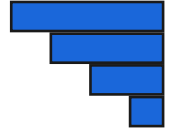


Benchmarking



Seminar:
Fortgeschrittene Fragestellungen zu parallelen und verteilten
Dateisystemen

David Büttner

11.12.2007

Prof. Dr. Thomas Ludwig
Julian Martin Kunkel
Olga Mordvinova
Parallele und Verteilte Systeme
Institut für Informatik
Ruprecht-Karls-Universität Heidelberg

Diese Folien und die dazugehörigen Notizen und Ausführungen entstanden im Rahmen des Seminars "Fortgeschrittene Fragestellungen zu parallelen und verteilten Dateisystemen".



Gliederung



- Einleitung
- Benchmarking: Was und Warum?
- Benchmarking: Von der Planung bis zu den Ergebnissen
- Benchmarking: Probleme und Schwierigkeiten
- Existierende Benchmarks
- Eigene Erfahrungen



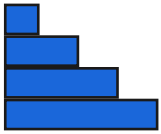
Einleitung

Voraussetzungen:

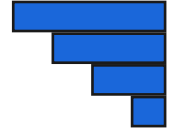
- Hochleistungsrechner: Hohe Leistung für Komplexität
- Benutzeranforderungen, Wünsche

Um über Benchmarks und Benchmarking zu reden, werden in den einleitenden Folien die Grundlagen erarbeitet. In welchem Umkreis wird Benchmarking (in unserem Zusammenhang) benötigt (-> HPC) und welche Bedürfnisse der HPC-Community sorgen eigentlich dafür, dass man Benchmarks braucht.

Leistungsmessung alleine ist nicht wichtig, wenn man dadurch nicht etwas verbessern kann bzw. möchte und wenn es niemanden gäbe, den die Ergebnisse interessieren könnten.



Hochleistungsrechner



Hardware & Software:

- sehr viele unterschiedliche Bauteile
- jeweils in vielen Varianten
- spezial HW & SW

unterschiedliche Konzepte

- Cluster
- SMP
- Kombinationen (!)

außerdem:

- hohe Kosten
- viel Wartungsaufwand

Komplexität sowohl in
Bezug auf
Rechenleistung als auch
bei parallelen und
verteilten Dateisystemen

Hochleistungsrechner und die dazugehörigen Ein/Ausgabe-Systeme treten heutzutage in vielen verschiedenen Formen und Größen auf. Die Komplexität der Systeme nimmt vor allem durch die Anzahl der unterschiedlichen System-Elemente zu. Außerdem werden immer mehr Spezialanfertigungen in HW sowie in SW entwickelt, die weiter für eine größere Vielfalt sorgen. Dies erhöht zusätzlich den Aufwand einen Hochleistungsrechner mit I/O-System zu verwalten und treibt die Kosten für ein solches in die Höhe.



Aufgaben & Anforderungen



Betreiber von Hochleistungsrechnern wollen:

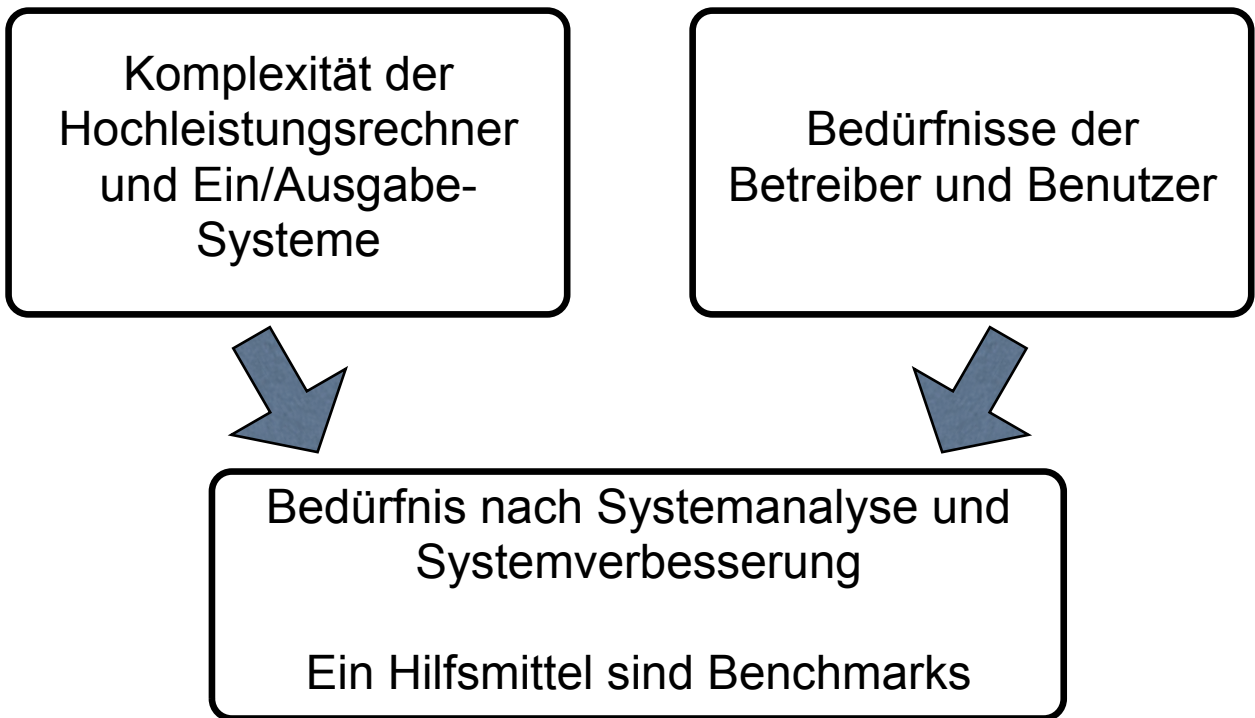
- möglichst gute Systemauslastung
- minimale Kosten für Anschaffung und Betrieb

Benutzer möchten:

- möglichst viel Rechenzeit
- schnelle Systeme
- viele Ergebnisse möglichst detailliert
- möglichst viele Ergebnis-Daten

Wie schon erwähnt sind die Anschaffung und der Betrieb von Hochleistungsrechnern sowohl teuer als auch Arbeitsintensiv. Die Betreiber von solchen sind daher sehr interessiert daran die Kosten minimal zu halten und möglichst viel aus dem System rauszuholen. Hierzu zählen eine möglichst ununterbrochene komplette Systemauslastung (s. später Probleme: äußere Einflüsse) und der Wunsch nach erfolgreichen Benutzern (Wissenschaft: Neue Ergebnisse, Wirtschaft: optimale Ergebnisse).

Notwendigkeit: Benchmark



Um die Wünsche von Betreiber und Benutzer möglichst gut zu erfüllen muss das System gut eingestellt sein. Das Problem ist, dass viele Benutzer unterschiedliche Wünsche haben und diese Wünsche nicht immer leicht gleichzeitig erfüllt werden können.

Um also gut auf die Wünsche eingehen zu können muss ein sehr komplexes System mit vielen "Schräubchen" gut eingestellt werden und zwar so, dass im Schnitt alle Anwendungen gut laufen können.

Da bei dem Umfang der Systeme eine perfekte Einstellung weder leicht zu sehen bzw. in manchen Fällen evtl. gar nicht möglich ist, brauchen die System-Administratoren Informationen über das Systemverhalten. Eine Möglichkeit solches Verhalten zu untersuchen und wichtige Informationen zu sammeln sind Benchmarks.



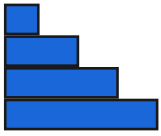
Notwendigkeit: Benchmark



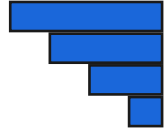
Situationen in denen Benchmarks wichtig sind:

- Leistungsmessung:
Gibt das System an Leistung das her, was es können sollte?
- Systemvergleich vor Neuanschaffung / Systemerweiterung
- Vergleich von unterschiedlichen Installationen bzw. Konfigurationen von gegebener HW und SW.
- Vergleich von unterschiedlichen Lösungsansätzen

Die aufgelisteten Situationen müssen nicht unbedingt alle sein, in denen Benchmarks und vor allem deren Ergebnisse wichtig sind, sie sind jedoch auf jeden Fall unter den wichtigen von diesen Situationen.



Was ist ein Benchmark?



benchmark[...] an example of sth which is used as a standard or point of reference for making comparisons



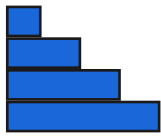
Benchmark & HPC



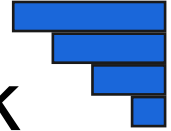
- Repräsentation von erwartetem Workload eines Systems
- Performance Test
- Test von Systemverhalten in unterschiedlichen Situationen
- Ein paralleles Programm (!)

"Durchschnitt" von normaler
Systemnutzung zusammen mit
Funktionalität zur Leistungsanalyse

Ein Benchmark ist ein paralleles Programm welches die zu erwartenden Anwendungen eines Systems repräsentiert und welches genutzt werden kann um die Systemleistung in Bezug auf diese Anwendungen zu analysieren. Die dabei erhobenen Daten können in unterschiedlichen Detailstufen erhoben werden.



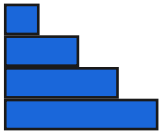
Dateisysteme-Benchmark



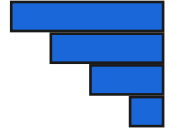
Benchmarks für ein Dateisystem (normal, parallel, verteilt):

- Repräsentation der erwarteten Dateizugriffe
 - > Zugriffsmuster
 - > Menge der verwendeten Daten
 - > Timing (!)
- Test der Performance in Bezug auf die erwarteten Dateizugriffe
- Test maximaler Performance (maximale Anzahl von übertragenen Daten pro Zeit)
- in parallelen Dateisystemen nicht nur Zugriff auf Festplatten
 - > Test für Zugriff auf Kommunikationsnetzwerk
 - > Test von zahlreichen Schichten des E/A-Pfades

Ein I/O Benchmark muss nicht nur auf die zu testende Menge von Daten achten, sondern auch, dass die Zugriffe den zu erwartenden Zugriffsmuster zeitlich richtig abgearbeitet werden. Sind in zu erwartenden Anwendungen die Zugriffe von allen am I/O-Prozess beteiligten Benutzerprozesse echt parallel, so muss dies in dem Benchmark auch so sein. Kommen wiederholte Zugriffe schnell bzw. mit langen Pausen hintereinander, muss das Benchmark dafür sorgen, dass dies auch im Test so passiert.



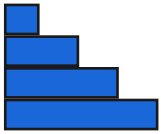
Benchmarking



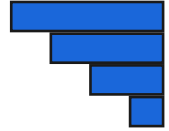
Benchmarking:

- Sammeln von Daten
- Test von erwarteter Systemauslastung
- Test von maximal möglicher Systemleistung

Benchmarks sollen durch Abstraktion von allgemeinen Benutzerprogrammen eine Möglichkeit bieten ein System zu analysieren und durch die gewonnenen Daten einen Einblick in evtl. vorhandene Problemstellen des Systems geben ("Bottlenecks").



Benchmarking

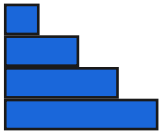


Durchführung:

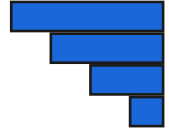
- Einfach: Messen von Zeiten
- Schwerer aber mit mehr Information: Messen von
 - > Gesamtzeit
 - > Einzelzeiten von Prozessen
 - > Zwischenzeiten aus einzelnen Teilen des Systems
- Noch besser: zusätzliches mitspeichern von Zusatzinformationen wie Ressourcenauslastung (graphische Darstellung)

Mit Gesamtzeiten alleine lässt sich evtl. feststellen, dass ein System nicht die erwartete Leistung vollbringt, aber es wird nicht leicht möglich sein dadurch alleine herauszufinden, warum das so ist.

Durch die Erhebung von zusätzlichen Zeiten und Daten wie Ressourcenauslastung zu bestimmten Zeiten, Informationen über die Folge von Abhängigkeiten zwischen Funktionen (welche Aktion löst welche Reaktion aus?) und ähnlichem lassen sich Bottlenecks viel leichter erkennen und es ist einfacher ihre Ursachen zu finden.



Benchmark & Praxis

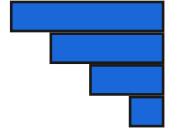


- Planung
- Implementation
- Durchführung
- Interpretation

Die folgenden Folien sollen eine Übersicht über die Entwicklung und Nutzung eines Benchmarks geben. Dabei soll immer überlegt werden, wie die einzelnen Phasen bearbeitet werden müssen um am Ende zu garantieren, dass man wissenschaftlich qualitative Ergebnisse erhält.



Planung



Wichtig:

Gute Analyse und Definition der zu testenden Szenarien!

Was will man eigentlich wissen?

- maximale Leistung des Systems / von Teilen des Systems
- Leistung bei erwarteter (normaler) Nutzung des Systems
 - > Wie sieht diese aus?
 - > Kann sie allgemeint gut repräsentiert werden?

Es ist wichtig sich in der Planung des Benchmarks viele Gedanken darüber zu machen, was man eigentlich wissen will.

Möchte man z.B. in Bezug auf die Ein/Ausgabe wissen wie viele Daten eine bestimmte Anzahl von Knoten überhaupt geschrieben bzw. gelesen werden können, muss man sich ausserdem im Klaren darüber sein, von welchem Typ diese Daten seien sollen. Wo kommen sie her, wo sollen sie hin, wie sind sie verteilt?

Möchte man das Verhalten des E/A-Systems in Bezug auf erwartete Benutzerprogramme testen, muss man sich eine Übersicht darüber machen, wie die Datenzugriffe der Benutzer aussehen werden. Wie viele Daten, in welchem Zeitraum werden sie gelesen/geschrieben? Welche logische, physikalische Verteilung besitzen sie, etc. (s. auch später das Kapitel "Probleme").



Erwartungen

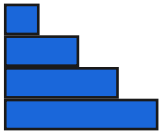
Vor Benchmarkausführung aufgestellte Erwartungen helfen dabei die Ergebnisse besser interpretieren zu können.

- mindestens erwartete Leistung
- obere Schranken für die Systemleistung

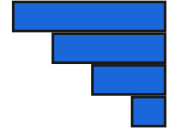
Woher?

- Aus theoretischen Überlegungen:
 - Was ist technisch überhaupt möglich?
 - Was können einzelne Systemteile individuell?
- Aus Erfahrungen:
 - > Erfahrungen mit bestimmten Benutzerprogrammen
 - > Erfahrungen auf anderen Systemen

Das Aufstellen von erwarteten Ergebnissen ist wichtig und hilfreich für die spätere Ergebnisinterpretation. Wenn man sich erst Gedanken über mögliche Leistungen (vor allem die erwartete) gemacht hat, sieht man leichter evtl. Abweichungen der tatsächlichen Leistung bei der Messung. Ausserdem (wie die nächste Folie noch beschreibt) braucht man die Überlegung um bei der Implementation des Benchmarks zu verhindern, dass die Art und Weise dieser eine neue (geringere) obere Schranke für die Systemleistung mit sich bringt, als sie von der Art der Systemnutzung eigentlich bedingt wäre.

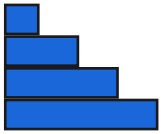


Implementierung

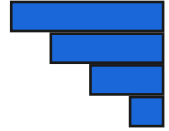


Eine Benchmarkimplementierung sollte:

- sauber und korrekt sein
- gut getestet werden
- den theoretischen Überlegungen entsprechen
(die getesteten Szenarien sollen die gewünschte Information liefern)
- keine neue (geringere) obere Schranke für die Systemleistung implizieren

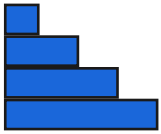


Benchmark-Eingaben

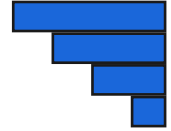


- müssen die Problemstellung repräsentieren
 - > Welche Eingabe testet was?
 - > Werden die erwarteten Zugriffsmuster gut simuliert?
 - > Resultiert die Eingabe in der richtigen Menge an Daten-Manipulation?
- sollten auch mal so gewählt werden, dass sie bekannte oder vermutete Problemsituationen testet
- dürfen nicht in Bezug auf die Ergebnisse gewählt werden !
- sollten für jeden Testlauf gut dokumentiert sein.

Zusätzlich zu den genannten Anforderungen an die Eingabe eines Benchmarks soll hier nochmals betont werden, dass es für wissenschaftlich wertvolle Ergebnisse wichtig ist, dass das Benchmark nicht so geschrieben und ausgeführt wird, dass es das System am Ende gut aussehen lässt! Es sollte ohne "Ergebniswunsch" die wirkliche Leistung des Systems testen.



Testumgebung



Die Testumgebung:

- entspricht erwarteter / zu testender Umgebung
- sollte mit Konfiguration für jeden Testlauf dokumentiert sein
- muss evtl. für jeden Testlauf neu aufgesetzt werden
- evtl. bekannte Tuning-Möglichkeiten gehen in **Planung** der Testumgebung mit ein!

Da bei einer bestimmten Hardwareumgebung oft mehrere (evtl. sogar sehr viele) Softwareumgebungen mit jeweils mehreren Konfigurationen getestet werden sollen ist es wichtig sich darüber Gedanken zu machen, wie diese jeweils aussehen sollen und wie die einzelnen Ergebnisse mit diesen in Verbindung gebracht werden (Dokumentation). Zusätzlich ist es wichtig sich zu überlegen, ob eine Testumgebung zwischen der Durchführung von zwei (evtl. identischen) Tests neu aufgesetzt werden muss (s. hierzu auch Kapitel "Probleme" (vor allem "Caches")).

In Bezug auf Tuning sollte in der Planung auch überlegt werden, welche Kenntnisse über Tuning vorhanden sind, ob diese angewandt werden sollen und ob diese evtl. nur dem Benchmark was bringen oder auch allen repräsentierten Benutzerprogrammen. Auch das Wissen über "Nichts-Wissen" von Tuning sollte mit in die Überlegungen und später in die Interpretation der Daten eingehen.



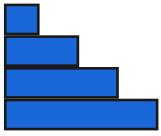
Allgemein

Gute Vorbereitung von einem Benchmark ist wichtig und beinhaltet:

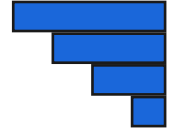
- klare Fragestellungen: Was soll getestet werden?
- gute Implementierung des Benchmarks
- Planung von Testfällen und ihre theoretische Analyse
- Durchführung mit geplanten Testumgebungen
- Interpretation der Ergebnisse

Gute Vorbereitung von einem Benchmark ist wichtig und beinhaltet

- Planung und Analyse des zu Testenden
 - gute Implementation eines Benchmarks
 - Planung von Testfällen und ihre theoretische Analyse
 - Durchführung mit geplanten Testumgebungen
 - Interpretation der Ergebnisse
- > Ausschluss bzw. Minimierung von Problemen und äußeren Einflüssen (s. nächsten Abschnitt)



Benchmarkingprobleme



Probleme und Schwierigkeiten: Welche können auftreten?

- keine Standardmethoden für Benchmarks
- Probleme in der Planung
- Probleme bei der Implementierung
- Schwierigkeiten bei guter Simulation von Zugriffsmustern
- Probleme in der Ausführung
- allgemeine Probleme
- Einfluss des Benchmarks auf Systemverhalten

Die bisherigen Betrachtungen besprechen die Planung und Umsetzung eines Benchmarks. In den folgenden Folien soll besprochen werden, welche Probleme in den ganzen Phasen auftreten können und auf was besonders geachtet werden muss. Da es keine Standardmethoden für Benchmarks gibt, ist es auch nicht möglich eine "Musterlösung" für die Probleme anzugeben. Es existieren jedoch viele Veröffentlichungen zu den einzelnen Themen.

- Probleme der Planung
- Probleme der Implementierung
- gute Simulation von Zugriffsmustern
- Probleme in der Ausführung
 - Äußere Einflüsse
 - Caches (auch Planung)
- Allgemeine Probleme
 - hohe Komplexität von Systemen mit parallelen Dateisystemen
 - Schwierigkeiten der Dateninterpretation
 - Problem der Ergebnisvergleiche

WICHTIG: Einfluss des Benchmarks auf Systemverhalten (Benchmark darf an sich keinen Einfluss auf das Systemverhalten haben (ausser dem zu simulierenden))



Probleme: Planung

Was soll eigentlich gemessen werden?

- Was kann aus Benutzerprogrammen wegabstrahiert werden?

Welche Informationen sind interessant, welche überflüssig?

- soviel wie möglich, so wenig wie nötig
- zu viele (unnötige Informationen)
 - > verursachen mehr Probleme
 - > erschweren die Ergebnisinterpretation

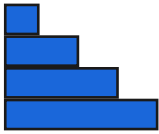
Was ist überhaupt eine "zu erwartende Workload"?

Wie kann eine solche repäsentiert werden?

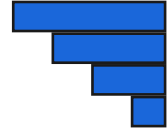
- viel Recherche, Erfahrung, etc.

Die Hauptprobleme in der Planung beinhalten vor allem die Entscheidungen darüber, was überhaupt das Ziel des Benchmarks sein soll. Die Vielfalt der Benchmarks, die schon alleine für E/A-Benchmarks existiert zeigt in ihren Ansätzen schon, dass es hier viele unterschiedliche Meinungen geben kann.

Es sollte auch darauf geachtet werden, dass nicht unnötig viele überflüssige Informationen gesammelt werden. Dies kann sonst weitere Probleme verursachen (s. Probleme: Systemeinfluss).



Probl.: Implementierung



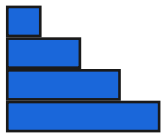
- Übliche Probleme des Programmierens...
- Übliche Probleme des parallelen Programmierens!
(deadlocks, races, Skalierbarkeit, etc.)
- Werden Zeiten richtig (sofort) gemessen?
- Sind die Zeiten für alle Prozesse gleich?
- Code einzelner Systemteile muss angefasst werden
(z.B. paralleles Dateisystem) um gewünschte Details zu erfassen.

Neben den bekannten Problemen des Programmierens und dem parallelen Programmierens kommen hier noch weitere Probleme hinzu.

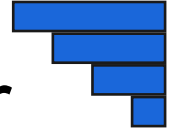
Vor allem 2 Bereiche machen neue Probleme:

Die Zeitnahme muss über das gesamte System konsistent geschehen und es muss dafür gesorgt werden, dass für alle Prozesse die gleiche "Zeit" herrscht. Da auch in der Hochleistungs-Ein-Ausgabe sehr kleine Zeiten eine Rolle spielen, ist die Interpretation der Daten sehr viel einfacher, wenn Zeit A gemessen bei einem Prozess gleich Zeit A bei einem anderen Prozess ist.

Zusätzlich reicht es meist nicht aus ein Benchmark als Benutzerprogramm zu schreiben ohne zusätzlich in anderen Teilen des Systems (z.B. den einzelnen Modulen von parallelen Dateisystemen) zusätzlichen Code einzufügen, der Traces erzeugt und wichtige Informationen mitprotokolliert. In Systemen die nicht als Open Source Code vorliegen, wird dies noch weiter erschwert. Nach Prof. Ludwig gibt es nur eine Hand voll Menschen, die Code-Instrumentierung auf Binärdatei-Ebene vornehmen und das auch verstehen und können.



Probleme: Zugriffsmuster

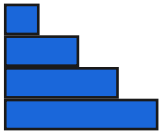


Zugriffsmuster sind sehr komplex:

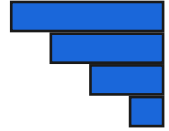
- Menge der Daten
- Zeitlicher Ablauf der Zugriffe
- Logische Organisation der Daten
- Physikalische Organisation der Daten

Wie kann garantiert werden, dass diese eingehalten werden ohne den normalen Benutzerprogrammcode (Berechnungen) mit zu übernehmen?

Bei der Bestimmung der erwarteten Zugriffsmuster von Benutzerprogrammen reicht es nicht sich nur anzuschauen, wie viele Daten gelesen/geschrieben werden und wann dies geschieht. Es muss auch überlegt werden, welcher Prozess welche Daten schreibt, da dies die Verteilung der Daten im parallelen Dateisystem beeinflussen kann.



Probleme: Ausführung



Während der Ausführung treten 2 Arten von Problemen auf:

- Äußere Einflüsse
- Leistungsoptimierende Mechanismen (z.B.: Caches)

Diese Probleme sollten natürlich auch schon in der Planungsphase mit bedacht werden.

Probleme: Äußere Einflüsse

Hochleistungsrechner oft nicht exklusiv nutzbar!
(Kosten, Rechte)

Netzwerk, Switches, etc. werden von anderen Knoten/Prozessen/Benutzern mitgenutzt

Hat das Einfluss auf das Benchmark?
Soll das mit betrachtet werden?

Wenn ja, sollte das nicht von Benchmark simuliert werden?
(Reproduzierbarkeit)

Ja



Probleme: Äußere Einflüsse



Netperf-Test:

TCP_RR (Request/Response) Test **ohne** zusätzliche Netzlast:

Trans. Rate per sec: 7992

TCP_RR Test **mit** zusätzlicher Netzlast durch TCP_STREAM Test:

Trans. Rate per sec: 45

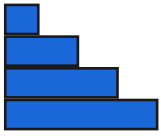


Probleme: Äußere Einflüsse

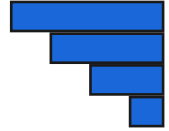


Weitere äußere Einflüsse sind:

- Betriebssystem Prozesse
(sollte bei guter Konfiguration gering sein)
- Bei normalen (Einzel-) Rechnern: andere Benutzerprozesse
- Ausfall von Hardware
Die Wahrscheinlichkeit, dass bei sehr großen Systemen Hardware kaputt geht ist nicht klein
(= Übliches Problem von parallelen Programmen)

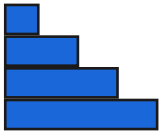


Probl.: Leistungsopt. Mechanismen

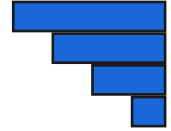


Leistungsoptimierende Mechanismen gibt es mittlerweile in vielen Teilen des Systems:

- Hardware (z.B. L1/2 Cache, Festplattencache, DMA)
- Software (z.B. Softwarecache des Dateisystems, Compileroptimierungen)



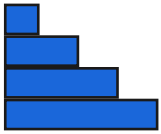
Probleme: Caches



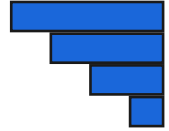
Frage: Soll Cache-Verhalten mit gemessen werden?

- Bei "Maximaler-Leistung-Messung" evtl. eher ohne?
- Wie beeinflussen Caches das Systemverhalten, wie werden sie in realen Benutzerprogrammen verwendet?
- Cold/Hot Caches: Sind die Zustände von einem Testlauf zum nächsten reproduzierbar?

Caches können in Bezug auf Benchmarks viele Probleme und Fragen aufwerfen. Es ist evtl. auch leider nicht immer möglich alle Caches zu ignorieren bzw. auszuschalten. Manche Caches gehören einfach zum System und werden immer genutzt. Dann stellt sich die Frage, wie können Fälle getestet werden, die den Cache "überrennen", wie grenzt man diese von Fällen ab, die ausschließlich im Cache z.B. des parallelen Dateisystems ablaufen?



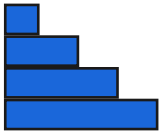
Probleme: Weitere...



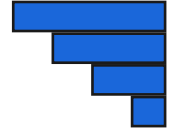
Zusätzliche allgemeine Probleme sind:

- Hohe Komplexität von Systemen mit parallelen Dateisystemen
- Schwierigkeiten mit der Dateninterpretation
- Problematik des Ergebnisvergleichs unterschiedlicher Benchmarks

Diese Themen werden auf den folgenden Folien behandelt. Es geht hier um allgemeinere Probleme, die eigentlich Einfluss auf alle Phasen des "Benchmarking" haben können.



Probleme: Komplexität



Hochleistungsrechner mit parallelen Dateisystemen haben:

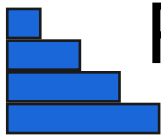
- viel Hardware
- sehr viele Software-Schichten mit sehr viel Code

Einzelne Ereignisse nicht leicht vorhersehbar, Probleme für theoretische Überlegungen.

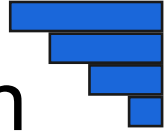
Keine Einzelpersonen kennen komplettes System

- Woher bekommt man Antworten?
- Welche Fragen soll man überhaupt stellen?

- hohe Komplexität von Systemen mit parallelen Dateisystemen
 - viel HW
 - viele Software-Schichten -> sehr viel Code
 - einzelne Ereignisse nicht mehr leicht vorhersagbar (Probleme für theoretische Überlegungen)
 - keine einzelnen Personen, die komplettes System (im Detail) überblicken (Woher bekommt man Antworten auf Fragen, die man evtl. noch nicht einmal richtig zu stellen weiss?)



Probl.: Dateninterpretation



Auch bei korrekter "Beschaffung" von Benchmark-Ergebnis-Daten können trotzdem noch Probleme auftreten:

- Unvollständigkeit der Daten
 - > unzureichende Dokumentation der Versuchsanordnung (Welches Tuning gab es?)

- Falsche Interpretation
 - > Ist jedes Tuning für alle Anwendungen gleich gut?
 - > Übersehen von (extremen) Abweichungen von möglichen Werten

Vor allem wenn die Benchmarkdaten nicht selber erstellt wurden, aber auch, wenn man ein fremdes Benchmark - evtl. zum ersten mal - einsetzt ist nicht immer klar, was die Ergebnisdaten einem sagen sollten. Mögliche Fehlverhalten im System müssen erkannt werden und deren Gründe richtig bestimmt.



Probleme: Vergleichbarkeit



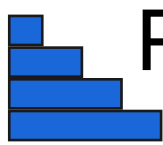
Wie sind Ergebnisse von unterschiedlichen Testläufen zu interpretieren?

Ergebnisse von unterschiedlichen Benchmarks sind evtl. überhaupt nicht sinnvoll zu vergleichen.

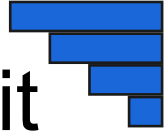
- Was messen die unterschiedlichen Benchmarks?
- Was sagen die einzelnen Ergebnisse wirklich aus?
- Sind die jeweils genutzten Testumgebungen gut dokumentiert?

Die Interpretation von Ergebnisdaten wird umso schwerer umso weniger man über das benutzte Benchmark weiss. Erfahrung hilft immer.

Unterschiedliche Benchmarks, die eigentlich gleiche Szenarien testen können die Ergebnisse in komplett unterschiedlicher Weise darstellen. Was sagen die Ergebnisse aus und wie genau sind sie entstanden. Sind sie direkt miteinander vergleichbar oder muss man sie nebeneinander anschauen?



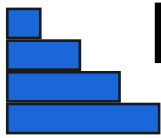
Probleme: Vergleichbarkeit



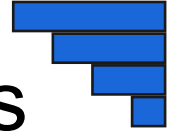
Ergebnisse von unterschiedlichen Systemen:

- Wieviel Tuning wurde auf welchem System gemacht?
- Hätte Tuning auf dem schlechteren System bessere Ergebnisse geliefert?
- Hat das Tuning was für die erwartete Systemnutzung gebracht oder war es "schön-tunen" der Maschine?

Problematisch wird es auch wenn man Benchmarkergebnisse von unterschiedlichen Systemen vergleicht. Hier kann vor allem Tuning dafür sorgen, dass für genau das getestete Szenario das System perfekt abschneidet, aber alle allgemeinen Typen des gleichen Problems doch schlechter abschneiden würden, als auf einem anderen System, bei dem dieses Tuning nicht vorgenommen wurde.



Probl.: Benchmarkeinfluss

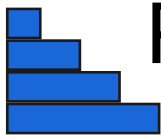


Eines der wichtigsten Voraussetzungen für ein gutes Benchmark ist:

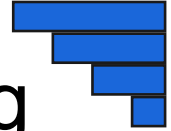
Der Benchmark-Code an sich darf ausser der zu simulierenden Systembelastung keinen Einfluss auf das selbe haben!

- Es darf kein weiterer Overhead entstehen
- Der extra Rechenaufwand durch Code z.B. zur Trace-Generierung muss möglichst gering gehalten werden!

Zusätzlicher Code in den einzelnen Systemteilen, der evtl. sehr oft aufgerufen wird (Trace-Generierung) muss so gering und effizient gehalten werden wie möglich. Vor allem wenn kleine Verzögerungen bei z.B. Daten-Operationen dafür sorgen, dass ein I/O-Server Zeit hat Platz in seinem Software-Cache zu machen, sind die Ergebnisdaten nicht mehr repräsentativ für die zu erwartende Systemauslastung. In dieser würden die Daten-Operationen, wenn auch geringfügig, etwas schneller aufeinander folgen und so evtl. insgesamt viel größere Engstellen aufzeigen, als mit dem Zusatzcode.



Probl.: Zusammenfassung

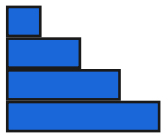


Probleme und Schwierigkeiten treten überall auf!

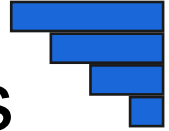
Größere und komplexere Systeme bringen mehr dieser Probleme mit sich, aber auch einen größeren Bedarf an guten Benchmarks zur Systemanalyse.

- Mehr Systemteile
- Mehr Konfigurationsmöglichkeiten
- Mehr Möglichkeiten etwas falsch zu machen!

Es gibt viele Dinge auf die man achten muss, die Probleme machen können. Größere komplexere Systeme bringen mehr Probleme mit sich, aber auch einen größeren Bedarf an Benchmarks (mehr Ecken an denen man drehen will)



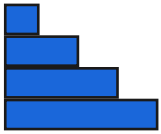
Existierende Benchmarks



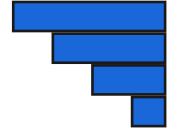
Es gibt eine Vielzahl von Benchmarks:

- PRIOMark
- PIO_Bench
- IOR
- iozone
- b_eff und b_eff_io

Eine e-mail Umfrage zu Erfahrungen in Bezug auf Benchmarking von parallelen und verteilten Dateisystemen hat leider keine Antworten erhalten ...



b_eff und b_eff_io



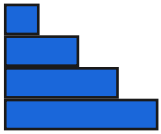
Bandwidth Efficient (Input/Output) Benchmark

- nutzt MPI(-I/O) Funktionen
- "Charakteristisches Ergebnis" ist ein Durchschnitt von vielen Testläufen
- "Ein System kann in einer festen (relativ kurzen) Zeit komplett getestet werden."

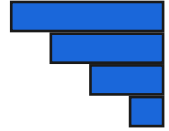
Sehr viele Benchmarks für parallele Dateisysteme sind als parallele Programme realisiert, die den MPI und MPI2 Standard als Schnittstelle für Kommunikation und I/O benutzen.

b_eff und b_eff_io sind zwei Teile einer Benchmark-Suite die durch sehr viele, relativ kurze Einzeltests, das System untersucht und aus den einzelnen Ergebnissen ein Gesamtergebnis, das sog. "Charakteristische Ergebnis", berechnet.

Die Dauer des Benchmarks wird konstant gehalten, indem während der Laufzeit entschieden wird, wie viele Durchläufe jeder Test bekommt. Die "Macher" von iozone sind der Meinung, dass ein System in relativ kurzer Zeit komplett getestet werden kann und auch jeder Cache im I/O-Pfad in wenigen Minuten "überannt" werden kann.



b_eff

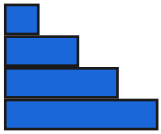


Nachrichtenaustausch/Netzwerkteil der Benchmark-Suite:

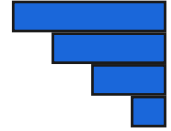
- viele kleine/kurze Tests
- jeweils mehrere Kommunikations-Routinen
- (echt) parallele Kommunikation

- x communication patterns * 21 Nachrichtengrößen *
3 Komm.-methoden * 3 Durchläufe = 9261 Tests
 - > 6 ring patterns
 - > 30 random patterns
 - > 13 weitere

Durch die Nutzung von mehreren Kommunikations-Routinen für jeden Test wird erreicht, dass evtl. vorhandene Optimierungen in der Optimierung einer dieser Routinen ausgeglichen werden. Die Frage ist natürlich, ob man eigentlich die Ergebnisse unter Einbezug genau solcher Optimierungen wissen wollte oder nicht. In wenigen Minuten werden sehr viele Tests durchgeführt.



b_eff_io



I/O Teil der Benchmark-Suite:

- 315 Tests:
 - 5 I/O Patterns
 - 7 chunk-sizes
 - 3 unterschiedliche Anzahlen von I/O Prozessen
 - Tests auf "write, read und reread"
- Messung von wohlgeformten und nicht wohlgeformten Datenmengen ($= 2^x + 4\text{kB}$)

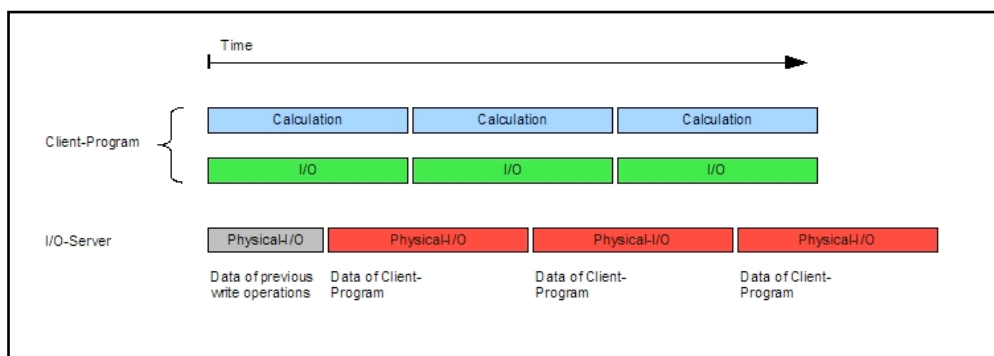
Der Eingabe/Ausgabe Teil der Benchmark-Suite ist so ausgelegt, dass alle Tests zusammen in unter 30 Minuten beendet sind.
Eine in anderen Benchmarks nicht gefundene Funktionalität ist der Test von nicht wohlgeformten Datenmengen bei den Zugriffen. Bei einem Vortrag von Rolf Rabenseifner vorgestellte Ergebnisse zeigen auch, dass dieser Unterschied in der Datengröße auch wirklich Unterschiede in den Messergebnissen zur Folge haben kann.

Eigene Erfahrungen

"Benchmarking Non-Blocking Input/Output on Compute Clusters"

Konzept schon vorgestellt von Michael:

- Während I/O Phase des Programms werden weiter Berechnungen ausgeführt



Der MPI-2 Standard erlaubt mit den `MPI_File_I...` Funktionen nicht blockierende E/A Operationen. Diese wurden zu Beginn meiner Bachelorarbeit in MPICH2 noch nicht implementiert, d.h. es wurden in diesen Funktionen einfach die entsprechenden blockierenden Varianten aufgerufen. Um zu sehen, ob eine Umsetzung dieser Idee überhaupt etwas bringen kann bzw. lohnenswert ist, wurde ein Benchmark geschrieben, der genau diese Situation testet und zeigt, dass die Benutzung solcher Funktionen wirklich eine Verbesserung in der Programmlaufzeit erzielen kann.



Eigene Erfahrungen



Planung:

- Gemessen werden soll maximal erreichbarer Speedup
- Erst feststellen, wie viel I/O = wie viel Berechnung
- Dann sequenzielle Variante

Theorie:

- Maximaler Speedup: 2

Testläufe:

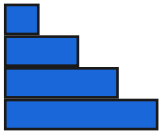
- Theoretisches Maximum kann erreicht werden
- Meist weniger Speedup, aber doch immer > 1



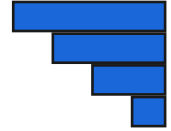
Zusammenfassung



- Benchmarks helfen Hochleistungsrechner in Bezug auf bestimmte Probleme zu bewerten.
- Viele detaillierte Informationen helfen Bottlenecks im E/A-Pfad zu erkennen und zu beheben.
- Das erstellen eines Benchmarks ist mit vielen Problemen und Schwierigkeiten verbunden.
- Die Interpretation von Benchmark Ergebnissen kann unter Umständen sehr komplex und schwierig sein.



Fragen?



Gibt es Fragen?

Bibliography

- [1] Hans Meuer (Univ. of Mannheim), Jack Dongarra (Univ. of Tennessee), Erich Strohmeier (NERSC/LBNL), and Horst Simon (NERSC/LBNL). TOP500.org. <http://top500.org>, April 2007.
- [2] A. S. Hornby. Oxford Advanced Learner's Dictionary of Current English. Oxford University Press, 5 edition, 1995.
- [3] Thomas Ludwig. Folien zur Vorlesung: Hochleistungs E/A Systeme. <http://pvs.informatik.uni-heidelberg.de/Teaching/HEAS-0607/heas-0607.pdf>, 2006.
- [4] Thomas Ludwig. Materialien zur Vorlesung: Cluster Computing. <http://pvs.informatik.uni-heidelberg.de/Teaching/CC-06/index.html>, 2006.
- [5] Michael Krietemeyer, Daniel Versick, and Djamshid Tavangarian. THE PRIOMark PARALLEL I/O-BENCHMARK. In International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria, February 2005.
- [6] R. Rabenseifner. Invited Talk in the Lecture: "Hochleistungs-Eingabe/Ausgabe-Systeme, Eective File-I/O Bandwidth Benchmark (b_e_io) and Other I/O Benchmarks. http://www.hlr.de/people/rabenseifner/publ/iwr_Jan2007_b_eff_io_slides.pdf, 2007.
- [7] R. Rabenseifner and Alice E. Koniges. Eective Communication and File-I/O Bandwidth Benchmarks. In Proceedings, Santorini, 8th European PVM/MPI Users' Group Meeting, Greece, September 2001.
- [8] Frank Shorter. Design and Analysis of a Performance Evaluation Standard for Parallel File Systems. <ftp://ftp.parl.clemson.edu/pub/techreports/2003/PARL-2003-001.ps>, August 2003.
- [9] William Gropp, Ewing Lusk, and Thomas Sterling. Beowulf Cluster Computing with Linux. The MIT Press, 2 edition, 2003.
- [9] David Büttner, Benchmarking of Non-Blocking Input/Output on Compute clusters, Ruprecht-Karls Universität Heidelberg April, 2007