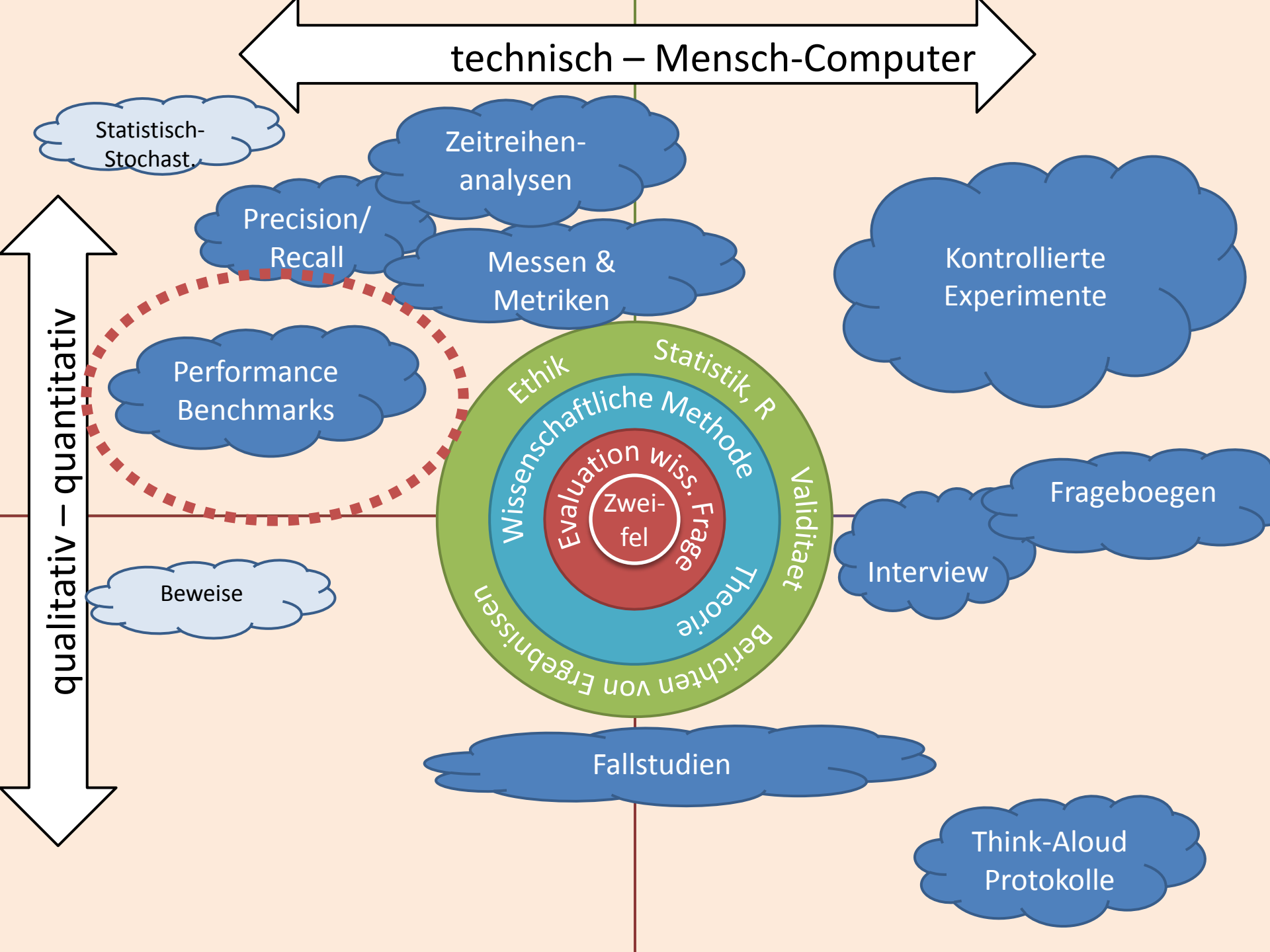
Empirische Methoden für Informatiker
Teil 2: Performance Messen

Christian Kästner


technisch – Mensch-Computer
qualitativ – quantitativ
Statistisch-Stochast.
Zeitreihen-analysen
Precision/ Recall
Messen & Metriken
Kontrollierte Experimente
Performance Benchmarks
Ethik
Statistik, R
Validitaet
Theorie
Berichten von Ergebnissen
Wissenschaftliche Methode
Evaluation wiss. Frage
Zwei-fel
Frageboegen
Interview
Beweise
Fallstudien
Think-Aloud Protokolle

1. Evaluierung in Bachelorarbeit o.ae.?
2. Themenwuensche

Aufgabe

- Bestimmen Sie die schnellste Sortierfunktion
- Gruppe 1: Mergesort vs. Quicksort
- Gruppe 2: Quicksort Rekursiv vs. Quicksort Iterativ
- Gruppe 3: Quicksort Java vs. Quicksort Scala
- Gruppe 4: Quicksort Oracle VM vs. Quicksort JRockitVM

- Ermitteln Sie welcher von 2 USB-Sticks die schnellere Schreib-/Lesegeschwindigkeit hat

Agenda

- Benchmarks und ihre Probleme
- Wiederholungen, Initial State
- Statische rigorose Benchmarks
- Benchmark-Suiten
- Diskussion und Tradeoffs zur Genauigkeit
- Berichten von Benchmarks

Messen

Warum Performanceanalyse?

- Alternativen vergleichen
- Einfluss eines Features
- System Tuning
- Relative Performance erkennen (ueber Zeit)
- Absolute Performance fuer ausgewaehlte Faelle
- Erwartungen setzen
- Analyse von Systemverhalten

Analysetechniken

- Messen
  - keine vereinfachenden Annahmen
  - i.d.R. am glaubwuerdigsten
  - inflexibel, spezielles System
- Simulation
  - abstraktion
  - flexibel
- Analytisches Modellieren
  - mathematische Beschreibung des Systems
  - starke Abstraktion, i.d.R. kaum glaubwuerdig
  - insb. zur fruehen Validierung

Benchmark

- Ausfuehren realer Programme/Hardwarekomponenten in realen Umgebungen (keine analytische Simulation)
- Messen von Performance, Speicherverbrauch, usw.
- Automatisierbar
- Kein menschlicher Einfluss

Benchmark Parameter

- Programmimplementierung
- Eingabedaten
- Umgebung

Was messen?

- CPU-Zyklen
- MIPS, FLOPS, SPEC, QUIPS
- Ausfuehrungszeit
- Transaktionen pro Sekunde

- Zielgerichtete Metrik waehlen
- Ausfuehrungszeit typisch

Mess-Ungenaugigkeiten (Stoervariablen)

- Hintergrundprozesse
- Hardwareunterschiede
- Temparaturunterschiede
- Eingabedaten, zufaellig?
- Heap-Size
- Hardware-Plattform
- System-Interrupts
- Parallelitaet in Single- und Multicore-Systemen
- Garbage Collection

Praesentation der eigenen Messergebnisse

- Vorgehen
- Ergebnisse

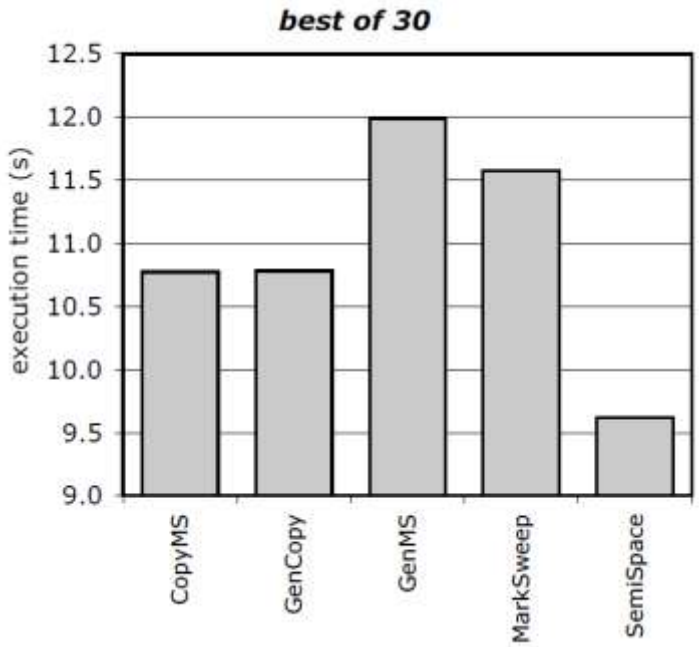
- Fuer die Zuhoerer:
  - Vertrauen Sie den Ergebnissen?

Mess-Ungenaugigkeiten (Stoervariablen)

- Hintergrundprozesse
- Hardwareunterschiede
- Temparaturunterschiede
- Eingabedaten, zufaellig?
- Heap-Size
- Hardware-Plattform
- System-Interrupts
- Parallelitaet in Single- und Multicore-Systemen
- Garbage Collection

Typisches Vorgehen: Bester Wert

- Wiederholen
- Berichten: Besten Wert, zweitbesten Wert, oder schlechtesten Wert



Typisches Vorgehen: Mittelwert

- Messung wiederholen (wie oft?)
- Mittelwert bilden

- Grundlage f. Mittelwert: Gesetz der grossen Zahlen & Zentraler Grenzwertsatz

Mittelwerte

- Artithmetisches Mittel

$$\bar{x}_{arithm} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

```
mean(c(1,4,6,10)) = 5.25
mean(c(-5,3,4,6,50)) = 11.6
```

- Median: Der Wert in der Mitte
  - Bei gerader Groesse das arithmetische Mittel der beiden mittleren Werte
  - Robust gegen Ausreisser

```
median(c(1,4,6,10)) = 5
median(c(-5,3,4,6,50)) = 4
```

- Gestutzte Mittel
  - Auf beiden Seiten 10% der Daten entfernen, dann mitteln

Median

- Median statt Arithmetisches Mittel, wenn
  - ordinale Beobachtungen (Rangdaten)
  - nur wenig Messwerte
  - asymetrische Verteilung
  - bei Verdacht auf Ausreisser

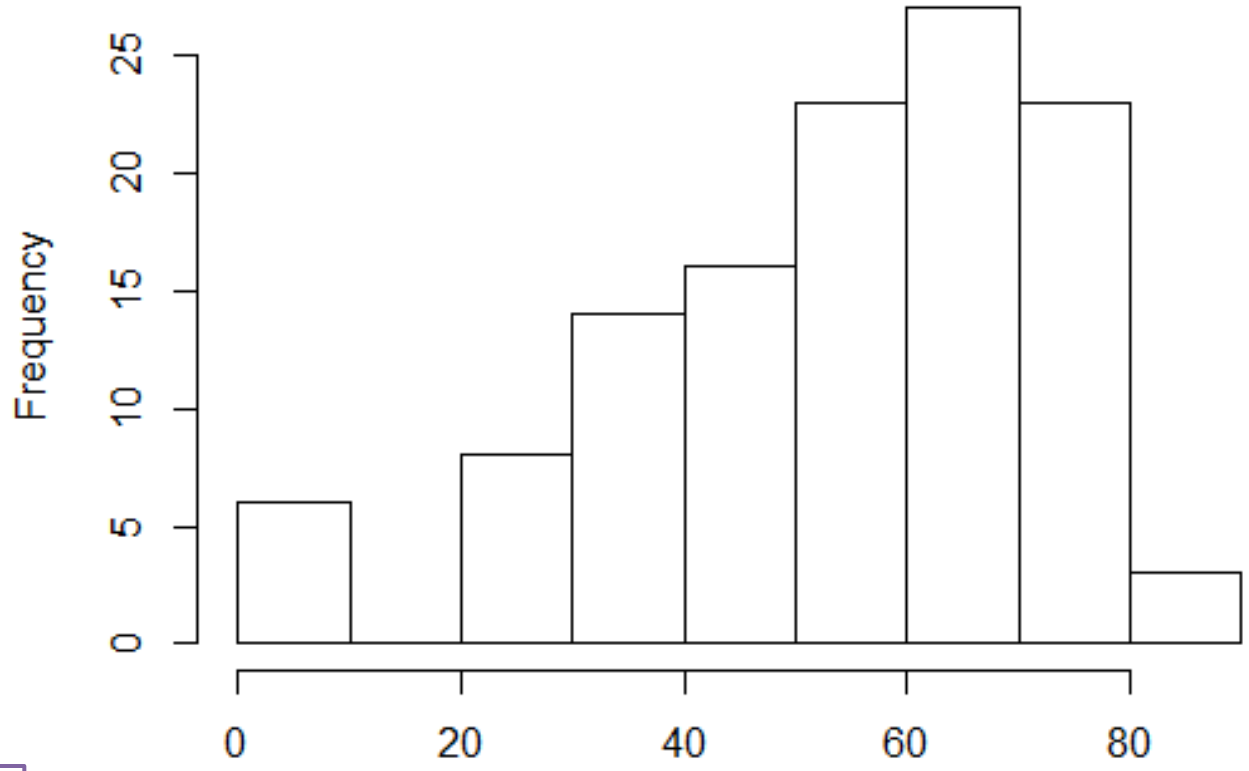
Aber

- Wie viele Messungen?
  - Reichen 3, 10, 50?
  - Oder gar 100, 10000?
- Direkt hintereinander messen oder verschraenkt?
  - AAABBB oder ABABAB
- Mehre Iterationen in einem Lauf oder mehere Laeufe?
- Messungen unabhaengig?
- Reicht Mittelwert?

Daten visualisieren

- Ueberblick verschaffen
- Verteilung und Ausreisser einschaetzen

Histogramme

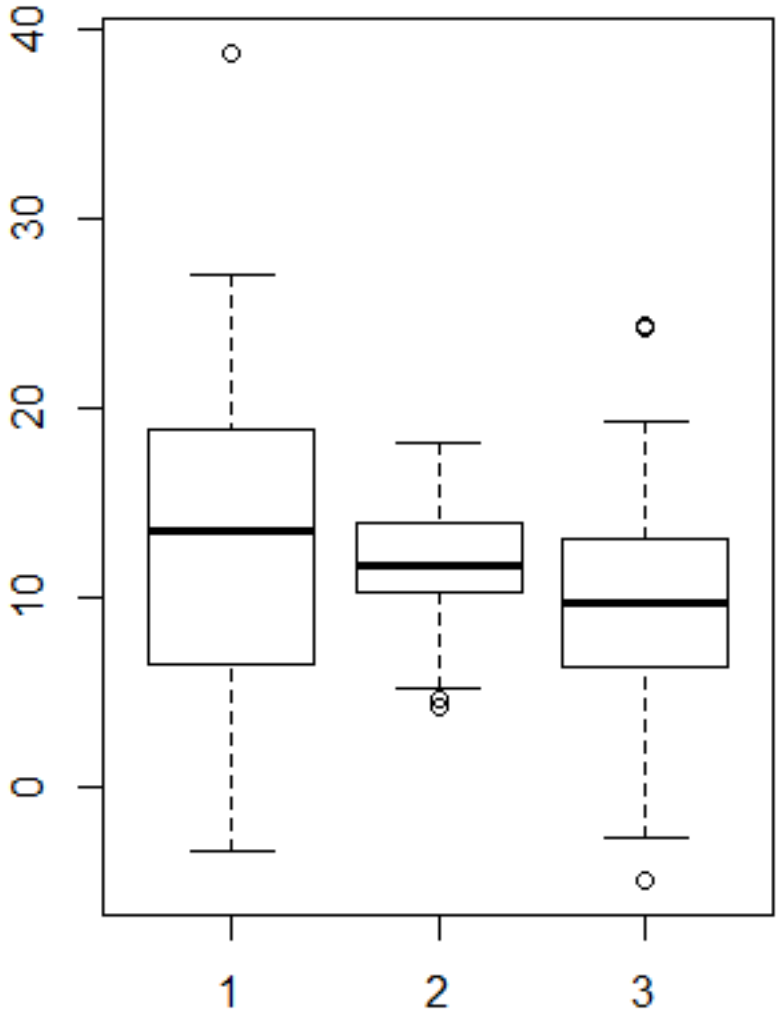


hist(c)

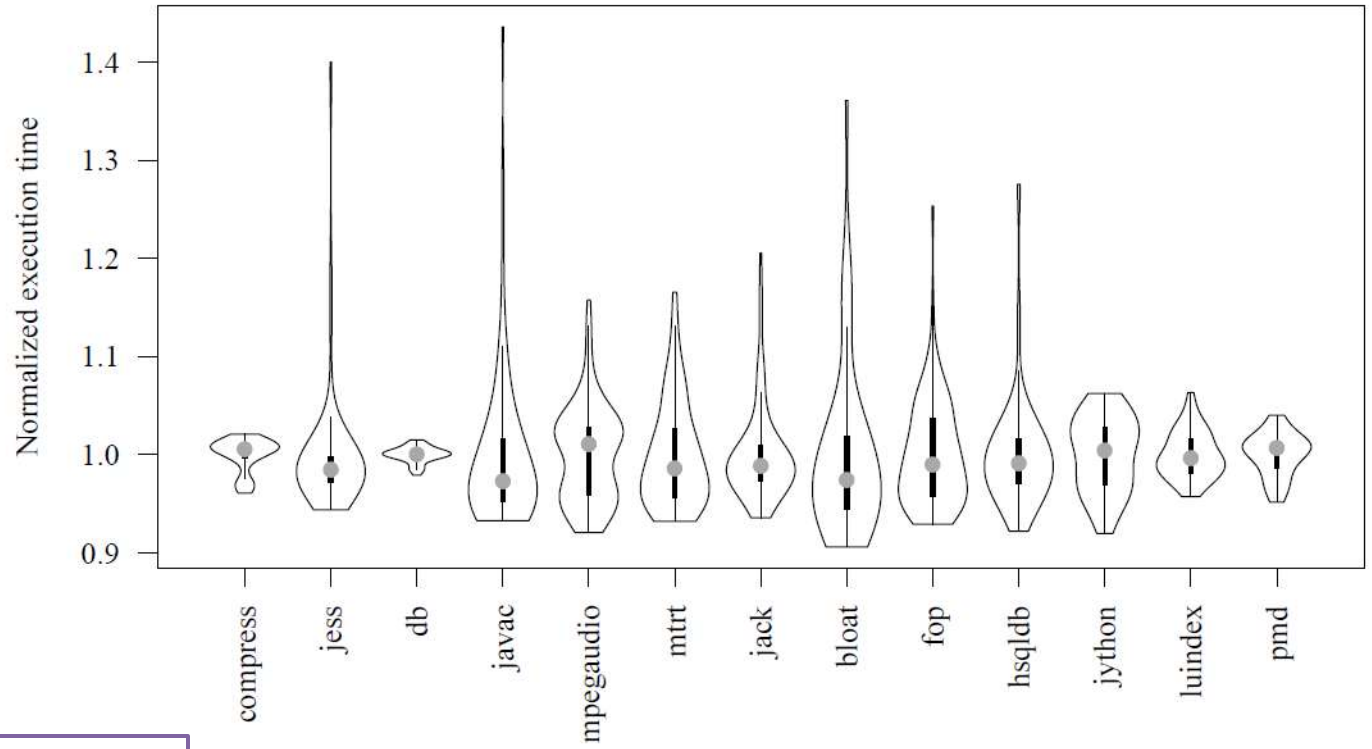
Verteilungen berichten

- Boxplot zeigt
  - Median als breite Linie
  - Quartile als Box
    (50% aller Werte in der Box)
  - Whiskers
  - Ausreisser als Punkte
- Graphische Darstellung von Verteilungen

boxplot(c)



Violin-Plots




```
library(vioplot)
vioplot(c)
```

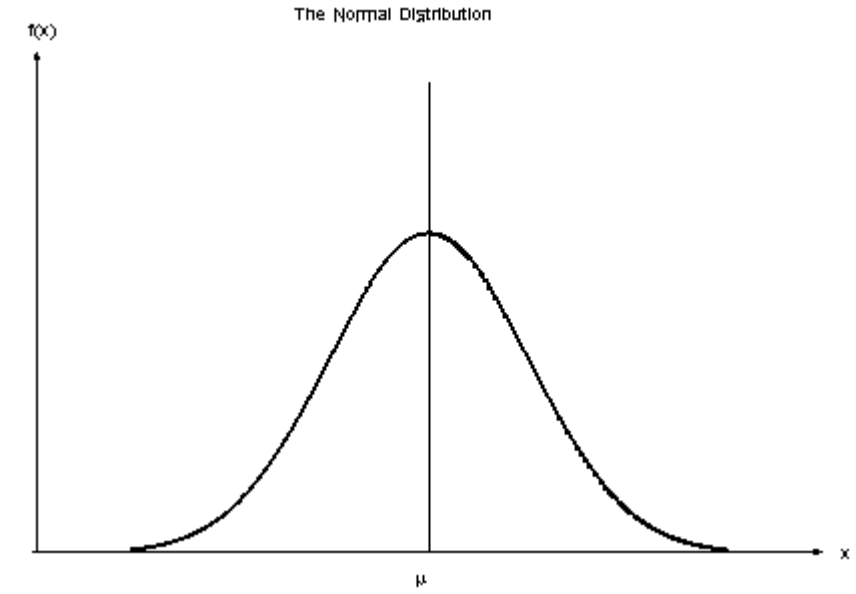
Intuition: Fehlermodell

- 1 Zufaelliger Fehler, Einfluss +/- 1
- Echter Mittelwert: 10
- Messwerte: 9 (50%) und 11 (50%)

- 2 Zufaellige Fehler, je +/- 1
- Messwerte: 8 (25%), 10 (50%) und 12 (25%)

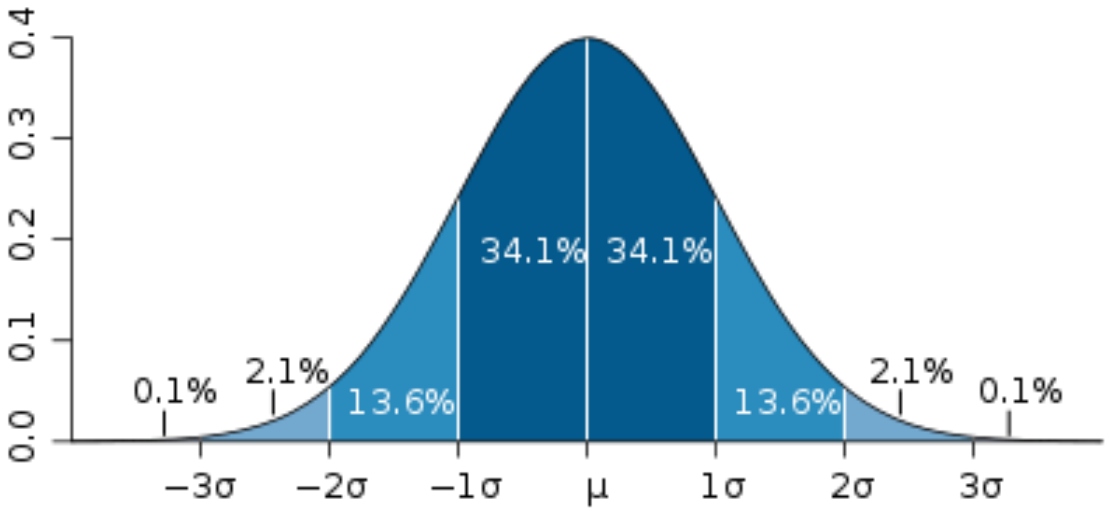
- 3 Zufaellige Fehler, je +/- 1
- Messwerte: 7 (12.5%), 9 (37.5), 11 (37.5), 12 (12.5)

Normalverteilung



Standardabweichung

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n}}$$



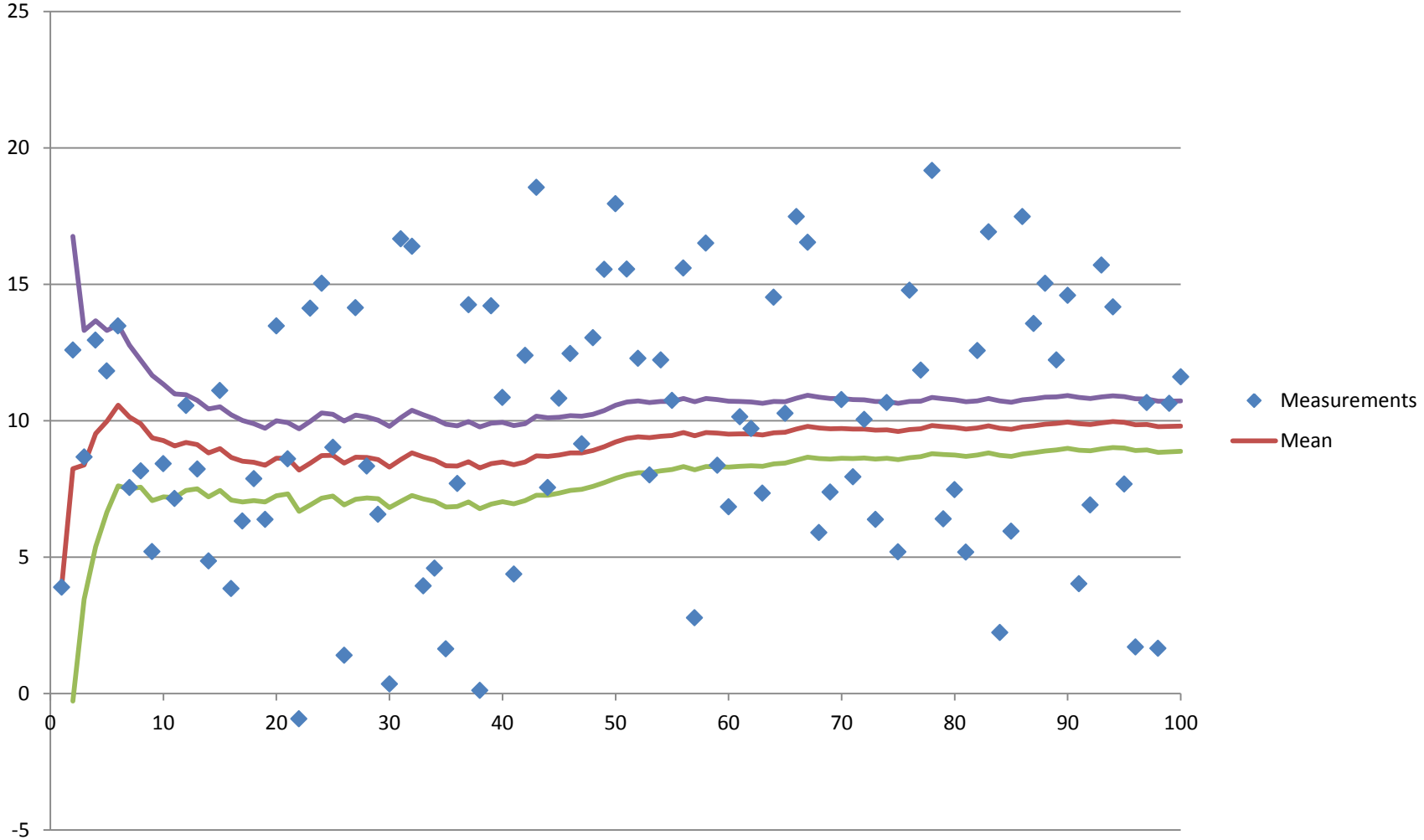
Konfidenzinterval - Formal

$$\left[\bar{x} - z_{(1-\frac{\alpha}{2})}\frac{\sigma}{\sqrt{n}} \; ; \; \bar{x} + z_{(1-\frac{\alpha}{2})}\frac{\sigma}{\sqrt{n}}\right]$$

$$\left[\bar{x} - t_{(1-\frac{\alpha}{2};n-1)}\frac{s}{\sqrt{n}} \; ; \; \bar{x} + t_{(1-\frac{\alpha}{2};n-1)}\frac{s}{\sqrt{n}}\right]$$

Konfidenzinterval

Daten sammeln bis in Ziel-Konfidenzintervall, z.b. +/- 10%



Konfidenzintervall

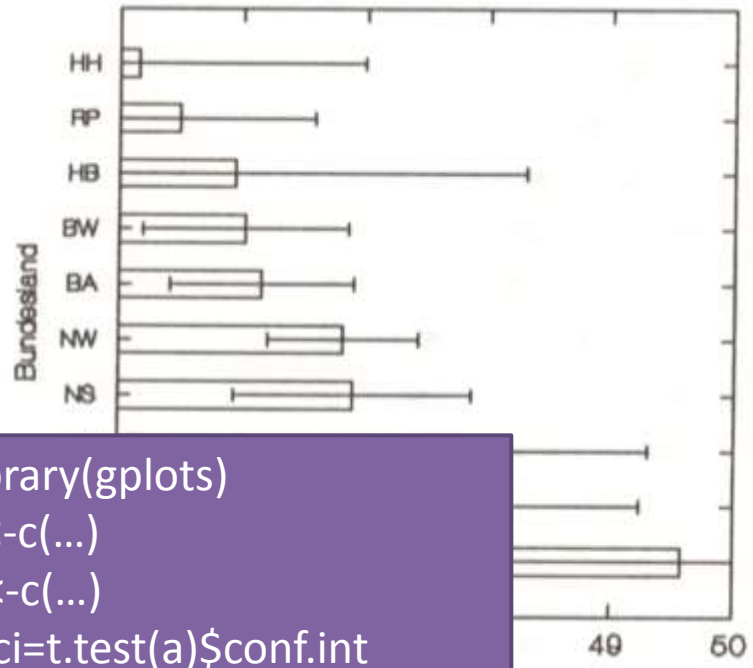
- Bei unabhaengigen Messungen sind Messergebnisse normalverteilt (Zentraler Grenzwertsatz)

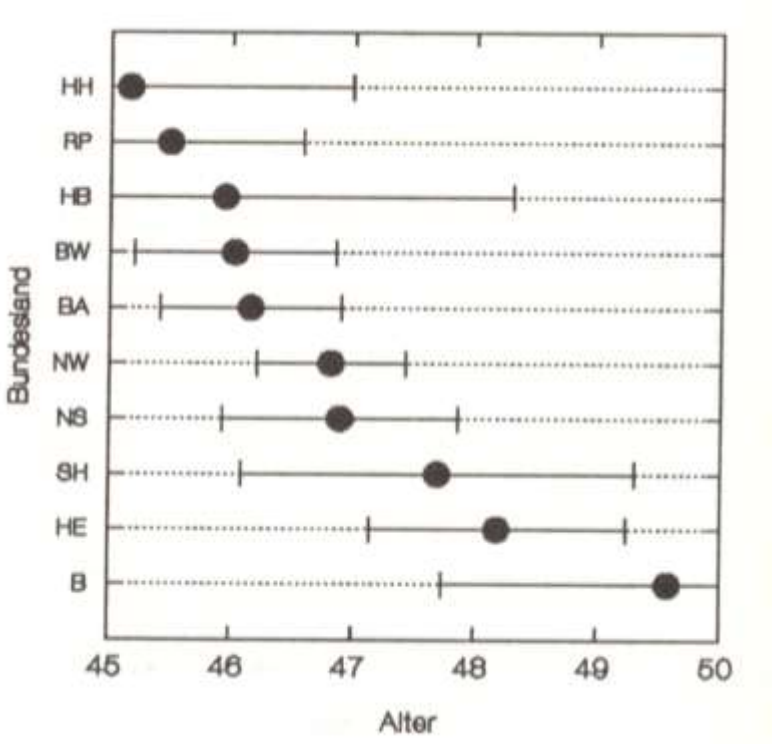
```
> t.test(data, conf.level=.95)
...
95 percent confidence interval:
 8.870949 10.739207
```

- Konfidenzlevel 95% =>
  mit 95% Wahrscheinlichkeit liegt der wirkliche Mittelwert in dem Intervall*
  - Mittelwert der Stichprobe vs. Mittelwert der Grundgesamtheit

*Technisch korrekter: Bei grosser Anzahl von Wiederholungen des Experiments liegt in 95% der Faelle der wirkliche Mittelwert in dem jeweils berechneten Konfidenzinterval

Mit Konfidenzintervallen/Stdabw. Berichten






```
library(gplots)
a<-c(...)
b<-c(...)
a.ci=t.test(a)$conf.int
b.ci=t.test(b)$conf.int
barplot2(
   c(mean(a),mean(b)),
   ci.l=c(a.ci[1],b.ci[1]),
   ci.u=c(a.ci[2],b.ci[2]),
   plot.ci=TRUE)
```

from Schell. Graphisch gestuetzte Datenanalyse. Oldenburg Verlag, 1994.

Beschriftung

- Welche Abweichung wird gezeigt?
  - Standardabweichung s (SD)
  - Konfidenzintervall (CI)
  - Standardfehler des arithm. Mittels (SE) = s/sqrt(n)
- Keine eindeutige Notation -> Achsen beschriften/Legende

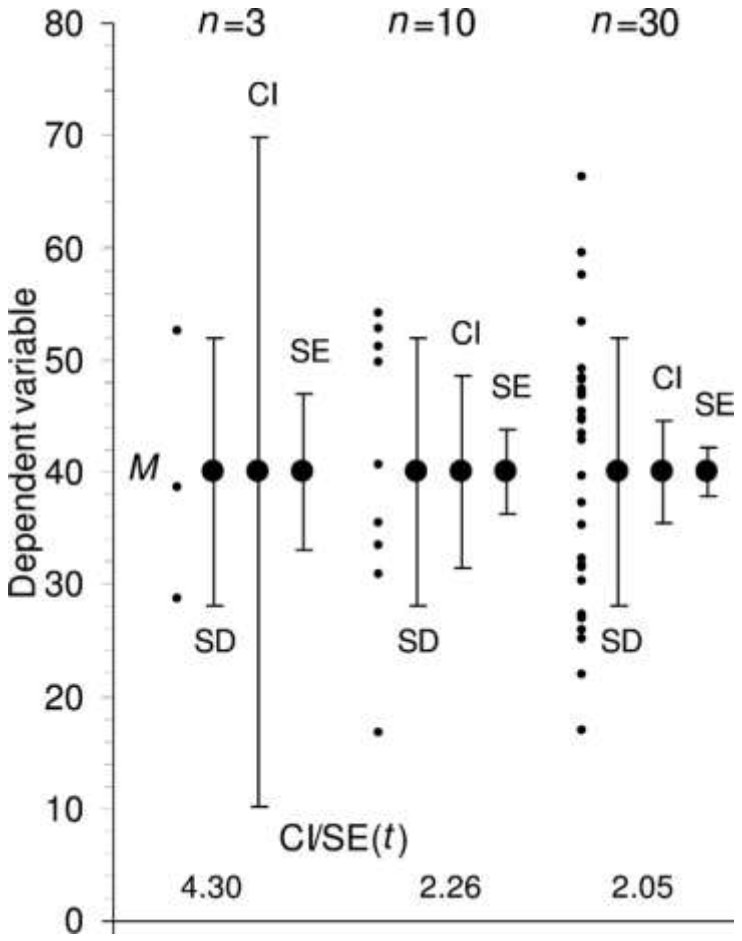
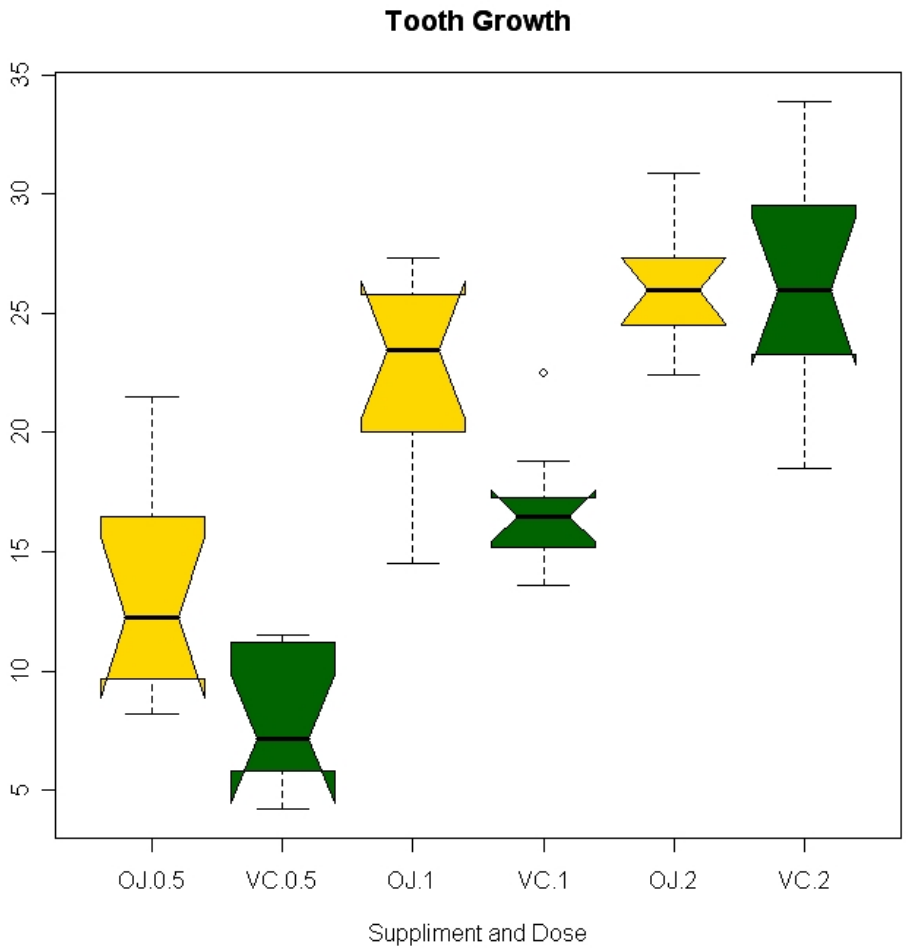


Bild-Quelle: http://betterposters.blogspot.com/2012/01/error-bars.html

Notched-Boxplot

- 95% Konfidenzintervalle des Medians

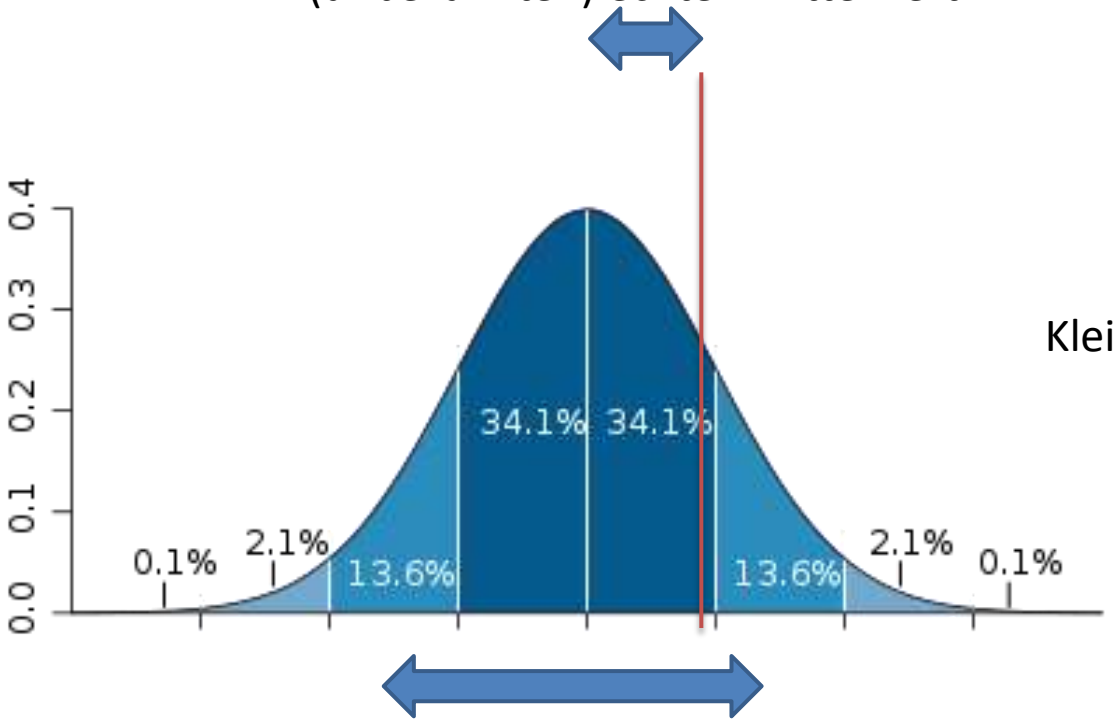



```
boxplot(c, notch=TRUE)
```

Genauigkeit vs. Präzision

Genauigkeit:
Abweichung gemessener Mittelwert vom (unbekannten) echten Mittelwert

insb. Zeitmessungen, typ. Vertrauen



Aufloesung:
Kleinster messbarer Unterschied

Präzision:
Streuung um den Stichprobenmittelwert (Wiederholbarkeit)

Ursache von Messfehlern kaum zuortbar

Zufaellige vs. Systematische Fehler

- Systematische Fehler – Fehler des Experiments / der Messmethode
  - CPU Speed: Messung bei unterschiedliche Temperaturen
  - Zustand nicht zurueckgesetzt fuer zweite Messung
  - Geringe Varianz, bis konstant ueber alle Messungen
  - Im Design ausschliessen, braucht Erfahrung
  - -> Genauigkeit
- Zufaellige Fehler
  - Nicht kontrollierbar
  - Stochastische Methoden
  - -> Präzision

Wie messen?

Intervall-Timer

- Differenz aus Startzeit und Endzeit
  - System Pulses
  - Wall Clock

- Genauigkeit der Messung

System. nanoTime();
System. currentTimeMillis();

```
long begin = System. nanoTime();
… execute  …
long end = System. nanoTime();
System.out.println(“Execution required “+(end-begin) “ ns”);
```

Timer Overhead

- Zeitanfrage verbraucht auch Aufwand

Zeitanfrage

Ereignis beginnt

Ereignis endet & Zeitanfrage

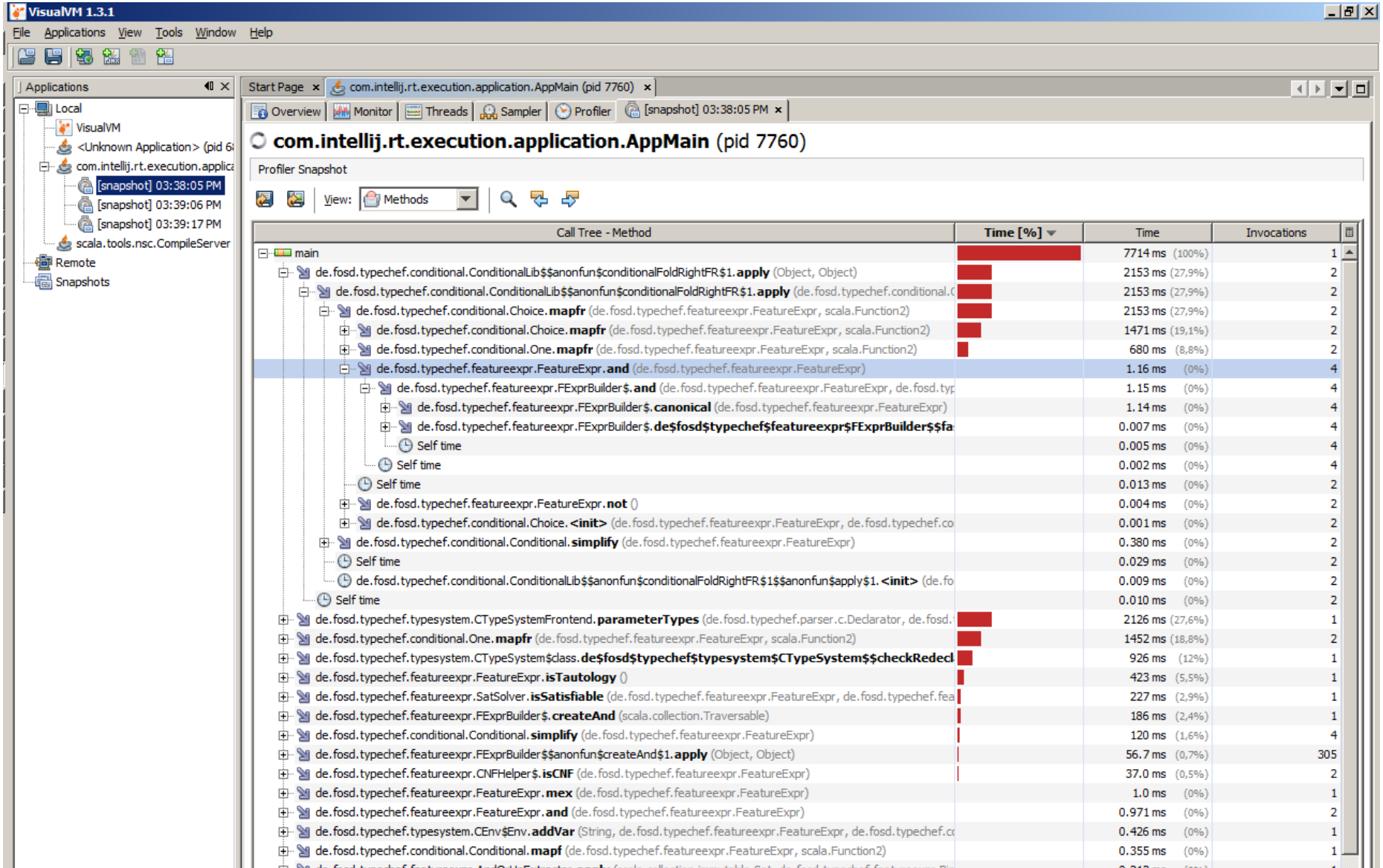
Zeit ermittelt

Endzeit gespeichert

Mindestens ein Speicherzugriff
ggf. Anfrage ans Betriebssystem

Gemessene Zeit sollte mind. 100-1000fach groesser als Overhead sein

Profiling


VisualVM 1.3.1
File Applications View Tools Window Help
Applications
Local
VisualVM
<Unknown Application> (pid 6
com.intellij.rt.execution.applica
[snapshot] 03:38:05 PM
[snapshot] 03:39:06 PM
[snapshot] 03:39:17 PM
scala.tools.nsc.CompileServer
Remote
Snapshots
Start Page
com.intellij.rt.execution.application.AppMain (pid 7760)
Overview Monitor Threads Sampler Profiler [snapshot] 03:38:05 PM
com.intellij.rt.execution.application.AppMain (pid 7760)
Profiler Snapshot
View: Methods
Call Tree - Method
Time [%]
Time
Invocations
main 7714 ms (100%) 1
de.fosd.typechef.conditional.ConditionalLib$$anonfun$conditionalFoldRightFR$1.apply (Object, Object) 2153 ms (27,9%) 2
de.fosd.typechef.conditional.ConditionalLib$$anonfun$conditionalFoldRightFR$1.apply (de.fosd.typechef.conditional.C 2153 ms (27,9%) 2
de.fosd.typechef.conditional.Choice.mapfr (de.fosd.typechef.featureexpr.FeatureExpr, scala.Function2) 2153 ms (27,9%) 2
de.fosd.typechef.conditional.Choice.mapfr (de.fosd.typechef.featureexpr.FeatureExpr, scala.Function2) 1471 ms (19,1%) 2
de.fosd.typechef.conditional.One.mapfr (de.fosd.typechef.featureexpr.FeatureExpr, scala.Function2) 680 ms (8,8%) 2
de.fosd.typechef.featureexpr.FeatureExpr.and (de.fosd.typechef.featureexpr.FeatureExpr) 1.16 ms (0%) 4
de.fosd.typechef.featureexpr.FExprBuilder$.and (de.fosd.typechef.featureexpr.FeatureExpr, de.fosd.typ 1.15 ms (0%) 4
de.fosd.typechef.featureexpr.FExprBuilder$.canonical (de.fosd.typechef.featureexpr.FeatureExpr) 1.14 ms (0%) 4
de.fosd.typechef.featureexpr.FExprBuilder$.de$fosd$typechef$featureexpr$FExprBuilder$$fa 0.007 ms (0%) 4
Self time 0.005 ms (0%) 4
Self time 0.002 ms (0%) 4
Self time 0.013 ms (0%) 2
de.fosd.typechef.featureexpr.FeatureExpr.not () 0.004 ms (0%) 2
de.fosd.typechef.conditional.Choice.<init> (de.fosd.typechef.featureexpr.FeatureExpr, de.fosd.typechef.co 0.001 ms (0%) 2
de.fosd.typechef.conditional.Conditional.simplify (de.fosd.typechef.featureexpr.FeatureExpr) 0.380 ms (0%) 2
Self time 0.029 ms (0%) 2
de.fosd.typechef.conditional.ConditionalLib$$anonfun$conditionalFoldRightFR$1$$anonfun$apply$1.<init> (de.fo 0.009 ms (0%) 2
Self time 0.010 ms (0%) 2
de.fosd.typechef.typesystem.CTypeSystemFrontend.parameterTypes (de.fosd.typechef.parser.c.Declarator, de.fosd. 2126 ms (27,6%) 1
de.fosd.typechef.conditional.One.mapfr (de.fosd.typechef.featureexpr.FeatureExpr, scala.Function2) 1452 ms (18,8%) 2
de.fosd.typechef.typesystem.CTypeSystem$class.de$fosd$typechef$typesystem$CTypeSystem$$checkRedecl 926 ms (12%) 1
de.fosd.typechef.featureexpr.FeatureExpr.isTautology () 423 ms (5,5%) 1
de.fosd.typechef.featureexpr.SatSolver.isSatisfiable (de.fosd.typechef.featureexpr.FeatureExpr, de.fosd.typechef.fea 227 ms (2,9%) 1
de.fosd.typechef.featureexpr.FExprBuilder$.createAnd (scala.collection.Traversable) 186 ms (2,4%) 1
de.fosd.typechef.conditional.Conditional.simplify (de.fosd.typechef.featureexpr.FeatureExpr) 120 ms (1,6%) 4
de.fosd.typechef.featureexpr.FExprBuilder$$anonfun$createAnd$1.apply (Object, Object) 56.7 ms (0,7%) 305
de.fosd.typechef.featureexpr.CNFHelper$.isCNF (de.fosd.typechef.featureexpr.FeatureExpr) 37.0 ms (0,5%) 2
de.fosd.typechef.featureexpr.FeatureExpr.mex (de.fosd.typechef.featureexpr.FeatureExpr) 1.0 ms (0%) 1
de.fosd.typechef.featureexpr.FeatureExpr.and (de.fosd.typechef.featureexpr.FeatureExpr) 0.971 ms (0%) 2
de.fosd.typechef.typesystem.CEnv$Env.addVar (String, de.fosd.typechef.featureexpr.FeatureExpr, de.fosd.typechef.c 0.426 ms (0%) 1
de.fosd.typechef.conditional.Conditional.mapf (de.fosd.typechef.featureexpr.FeatureExpr, scala.Function2) 0.355 ms (0%) 1

Trace Generation

- Log der Ausfuehrungsschritte, mit Zeiten
- Injektion durch Quelltextmanipulation/Aspekte/…
- Verschiedene Ereignisse moeglich

- Oft sehr grosse Datenmengen
  - Selektion
  - Kompression
  - Sampling

Indirekte und Ad-hoc Messungen

- z.B. Messen von Systemlast durch Ausfuehrungszeit eines Programs

- Gezielte Auswahl und sorgfaeltige Begruendung

Durchsatz messen

- Feste Zeit
- planbar, ggf. schnelles Feedback, inkrementell Nutzbar

- Transaktionen pro Sekunde
- Schreibzugriffe pro Sekunde

Micro-Benchmarks

- Messung einzelner Anweisungen

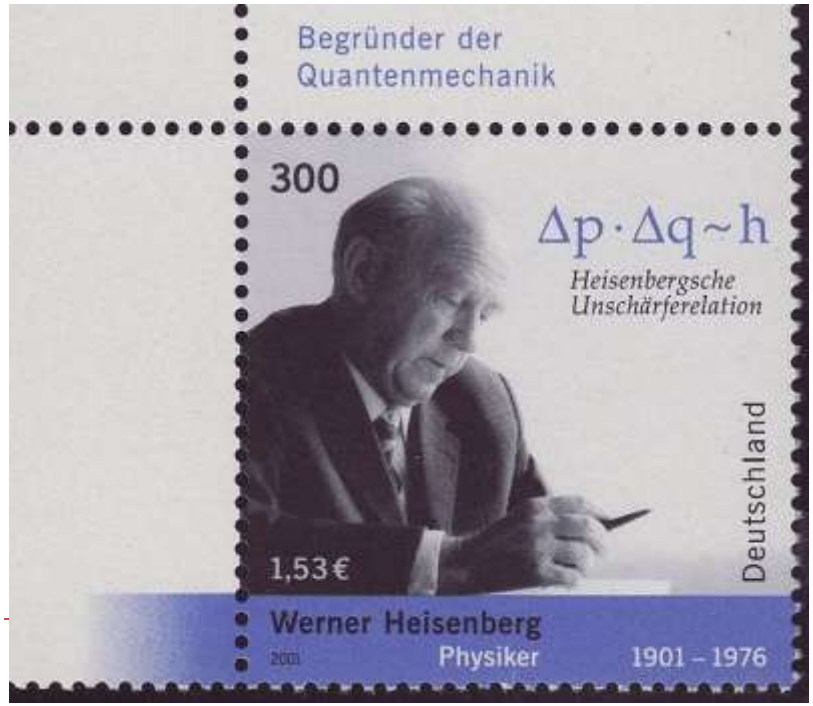
- Erfordern genaues Verstaendniss der Sprache (z.B. Optimierungen)
- Oft jenseits der Messgenauigkeit

Messbeeinflussung

- Die Messung selber kann das Ergebniss beeinflussen
- Insb. Quelltext-Instrumentierung

- Mehr Messungen -> Mehr Overhead -> Ungenauer

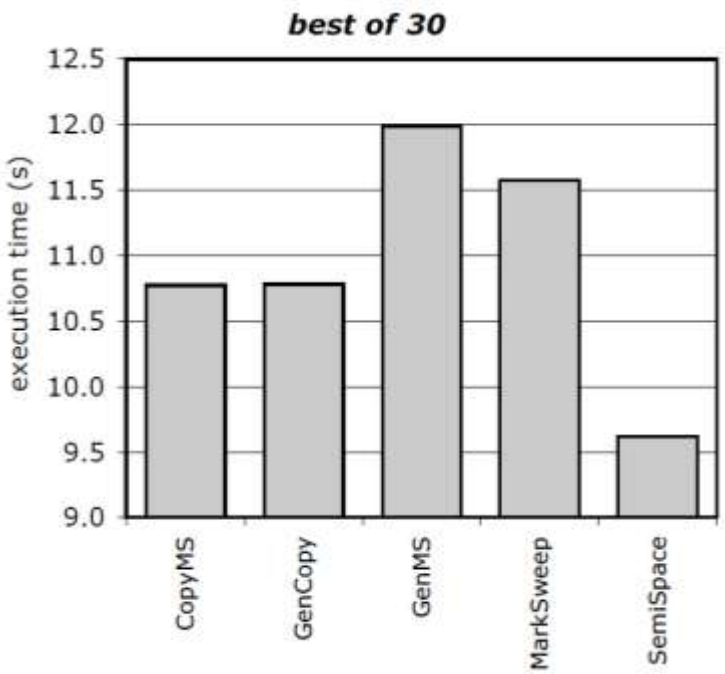
- Balance suchen

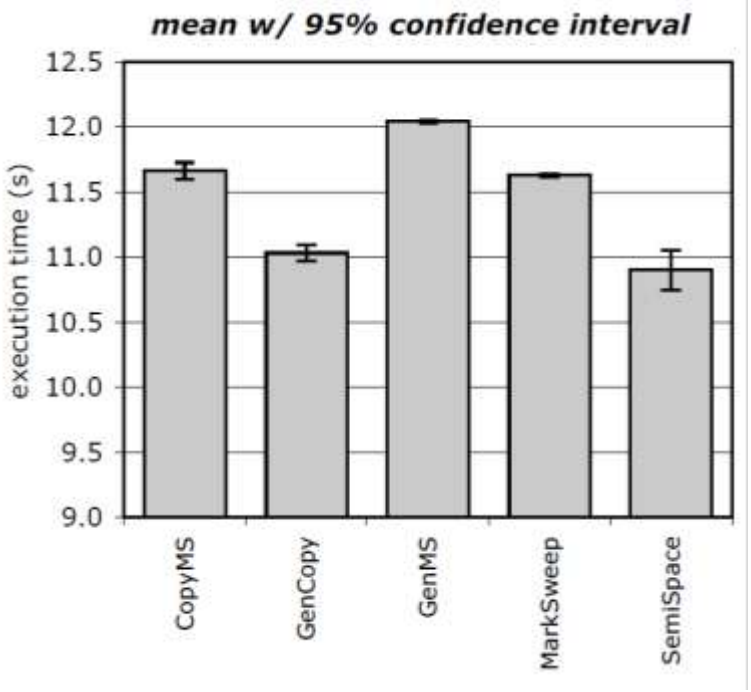


Vergleich von Messungen

Vergleich von Messergebnissen

- GenCopy schneller als GenMS?
- GenCopy schneller als SemiSpace?





Abhaengige vs. Unabhaengige Messungen

- Paarweise (abhaengige Messungen)
  - Vorher/Nachher Vergleich
  - Computer schneller mit zusaetzlicher Festplatte?
  - Neues Betriebssystem schneller?
  - Jeweils mit den gleichen Benchmarks/Eingabedaten
- Unabhaengige Messungen
  - etwa Eingabedaten je neu generiert
  - inkl. unterschiedlicher Anzahl an Messungen

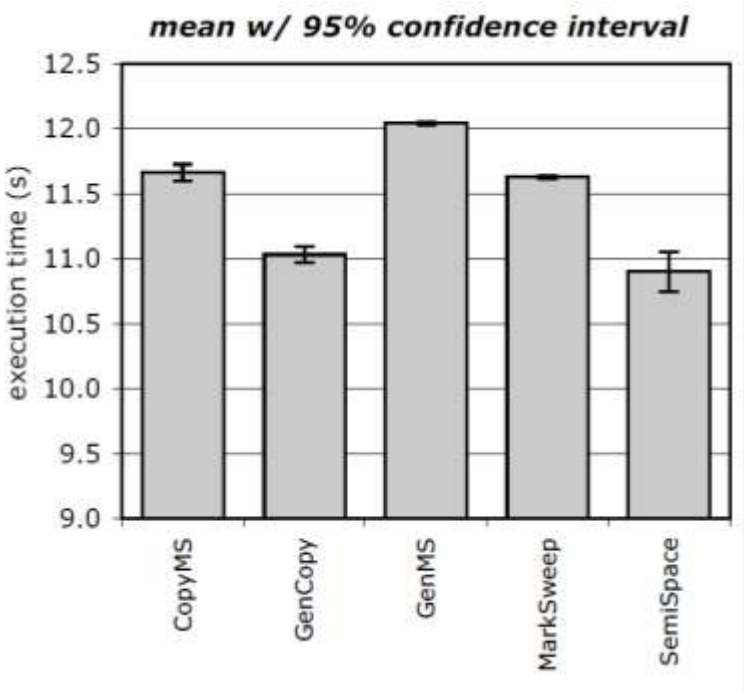
Signifikanzniveau

- Statistische Irrtumswahrscheinlichkeit
- Vor Durchfuehrung festlegen!
- Typisch
  - 0.05 signifikant
  - 0.01 sehr signifikant

- Statistisch signifikantes Ergebnis !=> Beweis
- Statistisch signifikantes Ergebnis !=> wichtiges Ergebniss
- Beruecksichtigt nur alpha-Fehler (dazu spaeter mehr)

Konfidenzintervalle Vergleichen

- Daumenregel: Ueberlappen sich die Konfidenzintervalle nicht ist der Unterschied signifikant



Bei Paarweisen (abhaengigen) Messungen

- z.B. Messpaare mit gleicher Eingabe
- etwa bei Vorher/Nachher-Vergleichen

- Differenz zwischen Messpaaren
- Konfidenzintervall fuer Differenzen bestimmen

- Signifikant wenn 0 *nicht* in Konfidenzinterval

t-Test

- Vorraussetzung: normalverteilte, metrische Daten
- Vergleich von 2 Messreihen
- Grundidee:
  - Annahme, dass Messreihen nicht unterschiedlich sind, also aus der gleichen Grundverteilung/Population kommen
  - t-Test bestimmt, wie wahrscheinlich das beobachtete Ergebnis ist, unter der Annahme, dass Messreihen nicht unterschiedlich sind
  - Wenn Wahrscheinlichkeit kleiner als 5% ist (Signifikanzniveau von 0.05), dann stimmt die Annahme wohl nicht

Berechnung t-Test

- Stichprobe 1: Umfang *n*, Mittelwert $\bar{x}$, Varianz ${s_x}^2$
- Stichprobe 2: Umfang *m*, Mittelwert $\bar{y}$, Varianz ${s_y}^2$

$$t = \sqrt{\frac{n*m}{n+m}} * \frac{\bar{x}-\bar{y}}{s} \qquad \text{mit} \qquad s^2 = \frac{(n-1){s_x}^2 + (m-1){s_y}^2}{n+m-2}$$

Signifikant wenn:

$$|t| > t(1-0.5*\alpha, n+m-2)$$

Anzahl Freiheitsgraden	P für zweiseitigen Vertrauensbereich							
	0,5	0,75	0,8	0,9	0,95	0,98	0,99	0,998
	P für einseitigen Vertrauensbereich							
	0,75	0,875	0,90	0,95	0,975	0,99	0,995	0,999
1	1,000	2,414	3,078	6,314	12,706	31,821	63,657	318,309
2	0,816	1,604	1,886	2,920	4,303	6,965	9,925	22,327
3	0,765	1,423	1,638	2,353	3,182	4,541	5,841	10,215
4	0,741	1,344	1,533	2,132	2,776	3,747	4,604	7,173
5	0,727	1,301	1,476	2,015	2,571	3,365	4,032	5,893
6	0,718	1,273	1,440	1,943	2,447	3,143	3,707	5,208
7	0,711	1,254	1,415	1,895	2,365	2,998	3,499	4,785
8	0,706	1,240	1,397	1,860	2,306	2,896	3,355	4,501

t-Test mit R

```
> t.test(x, y, conf.level=0.9)

        Welch Two Sample t-test

data:  x and y
t = 1.9988, df = 95.801, p-value = 0.04846
alternative hypothesis: true difference in means is not equal to 0
90 percent confidence interval:
 0.3464147 3.7520619
sample estimates:
mean of x mean of y
 51.42307  49.37383
```

`> t.test(x-y, conf.level=0.9)` (paired)

Beispiel

- Benchmark zweier Computer
- Aufgrund langer Laufzeit nur wenige Messungen

Messung	Zeit System 1 (in s)	Zeit System 2 (in s)
1	1011	894
	998	963
	1113	1098
	1008	982
	1100	1046
	1039	
	1003	
	1098	
	8	5
	1046,25	996,6
s_i	49,25	78,41

```
> t.test(x,y,conf.level=0.9)

        Welch Two Sample t-test

data:  x and y
t = 1.2682, df = 6.007, p-value = 0.2517
alternative hypothesis: true difference in
means is not equal to 0
90 percent confidence interval:
 -26.41009 125.71009
sample estimates:
mean of x mean of y
  1046.25    996.60
```

Ausblick: (multi-faktor-/multi-variate-) ANOVA

- Fuer Vergleiche zwischen mehr als 2 Messreihen

- Aber aufwendigere Verfahren
  - erfordert sehr viel mehr Messungen
  - benoetigt gute Statistikkenntnisse
  - teils unintuitive Interpretation

Validität (Threats to Validity)

- Konstruktvalidität (construct validity)
  - Messmethoden geeignet? Messen wir was wir behaupten zu messen? Systematische Fehler?
- Interne Validität
  - In sich schluessig? Alternativerklaerungen ausschliesbar? Stoervariablen kontrolliert?
- Externe Validität
  - Allgemeingültigkeit, Verallgemeinerungsfähigkeit?
- Statistische Validität
  - Statistische Methoden angemessen?
- dazu spaeter mehr

Beispiel fuer Diskussion Threats to Validity

level, XML-RPC's textual encoding of vectors can be more space efficient than the Java serialization mechanism used by sockets. However, the XML-RPC primitive might not be suitable for interactive Java applications because it frequently causes the transmission of very large packets (e.g., X-VV has an average packet size of 1455.90 bytes).

5.5 Threats to Validity

Any empirical study of the performance of local-remote software systems must confront certain threats to validity. During the empirical evaluation of communication primitive performance we were aware of potential threats to validity and we took steps to control the impact of these threats. Threats to internal validity concern factors that would present alternative explanations for the empirical results discussed in Section 5. The first threat to internal validity is related to the potential faults within the benchmarking framework that Section 3 describes. We controlled this threat by incorporating tools and libraries that are frequently used by practitioners, such as `tcpdump`, `capinfos`, `jvmstat`, `hprof`, and the `java.net` package. Since we have repeatedly used these tools without experiencing errors or anomalous results, we have a confidence in their correctness and we judge that they did not negatively impact the validity of our empirical study. Moreover, Allman also leveraged tools such as `tcpdump` without noticing problems that would compromise his results [2]. We also tested each benchmark in isolation in order to ensure that it regularly produced meaningful outcomes. Finally, we controlled threats to internal validity by using the same workstation and preventing additional user logins throughout experimentation.

Threats to external validity would limit the generalization of our approach and the empirical results to new benchmarks and execution environments. The experiments in this paper only focus on the use of four nano and four micro benchmarks and the performance measurements from these studies might be different from those produced by other nano, micro, macro, combined, and application-specific benchmarks. The experiments only evaluate sockets and XML-RPC and thus they provide no direct insights into the behavior of local-remote systems constructed with other communication primitives. Also, the client and server in the current benchmarks exchange a limited variety of parameters and return values. However, our framework supports the integration of other benchmarks and this paper reports on the results from using benchmarks that are similar to those that were described by Allman [2]. Another threat to external validity is related to the fact that the experiments measure the performance of sockets and XML-RPC in a single execution environment. This is because the computer hardware, JVM, operating system kernel, and other environmental factors were not varied during experimentation. Yet, it is our judgment that the execution environment is representative of one that is frequently used during the development and execution of a local-remote system.

Threats to construct validity concern whether the evaluation metrics accurately reflect the variables that the experiments were designed to measure. We judge that the response time metric is defined and measured in a fashion that will be useful to individuals who implement applications that perform intra-node communication. While network resource consumption is also a valuable metric, our current tools do not always support the accurate measurement of $W(B, P)$. As noted in Section 4.1, this is due to the fact that the high rates of data transfer rapidly exhaust the kernel buffer space that is reserved for packet capture. Before conducting further experiments we will attempt to control this threat to validity by modifying the GNU/Linux kernel and/or enhancing existing packet capture tools.

Philip F. Burdette, William F. Jones, Brian C. Blose, Gregory M. Kapfhammer. An Empirical Comparison of Java Remote Communication Primitives for Intra-Node Data Transmission. In ACM SIGMETRICS Performance Evaluation Review, Accepted, To Appear in 2012.
http://www.cs.allegheny.edu/~gkapfham/research/publish/burdette_per_paper.pdf

Wie oft messen?

Statistische Signifikanz vs. Oekonomische Signifikanz (Wirkung)

- Statische Signifikanz
  - Präzision der Messung gegenueber Zufallsfehlern
  - keine Aussage ueber groesse der Wirkung
- Mit sehr vielen Messungen auch statistische Signifikanz fuer kleine Unterschiede

- Primaer Wirkung betrachten (Unterschied der Mittelwerte), nicht statistische Signifikanz

Anzahl der Messungen

- Gewuensche Praezession +/- e%
- Geschaetztes s und $\bar{x}$
  - aus Vortest (kleine Messung)

$$n = \left(\frac{S * Z_{1-\alpha/2}}{e * \bar{x}}\right)^2$$

- Beispiel:
  - α=0.1, mean=7.94, s=2.14
  - e=0.05 => n = 105
  - e=0.025 => n = 418

$$n = \left(\frac{1.895 * s}{e * \bar{x}}\right)^2$$ fuer α=0.1

- Alternativ: So lange Messen bis Konfidenzintervall erreicht.

Experiment 2

- Gleicher Stil wie Experiment 1, andere Fragestellung
- Als Probandencode wieder 3 Buchstaben verwenden, am besten die gleichen
- Auf Pool-Rechnern:
  - S:\LEHRE\2012SS\Kaestner\exp2
- Zuhause:
  - Experimentdatei von Webseite runterladen
  - Zip Datei entpacken (!)
  - Experiment selber durchfuehren
  - Emailversand wird i.d.R. scheitern, bitte .zip Datei mit eurem Probandencode per Email an christian.kaestner@uni-marburg.de

Was messen?

Auswahl des Benchmarks

- Auswahl der Eingabedaten kann Ergebnisse massiv beeinflussen
  - Beispiel: teil-sortierte Daten fuer Sortieralgorithmen
  - Beispiel: kleine vs. sehr grosse Datenmengen (groesser als Hauptspeicher)
- Auswahl von Benchmarks muss **begruendet** sein

Typische Auswahlstrategien

- Wiederverwendung bestehender Benchmarks
- Guenstige/Kritische Faelle gezielt ausgewaehlt/konstruiert
- Typische reale Szenarien (belegen!)
- Echte industrielle Datensaetze
- Zufaellig generierte Daten (Parameter begruenden!)

- **Objektive Begruendung wichtig!**

Korpus

- Sammlung von Benchmarks, verwendet von vielen Forschern
- Oft gezielt als repraesentativ ausgewaehlt
- Foerdern gezielte und vergleichbare Forschung (siehe z.B. SAT)
- Beispiele
  - DaCapo Benchmark Suite fuer Java: Ausfuehren von grossen Open-Source Java-Programmen (AVR Simulator, Eclipse Tests, Jython Interpreter, XML Transformation, ...) http://dacapobench.org/
  - Matrix Market fuer Matrixmultiplikation http://math.nist.gov/MatrixMarket/
  - TCP Datenbank-Benchmark http://www.tpc.org/

Achtung: Auch Benchmark-Korpuse oft umstritten, z.B., DaCapo mit unterdurchschnittlicher Speicherbenutzung

Sort Benchmark Home Page

New: We are happy to announce the 2011 winners listed below. The new, 2011 records are listed in green. Congratulations to the winners!

The next deadline for submitting entries is April 1, 2012. There are no rule changes for 2012.

Background

Until 2007, the sort benchmarks were primary defined, sponsored and administered by Jim Gray. Following Jim's disappearance at sea in January 2007, the sort benchmarks have been continued by a committee of past colleagues and sort benchmark winners. The Sort Benchmark committee members include:

- Chris Nyberg of Ordinal Technology
- Mehul Shah of Hewlett-Packard
- Naga Govindaraju of Microsoft

Top Results

	Daytona	Indy
Gray	2011, 0.725 TB/min **TritonSort** 1,000,000,000,000 records in 8,274 seconds 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego	2011, 0.938 TB/min **TritonSort** 1,000,000,000,000 records in 6,395 seconds 52 nodes x (2 Quadcore processors, 24 GB memory, 16x500GB disks) Cisco Nexus 5096 switch Alex Rasmussen, Michael Conley, George Porter, Amin Vahdat, University of California, San Diego
Penny	2011, 286 GB **psort** 2.7 Ghz AMD Sempron, 4 GB RAM, 5x320 GB 7200 RPM Samsung SpinPoint F4 HD332GJ, Linux Paolo Bertasi, Federica Bogo, Marco Bressan and Enoch Peserico Univ. Padova, Italy	2011, 334 GB **psort** 2.7 Ghz AMD Sempron, 4 GB RAM, 5x320 GB 7200 RPM Samsung SpinPoint F4 HD332GJ, Linux Paolo Bertasi, Federica Bogo, Marco Bressan and Enoch Peserico Univ. Padova, Italy

Verfuegbarkeit und Reproduzierbarkeit von Benchmarks

	language	sources	items	unit	accessibility	reproducibility
⟨AlvesV05⟩	SDF	8	27	grammars	partial	approximate
⟨BaxterFNRSVMT06⟩	Java	17	56	projects	full	precise
⟨ChevanceH78⟩	COBOL	1	50	programs	none	none
⟨CollbergMS04⟩	Java	1	1132	JAR files	full	approximate
⟨CookL82⟩	Pascal	1	264	programs	none	none
⟨CranorESMC08⟩	P3P	3	?	policies	full	approximate
⟨GilM05⟩	Java	4	14	projects	full	precise
⟨HageK08⟩	Haskell	1	68000	compilations	none	approximate
⟨Hahsler04⟩	Java	1	988	projects	full	approximate
⟨Hautus02⟩	Java	?	9	packages	full	approximate
⟨KimSNM05⟩	Java	2	2	programs	full	approximate
⟨Knuth71⟩	Fortran	7	440	programs	none	none
⟨LaemmelKR05⟩	XSD	2	63	schemas	partial	none
⟨LaemmelP10⟩	P3P	1	3227	policies	full	trivial
⟨ReayDM09⟩	P3P	1	2287	policies	full	approximate
⟨SaalW77⟩	APL	6	32	workspaces	none	none
⟨Visser06⟩	XSD	9	9	schemas	full	approximate

from Jean-Marie Favre, Dragan Gašević, Ralf Lämmel and Ekaterina Pek, Empirical language analysis in software linguistics, In post-proceedings of the 3rd International Conference on Software Language Engineering (SLE 2010).

Stoerfaktoren Speicher

Variabilitaet zwischen Durchlaeufen

- Startzustand kann grossen Einfluss haben
- Mehre Durchlaeufe statt lange Messreihen (inkl. Computer neustarten)

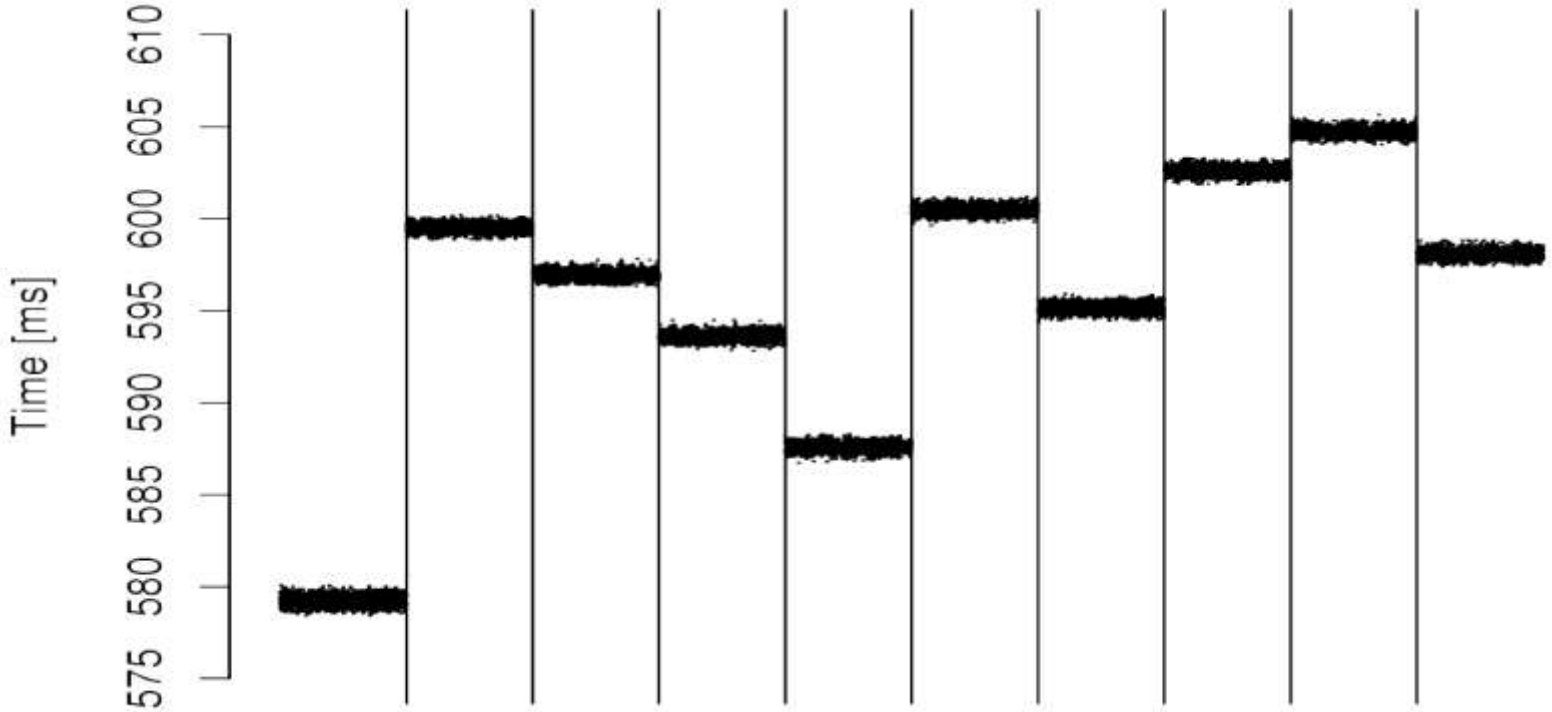
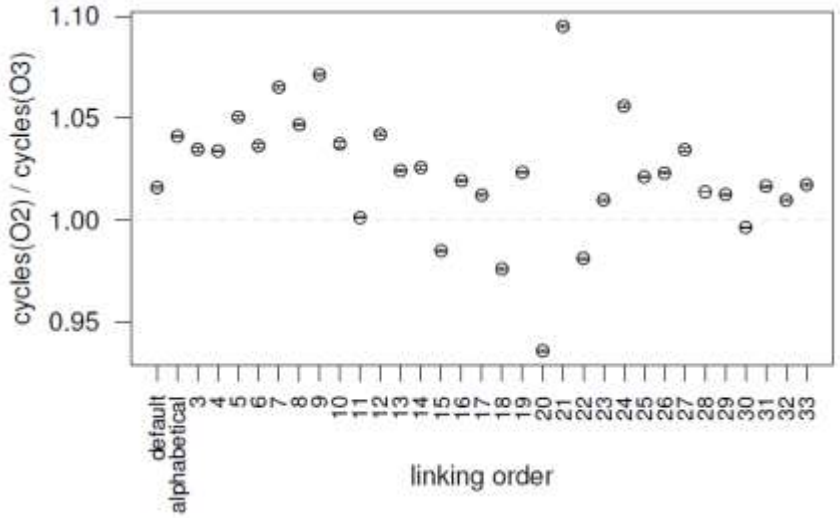
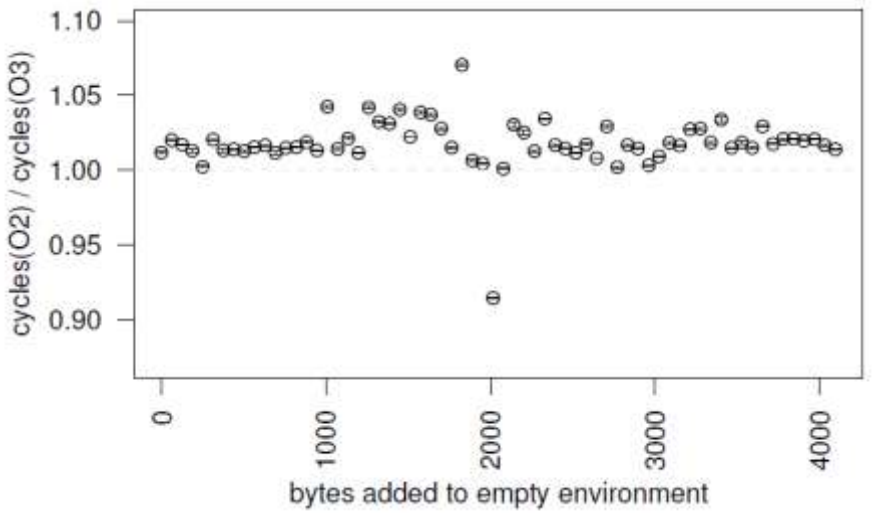


Abbildung: Kalibera, T. et al. 2005. **Benchmark precision and random initial state**. In Proc. SPECTS 2005

Performanceunterschiede durch Speicherlayout

- Speicherzuweisung (Virtuelle Adressen, Seitenzugriffe, Caching, …)
- Compiler-/Linker-Reihenfolge (Speicherlayout; teils zufaellig in Compiler)
- Groesse des UNIX-Environments (aendert Platzierung im Speicher)

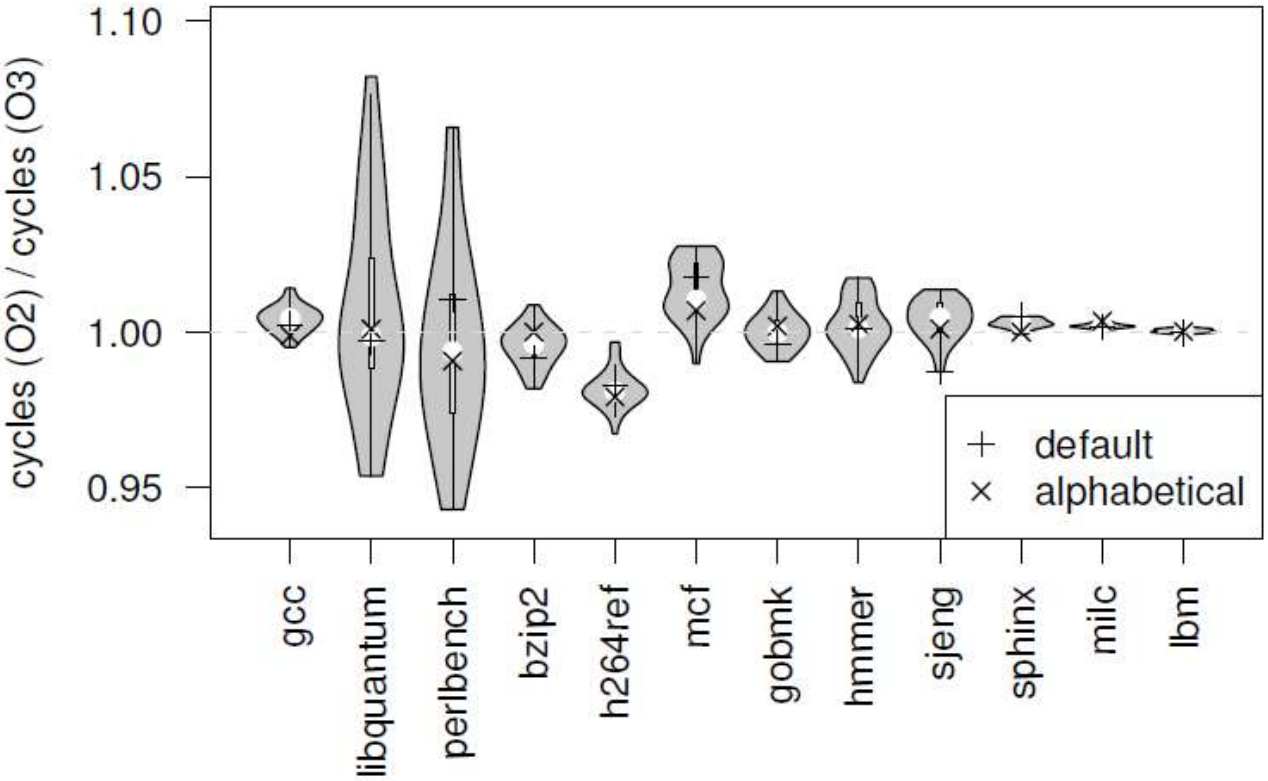




Abbildungen: T. Mytkowicz, et al. 2009. **Producing wrong data without doing anything obviously wrong!**. In Proc. ASPLOS. ACM, 265-276.

Measurement Bias

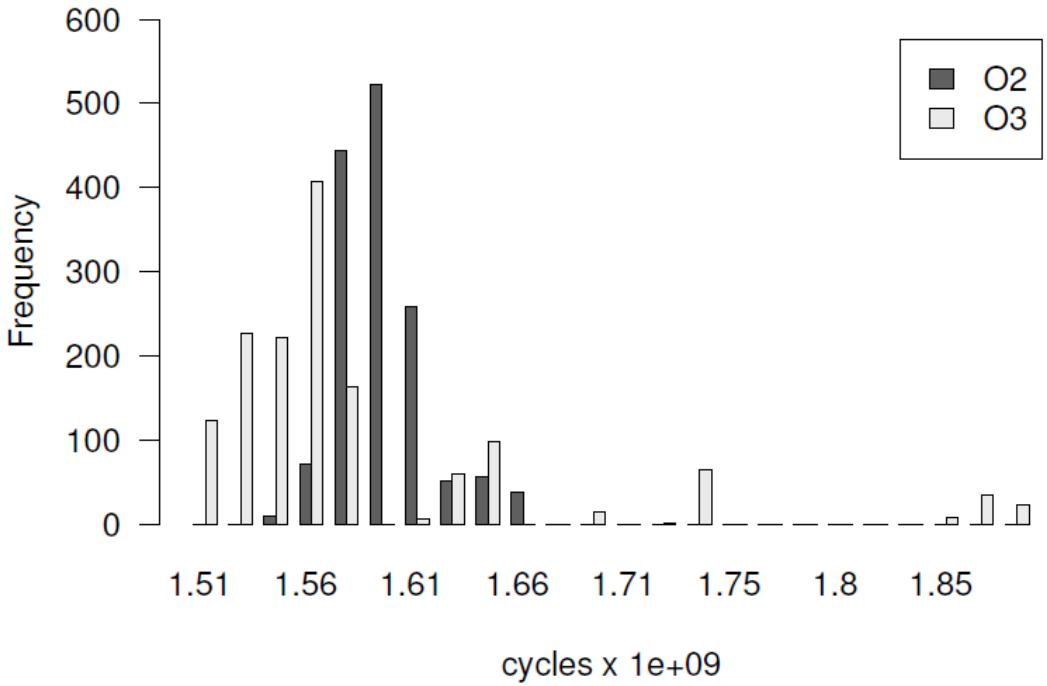
- Erhebliche Messfehler moeglich
- Insb. bei kleinen Unterschieden gefaehrlich



(a) Link Order

Fehler vermeiden

- Grosse, unterschiedliche Benchmarks
- Bewusst Messungen in unterschiedlichen Setups (Linkerreihenfolgen, Speicherbelegung etc)



Stoerfaktoren in Java

JIT Compilation - Warmup

- Uebersetzung und Optimierung nur bei Bedarf
  - time-based **sampling**
  - Verschiedene Optimierungslevel
- Messziel?
  - Typische Laufzeitperformance
  - Startup-Performance
- Erste Iteration nicht gewertet (Warmup)?
- Bei Messwiederholung JVM neu starten?

JIT Warmup Strategien

- Erste Iteration als Warmup
- Erste Iterationen als Warmup bis Konfidenzintervall in gewuenschter Groesse, z.b. +/- 2%
- Uebersetzten aller Methoden erzwingen, dann JIT abschalten
- Replay Compilation
  - Benchmark ausfuehren
  - Loggen welche Methoden wie uebersetzt werden
  - Vor Start des echten Benchmarks genau diese Methoden so uebersetzten
  - JIT abschalten
  - Alternativ weitere Anpassungen

Garbage Collection

- Garbage Collection nichtdeterministisch
- Kann explizit durchgefuehrt/abgeschaltet werden

- z.B. Garbage Collection durchfuehren vor Benchmarklauf

Verschraenktes Messen?

- In welcher Reihenfolge wird gemessen?
- aaabbb oder abababab?

Einflussgroessen kontrollieren

- Eine Heap-Konfiguration oder mehrere?

```
-Xmx512M -XX:PermSize=64M -XX:MaxPermSize=128M
```

- Ein System oder mehrere?
- Ein Betriebssystem oder mehrere?
- Eine JVM oder mehrere?

Java Benchmarks in der Forschung

methodology	A	B	C	D	E	F	G	H	I	J	K	L	M
reference	[4]	[18]	[21]	[22]	[25]	[1]	[2]	[20]	[7, 14]	[23]	[8]	[3]	[9]
Data analysis													
average performance number from multiple runs	√	√	√									√	
median performance number from multiple runs							√						
best performance number from multiple runs					√	√	√		√	√	√		
second best performance number from multiple runs													√
worst performance number from multiple runs										√			
confidence interval from multiple runs		√		√									
Experimental design													
one VM invocation, one benchmark iteration												√	
one VM invocation, multiple benchmark iterations		√	√			√					√		
multiple VM invocations, one benchmark iteration					√	√							√
multiple VM invocations, multiple benchmark iterations				√			√						
including JIT compilation			√			√	√		√				
excluding JIT compilation		√			√			√	√				
all methods are compiled before measurement		√			√						√		
replay compilation								√	√				
full-heap garbage collection before measurement		√							√				
a single hardware platform	√	√	√		√	√	√	√		√	√		√
multiple hardware platforms				√					√			√	
a single heap size						√	√			√			
multiple heap sizes	√	√	√	√	√			√	√		√	√	√
a single VM implementation	√	√	√	√	√	√	√	√			√	√	√
multiple VM implementations									√	√			
back-to-back measurements													
interleaved measurements									√				

Empfohlendes Vorgehen

nach A. Georges, D. Buytaert, and L. Eeckhout. 2007. Statistically rigorous Java performance evaluation. In Proc. OOPSLA. ACM, 57-76

- q Durchlaeufe mit je k verwendeten Messungen
- q mal die VM starten
  - w Durchlaeufe Warmup bis stabil (Stdev/Mean < 0.02)
  - dann k Messungen
- Pro Durchlauf Mittelwert bestimmen
- Konfidenzintervall ueber alle Durchlaeufe bilden

Was nun?

Kein perfekter Benchmark

- Mehr Messungen, mehr Scenarios, mehr Parameter => i.d.R. mehr Aufwand, aber genauer
- Alles kann nicht berücksichtigt werden
- Leben mit Ungenauigkeit

- **Balance Genauigkeit – Aufwand**
- Abhängig von Effektgroesse und Bedeutung des Ergebnisses
  - 2% vs. 8x Beschleunigung
  - Performanceverbesserung vs. Demonstration v. Praktikabilität
  - Exploration vs. Evaluierung

Diskussion: State of the Art

- Pragmatismus: Was ist zur Zeit akzeptiert
- Idealismus: Was sollte State of the Art sein?

- Akzeptierte Standards fehlen oft

Aufgaben

- Schlagen Sie ein Vorgehen vor, um folgende Loesungen zu evaluieren:
  - Die JavaScript-Engine von Browser X ist 20% schneller als die Vorgaengerversion
  - Der Algorithmus X kann eine angenaeherte Loesung fuer das Traveling-Salesman-Problem in linearer statt exponentieller Zeit berechnen
  - Der Vorschlag zur neuen Speicherverwaltung von Linux verringert den Speicherbedarf von Anwendungen um etwa 2%
  - Ein neues NoSQL-Datenbanksystem verspricht bessere Performance gegenueber relationalen Datenbanken
  - C-Programme sind schneller als Java-Programme

Messergebnisse berichten

Struktur

- Trennung: Ziele – Aufbau – Daten – Interpretation – Threats to Validity
- Ziele
  - Was soll gemessen werden und warum?
  - Welche Aussage soll belegt/widerlegt werden?
- Aufbau
  - Alle Informationen, die zur Replikation notwendig sind
  - Beschreibung Versuchskonzept
  - Beschreibung Daten (wenn moeglich verfuegbar machen)
  - Wie gemessen, wie oft?
  - Welche Einflussfaktoren kontrolliert?
  - Verwendete Hardware?

Struktur (II)

- Daten
  - Ergebnisse beschreiben, deskriptive Statistiken (Mittelwerte, Boxplots, Histogramme, Konfidenzintervalle, …)
  - Hypothesen testen (t-Test, p-Wert, …)
  - **Objektiv**
- Interpretation/Diskussion
  - Was sagen die Daten (wahrscheinlich) aus?
  - Warum belegen Sie die Hypothese?
  - **Subjektiv**

- Wieder: **Balance** Genauigkeit – Aufwand/Relevanz

Struktur (III)

- Validität (Threats to Validity)
  - Genauigkeit und Grenzen diskutieren
  - Konstruktvalidität (construct validity)
    - Messmethoden geeignet? Systematische Fehler?
  - Interne Validität
    - In sich schluessig? Alternativerklaerungen ausschliesbar? Stoervariablen kontrolliert?
  - Externe Validität
    - Allgemeingültigkeit, Verallgemeinerungsfähigkeit?
  - Statistische Validität
    - Statistische Methoden angemessen?
  - dazu spaeter mehr

Aufgaben

- Schreiben Sie einen kurzen Bericht ueber die Ergebnisse ihres QuickSort Benchmarks

Aufgaben

- Lesen Sie folgenden Artikel (insb. Abschnitt 4 – 4.3)
  - Boehm et al. Generalized Just-In-Time Trace Compilation using a Parallel Task Farm in a Dynamic Binary Translator. In PLDI'11
  - http://groups.inf.ed.ac.uk/pasta/papers/pldi044-bohm.pdf
- Wie wurde dort Performane evaluiert?
- Bewerten Sie die Evaluierung und den Bericht darueber. Wie wurden Benchmarks und Messverfahren ausgewaehlt und begruendet? Fehlen Daten? Fehlen Untersuchungen? Schlagen Sie ggf. Verbesserungen vor.
- Machen Sie sich Notizen fuer eine gemeinsame Diskussion.

Bewerten von Evaluierung

- Waehlen Sie eine der folgenden drei Veröffentlichungen:
  - Reichenbach, Immerman, Smaragdakis, Aftandilian, and Guyer. 2010. **What can the GC compute efficiently?: a language for heap assertions at GC time.** In OOPSLA'10
  - Garner, Blackburn, and Frampton, **A Comprehensive Evaluation of Object Scanning Techniques.** In ISMM ‘11
  - Curtsinger and Berger. **STABILIZER: Enforcing Predictable and Analyzable Performance.** Technical Report UMass-CS-TR2011-43
- Bonus:
  - Burdette, Jones, Blose, Kapfhammer. **An Empirical Comparison of Java Remote Communication Primitives for Intra-Node Data Transmission**. In ACM SIGMETRICS Performance Evaluation Review, To Appear in 2012.
  - Esmaeilzadeh, Cao, Xi, Blackburn, and McKinley. **Looking back on the language and hardware revolutions: measured power, performance, and scaling**. In ASPLOS '11

Bewerten von Evaluierung (forts.)

- Wie wurde dort Performane evaluiert?
- Bewerten Sie die Evaluierung und den Bericht darueber. Fehlen Daten? Fehlen Untersuchungen? Ist die Evaluierung angemessen? Schlagen Sie ggf. Verbesserungen vor.
- Stellen Sie ihre Ergebnisse kurz vor (Tafelbild/Flipchart/Folie).

Hinweise zu Leseaufgaben

- Fokus auf der Evaluierung
  - Querlesen des Rests reicht!
- Grobes Verstaendis entwickeln, was vorgeschlagen und evaluiert wird (oft reicht dafuer die Einleitung)
- Technische Details oft nachrangig

Zusammenfassung

- Verschiedene Messverfahren und Metriken
- Auswahl von Benchmarks
- Umgang mit Stoerfaktoren
  - Genauigkeit vs Präzision
  - Mehrfach messen, Einfluesse kontrollieren oder variieren
  - Mittelwerte, Verteilungen, Visuelle Darstellungen, Konfidenzintervalle, Signifanztests, …
- Validitaet
- Ergebnisse Berichten

Literatur

- A. Georges, D. Buytaert, and L. Eeckhout. 2007. **Statistically rigorous java performance evaluation**. In Proc. OOPSLA. ACM, 57-76
- Kalibera, T. and Bulej, L. and Tuma, P. 2005. **Benchmark precision and random initial state**. In Proc. SPECTS 2005
- T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney. 2009. **Producing wrong data without doing anything obviously wrong!**. In Proc. ASPLOS. ACM, 265-276.
- D Lilja. **Measuring Computer Performance: A practitioner's guide**. Cambridge University Press. 2000
- + Statistik-Buch