Rechnerarchitektur SS 2020

Amdahl's Law and Performance Measures

Jian-Jia Chen

TU Dortmund

Teilweise basierend auf Material von Michael Engel, Gernot A. Fink und R. Yahyapour

27 April 2020

Outline

Mehrprozessorsysteme

"The turning away from the conventional organization came in the middle 1960s, when the law of diminishing returns began to take effect in the effort to increase the operational speed of a computer ... Electronic circuits are ultimately limited in their speed of operation by the speed of light ... and many ... were already operating in the nanosecond range."

W. Jack Bouknight et al.: The Illiac IV System (1972)

"... sequential computers are approaching a fundamental physical limit on their potential power. Such a limit is the speed of light ..."

Angel L. DeCegama: The Technology of Parallel Processing (1989)

Mehrprozessorsysteme: Einführung

Beobachtung: Einschätzung, dass Fortschritte bei Uni-Prozessor-Architekturen sich ihrem Ende nähern, wurde verschiedentlich und zu verschiedenen Zeiten vertreten.

Allerdings: Allerdings: 1985-2000 höchste Leistungssteigerung bei Uni-Prozessoren zu beobachten (seit d. Einführung transistorbasierter Rechner Ende der 1950er)

$\Rightarrow$ Triebkraft: Verwendung/Technologie von Mikroprozessoren

Trotzdem: Bedeutung paralleler Prozessoren wächst und wird in der Zukunft weiter steigen

Mehrprozessorsysteme: Einführung II

Gründe für wachsende Bedeutung von MPs

- Mikroprozessoren werden vorherrschende Technologie für Uni-Prozessoren bleiben

  $\Rightarrow$ Weg zur Leistungssteigerung: Kombination mehrerer Prozessoren (seit einiger Zeit auch on-chip, z.B. Dual/Quad-Core-Prozessoren, MPSoCs)

  Annahme: Kosteneffektiver als Spezialprozessoren
- Stetiger Fortschritt bei größter Hürde für MP-Einsatz: *Software*
  - Groß bei Server-Anwendungen (haben häufig "natürlichen" Parallelismus)
  - Problematischer bei Desktop-Systemen/Anwendungen
  - Ex. erste weiter verbreitete Programmiersysteme (z.B. OpenMP)

Mehrprozessorsysteme: Einführung III

Motivation der Themenbehandlung

- Bereits seit 2000 Abschwächung der Leitstungssteigerung bei Uni-Prozessoren bemerkbar!

  Trotzdem: Steigerung der Integrationsdichte folgt (vorerst?) ungebrochen dem Moore'schen Gesetz ... betrifft aber mehr und mehr Intergration *weiterer Komponenten* on-chip!

- Vorstoss von MPs in breitere Märkte vollzogen (im Desktop/Server-Bereich: Multi-Core-Prozessoren)
- Parallelisierung *immer schon* wichtigste algorithmische Triebkraft bei der Leistungssteigerung:
  MPs wenden Prinzip nur auf anderer Granularitätsstufe an (i. Vgl. zu Pipelining)

$\Rightarrow$ Mehrprozessorsysteme werden in Zukunft weiter an Attraktivität gewinnen!

Taxonomie von Mehrprozessorsystemen

Idee der Leistungssteigerung durch Paralleles Rechnen bereits i.d. Anfängen der Rechnerentwicklung verfolgt

Taxonomie möglicher Architekturen nach Flynn (1966)
(betrachtet Parallelismus im Befehls- und Datenstrom)

Single instruction, single data (SISD)

- Entspricht "klassischem/normalem" Uni-Prozessor-System

Single instruction, multiple data (SIMD)

- Gleicher Befehl(-sstrom) wird von mehreren Prozessoren auf unterschiedlichen Daten(strömen) ausgeführt
- Eigene Datenspeicher pro Prozessor, aber gemeinsamer Befehlsspeicher und Kontrolleinheit (zum Holen/Ausgeben der Befehle)

Beispiel: Vektorrechner
Vergleichbar: Multimedia-Befehle $\hat{=}$ eingeschränkte Variante

Taxonomie von Mehrprozessorsystemen II

Single instruction, single data (SISD) ...

Single instruction, multiple data (SIMD) ...

Multiple instruction, single data (MISD)

- Ex. Mehrere Verarbeitungsknoten, in Matrix angeordnet, bearbeiten verschiedene Befehlsströme
- Ein Datenstrom durchläuft Berechnungsmatrix
- Bezeichnet als systolisches Array

Multiple instruction, multiple data (MIMD)

- Jeder Prozessor bearbeitet eigenen Befehls- und eigenen Datenstrom
- Verwendung "normaler" Mikroprozessoren möglich!
- Verbreitetstes Architektur-Schema paralleler Rechner

Taxonomie von Mehrprozessorsystemen: Diskussion

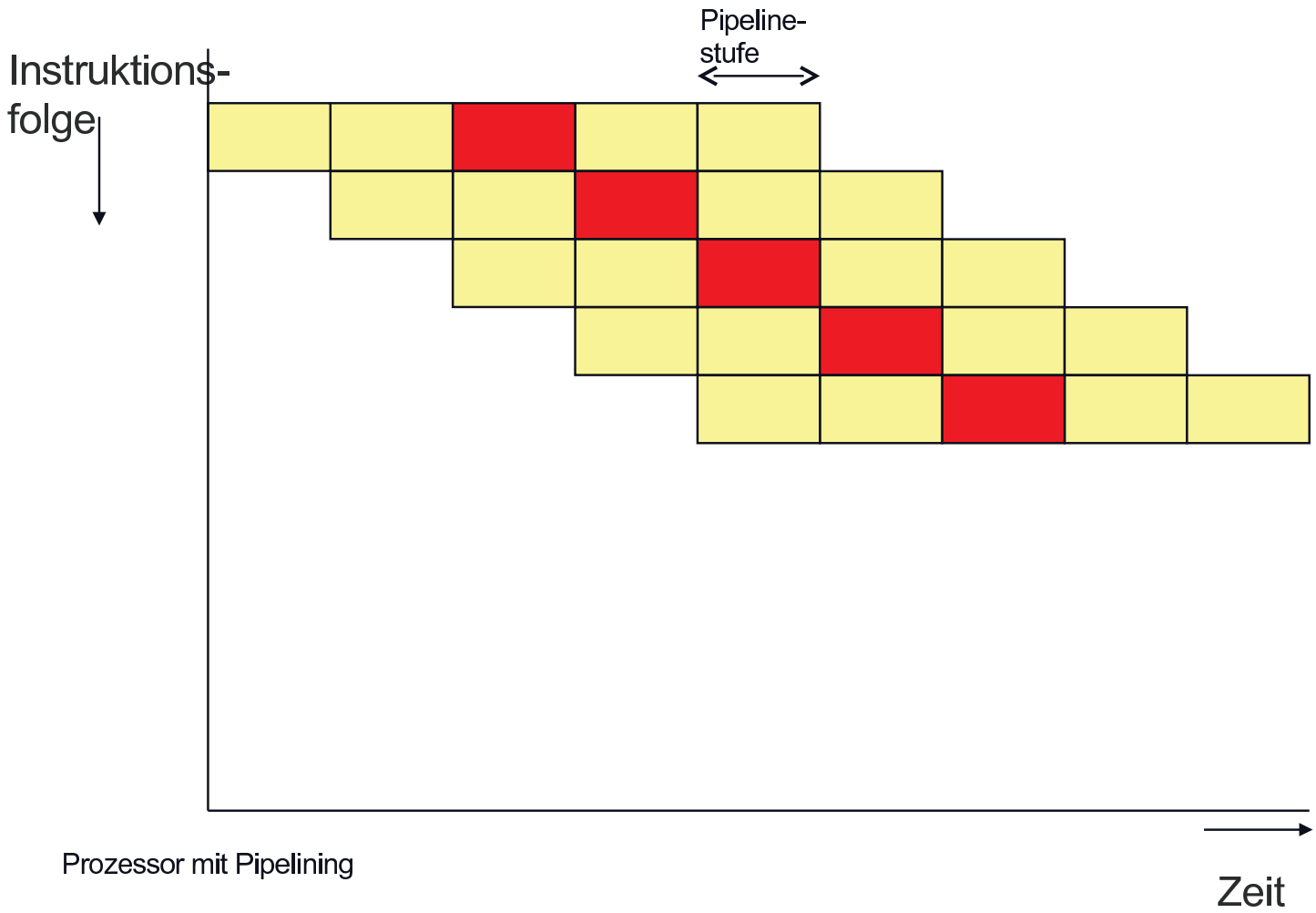
- Grobe Kategorisierung $\Rightarrow$ Ex. "hybride" Formen
- SIMD und MISD nur für spezielle Berechnungen geeignet ($\Rightarrow$ keine kommerziellen MISD-Systeme)
- Viele frühe MPs nach SIMD-Prinzip, seit 1990er praktisch verschwunden (Ausnahme: Vektorrechner)
- MIMD ist vorherrschende Architektur für "*general purpose*" Mehrprozessorsysteme
  - Große Flexibilität (gesamtes MP für eine(n) Anwendung/Benutzer bis parallele Bearbeitung vieler unterschiedlicher Tasks/Prozesse)
  - Können von Kosten-Leistungs-Vorteilen gängiger Mikroprozessoren profitieren

    $\Rightarrow$ praktisch alle heutigen MIMD-Systeme mit gleichen Mikroprozessoren wie Server/Desktops realisiert!

Welche Arten der Parallelität kennen wir?

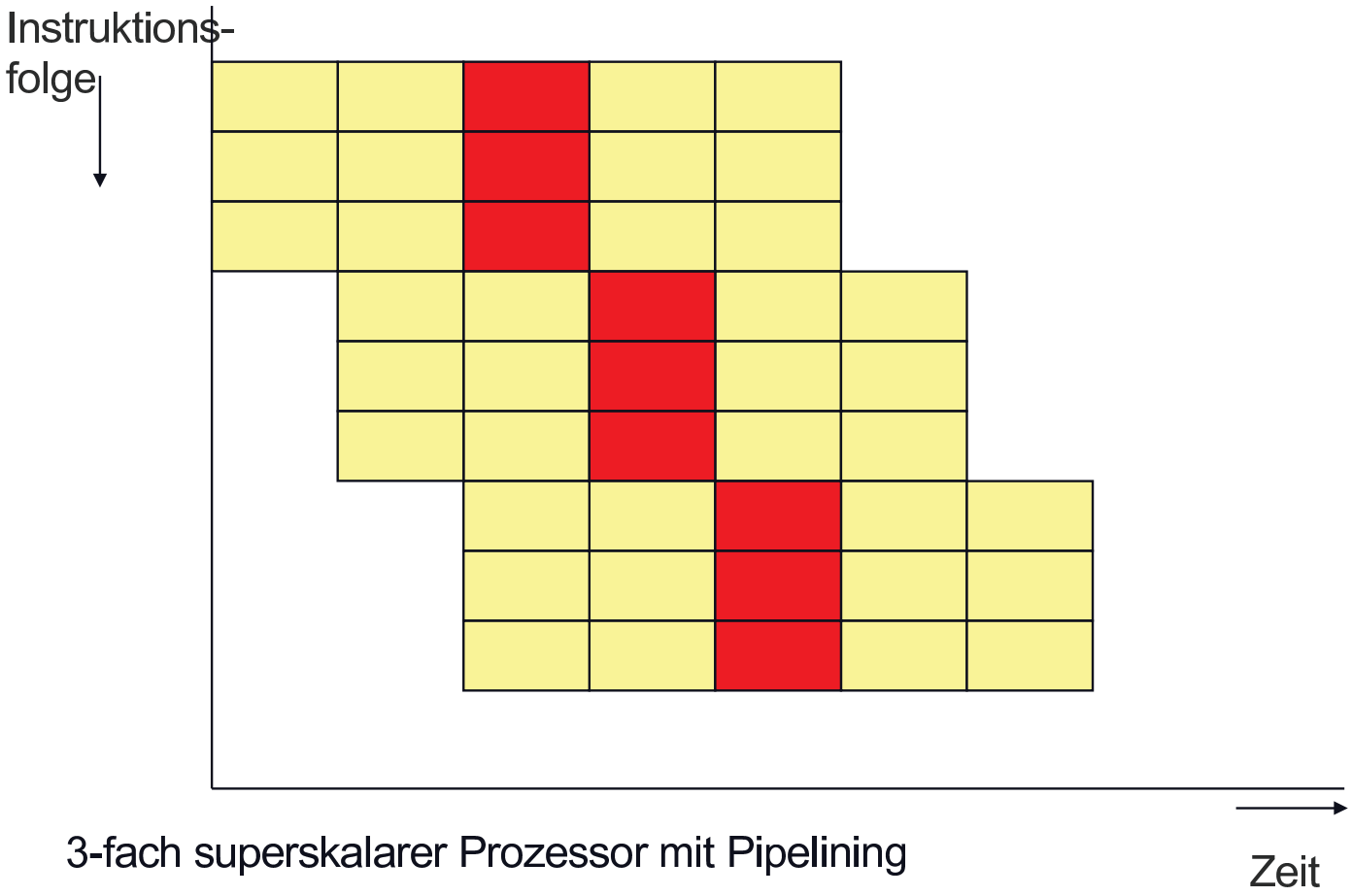
Parallelität: Pipelining

Quelle: R. Yahyapour, ehem. TU Dortmund



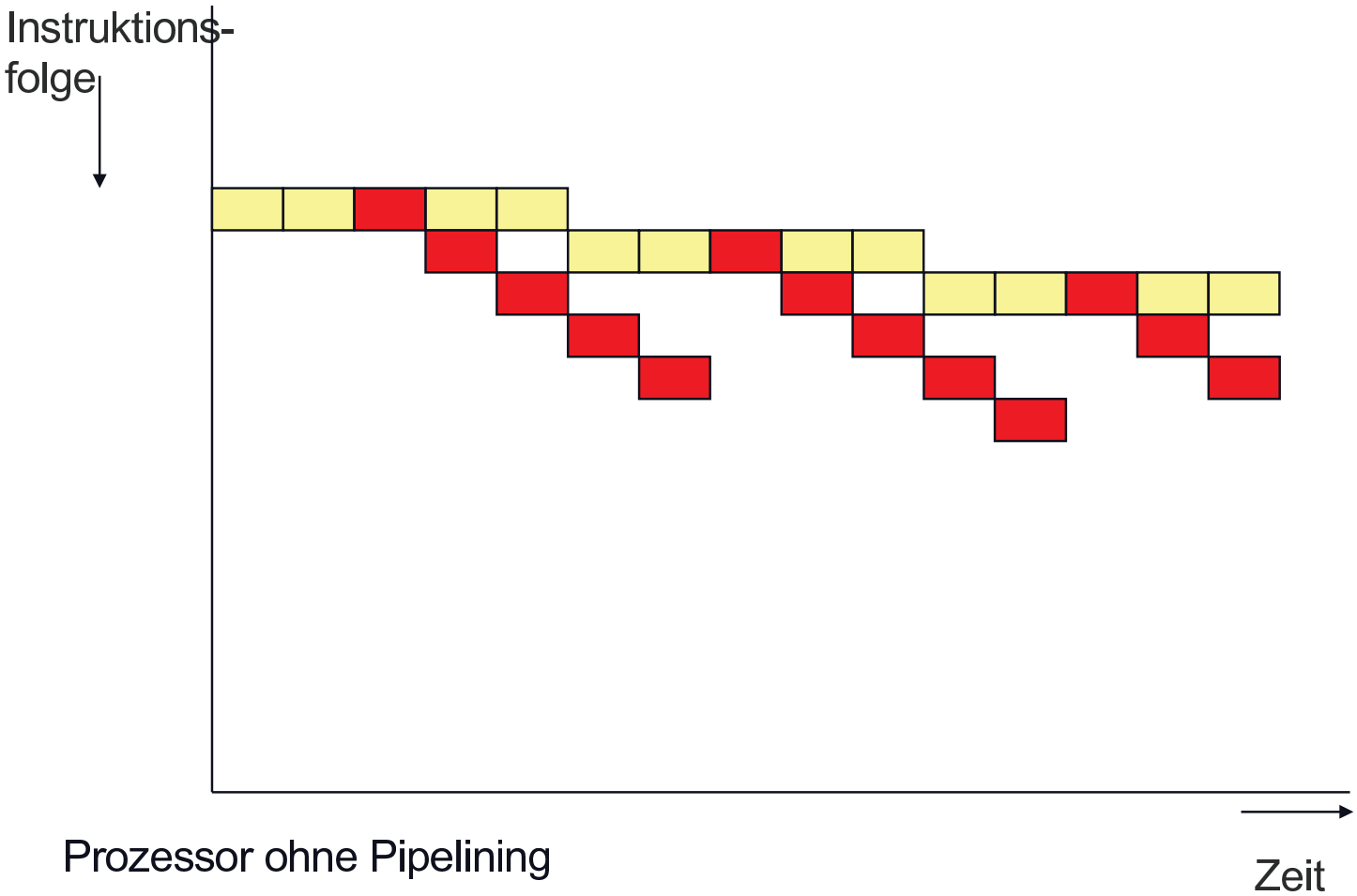
Parallelität: Superskalarität

Quelle: R. Yahyapour, ehem. TU Dortmund



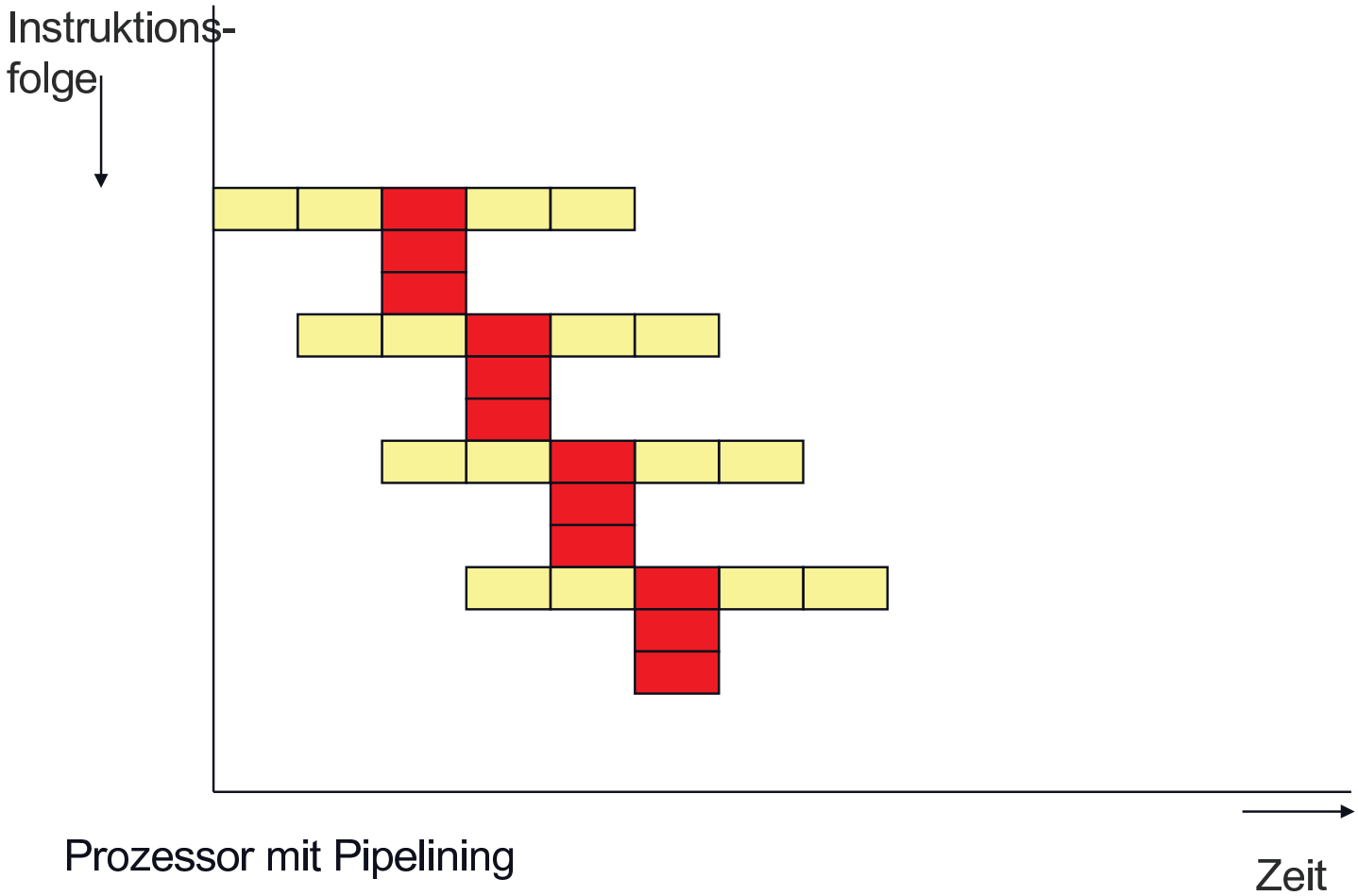
Parallelität: Vektorprozessoren

Quelle: R. Yahyapour, ehem. TU Dortmund



Parallelität: Very Long Instruction Word (VLIW)

Quelle: R. Yahyapour, ehem. TU Dortmund



Outline

Herausforderung der Parallelverarbeitung

Zwei wesentliche Herausforderungen ex. beim Versuch, Parallelverarbeitung zur Leistungssteigerung einzusetzen:

1. Leistungssteigerung durch zusätzliche Prozessoren
2. Kosten der notwendigen Kommunikation

Fragen:

- *Welchen Leistungsgewinn erzielt ein N-Prozessor-System gegenüber einem Uni-Prozessor?*

  ... im allgemeinen?? ... für eine best. Aufgabe?

- Wieviel Overhead entsteht durch Datenaustausch?

  ... wieso eigentlich Datenaustausch?

Gesetze von Amdahl und Gustafson machen darüber Aussagen!

Das Amdahl'sche Gesetz

Welchen Leistungsgewinn erzielt ein N-Prozessor-System gegenüber einem Uni-Prozessor?

Speedup

$$S_{\text{tot}} = \frac{\text{Laufzeit } \textit{ohne} \text{ Parallelisierung}}{\text{Laufzeit } \textit{mit} \text{ Parallelisierung}}$$

Nach Amdahl'schem Gesetz abhängig von:

- Anteil P des parallelisierbaren Codes des betreffenden Programms / der Anwendung
- Speedup S, der durch paralleles Rechnen erreicht wird (vereinfachende Annahme: Parallelisierung auf diesem Code [Anteil P] perfekt $\Rightarrow$ Speedup $S = N =$ Anz. der Prozessoren)

$$S_{\text{tot}} = \frac{1}{(1-P) + \frac{P}{S}} = \frac{1}{(1-P) + \frac{P}{N}}$$

Das Amdahl'sche Gesetz II

$$S_{\text{tot}} = \frac{1}{(1-P) + \frac{P}{S}} = \frac{1}{(1-P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
  - 80% parallelisierbarer Code $\Rightarrow$ Speedup ca. 4,7
  - 50% parallelisierbarer Code $\Rightarrow$ Speedup ca. 2
  - 20% parallelisierbarer Code $\Rightarrow$ Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \to \infty} S_{\text{tot}} = \frac{1}{(1-P)}$$

Das Amdahl'sche Gesetz II

$$S_{\text{tot}} = \frac{1}{(1-P) + \frac{P}{S}} = \frac{1}{(1-P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
  - 80% parallelisierbarer Code $\Rightarrow$ Speedup ca. 4,7
  - 50% parallelisierbarer Code $\Rightarrow$ Speedup ca. 2
  - 20% parallelisierbarer Code $\Rightarrow$ Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \to \infty} S_{\text{tot}} = \frac{1}{(1-P)}$$

Allein vom sequentiellen Programmanteil abhängig!

Das Amdahl'sche Gesetz II

$$S_{\text{tot}} = \frac{1}{(1-P) + \frac{P}{S}} = \frac{1}{(1-P) + \frac{P}{N}}$$

- Betrachten Beispiel (jeweils $N = 64$ Prozessoren):
  - 80% parallelisierbarer Code $\Rightarrow$ Speedup ca. 4,7
  - 50% parallelisierbarer Code $\Rightarrow$ Speedup ca. 2
  - 20% parallelisierbarer Code $\Rightarrow$ Speedup ca. 1,25
- Maximaler Speedup bei beliebig vielen Prozessoren:

$$\lim_{N \to \infty} S_{\text{tot}} = \frac{1}{(1-P)}$$

Allein vom sequentiellen Programmanteil abhängig!

Max. Speedup bei 80% parallel. Code: 5!
(d.h. 128 statt 64 Prozessoren kaum Gewinn [4,8 statt 4,7])

Das Gesetz von Gustafson (& Barsis)

... stellt quasi Gegenstandpunkt zum Amdahl'schen Gesetz dar:

$$S'_{\text{tot}} = N-(1-P)\cdot(N-1) = N-N+1+P\cdot(N-1) = (1-P)+P\cdot N$$

- Betrachten Beispiel (wieder jeweils $N = 64$ Prozessoren):
  - 80% parallelisierbarer Code $\Rightarrow$ Speedup ca. 51,4
  - 50% parallelisierbarer Code $\Rightarrow$ Speedup ca. 32,5
  - 20% parallelisierbarer Code $\Rightarrow$ Speedup ca. 13,6

Wieso dieser (scheinbare) Widerspruch zum Amdahl'schen Gesetz?

Das Gesetz von Gustafson (& Barsis)

... stellt quasi Gegenstandpunkt zum Amdahl'schen Gesetz dar:

$$S'_{\text{tot}} = N-(1-P)\cdot(N-1) = N-N+1+P\cdot(N-1) = (1-P)+P\cdot N$$

- Betrachten Beispiel (wieder jeweils $N = 64$ Prozessoren):
  - 80% parallelisierbarer Code $\Rightarrow$ Speedup ca. 51,4
  - 50% parallelisierbarer Code $\Rightarrow$ Speedup ca. 32,5
  - 20% parallelisierbarer Code $\Rightarrow$ Speedup ca. 13,6

Wieso dieser (scheinbare) Widerspruch zum Amdahl'schen Gesetz?

Annahme: Problemgröße wächst linear mit der Anzahl d. Prozessoren!

$$S'_{\text{tot}} = \frac{(1-P)+P\cdot N}{(1-P)+P}$$

Amdahl vs. Gustafson

Eine automobilistische Metapher: (Quelle: Wikipedia)

Amdahl's Law approximately suggests:

> *"Suppose a car is traveling between two cities 60 miles apart, and has already spent one hour traveling half the distance at 30 mph. No matter how fast you drive the last half, it is impossible to achieve 90 mph average before reaching the second city. Since it has already taken you 1 hour and you only have a distance of 60 miles total; going infinitely fast you would only achieve 60 mph."*

Gustafson's Law approximately states:

> *"Suppose a car has already been traveling for some time at less than 90mph. Given enough time and distance to travel, the car's average speed can always eventually reach 90mph, no matter how long or how slowly it has already traveled. For example, if the car spent one hour at 30 mph, it could achieve this by driving at 120 mph for two additional hours, or at 150 mph for an hour, and so on."*

Leistungsbewertung

Problem:

- Ermittlung des Speedup erfordert immer die Betrachtung eines Programms/einer Anwendung
- Die theoretische "Peak Performance" ist häufig weit entfernt von der realistischen Leistung eines Systems

Lösungsansatz: Benchmarks für die Bewertung der Rechenleistung

Beispiele:

- *NAS Parallel Benchmarks* (http://www.nas.nasa.gov/)
  "... set of programs designed to help evaluate the performance of parallel supercomputers. ... derived from computational fluid dynamics (CFD) applications and consist of five kernels ..."
- LINPACK (http://www.netlib.org/benchmark/hpl/)
  Ursprünglich numerische Programmbibliothek zum Lösen von lin. Gleichungssystemen

Achtung: Einfluss von Programmierung, Compiler, ...

Übungen

Ein sequentielles Programm $\mathcal{P}$ lässt sich in 5 Bereiche A bis E unterteilen, die aufgrund von Abhängigkeiten in dieser Reihenfolge ausgeführt werden müssen. Der Anteil der Bereiche an der Laufzeit ist in Tabelle 1 angegeben. Die Bereiche A, C und E lassen sich nicht parallelisieren. Bereich B kann in maximal 4 parallel ausführbare Einheiten transformiert werden. Für die Parallelisierbarkeit von Bereich D gibt es keine Beschränkung.

a) Wie viele Prozessorkerne werden benötigt, um einen Speed-Up von mindestens 4 zu erhalten? Geben Sie an, welche Gesetzmäßigkeit Sie angewendet haben.

b) Ausgehend von der Lösung in Teil a) soll in Bereich D nun ein doppelt so großes Problem gelöst werden. Die Gesamtlaufzeit des parallelisierten Programms soll sich hierbei nicht ändern. Benutzen Sie Gustavsons Gesetz um den resultierenden Speed-Up *bezüglich der Gesamtlaufzeit des parallelisierten Programmes aus Teil a)* zu berechnen.

Bereich	A	B	C	D	E
Laufzeitanteil	2%	20%	5%	70%	3%

Tabelle: Anteile der Bereiche an der Programmlaufzeit

Outline

Zusammenfassen der Performanz

- Am einfachstem ist es mit nur einer einzigen Zahl, die man mit anderen Rechnern vergleichen kann .
- Typischerweise führt man einige Tests (mit verschiedenen Konfigurationen oder Anwendungen) durch, und benutzt den Durchschnitt als Ergebnis.

Zusammenfassen der Performanz

- Am einfachstem ist es mit nur einer einzigen Zahl, die man mit anderen Rechnern vergleichen kann .
- Typischerweise führt man einige Tests (mit verschiedenen Konfigurationen oder Anwendungen) durch, und benutzt den Durchschnitt als Ergebnis.
- arithmetischer Durchschnitt:

$$ArithemeticMean(a_1, a_2, \ldots, a_N) = \frac{\sum_{i=1}^{N} a_i}{N}$$

- geometrischer Durchschnitt:

$$GeometricMean(a_1, a_2, \ldots, a_N) = \sqrt[N]{\prod_{i=1}^{N} a_i}$$

- harmonischer Durchschnitt:

$$HarmonicMean(a_1, a_2, \ldots, a_N) = \frac{N}{\sum_{i=1}^{N} \frac{1}{a_i}}$$

Geometrischer Durchschnitt im Detail

Geometrischer Durchschnitt ist vorteilhaft um die normalisierten Ergebnisse zu vergleichen.

$$\frac{GeometricMean(a_1, a_2, \ldots, a_N)}{GeometricMean(b_1, b_2, \ldots, b_N)} = \frac{\sqrt[N]{\prod_{i=1}^{N} a_i}}{\sqrt[N]{\prod_{i=1}^{N} b_i}}$$

$$= \sqrt[N]{\prod_{i=1}^{N} \frac{a_i}{b_i}}$$

a_i und b_i sind beide Anteile von einem Referenzdesign.

- *The geometric mean of the ratios is the same as the ratio of the geometric mean.*
- D.h., das Referenzdesign ist irrelevant.

arithmetischer/harmonischer Durchschnitt: Aufpassen!

Wir führen zwei Tests auf einem Rechner durch und bekommen die folgenden Ergebnisse:

- Test 1: 3 Sekunden (Die meisten Rechner brauchen 12 Sekunden)
- Test 2: 300 Sekunden (Die meisten Rechner brauchen 600 Sekunden)

Wie schnell ist die Maschine?

- $3 = (\frac{12}{3} + \frac{600}{300})/2$? arithmetischer Durchschnitt der normalisierten Anteile
- $2.67 = \frac{2}{\frac{3}{12}+\frac{300}{600}}$? harmonischer Durchschnitt der normalisierten Anteile
- $612/303 = \frac{600+12}{300+3}$?

Sematics of Means

- Es gibt viele Möglichkeiten, die Performanz anzugeben. Sie basiert auf den Daten (Zeiten) der Testreihen und der Referenz.
- Mann kann den Durchschnitt verschiedener Zeiten angeben
  - die gemessenen Zeiten,
  - die Rate (die Inverse von Zeiten),
  - das Verhältnis der Zeiten (gemessene Zeit durch Referenzzeit), oder
  - das Verhältnis der Rate (Referenzzeit durch gemessene Zeit)

Jede Möglichkeit hat eine eigene Bedeutung und beanwortet nur eine einzige Frage.

Schwierigkeiten mit Normalisierung

Performanz der Rechner wird oft durch Anteile oder Maße der Zeiten angegeben.

- Nach der Normalisierung bekommen wir einen Wert ohne Einheit.
- Was heißt es, den Durchschnitt einer Reihe Werte zu berechnen?
  - Gewichteter Durchschnitt (die Gewichtung entspricht der Laufzeit auf der Referenzmaschiene)
- Was drückt der arithmetische Durchschnitt der normalisierten Werte aus?
  - Angenommen die Referenzzeit ist die erwartete Zeit für jeden Benchmark. Dann repräsentiert der arithmetische Durchschnitt den Anteil der erwarteten Zeit ein Benchmark im Durchschnitt benötigt.

Schwierigkeiten mit Normalisierung

Performanz der Rechner wird oft durch Anteile oder Maße der Zeiten angegeben.

- Nach der Normalisierung bekommen wir einen Wert ohne Einheit.
- Was heißt es, den Durchschnitt einer Reihe Werte zu berechnen?
  - Gewichteter Durchschnitt (die Gewichtung entspricht der Laufzeit auf der Referenzmaschiene)
- Was drückt der arithmetische Durchschnitt der normalisierten Werte aus?
  - Angenommen die Referenzzeit ist die erwartete Zeit für jeden Benchmark. Dann repräsentiert der arithmetische Durchschnitt den Anteil der erwarteten Zeit ein Benchmark im Durchschnitt benötigt.

Untersuchen Sie nicht die durchschnittlichen Werte von normalisierten Ergebnissen wenn Sie sich nicht zu 100% sicher sind.

Beispiel mit Normalisierung (Szenario I)

Szenario I	Test T1	Test T2
Maschine A:	10 sec	100 sec
Maschine B:	1 sec	1000 sec
Referenz:	1 sec	100 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Szenario I	Test T1	Test T2	
Machine A:	0.1	1	
Machine B:	1	0.1	

Beispiel mit Normalisierung (Szenario I)

Szenario I	Test T1	Test T2
Maschine A:	10 sec	100 sec
Maschine B:	1 sec	1000 sec
Referenz:	1 sec	100 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Szenario I	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	1	2/(10+1)
Machine B:	1	0.1	2/(1+10)

Beispiel mit Normalisierung (Szenario I)

Szenario I	Test T1	Test T2
Maschine A:	10 sec	100 sec
Maschine B:	1 sec	1000 sec
Referenz:	1 sec	100 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Szenario I	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	1	2/(10+1)
Machine B:	1	0.1	2/(1+10)

Die beiden Maschinen schneiden gleich gut ab. Allerdings ist die Argumentation fehlerhaft.

Example with Normalization (Scenario II)

Scenario II	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	10 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario II	Test T1	Test T2	
Machine A:	0.1	0.1	
Machine B:	1	0.01	

Example with Normalization (Scenario II)

Scenario II	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	10 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario II	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	0.1	$2/(10+10) = 1/10$
Machine B:	1	0.01	$2/(1+100) = 2/101$

Example with Normalization (Scenario II)

Scenario II	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	10 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario II	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	0.1	$2/(10+10) = 1/10$
Machine B:	1	0.01	$2/(1+100) = 2/101$

In dieser Untersuchung schneidet Maschine A 5 mal besser ab als Maschine B. Allerdings ist die Argumentation fehlerhaft.

Example with Normalization (Scenario III)

Scenario III	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	1000 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario III	Test T1	Test T2	
Machine A:	0.1	10	
Machine B:	1	1	

Example with Normalization (Scenario III)

Scenario III	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	1000 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario III	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	10	$2/(10+1) = 2/11$
Machine B:	1	1	$2/(1+1) = 1$

Example with Normalization (Scenario III)

Scenario III	Test T1	Test T2
Machine A:	10sec	100sec
Machine B:	1 sec	1000 sec
reference:	1 sec	1000 sec

Performanz im Verhältnis zur Referenz (im Bezug auf die Zeit)

Scenario III	Test T1	Test T2	harmonischer Durchschnitt
Machine A:	0.1	10	$2/(10+1) = 2/11$
Machine B:	1	1	$2/(1+1) = 1$

In dieser Untersuchung schneidet Maschine A schlechter ab als Maschine B. Allerdings ist die Argumentation fehlerhaft.

Diese Maschine ist X mal so schnell wie jene Maschine

Wie können wir diesen Faktor X messen?

- Zunächst finden wir heraus, welche Programme auf Maschine A ausgeführt werden sollen
- Dann werden diese (oder ähnliche) Programme als Benchmark-Suite verwendet und auf Maschine B ausgeführt
- Danach bestimmen wir, wie oft wir diese Programme verwenden werden und bestimmen daraus den Gewichtungsfaktor, z.B. Prozentsatz der insgesamt ausgeführten Programme, Frequenz der Ausführung etc.
- gewichter Durchschnitt anhand der Gewichtungsfaktoren ...

Ist es immer sinnvoll?

Diese Maschine ist X mal so schnell wie jene Maschine

Wie können wir diesen Faktor X messen?

- Zunächst finden wir heraus, welche Programme auf Maschine A ausgeführt werden sollen
- Dann werden diese (oder ähnliche) Programme als Benchmark-Suite verwendet und auf Maschine B ausgeführt
- Danach bestimmen wir, wie oft wir diese Programme verwenden werden und bestimmen daraus den Gewichtungsfaktor, z.B. Prozentsatz der insgesamt ausgeführten Programme, Frequenz der Ausführung etc.
- gewichter Durchschnitt anhand der Gewichtungsfaktoren ...

Ist es immer sinnvoll? Nicht immer. Wir vergessen oft den Unterschied zwischen den folgenden Aussagen:

- die im Durchschnitt gesparte Zeit durch die Verwendung von Maschine A statt von Maschine B ist ...
- im Durchschnitt ist die relative Performanz von Maschine A um Vergleich zu Maschine B ...

Irreführende Aussagen

The rated mean time to failure of disks is 1,200,000 hours or almost 140 years, so disks practically never fail.
Stimmt die Aussage? Wenn nicht, was sind die Gründe dafür?

Aufgabe zu diskutieren.

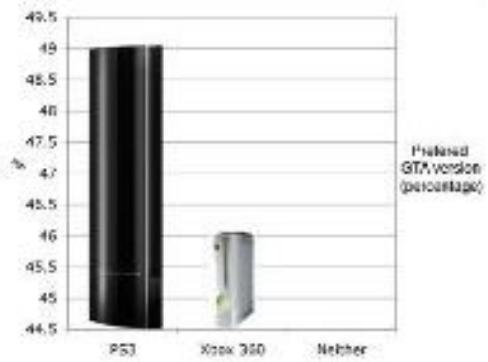
Misleading Graphs (I)

Statistics can be misleading. They are often used to prove a point, and can easily be twisted in favour of that point! Misleading graphs can appear. The following things are important to consider when looking at a graph:

1. Labels on both axes of a line or bar chart and on all sections of a pie chart
2. Source of the data
3. Scale: Does it start with zero? If not, is there a break shown
4. Scale: Are the numbers equally spaced?

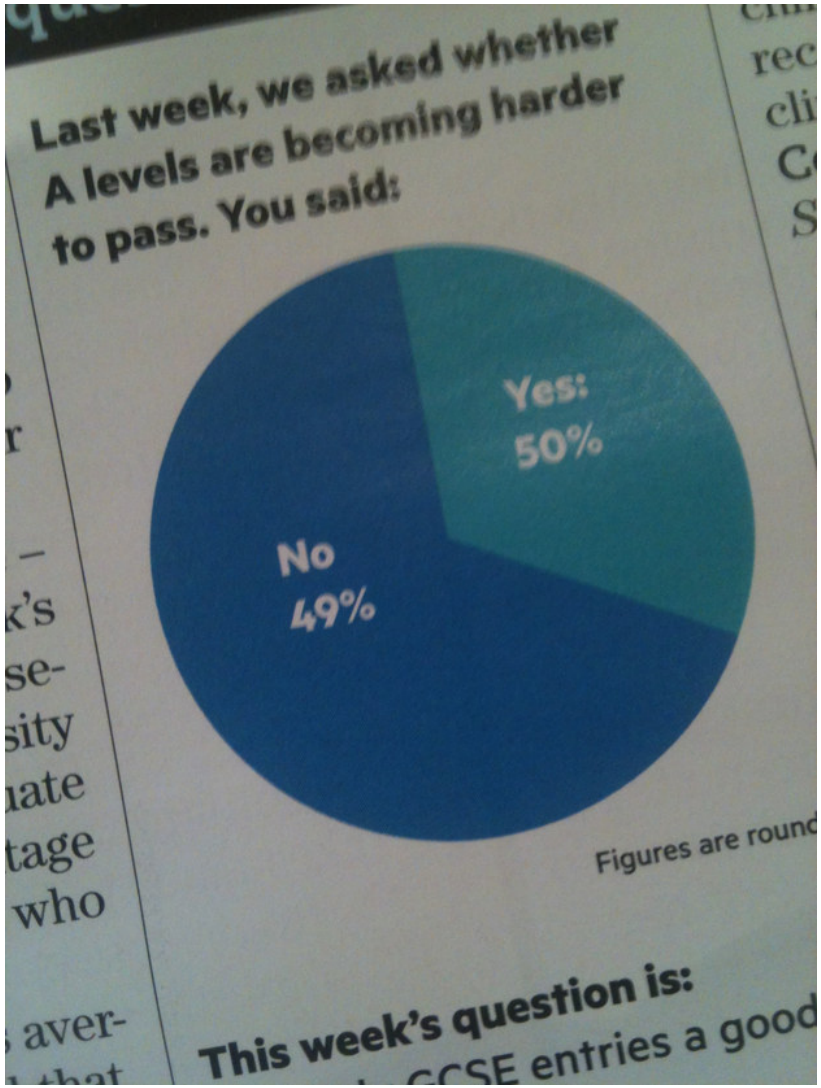
For misleading graphs, you can ask Google to get quite a lot of them. Here is a more serious reference:
http://faculty.atu.edu/mfinan/2043/section31.pdf

Out of the Topic



right: http://imgur.com/y9SzZaG

left: http://n4g.com/news/113664/gta-iv-ps3-to-outsell-xbox-360-version

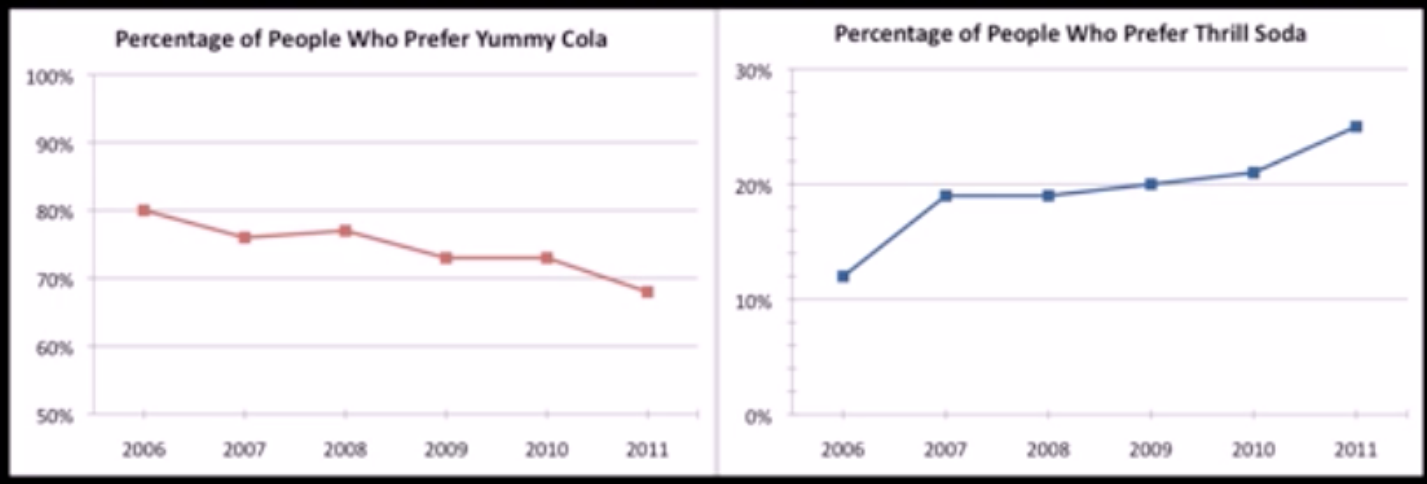


Misleading Graphs (II)

Thrill Soda hired a marketing company to help them promote their brand against Yummy Cola. The company gathered the following data about consumers' preference of soda:

Year	% of respondents who prefer Yummy Cola	% of respondents who prefer Thrill Soda	% of respondents who have no preference
2006	80%	12%	8%
2007	76%	19%	5%
2008	77%	19%	4%
2009	73%	20%	7%
2010	73%	21%	6%
2011	68%	25%	7%

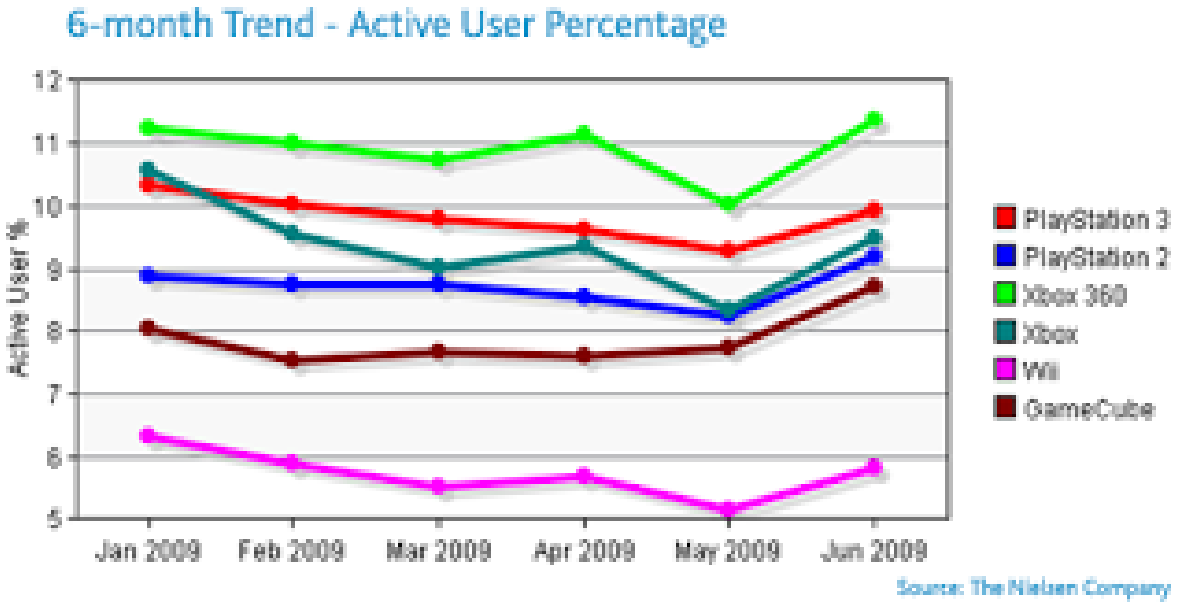
The advertising company created the following two graphs to promote Thrill Soda:



http://www.khanacademy.org

Misleading Graphs (III)

This graph from 2009 showed how game consoles were used:

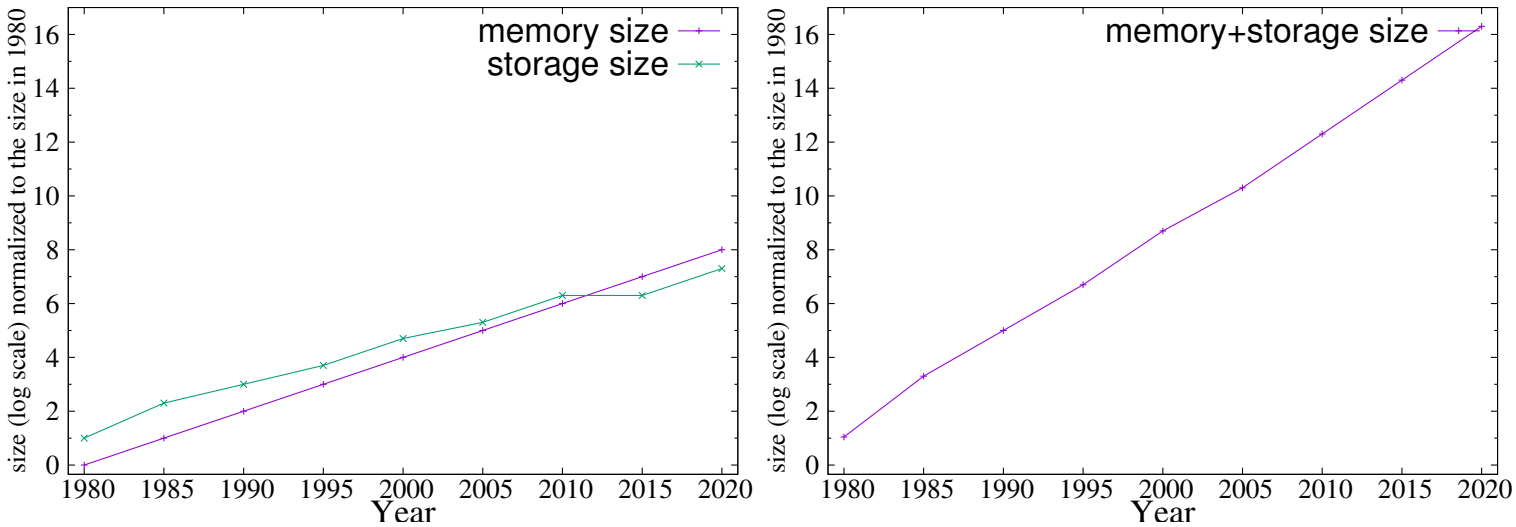


Lee Evans, in his article "Adventures in Misleading Graphs" put it best: Take a look at the top number on the graph. Only 11% of 360 owners actively use their 360 and only 10% of PS3 owners actively use their PS3.

- There are 50 million Wii owners. 6% of that number is 3 million.
- There are 30 million 360 owners. 11% of that number is 3.3 million.
- There are 20 million PS3 owners. 10% of that number is 2 million.

Incorrect Logscale Operations

Suppose that we have the following log-scale figures.



pay attention to the units/scales and operations!