärungen zum Urheberrecht

Im Rahmen des Proseminar-Themas Veritas - ein Machine Learning-Framework für die Performance-Coverage-Analyse von Proxy-Applications werden die wissenschaftlichen Untersuchungen des Forscherteams um Tanzima Z. Islam, Jayaraman J. Thiagarajan, Abhinav Bhatele, Martin Schulz und Todd Gamblin vom Center for Applied Scientific Computing am Lawrence Livermore National Laboratory in California zusammengefasst und veranschaulicht dargestellt. Der Zweck dieser Arbeit ist damit nicht die Bekanntgabe neuer wissenschaftlicher Erkenntnisse. Demzufolge ist dieses Dokument inhaltlich und chronologisch ähnlich zum Paper A Machine Learning Framework for Performance Coverage Analysis of Proxy Applications [ITB+16]. Grafiken und Zitate, die aus Quellen Dritter stammen, wurden als solche gekennzeichnet. Bei der Recherche nach Grafiken wurde darauf geachtet, dass diese zur Wiederverwendung gekennzeichnet sind. Für eventuelle Urheberrechtsverletzungen, welche sich aus der fälschlichen Kennzeichnung dieser Grafiken durch Dritte ergibt, ist der Autor dieses Dokuments demnach nicht verantwortlich.