Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Ausarbeitung

Visualisierung von Leistungsdaten

vorgelegt von

Joshua Schimmelpfennig

Seminar Effiziente Programmierung
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Studiengang: Mensch-Computer-Interaktion
Matrikelnummer: 6813643

Betreuer: Eugen Betke

Hamburg, 26.2.2019

Abstract

Diese Ausarbeitung bearbeitet das Thema der Datenvisualisierung im Kontext High-Performance-Computing (HPC). Dafür wird die Visualisierung im Allgemeinen zunächst aufgearbeitet um die Visualisierungspipeline als Tool für geordnete Visualisierungsprozesse vorzustellen. Anschließend werden Visualisierung im Kontext HPC betrachtet und die Anforderungen und Schwierigkeiten für die Visualisierung, insbesondere die Datenmenge und Ausmaße der Systemarchitekturen, beleuchtet und erklärt, wie sie durch hierachische Abstraktionsebenen und Interaktionstechniken gelöst werden können. Darauf folgt eine (kurze) Vorstellung von Vampir und einer Beispieloptimierung aus Vampirs Anleitung. Ergänzend zum Profiling wird Grafana speziell als Monitoring Tool vorgestellt. Abschließend finden Gestaltgesetze Erwähnung und es wird erläutert, wie diese die menschliche Wahrnehmung beeinflussen können.

Contents

1 Einleitung

Dieses Kapitel leitet in das Thema Datenvisualiserung ein und gibt einen Überblick über die Anwendungsbereiche von Visualisierungen.

1.1 Überblick

Visualisierung ist eine Form der Kommunikation unter Verwendung des visuellen Kanals des Menschen. Daten werden dabei in Form von geometrischen Objekten enkodiert. Im Gegensatz zu textuellen Modellierungen bietet die Visualisierung beim Ziel der Nachricht eine schnellere Informationsverarbeitung. Dies liegt vor allem an der parallelen Verarbeitung visueller Reize im menschlichen Wahrnehmungsapparat, während Texte sequentiell gelesen werden müssen [Hei12].

Ein Beispiel für die Verwendung von Datenvisualisierung wäre das Monitoring von Systemen (siehe Figure 1.1). Es ist durchaus möglich, die Systemzustände aus den */proc*-Files in Linux zu lesen (links). Ein Tool wie KSysGuard (rechts) ist dabei aber für die Informationsaufnahme und -verarbeitung deutlich schneller und übersichtlicher. Zusätzlich wird die temporale Komponente mit dargestellt.

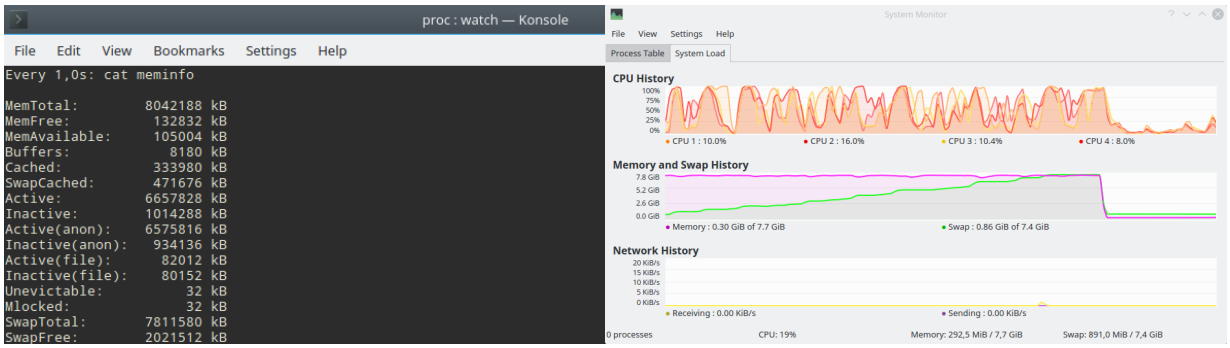


Figure 1.1: Monitoring des System Zustandes

Datenvisualisierung begegnen wir dabei in unserem täglichen Leben ständig. Beispiele dafür sind Routenkarten des öffentlichen Nahverkehrs, Projektionen der Temperaturänderungen in einem Wetterbericht oder einfach ein Schild auf der Autobahn, welches eine kommende Abfahrt ankündigt [WGD10].

Für den Menschen spielt der visuelle Sinn eine sehr wichtige Rolle und ist die Hauptquelle für Informationsgewinnung. Dadurch sind gerade Visualisierungen in der Lage, Entscheidungen von Menschen zu beeinflussen [WGD10]. Aus diesem Grund ist die Betrachtung von Gestaltungsgesetzten sinnvoll und wird später gesondert aufgegriffen.

1.2 Definition

Eine konkrete Definition liefert [FSGS18]. Demnach sei die (Daten-)Visualisierung, im Kontext der Informatik eine Transformation von Daten in eine Grafik, welche dem Leser bei der intuitiven Erkundung von Daten, der Verarbeitung der Daten im Gehirn und dem Verständnis von komplexen Zusammenhängen in den Daten unterstützen kann. Daher ist Visiualisierung in der aktuellen Zeit der zunehmenden Datenmengen wichtig, in der der Betrachter immer mehr Daten verarbeiten muss.

1.3 Datenvisualisierung und HPC

Visualisierung ist für den Bereich des High-Performance-Computings (HPC) interessant, da hier sehr große Datenmengen berachnet und verarbeitet werden. HPC wird im Folgenden im Fokus betrachtet, die Punkte Themen sich aber auch auf andere Gebiete anwenden.

Für HPC ist wichtig, dass Prozesse effizient auf den vorhandenen Ressourcen ausgeführt werden können. Eine höhere Effizienz in der Ausführung bedeutet mehr Freiraum für zusätzliche Berechnungen und dadurch schnellere oder präzisere Ergebnisse. Deswegen ist die Visualisierung mit ihren Eigenschaften sehr nützlich. Mögliche Einsatzgebiete für der Visualisierung im Kontext HPC umfassen das Monitoring von Programmverhalten und Ressourcenverbrauch, die Erkennung von fehlerhaftem und ressourcenintensivem Fehlverhalten oder als Tool für die allgemeine Performance Analyse von System und Software.

Daten werden oft in Form von Zeitreihendaten zu verschiedenen Zeitpunken aufgenommen und können so den Verlauf des Verhaltens darstellen. Erfassbare Daten sind, unter Anderen, CPU-Auslastung, Speicherverbrauch, Netzwerkommunikation, Hardware Counter, die beispielsweise float Operationen zählen, oder für das HPC sehr interessant, die Art und Dauer von MPI-Kommunikationen, welche für die Kommunikation zwischen den Prozessen verwendet wird.

2 Visualisierungsprozess

Dieses Kapitel stellt die Visualisierungspipeline mit ihren Komponenten vor und zeigt, wie sie verwendet werden kann.

2.1 Pipeline

Der Visualisierungsprozess nach [FSGS18] ist ein Hilfsmittel zur strukturierten Erstellung von Visualisierungen. Er hilft dabei, systematisch Probleme zu lösen und leitet den Anwender durch die einzelnen Schritte der Visualisierung durch Übersichtlich lässt er sich auch, wie in Figure 2.1 zu sehen, als Pipeline darstellen. Die Pipeline beschreibt die einzelnen Stationen, die beim Visualisieren durchlaufen werden.

Im ersten Schritt, Rohdaten, wird die Frage der Datenerhebung geklärt. Im nächsten Schritt werden die Daten vor der Visualisierung durch das Preprocessing in eine nutzbare, beziehungsweise für die Visualisierung benötigte, Form gebracht. Das anschließende Mapping stellt das Hauptproblem der Visualisierung dar, hier wird entschieden, wie die Daten in geometrische Objekte transformiert werden. Darauf folgt das Rendering. Hier wird das aus mathematische Formeln bestehende Mapping auf die Daten angewendet und eine tatsächliche sichtbare Grafik, die auf dem Bildschirm angezeigt werden kann, erstellt. In diesem Punkt finden Techniken, wie Rasterisierung, ihren Einsatz. Da dieser Prozess in der Regel vom verwendeten Computer übernommen wird, wird in dieser Ausarbeitung nicht näher drauf eingegangen.

Anschließend ist in der Station *Bild* die Visualisierung fertig und kann dem Betrachter dargeboten werden.

2.2 Qualität

“Eine Visualisierung ist dann gut, wenn die bildliche Darstellung das kommunikative Ziel erreicht hat, d.h. wenn der Informationstransport zum Betrachter gelungen ist.” ([FSGS18])

Figure 2.1: Visualisierungsprozess als Pipeline

Vor der Erläuterung des Visualisierungsprozesses sollten noch Bewertungskriterien für Datenvisualisierungen vorgestellt werden. [FSGS18] betont (u.A. im Zitat oben) den Transport der Information zum Betrachter. Der Erfolg dessen hängt von der Charakteristik der Daten, dem Ziel der Visualisierung, der Natur des Darstellungsmediums und der Wahrnehmungskapazität des Betrachters ab und sollte damit bei der Planung der Visualisierung immer mit einbezogen werden.

Für die Bewertung der Visualisierung zu jedem Schritt im Visualisierungsprozess, existieren die zwei Größen Expressivität (Ausdrucksfähigkeit) und Effektivität [WGD10]. Expressivität ist die zielführende Informationskonzentration der Visualisierung. Eine expressive Visualisierung beinhaltet alle für die Aufgabe wichtigen Informationen, aber nicht mehr. Zu viel Informationen können schaden, da sie die gewünschte Interpretation der Visualisierung behindern können. Eine Visualisierung ist effektiv, wenn sie in der Erzeugung effizient ist und sowohl schnell, als auch akkurat vom Betrachter wahrgenommen werden kann.

Dabei ist anzumerken, dass Expressivität und Effektivität vom zuvor genannten Informationstransport abhängig sind und bei unterschiedlichen Visualisierungszielen variieren können.

2.3 Datenerhebung

Der Punkt der Datenerhebung behandelt zwei Unterpunkte. Erstens muss entschieden werden, welche Daten erhoben werden sollen. Zweitens muss entschieden werden, in welcher Form und auf welcher Art die Daten gespeichert werden. Das kann von einem manuellen ablesen und eintragen von Werten in eine Excel Tabelle bis zum automatischen Profiling reichen. Auch das auslesen der zuvor angesprochenen */proc*-Files ist möglich.

Eine günstige Methode stellen hier schon vorhandene Tracing- bzw. ganze Profiling-Tools dar. Diese können zum Beispiel Prozesse und die Dauer dieser, verbrauchte Sytemressourcen und Kommunikationen zwischen Prozessen aufzeichnen. Zu unterscheiden sind zeit- und eventbasierter Erhebungen. Bei der zeitbasierten Erhebung werden Daten in festen Zeitintervallen erhoben. Bei der evenbasierten Erhebung lösen Events, wie zum Beispiel Funktionsaufrufe, das Speichern aus. Zusätzlich muss man bei manchen Tools auf die Instumentierung achten, in der im Programmcode festgelegt werden muss, welche Teile von dem Tool beobachtet werden soll.

Dies kann trotz Standardeinstellungen sinnvoll sein, da das Speichern überflüssig vieler Daten vermieden werden kann. Das kann bei größeren Programmen/Programmlaufzeiten zu Problemen durch Speicherbegrenzung führen. Beispiele für solche Tools stellen Vampir Trace und das neure Score-P dar [GWT].

2.4 Preprocessing

Der Teil des Preprocessings behandelt alle Datentransformationen, die *vor* der Visualisireung stattfinden müssen. An dieser Stelle werden die Datenformate in von der

Visualiserung nutzbare, fundamentale Datentypen konvertiert.

Daten müssen von falschen Eingaben, Formaten und Fehler bereinigt werden. Fehlende Daten werden auch hier behandelt, indem Datensätze mit fehlenden Daten komplett entfernt oder zum Beispiel mit dem Durchschnitt interpoliert werden.

Bei zu großen Datenmengen bieten sich Aggregationen und Reduktionen an. Auch Filterungen und Partitionierungen der Daten werden hier durchgeführt. Weitere Verfahren sind das Berechnen von Regressionsgeraden, Klassifizieren von Werten oder Clustern von Werten. Auch sie finden in der Visualiserungspipeline im Preprocessing statt.

Zum Schluss sollten die Daten so vorliegen, dass sie einfach nur noch in die Mapping-Funktion gepumpt werden müssen, um eine Visualisierung zu erzeugen.

2.5 Mapping

2.5.1 Definition

Das Mapping ist das Hauptproblem der Visualisierung. Hier werden die nun sauberen Daten auf geometrische Objekte und ihre qualitativen Eigenschaften (wie Form und Farbe) abgebildet. Bei der Suche nach einer geeigneten Abbildung muss auf die Effektivität und Expressivität der Abbildung geachtet werden. Weitere Punkte, die man bei der Wahl der Visualisierung beachten sollte, sind die Struktur der Daten (damit sie sich auch für die Visualisierung eignen), das Ziel der Visualisierung, die Fähigkeiten/Eigenschaften der Zielperson (wie Farbenblindheit und Alter) und der Kontext der Visualisierungspräsentation.

Gute Dos and Don'ts stellen die Gestaltgesetzte dar, welche die Physiologie des Menschen mit einbeziehen und weiter unter erläutert werden.

2.5.2 Beispiele

Anschließend folgt ein kleiner Überblick über bekannte Visualisierungen.

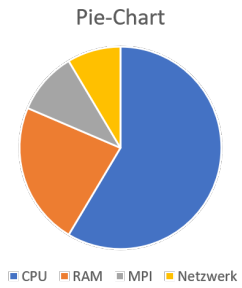
In Figure 2.2 (a) sieht man das allbekannte Pie-Chart. Es bietet einen schnellen Überblick über Verteilungen, ist aber ungenau. Eine Ableitung davon stellt das Ring-Chart dar, welches häufig für Auslastungen von Ressourcen verwendet wird.

In Figure 2.2 (b) ist ein stacked Bar-Chart zu sehen, welcher Verhältnisse zwar besser darstellen kann, die Größen allerdings nicht vergleichbar sind.

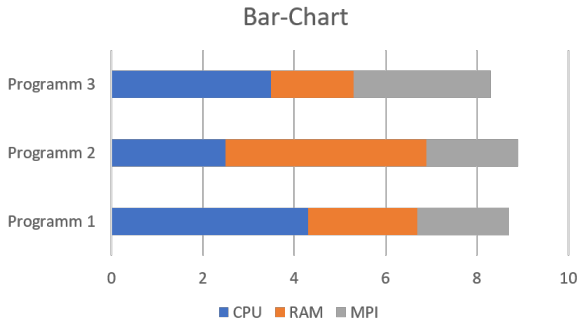
In Figure 2.2 (c) ist ein Star-Chart abgebildet, welches sehr gut für Parameterausprägung von Profilen geeignet ist. Auf den unterschiedlich skalierbaren Achsen können verschiedene Parameter abgemessen werden, die zusammen verbunden die klassischen Poligone bilden.

Eine für die Wissenschaft sehr wichtige Visualisierung ist der Boxplot, mit der Erweiterung des Whiskerplots. Bei einem Boxplot wird die Verteilung von Ausprägungen entlang einer Dimension dargestellt und findet häufig in der Statistik Anwendung.

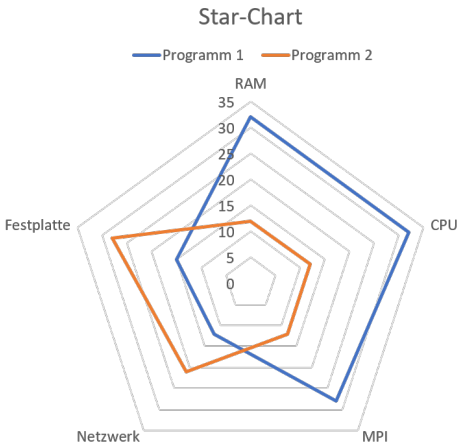
Wie in Figure 2.2 (d) zu sehen ist, besteht die Box des Boxplots aus Markern für die 3 Quartile des Datensatzes, wobei der Median das zweite Quartil Q_2 ist.



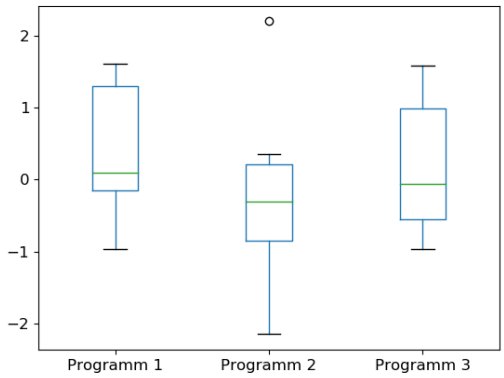
(a) Pie-Chart



(b) Stacked Bar-Chart



(c) Star-Chart



(d) Boxplot with Outliers

Figure 2.2: Einige Charts im Überblick

Die Definition des Medians lautet, dass genau 50% der Daten größer-gleich und kleiner-gleich als der Median sind. Somit ist der Median genau der Wert in der Mitte aller Werte des Datensatzes. Analog sind die anderen beiden Quartile Q_1 und Q_2 definiert, entsprechend mit 25% und 75% der Daten. Dadurch wird der Datensatz in vier 25%-Teile eingeteilt, wobei die mittleren Teile von der Box umgeben sind.

Als Whiskers werden die Striche bezeichnet, die seitlich von der Box hervor schauen, wobei zwei unterschiedliche Darstellungsmöglichkeiten existieren. Bei der einfachen Variante ziehen die Whiskers bis zu den Minimum/Maximum Werten des Datensatzes. Sollen aber Ausreißer dargestellt werden, sind die Whiskers maximal das 1.5-Fache der Box lang. Die Werte außerhalb der Whiskers werden dann als Ausreißer mit einem Punkt markiert.

3 Visualisierung für HPC

Dieses Kapitel behandelt Visualisierungen im Kontext von HPC und wie die dort entstehenden Schwierigkeiten gelöst werden können.

3.1 Anforderungen und Schwierigkeiten

Wie in der Einleitung schon angeschnitten, ist beim HPC wichtig, Prozesse möglichst schnell, effizient und mit wenig Overhead durchzuführen. Visualisiernugen haben hier ein Anwendungsgebiet, um bei den Datenmengen versteckte Bottlenecks aufdecken zu können [WBM+01]. Allerdings steht HPC bei der Visualisierung vor einem Problem, welches durch eine Beispielrechnung für den Mistral Cluster des DKRZ in Hamburg [Deu] gezeigt werden soll.

Angenommen, wir haben ein Profiling Tool, welches für eine Stunde 100 ints (mit je 4 bytes) pro Sekunde speichern soll. Mistral besitzt 3.300 Nodes mit 24 und 36 Kernen. Angenommen, wir benutzen davon 700 Nodes mit 24 paralellen Prozessen. Dann lautet die Beispielrechnung:

$$4 \text{ bytes} \cdot 100 \text{ int} \cdot 24 \text{ Prozesse} \cdot 700 \text{ Nodes} \cdot 3.600 \text{ Sekunden / Stunde} \approx 22.53 \text{ GiB/h}$$

Offensichtlich verbraucht nur eine Stunde Profiling sehr viel Speicher. Hinzu kommt die asymmetrische Systemstruktur durch 24 und 36 Kern Prozessoren. Wie kann man diese große Datenmengen handhaben und asymmetrische Strukturen visualisieren? Geht man Schritt für Schritt durch die Visualisierungspipeline, kann man die Probleme strukturiert lösen.

3.2 Skalierbarkeit von Visalisierungen

Angefangen mit der Datenerhebung stellt sich die Frage, wie die großen Datenmengen gespeichert werden sollten. Sinnvoll wäre, die Daten in eine Datenbank zu schreiben und dann später ein externes Programm darauf zugreifen zu lassen, um die eigentliche Visualisierung durchzuführen. Dadurch werden die Rechennodes nicht zusätzlich mit der Visualisierung belastet. Außerdem wäre eine Aggregation und Reduktion der Daten auf Kennmaße hier geeignet, wenn die eine solche Präzision nicht benötigt wird.

Im Preprocessing der Visualisierung sind auch Reduktionen und Aggregationen sinnvoll, da technisch bedingt nicht alle kleinen Datenpunkte angezeigt werden können.

Beim Thema Mapping sollte vor allem auf die Expressivität der Visualisierung geachtet werden. Was genau ist das Ziel und welche Informationen werden benötigt? Aufgrund der

Datenmenge sollte, abhängig von der gewünschten Präzision, nur ein gewisser Ausschnitt der Daten gezeigt werden. Auch hinsichtlich der Effektivität sollten nicht zu viele Daten enthalten sein, damit das Rendering schnell abläuft und die Visualisierung nicht überladen wirkt. Gleichzeitig sollen aber auch alle Details mit abgebildet werden, da kleinste Datenpunkte große Programmeinflüsse kodieren können.

Folglich wird eine Visualisierungstechnik benötigt, die in bestimmten fällen eine sehr hohe Datenauflösung bieten kann, um kleinste Unterschiede zum Beispiel auf Prozessebene sichtbar zu machen. Zum Anderen sollte sie aber auch Daten wie Gesamtauslastungen auf ganzen Nodes anzeigen können, damit ein Betachter einen groben Überblick über generell verbrauchte Ressourcen bekommen kann.

Dieses Problem wird durch hierarchische Strukturen mit unterschiedlichen Abstraktionsebenen auf den Daten gelöst [BWNH01]. Durch die unterschiedlichen Abstraktionsebenen können Daten in unterschiedlicher Granulität dargestellt werden, die an das jeweilige Ziel der aktuellen Visualisierung anpassbar sind. Gleichzeitig gibt die Hierarchie den Abstraktionsebenen eine zusammenhängende Struktur, sodass die Ebenen abhängig von ihrem Visualisierungsobjekt und Granulität untereinander geordnet sind. Zusätzlich kann die Visualisierung mit einer Interaktionstechnik, wie zum Beispiel dem Zoomen [BWNH01], kombinieren werden, um einem Betrachter zu erlauben, sich intuitiv durch die Ebenen bewegen zu können.

Dies ermöglicht die Skalierbarkeit in Vampir, wie in Figure 3.1 sichtbar ist. Deutlich sind die Ankerpunkte für die verschiedenen Abstraktionen in der Visualisierung, sowie die unterschiedlichen Informationen, die dargestellt werden. In der Abbildung kann so auf der hintersten Ebene eine Visualisierung der Auslastung aller Cluster sehen. Wenn nun auf Cluster 0 eine Abstratkionsebene tiefer gegangen wird, erscheint die mittlere Abbildung, in der für einen Cluster die einzelenen Prozesse mit der entsprechend ausgeführten Funktion angezeigt wird. Wenn nun in einen Prozess hineingezoomt wird, erhält der Betrachter wieder andere Informationen. In dem Fall ist es der Prozessverlauf mit Callebenen und darunter, auf der selben Zeitachse, die Frequenz für ausgeführte Berechnungen. Durch die spezifisch dargestellten Daten bleibt die Expressivität hoch. Weil nicht zu viele Informationen auf einmal dargestellt werden müssen, lassen sich die Daten schneller vom System rendern und vom Menschen verarbeiten. Daher ist auch die Effektivität hoch.

3.3 Beispiel Optimierung

Mögliche Tools für die Visualisierung auf HPC Systemen wäre das bereits erwähnte Vampir [WBM+01] von [GWT] und Grafana, Abbildung aus [Lab].

Mit Vampir lässt sich intuitive Optimierung betreiben. Dies wird auch in [WBM+01] gezeigt, hier werden aber Abbildungen und Beispiele aus dem Tutorial/Use-Case Beispiel von [GWT] verwendet.

Ein Benutzer kann sich die Leistung einzelner Prozesse betrachten und feststellen, dass eine Prozess unerwartet viele Cache Misses aufweist. Hier kann der Benutzer nun ansetzten und den Prozess optimieren. Anschließend kann er den sich den Prozess

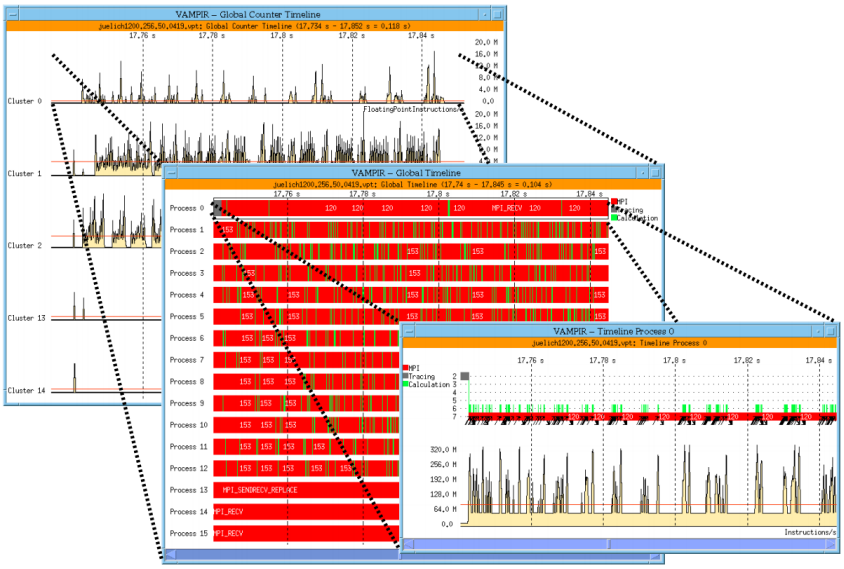


Figure 3.1: Hierarchieebenen in Vampir aus [BWNH01]

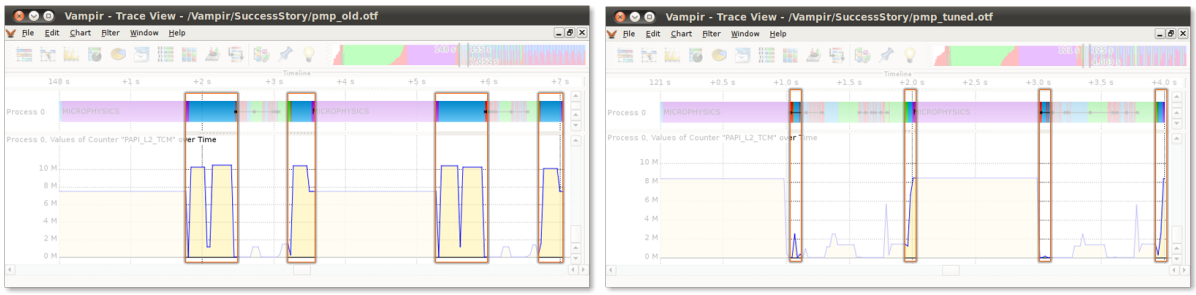


Figure 3.2: Optimierung von Cache Misses, Abbildung von [GWT]

nocheinmal ansehen und feststellen, dass nach der Optimierung weniger Misses vorhanden sind und die Optimierung anscheinend erfolgreich war.

In einem anderen Fall kann der Nutzer einen überproportionalen Anteil von MPI Kommunikationen auffinden und stellt fest, dass die Funktion, die für die Erstellung von Aufgabenpaketen zuständig ist, diese nicht gleichmäßig verteilt. Dies kann zur Folge haben, dass manche Prozesse mit ihrer Arbeit früher fertig sind und auf langsamere Prozesse warten müssen, was Berechnungszeit verschwendet. Eine Anpassung der Verteilungsfunktion bringt in diesem Beispiel eine Zeitersparnis von 33 % für diese Funktion.

Ein weiters Tool zum Monitoring ist Grafana [Lab]. In Figure 3.4 sind Login Nodes für Grafana aufgeführt. Die Auslastung ist durch eine blaue Linie in den Kästen visualisiert und, sollte die Last sehr hoch sein, färbt sich der grüne Kasten rot.

Weiter hat Grafana ein Dashboard (siehe Figure 3.4), auf dem mehrere Visualisierungen parallel nebeneinander angezeigt werden. Hier sind die RingCharts für die Auslastung sichtbar. Jede einzelne Visualisierung, wie auch die Nodes, beinhaltet eine feinere Auflösung ihrer Daten, wenn man hinein zoomt.

So wird das Dashboard nicht überladen und die Expressivität bleibt hoch, da alle

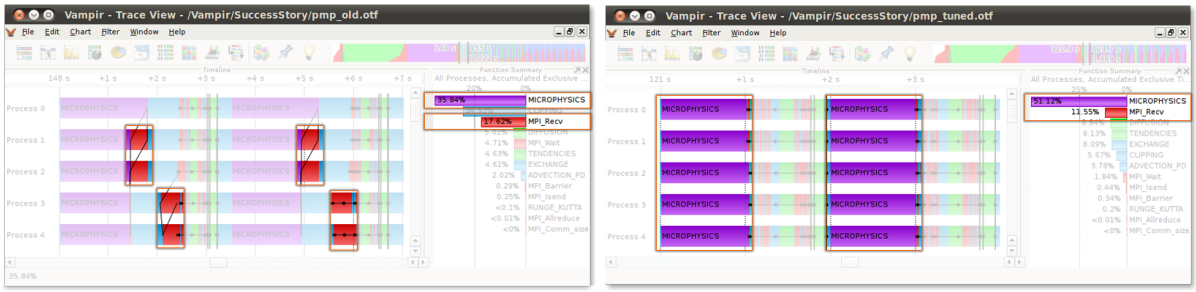
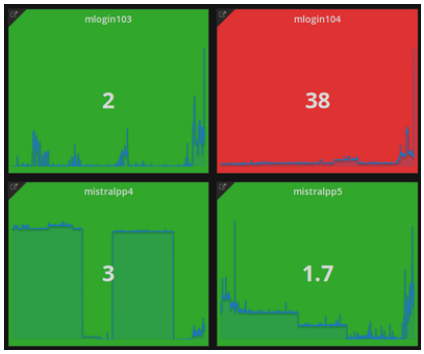


Figure 3.3: Optimierung anhand von MPI-Kommunikation, Abbildung von [GWT]



(a) Grafana Loginnodes, Abbildung von [Deu]

(b) Grafana Dashboard, Abbildung von [Lab]

Figure 3.4: Monitoring mit Grafana

wichtigen Daten fürs Monitoring erhalten bleiben. Außerdem ist die Effektivität hoch, weil die Visualisierungen dadurch schnell renderbar sind.

4 Gestaltgesetze

Dieses Kapitel gibt einen kurzen Überblick über Gestaltgesetzte und wie sie zum Einsatz kommen können.

Gestaltgesetze sind Richtlinien für Visualisierungen. Sie orientieren sich an der Physiologie des Menschen und halten Phänomene der menschlichen Wahrnehmung als Heuristiken fest. Gestaltgesetze können in jedem Lehrbuch zum Thema *Mensch-Computer-Interaktion* gefunden werden. Folgende Gestaltgesetze, Beispiele und Grafiken sind aus, beziehungsweise nachgezeichnet nach [Hei12].

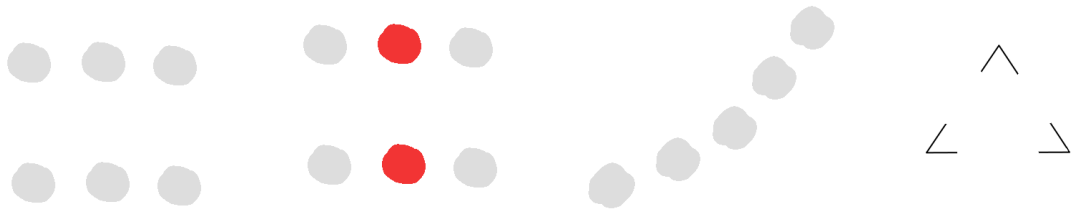
Nach dem **Gesetz der Nähe** werden Items, die temporal oder räumlich nah beieinander liegen, gruppiert. In Figure 4.1 (a) werden die oberen und unteren Punkte deswegen als Reihen wahrgenommen.

Das **Gesetz der Gleichartigkeit** beschreibt, dass Items mit gleichen Eigenschaften zusammen gruppiert werden, obwohl sie räumlich getrennt sind. Diese Eigenschaften sind Farbe, Helligkeit, Größe, Orientierung oder Form. Bei mehreren Gleichheiten zählt in der Regel die Reihenfolge der Aufzählung im letzten Satz. In Figure 4.1 (b) werden daher die roten Punkte als Einheit gesehen.

Nach dem **Gesetz der guten Fortsetzung**, werden Items als zusammengehörig gruppiert, wenn sie sich einfach harmonisch fortführen lassen. In Figure 4.1 (c) werden die Punkte deswegen als Kurve wahrgenommen.

Nach dem **Gesetz der Schließung** werden bei Bruchstücken von Konturen die Kanten gedanklich geschlossen, sodass die eingeschlossene Fläche zur Figur und der Rest zum Hintergrund wird. In Figure 4.1 (d) wird daher aus den Bruchstücken mental die Figur eines Dreiecks zusammengesetzt.

Das Prinzip der guten Gestalt besagt, dass beim Betrachten einer Grafik eine Gliederung durchgesetzt wird, die möglichst einfache, symmetrische und geschlossene Figuren erzeugt, wodurch ein Teil der Grafik stets der Vordergrund wird und der Rest zum Hintergrund übergeht. Ist dies nicht klar definiert oder gibt es vergleichbare Interpretationsalternativen, so entstehen Kippbilder, wie das bekannte Vase-Gesichter Beispiel.



(a) Gesetz der Nähe (b) Gesetz der Gleichartigkeit (c) Gesetz der guten Fortsetzung (d) Gesetz der Schließung

Figure 4.1: Visualisierung einiger Gestaltgesetze

5 Zusammenfassung

Zusammengefasst, die wichtigsten Punkte und Aspekte, die man aus dieser Ausarbeitung mitnehmen sollte:

- Visualisierungen erleichtern die Wahrnehmung, Kommunikation und das Verständnis von komplexen Daten
- Visualisierungen ermöglichen die explorative Performance Analyse bei parallelen Programmen
- Die Qualität einer Visualisierung kann durch Expressivität und Effektivität gemessen werden
- Die Visualisierungspipeline ist ein Tool zur strukturierten Bewältigung von Visualisierungsproblemen
- Visualisierungen sind skalierbar und für HPC sinnvoll verwendbar durch hierarchische Strukturen und Abstraktionen
- Interaktionsmöglichkeiten erlauben intuitive Navigation durch Visualisierungshierarchien
- Gestaltgesetze können Visualisierungen optimieren

List of Figures

Bibliography

[BWNH01] Holger Brunst, Manuela Winkler, Wolfgang E. Nagel, and Hans Christian Hoppe. Performance optimization for large scale computing: The scalable VAMPIR approach. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2074, pages 751–760. Springer, Berlin, Heidelberg, 2001.

[Deu] Deutsche Klimarechenzentrum GmbH (DKRZ). Mistral — User Portal. `https://www.dkrz.de/up/de-systems/de-mistral`. Date Accessed: 2019-02-19.

[FSGS18] Peter Fischer-Stabel, Christoph Göttert, and Jens Schneider. *Datenvisualisierung : vom Diagramm zur Virtual Reality*. UVK Verlag, München, 2018.

[GWT] GWT-TUD GmbH. Vampir 9.6. `https://vampir.eu/`. Date Accessed: 2019-02-19.

[Hei12] Andreas M. Heinecke. *Mensch-Computer-Interaktion (Basiswissen für Entwickler und Gestalter)*. Springer, 2012.

[Lab] Grafana Labs. Grafana : The open platform for beautiful analytics and monitoring. `https://grafana.com`. Date Accessed: 2019-02-22.

[WBM+01] Chris Weaver, Kenneth C. Barr, Eric Marsman, Dan Ernst, and Todd Austin. Performance analysis using pipeline visualization. In *2001 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2001*, pages 18–21, 2001.

[WGD10] Matthew Ward, Grinstein Georges, and Keim Daniel. *Interactive Data Visualization: Foundations, Techniques, and Applications*, volume 136. CRC Press, Taylor & Francis Group, an AK Peters book, Boca Raton, second edi edition, 2010.