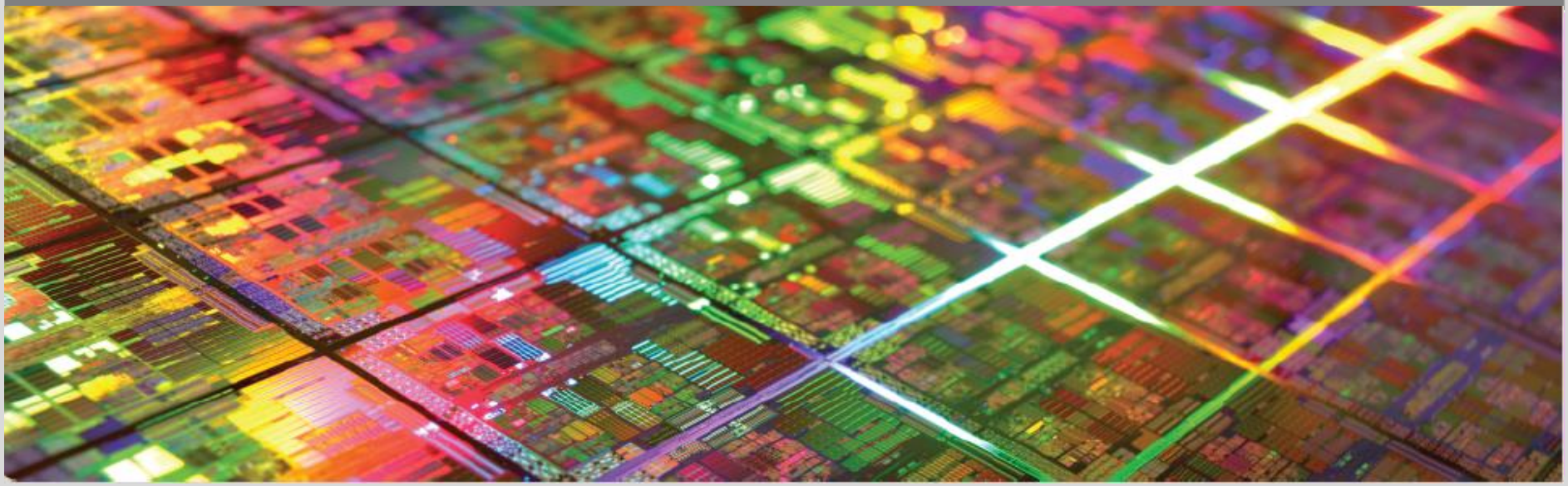
Karlsruher Institut für Technologie

Rechnerstrukturen

Vorlesung im Sommersemester 2013

Prof. Dr. Wolfgang Karl

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Einführung in den Entwurf eingebetteter Systeme
- 1.4 Energieeffizienter Entwurf – Grundlagen
- 1.5 Bewertung der Leistungsfähigkeit eines Rechners

Bewertung der Leistungsfähigkeit

Ziele

- Auswahl der Rechenanlage
- Veränderung der Konfiguration einer bestehenden Anlage
- Entwurf von Anlagen

Bewertung der Leistungsfähigkeit

Was heißt es: Ein Rechner ist schneller als ein anderer Rechner?

- **Der Benutzer eines Arbeitsplatzrechners:**
  - „Ein Rechner A ist schneller als ein Rechner B, wenn ein Programm auf A weniger Zeit benötigt.“
  - Reduzierung der **Antwortzeit (response time)** oder **Ausführungszeit (execution time), Latenz**
    - Zeit zwischen dem Beginn und dem Ende eines Ereignisses, einer Aufgabe
  - A ist n-mal schneller als B:

$$\frac{\text{Ausführungszeit(B)}}{\text{Ausführungszeit(A)}} = n$$

Bewertung der Leistungsfähigkeit

Was heißt es: Ein Rechner ist schneller als ein anderer Rechner?

- **Der Rechenzentrumsleiter:**
  - „Ein Rechner A ist schneller als ein Rechner B, wenn A in einer Stunde mehr Aufträge (Jobs) erledigt.“
  - Erhöhung des **Durchsatzes (throughput)**
    - Anzahl der ausgeführten Aufgaben in einem gegebenen Zeitintervall
    - Durchsatz von A ist m-mal höher als der von B:
      - Die Anzahl der erledigten Aufgaben auf A ist m-mal die Anzahl der erledigten Aufgaben auf B.

Bewertung der Leistungsfähigkeit

Definitionen:

- **Ausführungszeit (execution time)**

- **Wall-clock time, response time, elapsed time**
  - Latenzzeit für die Ausführung einer Aufgabe
  - Schließt den Speicher- und Plattenzugriff, Ein-/ Ausgabe etc. mit ein.

- **CPU Time**
  - Zeit, in der die CPU arbeitet
  - User CPU Time: Zeit, in der die CPU ein Programm ausführt
  - System CPU Time: Zeit, in der die CPU Betriebssystemaufgaben ausführt, die von einem Programm angefordert werden

  - Beispiel: UNIX time Kommando:
    - 90.7u, 12.9s, 2:39 65%
    - % CPU time an der Elapsed time: (90.7s + 12.9s) / 159s = 0.65

Bewertung der Leistungsfähigkeit

Verfahren

- Auswertung von Hardwaremaßen und Parametern
- Laufzeitmessungen bestehender Programme
- Messungen während des Betriebs von Anlagen
- Modelltheoretische Verfahren

Bewertung der Leistungsfähigkeit

Einfache Hardwaremaße

- **Gleichung für die Leistung der CPU**
  - Der Rechner läuft mit fester Taktrate, angegeben durch
    - Dauer eines Taktzyklus (z. B. 1ns)
    - Taktfrequenz (z. B. 1 GHz)
  - Die CPU-Zeit einer Programmausführung kann dargestellt werden mit

$$\text{CPU-Zeit} = \text{Anzahl Taktzyklen für das Programm} * \text{Taktzyklusdauer} = \frac{\text{Anzahl Taktzyklen für das Programm}}{\text{Taktfrequenz}}$$

Bewertung der Leistungsfähigkeit

Einfache Hardwaremaße

- **Gleichung für die Leistung der CPU**
  - Einführung der Maßzahl CPI (clock cycles per instruction): Mittlere Anzahl der Taktzyklen pro Befehl
    - mit IC (instruction count), der Anzahl der ausgeführten Befehle eines Programms

$$\text{CPI} = \text{Anzahl Taktzyklen für das Programm} / \text{IC}$$

  - Damit:

$$\begin{aligned}\text{CPU-Zeit} &= \text{IC} \times \text{CPI} \times \text{Taktdauer} \\ &= \text{IC} \times \text{CPI} \ / \ \text{Taktfrequenz}\end{aligned}$$

Bewertung der Leistungsfähigkeit

Einfache Hardwaremaße

- **Maßzahlen für die Operationsgeschwindigkeit**
  - **MIPS** (Millions of Instructions Per Second):

$$\text{MIPS} = \frac{\text{Anzahl der ausgeführten Instruktionen}}{10^6 \times \text{Ausführungszeit}}$$

  - **MFLOPS** (Millions of Floatingpointoperations Per Second):

$$\text{MFLOPS} = \frac{\text{Anzahl der ausgeführten Gleitkommainstruktionen}}{10^6 \times \text{Ausführungszeit}}$$

Bewertung der Leistungsfähigkeit

Einfache Hardwaremaße

- **Maßzahlen für die Operationsgeschwindigkeit**
- **Probleme**
  - Abhängigkeit von ISA und ausgeführter Befehlssequenz
    - Vergleich von Rechnern mit unterschiedlicher ISA
  - Unterschiedliche MIPS/MFLOPS-Zahlen bei verschiedenen Programmen
  - MIPS kann umgekehrt zur tatsächlichen Rechenleistung variieren
    - Beispiel: Gleitkommarechnung in Hardware bzw. mit Software-Routinen
  - MIPS/MFLOPS Angaben von Herstellern oft nur best-case-Annahme: theoretische Maximalleistung

Bewertung der Leistungsfähigkeit

Einfache Hardwaremaße

- **Zusammenfassung**
  - Vergleich von Rechnern bezüglich ihrer Leistung ohne großen Aufwand
  - Maßzahlen bewerten nur spezielle Aspekte
  - Kritische Betrachtung der Leistungsangabe unbedingt notwendig!
  - Angabe einer hypothetische Maximalleistung!

Laufzeitmessung bestehender Programme

Benchmarks

- Bewertung der Leistungsfähigkeit aufgrund von Messungen mit Hilfe von einem Programm oder einer Programmsammlung
  - Programme liegen im Quellcode vor
    - Übersetzung notwendig
  - Messung der Ausführungszeiten
    - In die Bewertung fließt „Güte“ des Compiler und Betriebssoftware ein
  - Zugriff auf die Maschinen notwendig

Laufzeitmessung bestehender Programme

Benchmarks

- **Kernels**
  - Rechenintensive Teile realer Programme
    - Vorwiegend numerische Algorithmen
  - Beispiele:
    - Lawrence Livermore Loops:
      - Zur Bewertung vektorisierender Compiler
    - BLAS (Basic Linear Algebra Subprograms)
      - Wenig Aufwand, aber nur bedingt aussagekräftig
    - LINPACK Softwarepaket:
      - Lösung eines Systems linearer Gleichungen

Laufzeitmessung bestehender Programme

Benchmarks

- **Kernels**
  - LINPACK Softwarepaket:
    - TOP500 Liste (http://www.top500.org)

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- Ziel: Vergleichbarkeit von Rechnern (inkl. Betriebssystem und Compiler)

- Anforderungen:
  - Gute Portierbarkeit
  - Repräsentativ für typische Nutzung der Rechner

- Sammlung von Benchmark-Programmen (Benchmark Suites)
  - Ausgeglichene Bewertung durch die unterschiedlichen Eigenschaften der Programme

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **Standardisierungsorganisationen**
  - TPC (Transaction Processing Performance Council)
    - Mitte der 80‘er Jahre, http://www.tpc.org
    - Zusammenschluss von Datenbank- und Rechnerherstellern
    - Ziel: Bewertung von Datenbanksystemen

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **Standardisierungsorganisationen**
  - SPEC (Standard Performance Evaluation Corporation)
  - Gegründet 1988, http://www.spec.org
    - Zusammenschluss von mehr als 40 Firmen (Rechnerhersteller)
    - Festlegung von Richtlinien für eine gemeinsame Rechnerbewertung
  - SPEC's Structure
    - *“SPEC is a non-profit corporation whose membership is open to any company or organization that is willing to support our goals (and pay our nominal dues). Originally just a bunch people from workstation vendors devising CPU metrics, SPEC has evolved into an umbrella organization encompassing three diverse groups.”*

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- Struktur:
  - Open Systems Group (OSG)
    - CPU: SPECmarks CPU Benchmarks
    - Java: Client- und Serverseitige Benchmarks: JVM98, JVM2008, JBB2000, JBB2005, jAppServer Java Enterprise Application Server benchmarks
    - Mail: SPECmail2001, Consumer Internet Service Provider (ISP) mail server benchmark
    - Power: SPECpower_ssj2008, der SPEC benchmark zur Evaluirung der Energieeffizienz für Server
    - SIP: SPEC Benchmark zum Vergleich von Servern, die das Session Initiation Protokoll (SIP) verwenden
    - SFS: SFS93 (LADDIS), SFS97, SFS97_R1, and SFS2008
    - Virtualization: Entwicklung der ersten Generation eines SPEC Benchmarks zum Vergleich der Virtusalisierungsleistung für Data Centers
    - WEB: WEB96, WEB99, WEB99_SSL, and WEB2005, die web server benchmarks.

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- Struktur:
  - The High-Performance Group (HPG)
    - Benchmark Suite, die HPC Anwendungen repräsentieren
    - Zielarchitekturen: symmetrische Multiprozessorsysteme, Workstation Cluster, Parallelrechner mit verteiltem Speicher, Vektorrechner
  - The Graphics and Workstation Performance Group (GWPG):
    - graphics and workstation performance benchmarks and reporting procedures
    - SPECapc (Application Performance Characterization): CAD/CAM, Digital Content Creation, Visualisierungsanwendungen
    - SPECgpc (Graphics Performance Characterization group): Benchmarks zur Leistungsbewertung von Graphicsystemen, die unter OpenGL oder anderen APIs laufen, SPECviewperf(r)

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
  - SPEC CPU Benchmarks
    - Strenge, genau festgelegte Regeln
    - Ab CPU95: vollautomatische Messung und Protokollierung
    - Regelmäßige Aktualisierungen (CPU92, CPU95, CPU2000)
      - Laufzeiten werden zu kurz
      - Caches werden größer: größere Datensätze
      - Mehr Praxisnähe: Programme mit schlechterer Datenlokalität
  - SPEC CPU 2006
    - 12 nichtnumerische Programme in C/C++ (CINT2006): http://www.spec.org/cpu2006/CINT2006/
    - 14 numerische Programme in FORTRAN/C (CFP2006): http://www.spec.org/cpu2006/CFP2006/
    - Referenzmaschine:
      - Historisches Sun System, eine "Ultra Enterprise 2 mit einem 296 MHz UltraSPARC II Prozessor

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- SPEC CPU 2000

	Geschwindigkeit	**Durchsatz**
Aggressive Optimierung	SPECint2000	SPECint_rate2000
	SPECfp2000	SPECfp_rate2000
Konservative Optimierung	SPECint_base2000	SPECint_rate_base2000
	SPECfp_base2000	SPECfp_rate_base2000

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- SPEC CPU 2000 Benchmark-Metrik Geschwindigkeit

$$\text{SPECratio} = \frac{\text{Referenzzeit}_x}{\text{Laufzeit}_x \text{ auf Testsystem}} \quad \text{Benchmark x}$$

- Endwerte: je ein geometrisches Mittel der SPECratio‘s über alle CINT2006 und CFP2006 Benchmarks
  - SPECint2006, SPECfp2006
    - Aggressive, individuelle Optimierungen erlaubt
  - SPECint_base2006, SPECfp_base2006
    - Nur mit konservativer Standardoptimierung
    - Identische Compileroptionen für alle Programme

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- SPEC CPU 2000 Benchmark-Metrik Geschwindigkeit
  - Warum Geometrisches Mittel

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Execution time ratio$_i$:
Ausführungszeit des Programms *i* einer Last von *n* Programmen, normalisiert bezüglich der Referenzmaschine

  - Eigenschaft des geometrischen Mittels:

$$\frac{\text{geometrisches Mittel } (X_i)}{\text{geometrisches Mittel } (Y_i)} = \text{geometrisches Mittel}\left(\frac{X_i}{Y_i}\right)$$

Geometrisches Mittel ist konsistent, unabhängig von Referenzmaschine!

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- SPEC CPU 2000 Benchmark-Metrik Geschwindigkeit
  - Warum Geometrisches Mittel

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Execution time ratio$_i$:
Ausführungszeit des Programms *i* einer Last von *n* Programmen, normalisiert bezüglich der Referenzmaschine

  - Eigenschaft des geometrischen Mittels:

$$\frac{\text{geometrisches Mittel } (X_i)}{\text{geometrisches Mittel } (Y_i)} = \text{geometrisches Mittel}\left(\frac{X_i}{Y_i}\right)$$

Geometrisches Mittel ist konsistent, unabhängig von Referenzmaschine!

Laufzeitmessung bestehender Programme

Standardisierte Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**
- SPEC CPU 2000 Benchmark-Metrik Durchsatz

$$\text{SPECrate}_x = n_x \times \frac{\text{Sekunden pro Stunde}}{\text{Laufzeit}_x \text{ von } n_x \text{ Kopien auf Testsystem}} \times \frac{\text{Referenzzeit}_x}{\text{längste Laufzeit}}$$

- Endwerte: je ein geometrisches Mittel der SPECrate‘s über alle CINT2000 bzw. CFP2000 Benchmarks
  - SPECint_rate2000, SPECfp_rate2000
  - SPECint_base2000, SPECfp_base2000
    - n_x kann frei gewählt werden, muss aber dokumentiert werden

Benchmarks

SPEC (Standard Performance Evaluation Corporation)

Processor	Alpha 21364	AMD Athlon XP	HP PA-8700	IBM Power 4+	Intel Itanium 2	Intel XeonMP	Intel Xeon	MIPS R14000	Sun UltraSPARC III
System or Motherboard	Alpha GS1280/7	ASUS A7N8X	HP9000 C3750	pSeries 650 6M2	HP RX2600	Dell PwrEdg 6650	Dell Prec. 350	SGI 3200	Sun Blade 2050
Clock Rate	1.15GHz	2.17GHz	870MHz	1.45GHz	1.0GHz	2.0GHz	3.06GHz	600MHz	1.05GHz
External Cache	None	None	None	16MB	None	None	None	8MB	8MB
164.gzip	583	1,026	588	673	583	758	1,138	322	433
175.vpr	822	653	688	902	704	625	606	572	460
176.gcc	859	755	906	914	1,014	1,100	1,236	445	577
181.mcf	712	420	494	1,391	834	599	773	783	659
186.crafty	982	1,292	751	884	781	712	1,179	502	558
197.parser	514	905	495	381	660	778	1,025	409	488
252.eon	958	1,483	592	1,150	1,004	920	1,387	507	527
253.perlbmk	768	1,306	619	712	815	952	1,381	367	540
254.gap	636	1,059	339	936	680	722	1,417	308	372
255.vortex	1,094	1,608	1,196	1,428	1,193	1,118	1,658	679	738
256.bzip2	824	840	534	965	759	712	856	493	629
300.twolf	1,018	887	911	1,198	880	1,009	900	645	570
SPECint_base2000	795	960	642	909	810	816	**1,085**	483	537
168.wupside	883	1,131	446	1,532	1,003	816	1,406	434	659
171.swim	3,590	1,006	931	1,417	3,205	848	1,837	529	980
172.mgrid	708	799	621	850	1,720	449	1,047	379	487
173.applu	1,518	654	702	979	2,033	496	1,168	381	310
177.mesa	928	1,103	694	737	642	814	1,165	425	543
178.galgel	2,105	738	1,603	3,186	2,505	1,200	1,536	1,398	1,713
179.art	2,014	495	670	1,864	4,226	1,147	716	1,436	9,389
183.equake	519	730	413	2,098	1,871	449	1,291	347	645
187.facerec	1,105	1,008	430	1,515	1,152	762	1,315	647	958
188.ammp	735	587	553	923	788	729	644	573	509
189.lucas	1,522	853	448	1,306	1,206	682	1,522	442	371
191.fma3d	1,019	850	404	898	747	551	1,089	306	400
200.sixtrack	469	538	471	621	894	376	564	298	366
301.aspi	1,242	705	696	966	678	695	833	406	471
SPECfp_base2000	1,124	776	600	1,221	**1,356**	677	1,092	499	701

Benchmarks

SPEC (Standard Performance Evaluation Corporation)

Processor	AMD 1-core Opteron 854	Intel 1-core Xeon	AMD 2-core Opteron 8224SE	Intel 2-core Xeon 5160	AMD 4-core Opteron 8360SE	Intel 4-core Xeon X7350	Intel 4-core Core 2 Quad QX9650
Bit-width	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit
Cores/chip x Threads/core	1 x 1	1 x 2	2 x 1	2 x 1	4 x 1	4 x 1	4 x 1
Clock Rate	2.80GHz	3.80GHz	3.20GHz	3.03GHz	2.50GHz	2.93GHz	3.00GHz
Cache: L1-L2-L3 - I/D or Unified	64K/64K - 1M - N/A	12K/16K - 2M - N/A	2 x 64K/64K - 2 x 1M - NA	2 x 32K/32K - 4M - NA	4 x 64K/64K - 4 x 512K - 2M	4 x 32K/32K - 2 x 4M - NA	4 x 32K/32K - 2 x 6M - NA
Execution Rate/Core	3 instructions	3 instructions	3 instructions	1 complex + 3 simple	3 instructions	1 complex + 3 simple	1 complex + 3 simple
Pipeline Stages	12 int / 17 fp	31	12 int / 17 fp	14	12 int / 17 fp	14	14
Out of Order	72	126	72	96	72	96	96
Memory bus	6.4 GB/s	800 MHz	10.6 GB/s	1333 MHz	10.6 GB/s	1066 MT/s	1333 MHz
Package	uPGA 940	LGA-775	LGA-1207	LGA-771	LGA-1207	LGA-771	LGA-775
IC Process	90nm 9M	90nm 7M	90nm 9M	65nm 8M	65nm 8M	65nm 8M	45nm
Die Size	106mm²	109mm²	227mm²	143mm²	283mm²	2 x 143mm²	2 x 107mm²
Transistors	120M	169M	233M	291M	463M	2 x 291M	2 x 410M
List Price (intro)	$1,514	$903	$2,149	$851	N/A	$2,301	$999
Power (Max)	93W	110W	120W	80W	120W	130W	130W
Availability	3Q05	3Q05	3Q07	3Q06	1Q08	3Q07	4Q07
Scalability	2-4 chips	1-2 chips	1-4 chips	1-2 chips	2-4 chips	1–4 chips	1 chip
SPECint/fp2006 [cores]	11.2/12.1 [2]	11.4/11.7 [2]	14.1/14.2 [8]	19.7*/18.3* [4]	N/A	21.7*/18.9* [16]	22.3*/21.4* [4]
SPECint/fp2006_rate [cores]	41.4/45.6 [4]	20.9/18.8 [2]	105/96.7 [8]	60.8/45.1 [4]	163/149 [16]	184*/108* [16]	69.0*/49.9 [4]

Quelle: MDR, Microprocessor Report, January 2008

Benchmarks

- **SPEC (Standard Performance Evaluation Corporation)**

Processor	Intel Itanium 2 9050	Intel Itanium 9150M	IBM Power6	IBM Power5+	Fujitsu SPARC64 VI	Sun UltraSPARC IV+	Sun UltraSPARC T2
Bit-width	64-bit	64-bit	64-bit	64-bit	64-bit	64-bit	64-bit
Cores/chip x Threads/core	2 x 2	2 x 2	2 x 2	2 x 2	2 x 2	2 x 1	8 x 8
Clock Rate	1.60GHz	1.67GHz	4.70GHz	2.20GHz	2.40GHz	1.95GHz	1.40GHz
Cache: L1-L2-L3 I/D or Unified	2 x 16K/16K - 1M/256K - 12M(on)	2 x 16K/16K - 1M/256K - 12M(on)	2 x 64K/64K - 2 x 4M - 32M(off)	2 x 64K/32K 1.92M - 36M(off)	2 x 128K/128K - 5M - NA	2 x 64K/64K - 2M - 32M(off)	8 x 8K/16K - 4M - NA
Execution Rate/Core	6 issue	6 issue	7 issue	5 issue	4 issue	4 issue	16 issue
Pipeline Stages	8 stages	8 stages	13 stages	15 stages	15 stages	14 stages	8 int / 12 fp
Out of Order	None	None	"Limited"	200	64	None	None
Memory B/W	8.5GB/s	10.6GB/s	75GB/s	12.8GB/s	8GB/s	4.8GB/s	42.7GB/s
Package IC Process	mPGA-700 90nm 7M	mPGA-700 90nm 7M	N/A 65nm 10m	MCM-5370 pins 90nm 10m	412 I/O pins 90nm 10M	FC-LGA 1368 90nm 9M	1831 pins 65nm
Die Size	596mm²	596mm²	341mm²	245mm²	421mm²	335mm²	342mm²
Transistors	1.72 billion	1.72 billion	790 million	276 million	540 million	295 million	503 million
List Price (intro)	$3,692	$3,692	N/A	N/A	N/A	N/A	N/A
Power (Max)	104W	104W	~100W	100W	120W	90W	84W
Availability	3Q06	4Q07	2Q07	4Q05	2Q07	3Q06	3Q07
Scalability	1–64 chips	8-128 chips	2–32 chips	1–32 chips	4–64 chips	1–72 chips	1 chip
SPECint/fp2006 [cores]	14.5/17.3 [2]	N/A	17.8/18.7 [1]	10.5/12.9 [1]	9.7/11.1 [32]	N/A	N/A
SPECint/fp2006_rate [cores]	1534/1671 [128]	1832/N/A [128]	420/379 [16]	197/229 [16]	1111/1160 [128]	1120/N/A [144]	73.1/58.1

Quelle: MDR, Microprocessor Report, January 2008

Benchmarks

SPEC (Standard Performance Evaluation Corporation)

Processor	Intel 2-core Xeon X5270[1]	AMD 2-core Opteron 8224SE	Intel 4-core Xeon W5590	AMD 4-core Opteron 8393SE[2]	Intel 4-core Xeon X5570[3]	AMD 6-core Opteron 8439SE	Intel 6-core Xeon X7460[4]
Bit-Width	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit	32/64-bit
Cores/Chip x Threads/core	2 x 1	2 x 1	4 x 2	4 x 1	4 x 2	6 x 1	6 x 1
Clock Rate	3.50GHz	3.20GHz	3.33GHz	3.10GHz	2.93GHz	2.80GHz	2.67GHz
Cache: L1-L2-L3 - I/D or Unified	2 x 32K/32K - 6M - NA	2 x 64K/64K - 2 x 1M - NA	4 x 32K/32K - 4 x 256K - 8M	4 x 64K/64K - 4 x 512K - 6M	4 x 32K/32K - 4 x 256K - 8M	6 x 64K/64K - 6 x 512K - 6M	6 x 32K/32K - 3 x 3M - 16M
Execution Rate/Core	1 cmplx + 3 simple	3 instructions	1 cmplx + 3 simple	3 instructions	1 cmplx + 3 simple	3 instructions	1 cmplx + 3 simple
Pipeline Stages	14	12int / 17fp	16	12int / 17fp	16	12int / 17fp	14
Out of Order	96	72	128	72	128	72	96
Memory Bus	1333MHz	10.6GB/s	16GB/s	8GB/s	16GB/s	17GB/s	1064MHz
Package	LGA-771	LGA-1207	LGA-1366	LGA-1207	LGA-1366	LGA-1207	LGA-771
IC Process / Metal Layers	45nm / 10M	90nm / 10M	45nm / 10M	45nm / 10M	45nm / 10M	45nm / 10M	45nm / 10M
Die Size	107mm²	227mm²	263mm²	258mm²	263mm²	346mm²	503mm²
Transistors	410M	233M	731M	758M	731M	904M	1900M
List Price (Intro)	$1,172	$2,149	$1,600	$2,649	$1,386	$2,649	$2,729
Power (Max)	80W	120W	130W	137W (TDP)	95W	137W (TDP)	130W
Availability	3Q08	3Q07	3Q09	2Q09	1Q09	3Q09	4Q08
Scalability	1–2 chips	1–4 chips	1–2 chips	1–8 chips	1–2 chips[3]	1–8 chips	1–4 chips
SPECint/fp2006 [cores]	26.5/25.5 [4]	14.1/14.2 [8]	34.2[a]/40.4[a] [8]	19.7/23.6 [8]	32.1[a]/45.0 [8]	18.3/23.3 [12]	22.0/22.3 [24]
SPECint/fp2006_rate [cores]	85.3/57.7 [4]	105/96.7 [8]	255/204 [8]	232/204[2] [16]	240/197 [8]	629/473 [48]	274[4]/142 [24]
x86 Codename	Wolfdale	Santa Rosa	Gainestown	Shanghai	Gainestown	Istanbul	Dunnington
Microarchitecture	Core	K8	Nehalem	K10	Nehalem	K10	Core

Quelle: MDR, Microprocessor Report, January 2010

Benchmarks

SPEC (Standard Performance Evaluation Corporation)

Processor	Intel Itanium 2 9050	Intel Itanium 9150M[5]	IBM POWER5+	IBM POWER6+[6]	Fujitsu SPARC64 VI	Fujitsu SPARC64 VII	Sun UltraSPARC T2+[7]
Bit-Width	64-bit	64-bit	64-bit	64-bit	64-bit	64-bit	64-bit
Cores/Chip x Threads/core	2 x 2	2 x 2	2 x 2	2 x 2	2 x 2	4 x 2	8 x 8
Clock Rate	1.60GHz	1.67GHz	2.20GHz	5.00GHz	2.40GHz	2.52GHz	1.60GHz
Cache: L1-L2-L3 - I/D or Unified	2 x 16K/16K - 1M/256K - 12M(on)	2 x 16K/16K - 1M/256K - 12M(on)	2 x 64K/32K 1.92M - 36M(off)	2 x 64K/64K - 2 x 4M - 32M(off)	2 x 128K/128K - 6M - N/A	4 x 64K/64K - 6M - N/A	8 x 16K/8K - 4M - NA
Execution Rate/Core	6 issue	6 issue	5 issue	7 issue	4 issue	4 issue	16 issue
Pipeline Stages	8	8	15	13	15	15	8int / 12fp
Out of Order	None	None	200	"limited"	64	64	None
Memory Bus	8.5GB/s	10.6GB/s	12.8GB/s	75GB/s	8GB/s	8GB/s	42.7GB/s
Package	mPGA-700	mPGA-700	MCM-5370 pins	N/A	412 I/O pins	412 I/O pins	1831 pins
IC Process / Metal Layers	90nm / 7M	90nm / 7M	90nm / 10M	65nm / 10M	90nm / 10M	65nm / 11M	65nm / 8M
Die Size	596mm²	596mm²	245mm²	341mm²	421mm²	400mm²	342mm²
Transistors	1.72B	1.72B	276M	790M	540M	600M	503M
List Price (Intro)	$3,692	$3,692	N/A	N/A	N/A	N/A	N/A
Power (Max)	104W	104W	100W	>100W	120W	135W	95W[7]
Availability	3Q06	4Q07	4Q05	4Q08[6]	2Q07	3Q08	3Q09
Scalability	1–64 chips	8–128 chips	1–32 chips	2–32 chips	4–64 chips	4–64 chips	1–4 chips
SPECint/fp2006 [cores]	14.5/17.3 [2]	N/A	10.5/12.9 [1]	15.8/20.1 [1]	9.7/21.7 [32]	11.5/13.0 [1]	N/A
SPECint/fp2006_rate [cores]	1534/1671 [128]	2893/N/A [256]	197/229 [16]	1866/1822 [64]	1111/1160 [128]	2088/1861 [256]	338/254 [32]
Development Status	Inactive	active	Inactive	active	Inactive	active	active

All SPEC scores are base. [a]Score measured in single-thread mode.

NOTES:

[1]Higher-numbered X5272 at lower 3.40GHz frequency has a faster 1600MHz front-side bus and posts a slightly higher score (26.6) on the CINT2006 benchmark. Among Intel's dual-core processors, the 2.93GHz E7220 scales to 4 chips and consequently boasts the highest SPECrate INT/FP numbers (139/89.4).

[2]Posted 8-chip [32-core] SPEC intrate score of 338, but no official 8-chip SPEC fprate score.

[3]Although 2 chips is the design limit for an MP board, the X5570 is used in "virtual SMP" systems based on scalable blade architectures that, in principle, can go as high as 32K chips. Highest posted SPEC intrate/fprate score is for 32 chips [128 cores]: 3147/2553.

[4]Unisys ES7000 Model 7600R has a posted 96-core [16 chip] CINT2006rate number of 999 (but no other SPEC measures).

[5]Dual-core Itanium 2 9040 at 1.60GHz in SGI Altrix 4700 Bandwidth System has posted 512 chip [1024 core] CFP2006 INTrate/FPrate scores of 9031/10583.

[6]POWER6+ at 5.0GHz officially introduced in April 2009, but began shipping in 4Q08.

[7]Wattage at 1.40GHz, no published update with 3Q09 upgrade to 1.6GHz frequency.

Quelle: MDR, Microprocessor Report, January 2010

Literatur

- Hennessy/Patterson: A Quantative Approach: Kap. 1.5 – 1.9

Bewertung der Leistungsfähigkeit

Messung während des Betriebs von Anlagen

- **Monitore**
  - Aufzeichnungselemente, die zum Zweck der Rechnerbewertung die Verkehrsverhältnisse während des normalen Betriebs beobachten und untersuchen.
  - Hardware-Monitore
    - Unabhängige physikalische Geräte
    - Keine Beeinflussung
  - Software-Monitore
    - Einbau in das Betriebssystem
    - Beeinträchtigung der normalen Betriebsverhältnisse

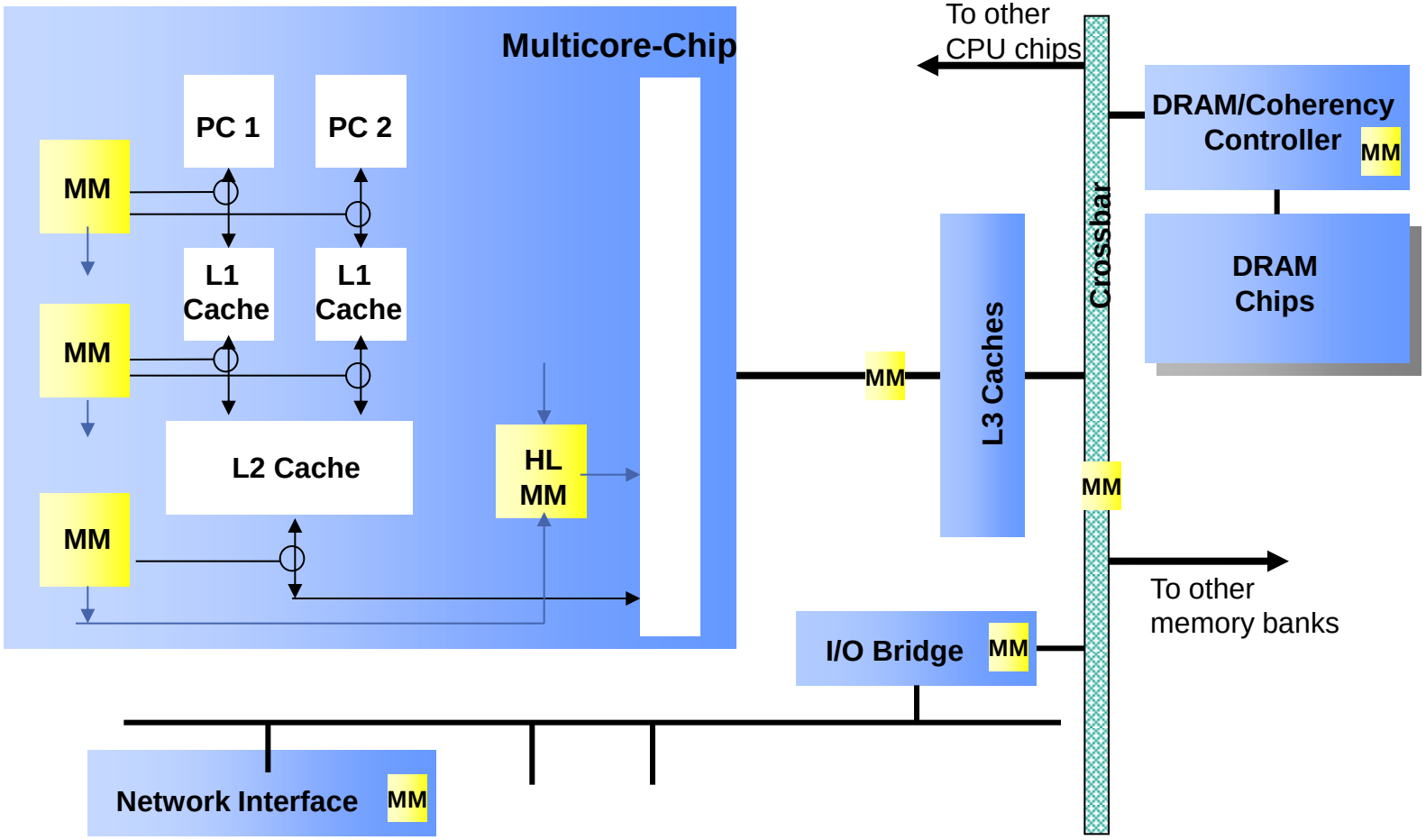
Bewertung der Leistungsfähigkeit

Messung während des Betriebs von Anlagen

- **Monitore**
  - Aufzeichnungstechniken:
    - Kontinuierlich oder sporadisch
    - Gesamtdatenaufzeichnung (Tracing)
    - Realzeitauswertung
    - Unabhängiger Auswertungslauf (Post Processing)

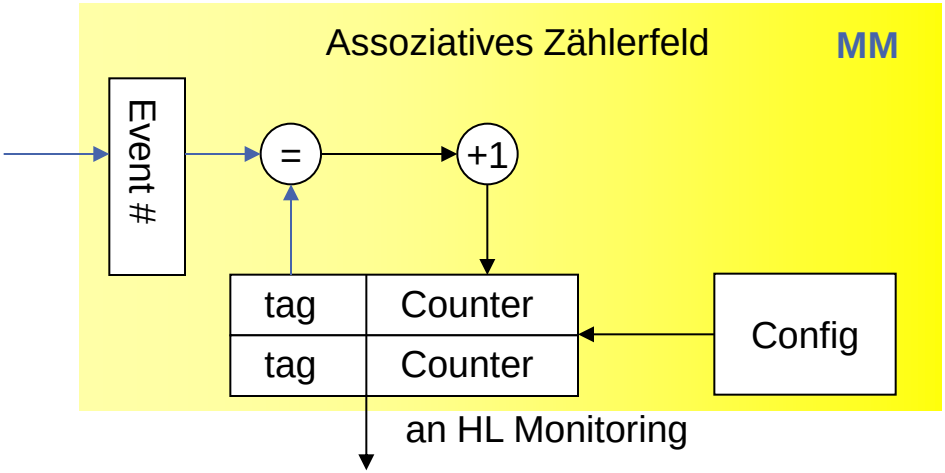
Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Knoten:



Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- HW-Ebene: Monitorkomponenten
  - Grundlegender Aufbau eines Monitor Moduls (MM)
  - Assoziatives Zählerfeld
  - Zählen von Ereignissen
  - Korrelation von Ereignissen
  - Auslagern eines Tag/Zählerfeldes an HL Monitoring bei Zählerüberlauf / Verdrängung

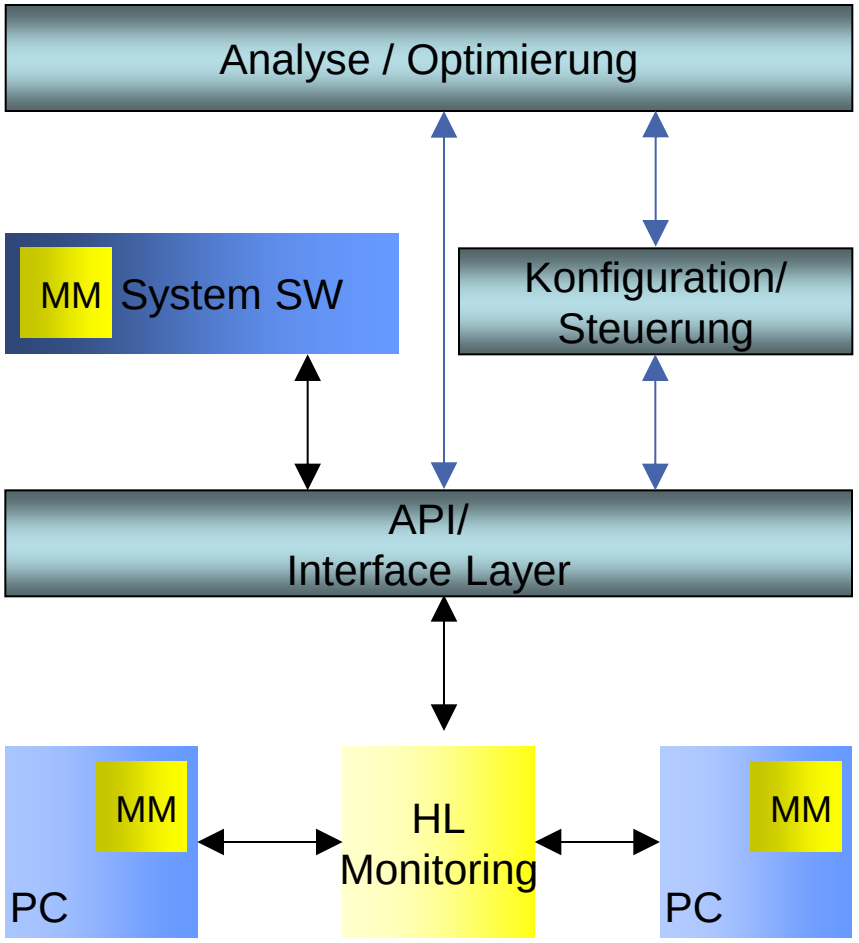


Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Monitoring-Infrastruktur
- Monitor Module (HW, SW)

- API / Interface Layer
  - Einheitliche Schnittstelle
  - Einfache Kontrolle und Steuerung
  - Verwaltung

- Kommunikationsprotokoll
  - Leichtgewichtig und erweiterbar
  - Nachrichtenorientiert
    - Ereignisse
    - Konfiguration
  - Anstoßen der Aufzeichnung



Beobachtung des Laufzeitverhaltens

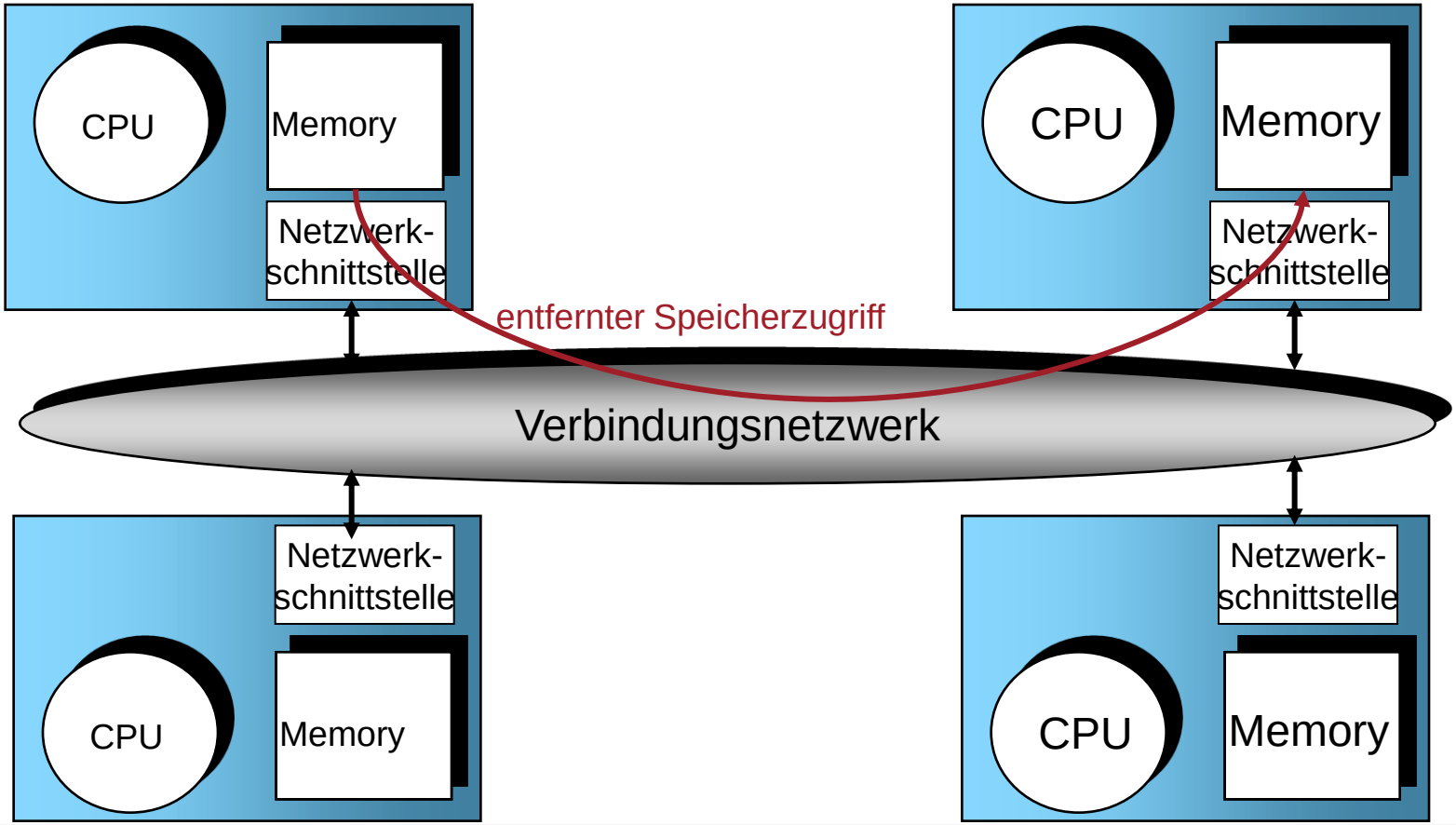
- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Koordiniertes, kooperatives und systemweites Monitoring
- Merkmale
  - Geringer HW-Aufwand
  - Geringer Kommunikationsbedarf
  - Zeitliche, inhaltliche Auflösung
  - On-line Auswertung
- Auswerten der gesammelten Informationen
  - Zusammenspiel mit Werkzeugen zur Optimierung, Lastverteilung, Energieverbrauch, …
- Herausforderungen
  - Korrelation von Ereignissen
  - Proaktivität vs. Reaktivität
  - Steuerung des adaptiven Verhaltens

Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Realisierungen in HW
  - SMiLE-Projekt
    - Ziel: Datenlokalitätsoptimierung – Aufspüren entfernter Speicherzugriffe in einem Cluster-System mit NUMA-Eigenschaften
  - OpenSPARC
  - HyperTransport
    - Integration in HTX-Board (FPGA) zur Beobachtung des Kommunikationsverhaltens
  - DodOrg (Digital On-Demand Computing Organism)
    - Ziel: Systemarchitektur mit Merkmalen der Selbstorganisation
    - Hierarchisches Monitor-Konzept

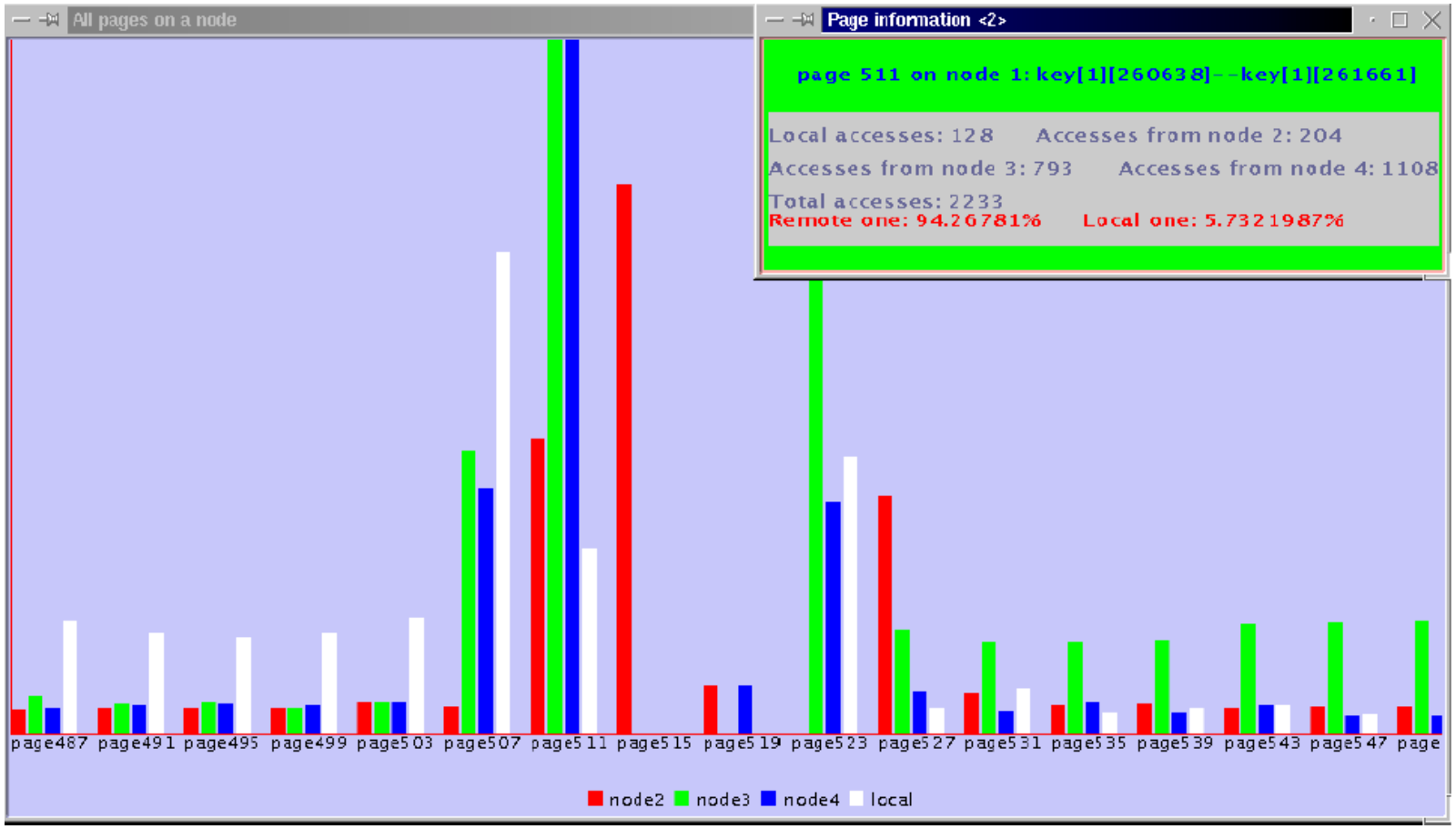
Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- NUMA-System



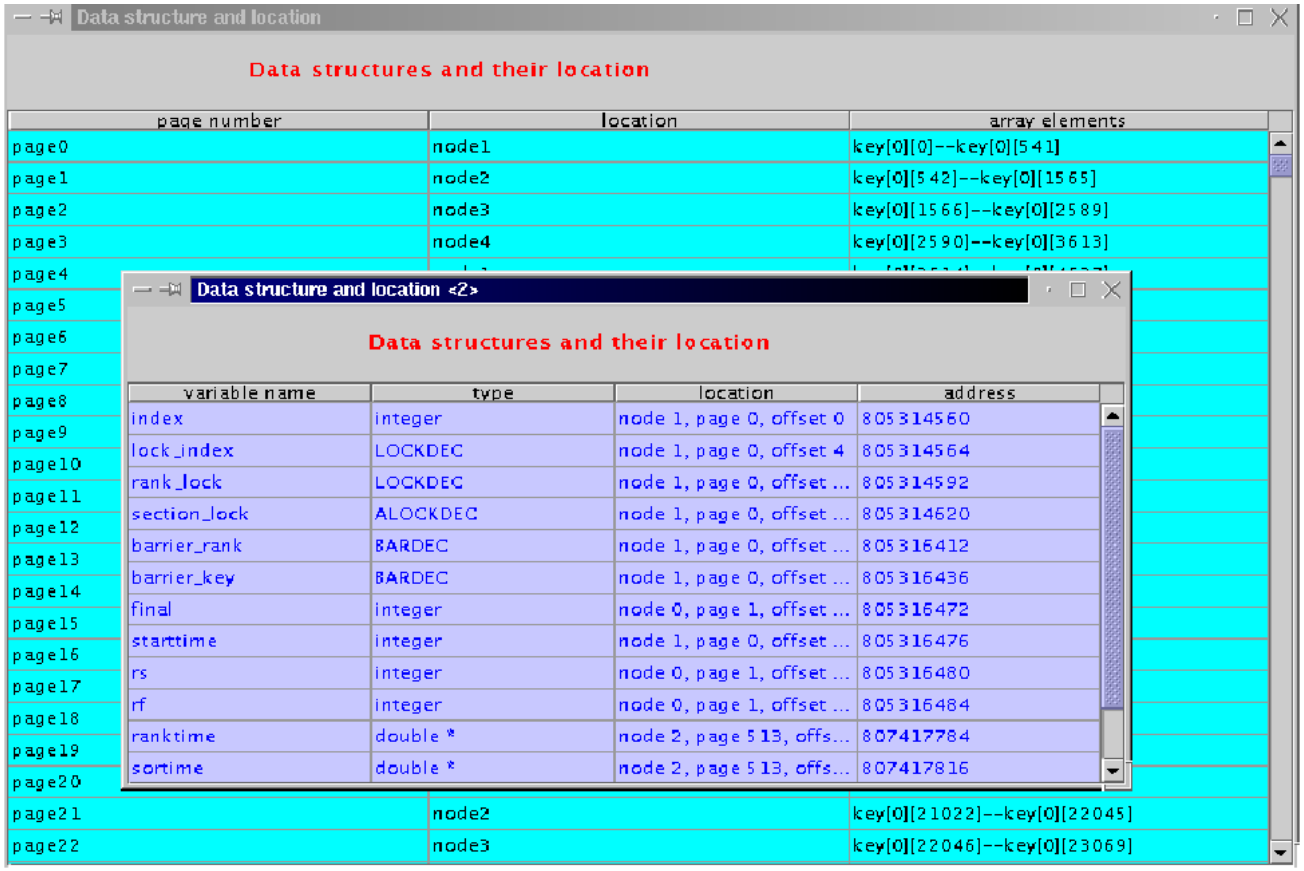
Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Visualisierung: Lokale und entfernte Speicherzugriffe



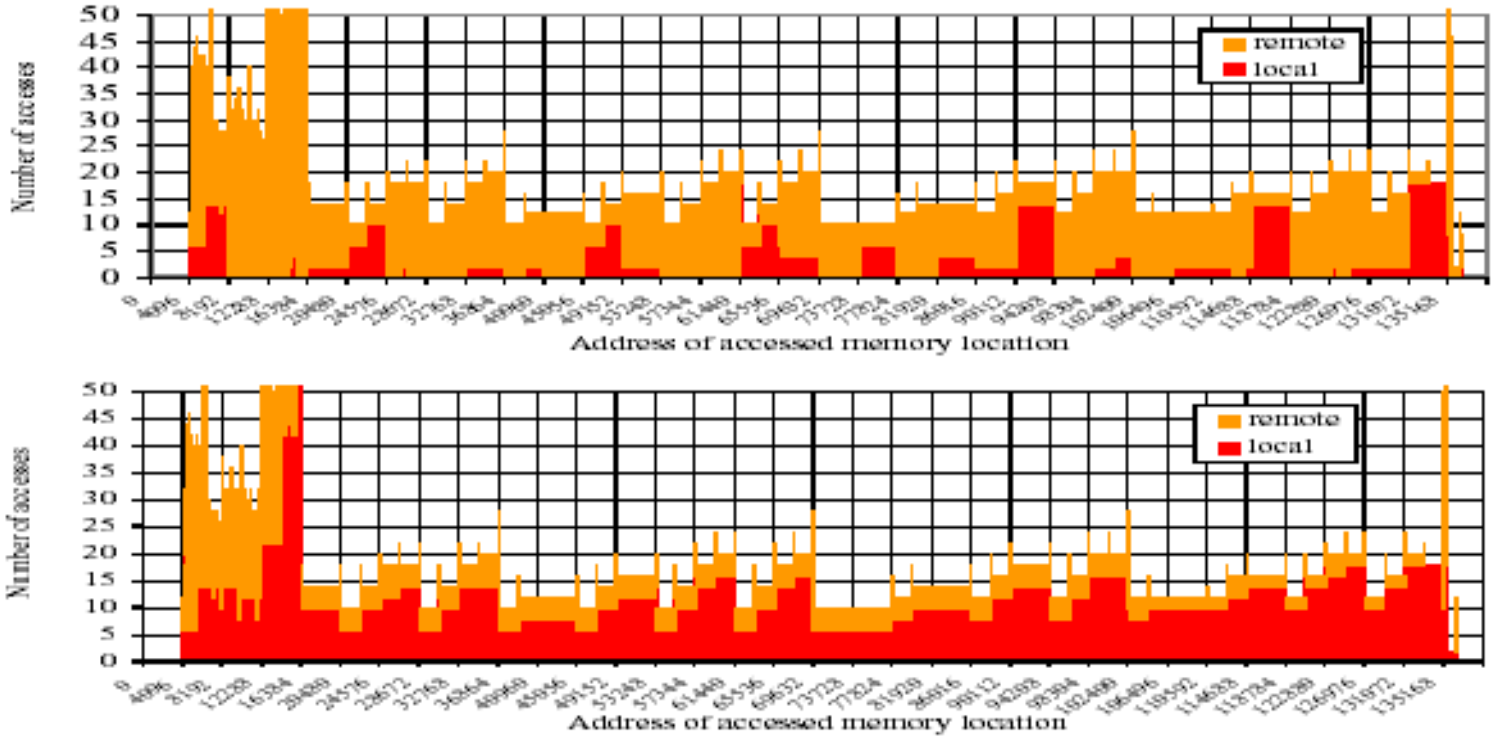
Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Visualisierung: Rückführung auf Quellcode



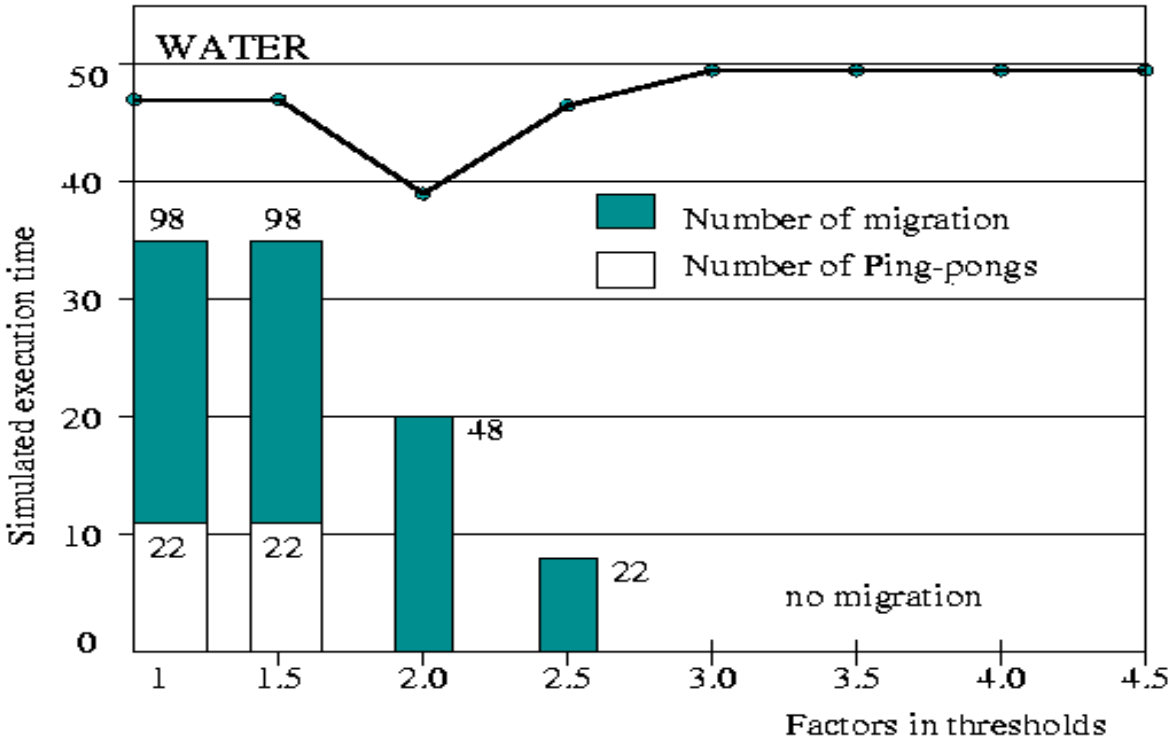
Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Beispiel LU Zerlegung: Verbesserung des Datenlokalität



Beobachtung des Laufzeitverhaltens

- **Fallstudie: CAPP Systemweite Monitoring Infrastruktur**
- Adaptive Optimierungskomponente:
  - Automatische Seitenmigration



Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- Unabhängig von der Existenz eines Rechners

- **Modellbildung**
  - Annahmen über die Struktur und Betrieb eines Rechners und über die Prozesse
  - Darstellung der für die Analyse relevanten Merkmale des Systems:
    - Systemkomponenten
    - Datenverkehr zwischen den Systemkomponenten
  - Abstrahierung komplexer Systeme
    - Nur die interessierenden Größen werden erfasst
  - Ziel:
    - Aufdecken von Beziehungen zwischen Systemparametern
    - Ermitteln von Leistungsgrößen (Auslastung von Prozessoren und Kanälen, mittlere Antwortzeiten, Warteschlangenlängen, …)

Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

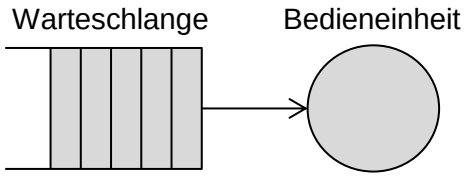
- **Analytische Methoden**
  - versuchen auf mathematischem Weg, Beziehungen zwischen relevanten Leistungskenngrößen und fundamentalen Systemparametern herzuleiten
  - oft nur minimaler Aufwand, aber dafür weniger aussagekräftig

  - Warteschlangenmodelle
    - Leistungsanalyse von Rechensystemen
  - Petrinetze
    - theoretische Untersuchungen
  - Diagnosegraphen
    - Zuverlässigkeitsanalysen
  - Netzwerkflussmodelle
    - Kapazitätsüberlegungen

Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- **Analytische Methoden**
- Beispiel Warteschlangenmodelle
  - Gesetz von Little: $k = \lambda * t$ bzw. $Q = \lambda * w$
    - k: mittlere Anzahl der Aufträge
    - λ: Durchsatz (mittlere Anzahl von Aufträgen, die pro Zeiteinheit bedient werden)
    - t: Antwortzeit (Verweilzeit, Gesamtheit der Zeit, die ein Auftrag im Wartesystem verbringt)
    - Q: mittlere Warteschlangenlänge
    - w: Wartezeit (Zeit, die angibt, die ein System im Wartesystem verbringt)
    - Voraussetzung: statistisches Gleichgewicht:
      - Die Rate, mit der die Aufträge ankommen ist gleich der Rate, mit der die Aufträge abgehen



Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- **Simulation**
  - Vorgänge in einem Rechensystem werden nachgebildet
  - Verwendung üblicher Programmiersprachen oder spezieller Simulationssprachen
  - Verhalten des Simulationsmodells in Bezug auf die relevanten Parameter entspricht weitgehend dem Verhalten des realen Systems
  - Ermittlung der für die Leistungsbewertung interessierenden Größen

Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- **Simulation**
  - Deterministische Simulation
    - Alle an dem Modell beteiligten Größen sind exakt definiert oder berechenbar
  - Stochastische Simulation
    - Verwendung von zufallsabhängigen Größen
    - Oft: Einsatz von Zufallsgeneratoren
  - Aufzeichnungsgesteuerte Simulation
    - Verwendung von gemessenen Werten

Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- **Simulation**
  - Nachteile:
    - Vorbereitung und Ausführung der Simulationsmodelle zeitaufwendig und teuer
    - Planung der Experimente muss sorgfältig durchgeführt werden
    - Auswertung und Interpretation der Ergebnisse nicht immer einfach
  - Beispiel:
    - SimpleScalar Tool Set (http://www.simplescalar.com)
    - „Standard“-Werkzeug zur Simulation von superskalaren Mikroprozessoren

Bewertung der Leistungsfähigkeit

Modelltheoretische Verfahren

- Simulation vs. Analytische Methoden
  - Simulation:
    - Realistischere Annahmen über das System bei der Simulation
    - Berücksichtigung vieler verschiedener Systemgrößen
    - Abdeckungen verschiedener Anwendungsbereiche
  - Können sich gegenseitig gut ergänzen

Bewertung der Leistungsfähigkeit

Zusammenfassung

	Auswertung von Hardwaremaßen und Parametern	**Laufzeit-messungen bestehender Programme**	**Messungen während des Betriebs der Anlagen**	**Modell-theoretische Verfahren**
Rechnerauswahl	Maßzahlen für die Operations-geschwindingkeit Kernprogramme	Benchmarks		
Rechner-„Tuning“			Hardware-Monitore Software-Monitore	
Rechnerentwurf				Analytische Methoden Simulation

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Zuverlässigkeit (dependability)**
  - bezeichnet die Fähigkeit eines Systems, während einer vorgegebenen Zeitdauer bei zulässigen Betriebsbedingungen die spezifizierte Funktion zu erbringen.
  - Ziel

- **Fehlertoleranz (fault tolerance)**
  - bezeichnet die Fähigkeit eines Systems, auch mit einer begrenzten Anzahl fehlerhafter Subsysteme die spezifizierte Funktion (bzw. den geforderten Dienst) zu erbringen.
  - Technik

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Sicherheit (safety)**
  - bezeichnet das Nichtvorhandensein einer Gefahr für Menschen oder Sachwerte. Unter einer Gefahr ist ein Zustand zu verstehen, in dem (unter anzunehmenden Betriebsbedingungen) ein Schaden zwangsläufig oder zufällig entstehen kann, ohne dass ausreichende Gegenmaßnahmen gewährleistet sind.
- **Vertraulichkeit (security)**
  - betrifft Datenschutz, Zugangssicherheit.

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- Zuverlässigkeit ist durch **Zuverlässigkeitskenngrößen** zu quantifizieren:
  - Beispiele: Verfügbarkeit, Überlebenswahrscheinlichkeit, …

- **Anforderung des Benutzers**
  - Bei Erneuerung, Erweiterung oder Wechsel des Systems sollen die Auswirkungen der Änderungen auf die Anwendungen nicht nennenswert wahrnehmbar sein.
  - **Wartungsfreundlichkeit**
    - System: Instandhaltung notwendig?
    - Komponenten: Ersatz?
    - Datenbestände: langfristig lesbar?

Zuverlässigkeit und Fehlertoleranz

Begriffsbildung

- **Nutzungsdauer** eines Rechners
  - Mindestens 5 Jahre oder 40000 h, Dauerbetrieb notwendig?
  - Sehr kleine Ausfallraten der Komponenten ($< 10^{-9}h^{-1}$)
  - Probleme: Nachweisbarkeit, Testmöglichkeiten

- **Ausfall** durch
  - Hardwarekomponenten
  - Software (Programmfehler)
  - Menschliche Eingriffe

- **Sicherheitsrelevante Anwendungen**
  - erfordern hohe **Verfügbarkeit**

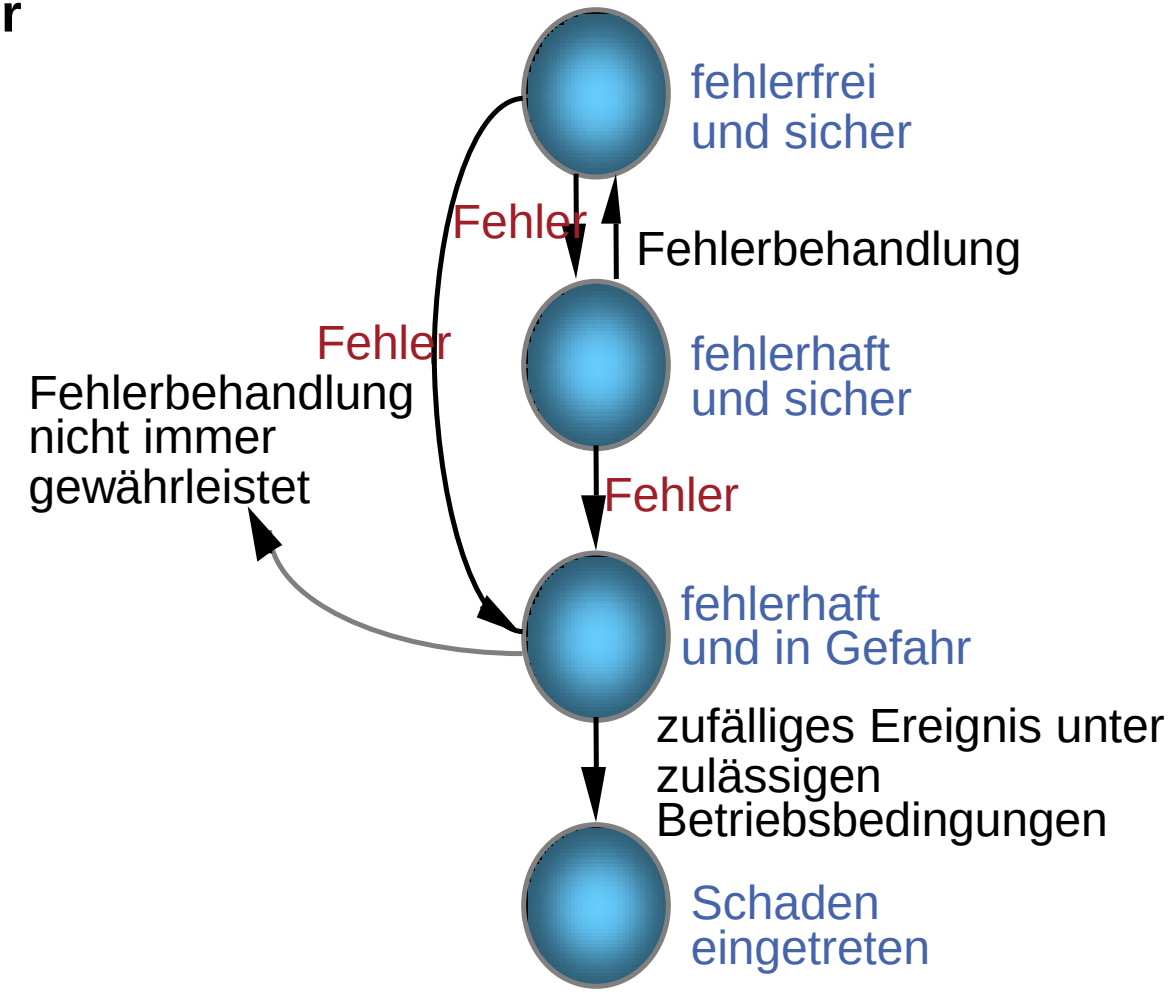
Zuverlässigkeit und Fehlertoleranz

Fragen:

- Wie zuverlässig sind heutige Rechensysteme?
- Nutzen redundante, also fehlertolerante Strukturen zur Verbesserung der Zuverlässigkeit?
- Welche Verbesserung der Zuverlässigkeit lässt sich durch derartige Fehlertoleranzmaßnahmen überhaupt erreichen?

Zuverlässigkeit und Fehlertoleranz

Fehler



Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Funktionsausfälle**
  - Unzulässige bzw. aussetzende Funktion einer Komponente
- **Fehlzustände** (unzulässiger Zustand) einzelner Komponenten des Rechensystems oder Störungen
  - Sind verantwortlich für den Ausfall
  - Werden durch verschiedenste Fehlerursachen erzeugt
- **Wirkungskette**:
  - **Fehler → Fehlzustand → Ausfall**
  - Fehlerausbreitung verhindern!
- **Ziel der Fehlertoleranz**:
  - Tolerierung der Fehlzustände von Teilsystemen (Komponenten)
  - Erhöhung der Zuverlässigkeit
  - Behebung der Fehlzustände vor dem Ausfall des Systems

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Ursachen**
  - **Fehler beim Entwurf**
    - Führen dazu, dass ein von vornherein fehlerhaftes System konzipiert wird
      - Spezifikationsfehler
      - Implementierungsfehler
      - Dokumentationsfehler
  - **Herstellungsfehler**
    - Verhindern, dass aus einem korrekten Entwurf ein fehlerfreies Produkt entsteht

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Ursachen**
  - **Betriebsfehler**
    - Erzeugen während der Nutzungsphase eines Rechensystems einen fehlerhaften Zustand in einem vormals fehlerfreien System
    - **Störungsbedingte Fehler**
      - Störungen mechanischer, elektrischer, magnetischer, elektromagnetischer oder thermischer Art sind auf äußere Einflüsse zurückzuführen, denen keine Ursachen im Rechensystem selbst zugrunde liegt
    - **Verschleißfehler**
      - Treten mit zunehmender Betriebsdauer in der HW auf
    - **Zufällige physikalische Fehler**
    - **Bedienungsfehler**
      - Bewusste oder unbewusste Fehleingaben des Benutzers
    - **Wartungsfehler**

Zuverlässigkeit und Fehlertoleranz

Fehler:

- **Fehlerentstehungsort**
  - **Hardwarefehler**
    - Umfassen alle Entwurfs-, Herstellungs- und Bedienfehler
  - **Softwarefehler**
    - Umfassen alle Fehler, die in Programmteilen entstehen

- **Fehlerdauer**
  - **Permanente Fehler**
    - bestehen ab ihrem Auftreten so lange ununterbrochen auf, bis geeignete Reparatur- oder Fehlertoleranzmaßnahmen ergriffen werden
  - **Temporäre Fehler**
    - Treten nur vorübergehend auf
    - Entstehen eventuell mehrmals spontan und verschwinden wieder

Zuverlässigkeit und Fehlertoleranz

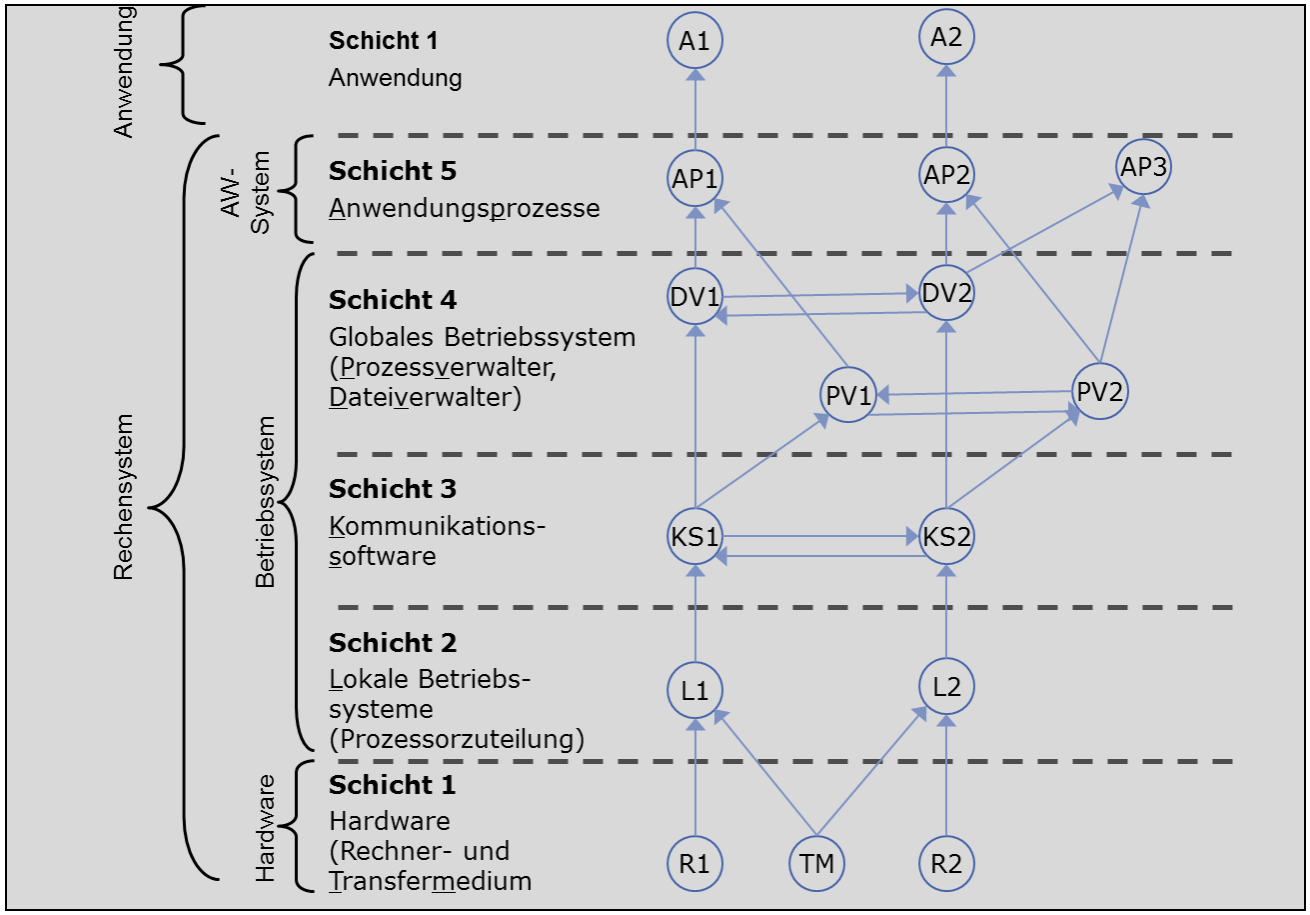
Struktur-Funktions-Modell

- Definition:
  - Das **Struktur-Funktions-Modell** ist ein gerichteter Graph, dessen Knoten die Komponenten und dessen Kanten die Funktionen eines Systems repräsentieren. Eine gerichtete Kante von der Komponente K_i zur Komponente K_j bedeutet, dass K_i eine Funktion erbringt, die von K_j benutzt wird.
  - Eine **Komponentenmenge** heißt **System**, wenn die nach außen erbrachten Funktionen in einer äußeren Spezifikation festgelegt sind. Ein System, das Teilmenge eines anderen ist, heißt **Subsystem**.
  - **Schichtenmodell**:
    - Die Komponenten werden in disjunkte Schichten partitioniert, für die es eine Totalordnung gibt. Funktionszuordnungen sind nur von niedrigeren an höhere Schichten (eine Halbordnung) und innerhalb von Schichten möglich.

Zuverlässigkeit und Fehlertoleranz

Struktur-Funktions-Modell

- **Schichtenmodell** eines 2-Rechensystems



Zuverlässigkeit und Fehlertoleranz

Definitionen:

- Ein **Fehlermodell** beschreibt die möglichen Fehlzustände eines Systems, beispielsweise durch Angabe der Komponentenmengen, die zugleich von einer Fehlerursache betroffen sein können und durch Angabe des möglichen fehlerhaften Verhaltens dieser Komponenten.
- **Binäres Fehlermodell**:
  - Binäre **Fehlerzustandsfunktion Z** gibt für jede Komponente und das System an, ob sie fehlerfrei sind (wahr = kein Fehler, falsch = Fehler):

$$Z : (S \cup \{S\}) \rightarrow \{wahr, falsch\}$$

  - Ein System, das nur dann fehlerfrei arbeitet, wenn es seit der Inbetriebnahme fehlerfrei war, erfüllt:

$$Z(S,t) = \bigwedge_{t_0 \leq t} Z(S,t_0)$$

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Nichtredundantes System**
  - Ein System, das nur dann fehlerfrei ist, wenn alle seine Komponenten fehlerfrei sind, wird charakterisiert durch

$$Z(S) = Z(K_1) \wedge \ldots \wedge Z(K_n)$$

- **Systemfunktion** $f(K_1, \ldots, K_n)$
  - gibt an, wie sich die Funktion des Systems aus den Funktionen der einzelnen Komponenten ableitet.
  - Systemfunktion für ein nichtredundantes System

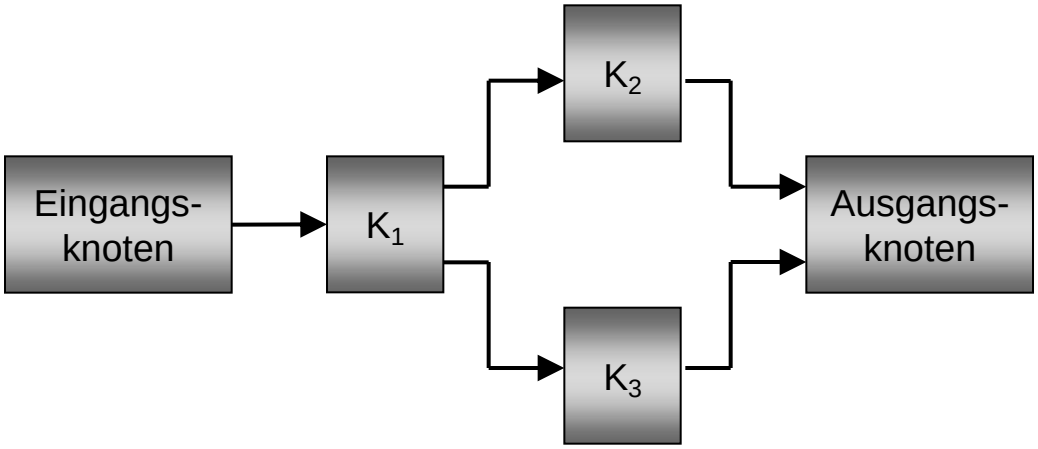
$$S = K_1 \wedge \ \ldots \ \ \wedge K_n$$

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Zuverlässigkeitsblockdiagramm**
  - Die Systemfunktion lässt sich grafisch durch ein Zuverlässigkeitsblockdiagramm darstellen:
  - Gerichteter Graph mit einem Eingangs- und einem Ausgangsknoten
  - Beispiel für Systemfunktion

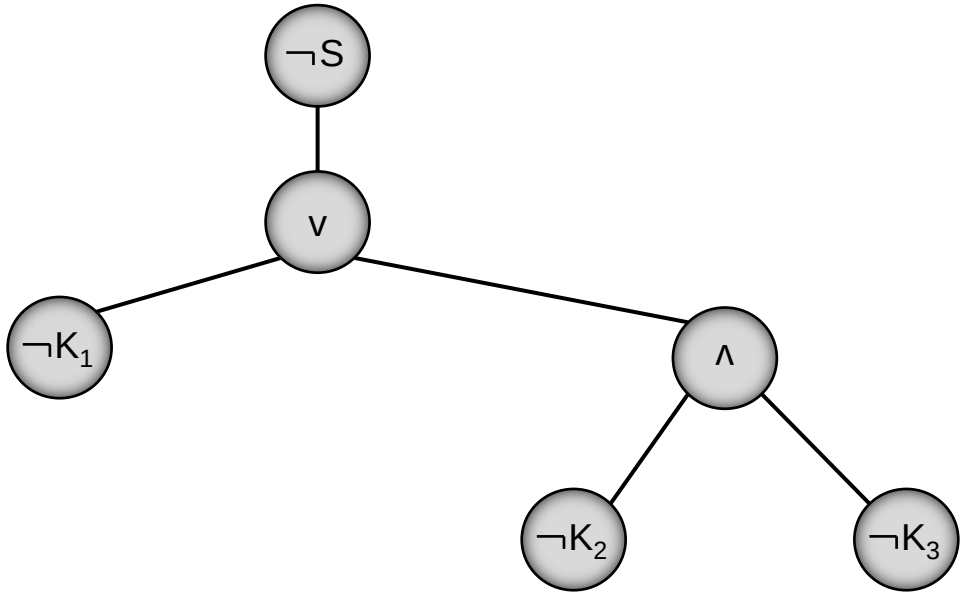
$$S = K_1 \wedge (K_2 \vee K_3)$$
$$Z(S) = Z(K_1) \wedge (Z(K_2) \vee Z(K_3))$$



Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbaum**
  - Strukturbaum der Negation der Systemfunktion
  - Stellt graphisch dar, wie sich Fehler des Systems auf Fehler der Komponenten zurückführen lassen
  - Beispiel: **Fehlerbaum** für $S = K_1 \wedge (K_2 \vee K_3)$ d. h. $\neg S = \neg K_1 \vee (\neg K_2 \wedge \neg K_3)$



Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbereich B:**
  - Ein **Fehlerbereich** B $\subset$ S ist eine Menge von Komponenten, die zugleich fehlerhaft sein können, ohne dass das System S insgesamt fehlerhaft wird.
  - D. h.: aus $\forall K \in S - B : Z(K) =$ wahr

    folgt: $Z(S) =$ wahr
  - Beispiel: $S = K_1 \wedge (K_2 \vee K_3) \rightarrow B_1 = \{K_2\}$ und $B_2 = \{K_3\}$

Zuverlässigkeit und Fehlertoleranz

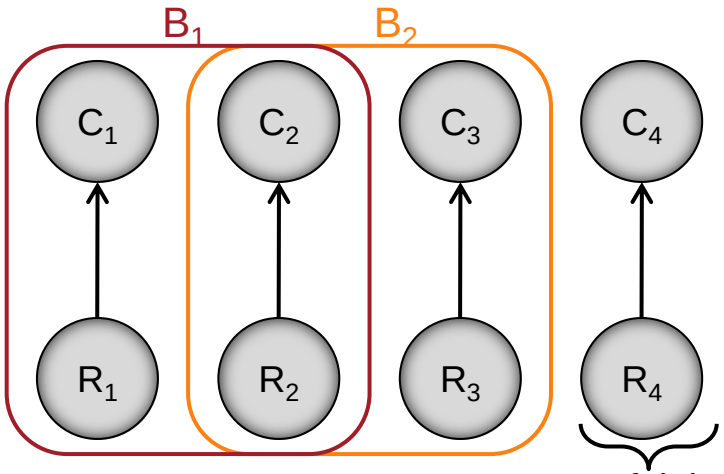
Binäres Fehlermodell:

- **Fehlerbereich B:**
  - **Einzelfehlerbereich:**
    - Wenn für ein System eine Menge von Fehlerbereichen Γ definiert ist, so bezeichnen wir eine Menge von Komponenten, die genau den gleichen Fehlerbereichen angehören, als Einzelfehlerbereich.
  - **Perfektionskern**
    - das Komplement der Vereinigung aller Fehlerbereiche
  - Es wird von den einzelnen Komponenten dieser Bereiche abstrahiert. Dadurch können sich Systemmodelle erheblich vereinfachen.

Zuverlässigkeit und Fehlertoleranz

Binäres Fehlermodell:

- **Fehlerbereich B:**
  - Beispiel: Für ein System S = {R_1,R_2,R_3,R_4,C_1,C_2,C_3,C_4} könnte die Fehlerbereichsannahme wie folgt lauten:
    - Γ = {B_1, B_2
    - Fehlerbereiche B_1 = {R_1,R_2,C_1,C_2} und B_2 = {R_2,R_3,C_2,C_3}
    - Einzelfehlerbereiche E_1 = {R_1,C_1}, E_2 = {R_2,C_2} und E_3 = {R_3,C_3}
    - Perfektionskern P_1 = {R_4,C_4}



Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Steht das mögliche Ausfallverhalten bestimmter Komponenten eines Rechensystems im Vordergrund der Betrachtung, dann ist das eingeführte binäre Fehlermodell zu allgemein
- Es treten nur bestimmte Fehlfunktionen auf.

- Die aus bestimmten Fehlfunktionsannahmen hervorgehenden Einschränkungen des fehlerhaften Verhaltens können den Redundanzaufwand für ein Fehlertoleranzverfahren teilweise erheblich reduzieren.

- Um zu erreichen , dass auf einzelne Komponenten nur bestimmte Fehlfunktionsannahmen zutreffen, kann es notwendig sein, die Komponenten selbst fehlertolerant zu gestalten.

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- **Teilausfall**: Von einer fehlerhaften Komponente fallen eine oder mehrere, aber nicht alle Funktionen aus.
- **Unterlassungsausfall**: Eine fehlerhafte Komponente gibt eine Zeit lang keine Ergebnisse aus. Wenn jedoch ein Ergebnis ausgegeben wird, dann ist dieses korrekt.
- **Anhalteausfall**: Eine fehlerhafte Komponente gibt nie mehr ein Ergebnis aus.
- **Haftausfall**: Eine fehlerhafte Komponente gibt ständig den gleichen Ergebniswert aus.
- **Binärstellenausfall**: Ein Fehler verfälscht eine oder mehrere Binärstellen des Ergebnisses.

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Systeme, die nur eine bestimmte Art von Ausfallverhalten aufweisen
  - **Fail-stop-System**:
    - Ein System, dessen Ausfälle nur *Anhalteausfälle* sind
  - **Fail-silent-System**:
    - Ein System, dessen Ausfälle nur *Unterlassungsausfälle* sind
  - **Fail-safe-System**:
    - Ein System, dessen Ausfälle nur *unkritische Ausfälle* sind

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- Der Ausfall einer ursächlich fehlerhaften Komponente K kann auch die Fehlerursache für Fehler in anderen Komponenten darstellen, wenn diese Funktionen auf K zugreifen: **Folgefehler**
- **Maßnahmen der Fehlereingrenzung**

Zuverlässigkeit und Fehlertoleranz

Ausfallverhalten

- **Maßnahmen der Fehlereingrenzung**
  - **Vertikale Fehlereingrenzung von höheren auf niedrigere Schichten**
    - Niedrigere Schichten prüfen die Funktionsaufrufe vor ihrer Ausführung
    - Beispiel: jeder unzulässige Befehlscode führt zu einer Fehlermeldung.
  - **Vertikale Fehlereingrenzung von der Hardware auf höhere Schichten**
    - Beispiel: Fehlerkorrekturcode im Arbeitsspeicher
  - **Vertikale Fehlereingrenzung von niedrigeren auf höhere Software-Schichten**
    - Durchführen von Plausibilitäts- und Konsistenzprüfungen der Ergebniswerte in höheren Schichten.
    - Es können viele, aber nicht alle Fehler erkannt werden
  - **Horizontale Fehlereingrenzung in lokalen Schichten:**
    - **z.B. (räumliche, elektrische, thermische, ...) Isolierung der Knoten.**
  - **Horizontale Fehlereingrenzung in globalen Schichten:**
    - Hauptproblem der Fehlereingrenzung
    - erfordert mitunter aufwendige Fehlertoleranzverfahren.

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzanforderungen

- Hohe **Überlebenswahrscheinlichkeit**
  - Beispielsweise zwecks Erfolg bei einer kurzzeitige Mission (z.B. 10-stündiger Flug)
- Hohe **mittlere Lebensdauer**
  - z.B. bei begrenzten Reparaturmöglichkeiten in unzugänglichen Rechensystemen
- Hohe **Verfügbarkeit**
  - z.B. im interaktiven Rechenzentrums- oder Nutzerbetrieb
- Hohe **Sicherheitswahrscheinlichkeit**
  - Schutz von Menschen, Maschinen, Daten
- Hohe **Sicherheitsdauer**

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzanforderungen

- **Vorgehensweise zur Erfüllung der Anforderungen**
  - **Fehlervermeidung**
    - Perfektionierung, Verwendung von zuverlässigen Komponenten, sorgfältiger Entwurf
  - **Fehlertoleranz**
    - Erfordert Redundanz und damit Zusatzaufwand

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Gesichtspunkte bei der Konstruktion**
  - Ableiten einer **Fehlervorgabe**
    - aus den angenommenen Fehlerraten der Komponenten und den aus den Zuverlässigkeitsanforderungen an das Gesamtsystem
    - Fehlervorgabe besteht aus **Fehlermodell** und der **Menge der zu tolerierenden Fehler**
  - **Menge der zu tolerierenden Fehler**
    - gibt an, welche der im Fehlermodell vorgesehenen Fehler zu tolerieren sind.
    - Häufig wird die Menge der zu tolerierenden Fehler bezüglich einer ***Fehlerbereichsannahme*** formuliert;
      - in diesem Fall ist festzulegen, wie viele ***Einzelfehlerbereiche*** gleichzeitig fehlerhaft werden können und
      - welche ***Fehlfunktionen*** zu behandeln sind.

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Gesichtspunkte bei der Konstruktion**
  - Zur Behandlung von mehreren nacheinander auftretenden Fehlern muss jeweils ein Zeitintervall gegeben sein, in dem keine zusätzlichen Fehlerauftreten, bevor die Fehlerbehandlung abgeschlossen ist
  - **Zeitredundanz**
    - Zeitintervall in dem keine weiteren Fehler auftreten, bevor die Fehlerbehandlung abgeschlossen ist
  - **Fehlerbehandlungsdauer**
    - Zeit, die das Fehlerbehandlungsverfahren benötigt, um den Fehler zu behandeln
    - Fehlerbehandlungsdauer muss kleiner als die Zeitredundanz sein

Zuverlässigkeit und Fehlertoleranz

Fehlertoleranzverfahren

- **Zusätzliche Anforderungen**
  - Nachweis der Fehlertoleranzfähigkeit
    - Verifikation, Validierung, Durchführung einer Anfälligkeitsanalyse
  - Geringer Betriebsmittelbedarf (geringe Kosten)
  - Schnelle Ausführung von Fehlertoleranzverfahren (Leistung)
  - Unabhängigkeit von der Anwendungssoftware (Transparenz)
  - Unabhängigkeit vom Rechensystem

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Zuverlässigkeit, Sicherheit einer Rechensystems**
  - Quantifizierbar mittels stochastischer Modelle
  - Man betrachtet die kontinuierliche Variable Zeit zwischen dem Zeitpunkt, ab dem die Zuverlässigkeitsbetrachtung beginnen soll (Zeitpunkt Null), bis zum Auftreten eines betrachteten Effekts

  - Nichtnegative Zufallsvariablen:
    - **Lebensdauer L** – besitzt die **Dichte $f_L(t)$**
    - **Fehlerbehandlungsdauer B** – besitzt die **Dichte $f_B(t)$**
    - **Sicherheitsdauer D** – besitzt die **Dichte $f_D(t)$**

  - Korrespondierende **Verteilungsfunktionen**

$$F_x(t) := \int_0^t f_x(s)ds \quad \text{mit x = L, B und D}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Fehlerwahrscheinlichkeit $F_L(t)$**
  - Bezeichnet die Wahrscheinlichkeit, dass ein zu Beginn fehlerfreies System im Zeitintervall [0,t] fehlerhaft wird

- **Überlebenswahrscheinlichkeit R(t)**
  - Gibt an, mit welcher Wahrscheinlichkeit ein zu Beginn (also zum Zeitpunkt t=0) fehlerfreies System bis zum Zeitpunkt t ununterbrochen fehlerfrei bleibt
  - Man setzt $R(t) := 1 - F_L(t)$
  - Es gilt für die Verteilungsfunktionen nichtnegativer Zufallsvariablen FL, dass diese in t monoton wachsen und es gilt:
    - $F_L(t)=0$ und $\lim_{t\to\infty} F_L(t) = 1$
    - damit folgt: $R(0)=1, \lim_{t\to\infty} R(t) = 0$ und R ist in t monoton fallend

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Mittlere Lebensdauer E(L):**
  - Bezeichnet für ein zu Beginn fehlerfreies System den Erwartungswert der Zeitdauer bis zum ersten Fehler. Es gilt:

$$E(L) = \int_0^\infty R(t)dt$$

- **Ausfallrate z(T)**
  - Die Ausfallrate bezeichnet den Anteil der in einer Zeiteinheit ausfallenden Komponenten bezogen auf den Anteil der noch fehlerfreien Komponenten
  - Damit definiert man:

$$z(t) := \frac{t_L(t)}{T(t)}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Fehlerwahrscheinlichkeit $F_L(t)$:**
  - Ist nur die Ausfallrate bekannt, so ergibt sich die Fehlerwahrscheinlichkeit aus der Anfangswertaufgabe

$$\frac{d}{dt} F_L(t) = f_L(t) = z(t) \cdot (1 - F_L(t))$$

  - mit der Anfangsbedingung $F_L(0) = 0$
  - Die Anfangswertaufgabe hat die Lösung:

$$F_L(t) = 1 - e^{-\int_0^t z(s)ds}$$

  - Bei einer konstanten Ausfallrate $z(t) = \lambda$ ist die Fehlerwahrscheinlichkeit folglich exponentialverteilt mit Parameter λ

$$F_L(t) = 1 - e^{-\lambda t}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Verfügbarkeit V:**
  - Die Verfügbarkeit bezeichnet die Wahrscheinlichkeit, ein System zu einem beliebigen Zeitpunkt fehlerfrei anzutreffen.
  - Es interessiert der zeitliche Anteil der Benutzbarkeit des Systems an der Summe der Erwartungswerte von Lebensdauer L und Behandlungsdauer B, wenn während B das System repariert und wieder funktionsfähig wird.
  - Es gilt:

$$v := \frac{E(L)}{E(L) + E(B)}$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Sicherheit einer Rechensystems**
  - Geht man Ausfällen aus, die die Sicherheit beeinträchtigen, dann ergeben sich analog zu den bisher betrachteten Größen solche mit Sicherheitsrelevanz
  - **Gefährdungswahrscheinlichkeit $F_D(t)$**
    - Wahrscheinlichkeit, dass ein zu Beginn sicheres System im Zeitintervall [0,t] in einen gefährlichen Zustand gerät
  - **Sicherheitswahrscheinlichkeit $S(t):=1- F_D(t)$**
    - Wahrscheinlichkeit, dass ein zu Beginn sicheres System bis zum Zeitpunkt t ununterbrochen in einem sicheren Zustand bleibt
  - **Mittlere Sicherheitsdauer E(D)**

$$E(D) = \int_0^\infty t \cdot f_D(t)dt = \int_0^\infty S(t)dt$$

    - Erwartungswert der Zeitdauer, bis ein gefährlicher Zustand auftritt

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit φ**
  - Da die Zuverlässigkeitsbewertung für die Überlebenswahrscheinlichkeit R und die Verfügbarkeit V einer Komponente bzw. eines Systems analog erfolgt, führen wir für beide Größen den Oberbegriff Funktionswahrscheinlichkeit ein.
  - Ausgehend von gegebenen Funktionswahrscheinlichkeiten $\varphi(K_1)$, …, $\varphi(K_n)$ der Komponenten ist die Funktionswahrscheinlichkeit $\varphi(S)$ des Systems S zu bestimmen. Diese muss alle möglichen Kombinationen von Werten der Fehlerzustandsfunktion aller Komponenten berücksichtigen
  - **Funktionswahrscheinlichkeit des Systems S = f(K_1, …, K_n)**

$$\varphi(S) = \sum_{(K_1,\ldots,K_n)\in f^{-1}(wahr)} \varphi(\wedge_{i=1}^{n} K_i)$$

    - wobei $K_i \in$ {wahr,falsch} den Fehlzustand der jeweiligen Komponente angibt und f^{-1}(wahr) die Menge der Kombinationen von Fehlzuständen der Komponenten des Systems S beschreibt, für die der Fehlerzustand „wahr“ ist

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Nichtfunktionswahrscheinlichkeit**

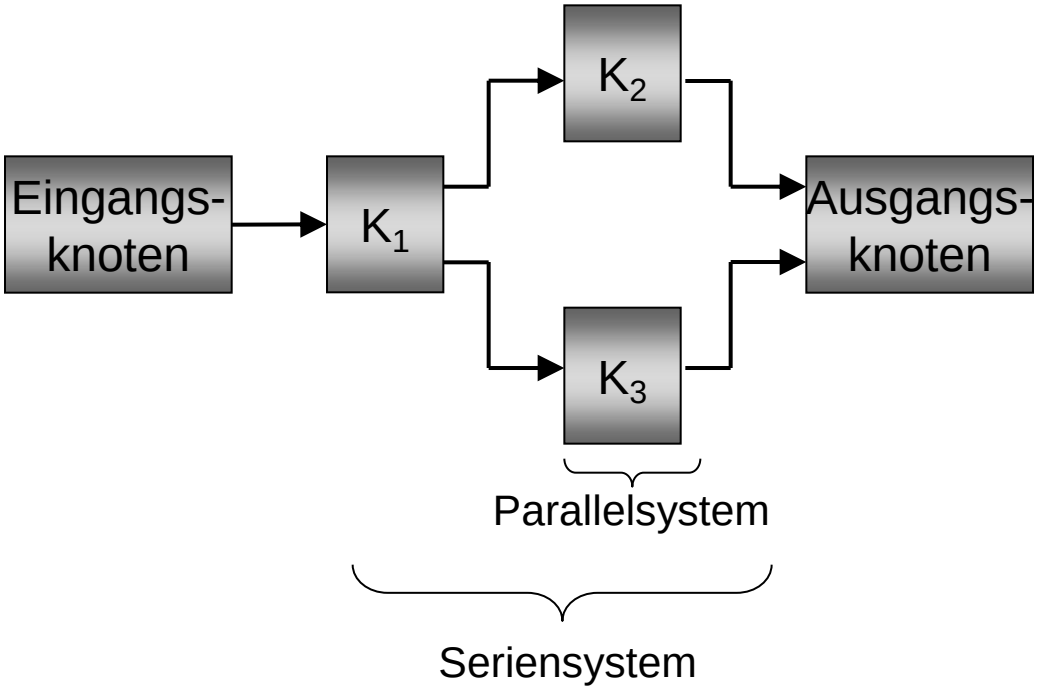
$$\varphi(\neg K) = 1 - \varphi(K)$$

Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit für Seriensystem und Parallelsystem**
  - Zuverlässigkeitsdiagramm

$$S = K_1 \wedge (K_2 \vee K_3)$$



Zuverlässigkeit und Fehlertoleranz

Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit**
  - **Seriensystem**

$$\varphi(\wedge_{K\in\Lambda}) = \prod_{K\in\Lambda} \varphi(K)$$

  - **Parallelsystem**

$$\varphi(\vee_{K\in\Lambda}) = \sum_{\varnothing\neq A\in\Lambda} (-1)^{1+\#A} \cdot \varphi(\wedge_{K\in A} K)$$

K steht für einzelne Komponenten und Λ für eine endliche Menge von Komponenten oder Systemfunktionen

  - **System** $S = K_1 \vee K_2$

$$\varphi(S) = \varphi(K_1 \vee K_2) = \varphi(K_1) + \varphi(K_2) - \varphi(K_1 \wedge K_2)$$

Zuverlässigkeit und Fehlertoleranz

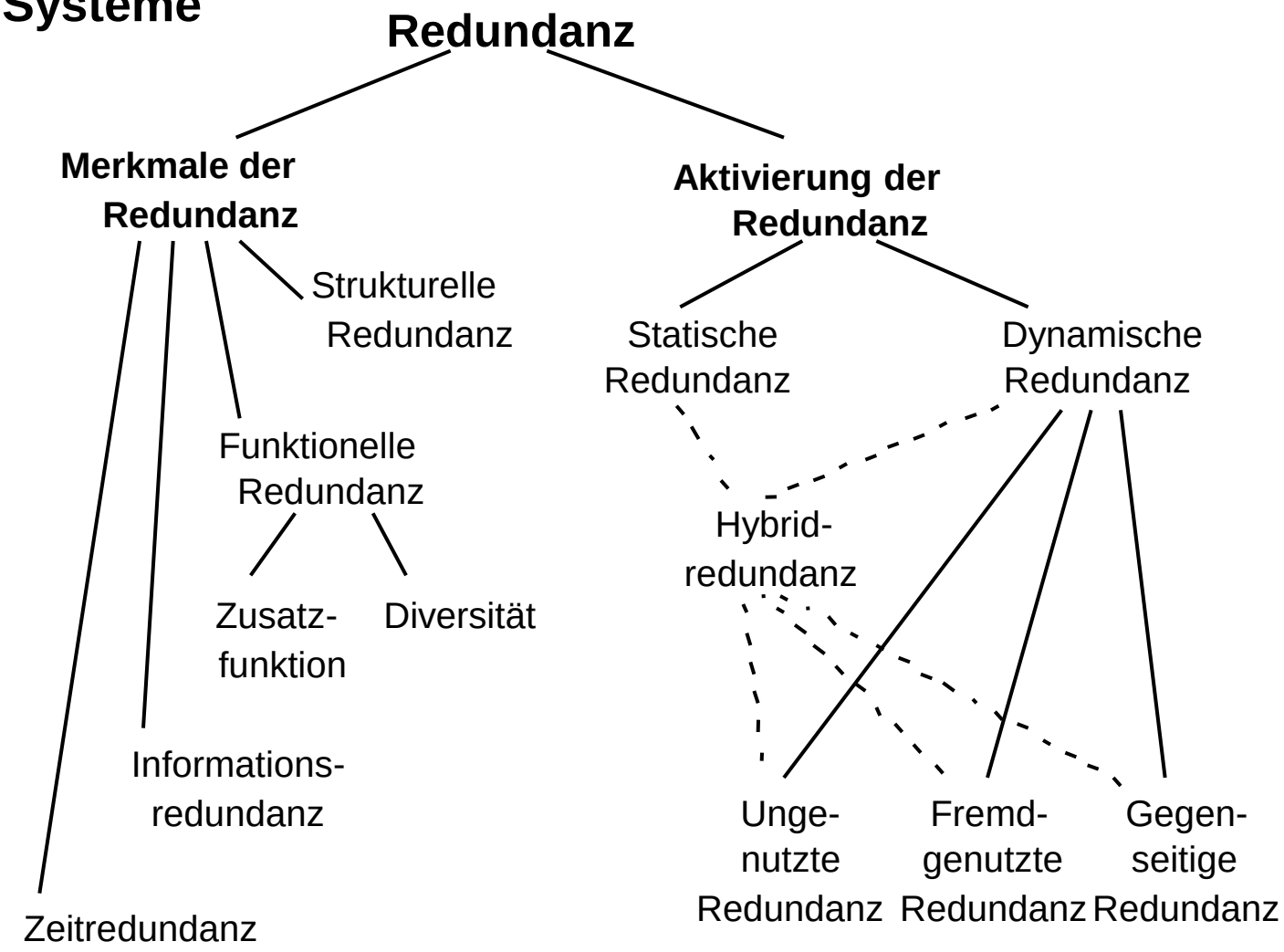
Zuverlässigkeitskenngrößen

- **Funktionswahrscheinlichkeit**
  - **Zuverlässigkeitsverbesserung**

$$\Phi_{S_1 \to S_2} = \frac{\varphi(\neg S_1)}{\varphi(\neg S_2)} = \frac{1-\varphi(S_1)}{1-\varphi(S_2)}$$

Zuverlässigkeit und Fehlertoleranz

Redundante Systeme



Zuverlässigkeit und Fehlertoleranz

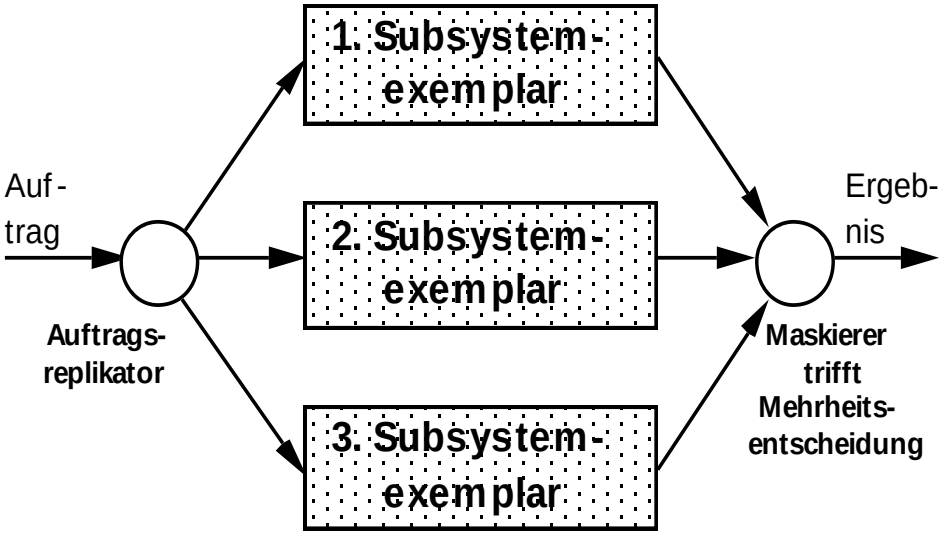
- **Redundanz**
- **Dynamische Redundanz (dynamic redundancy)**
  - bezeichnet das Vorhandensein von redundanten Mitteln, die erst nach Auftreten eines Fehlers aktiviert werden, um eine ausgefallene Nutzfunktion zu erbringen.
  - Typisch für dynamische strukturelle Redundanz ist die Unterscheidung in Primär- und Ersatzkomponenten (bzw. Sekundär- oder Reservekomponenten).
  - Grundstruktur eines dynamisch strukturell redundanten Systems

Zuverlässigkeit und Fehlertoleranz

- **Redundanz**
- **Dynamische Redundanz (dynamic redundancy)**
  - Bevor Ersatzkomponenten aktiviert werden, lassen diese sich auf eine der folgenden Arten verwenden:
  - **Ungenutzte Redundanz**
    - Ersatzkomponenten führen keine sonstigen Funktionen aus und bleiben bis zur fehlerbedingten Aktivierung passiv.
  - **fremdgenutzte Redundanz:**
    - Ersatzkomponenten erbringen nur Funktionen, die nicht zum betreffenden Subsystem gehören und im Fehlerfall bei niedrigerer Priorisierung ggf. verdrängt werden.
  - **gegenseitige Redundanz:**
    - Ersatzkomponenten erbringen die von einer anderen Komponente zu unterstützenden Funktionen, die Komponenten stehen sich gegenseitig als Reserve zur Verfügung.
      Dies ermöglicht einen abgestuften Leistungsabfall (graceful degradation).

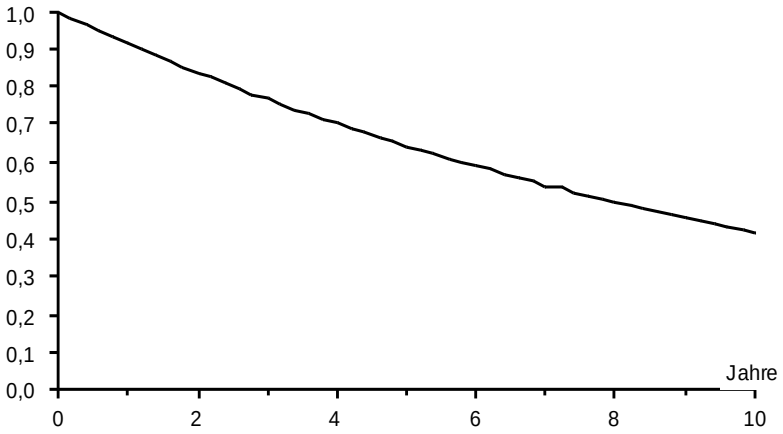
Zuverlässigkeit und Fehlertoleranz

- **Redundanz**
- **Statische Redundanz (static redundancy)**
  - bezeichnet das Vorhandensein von redundanten Mitteln, die während des gesamten Einsatzzeitraums die gleiche Nutzfunktion erbringen.
  - Beispiel der statischen strukturellen Redundanz: **n-von-m-System**
    - 2-von-3-System:



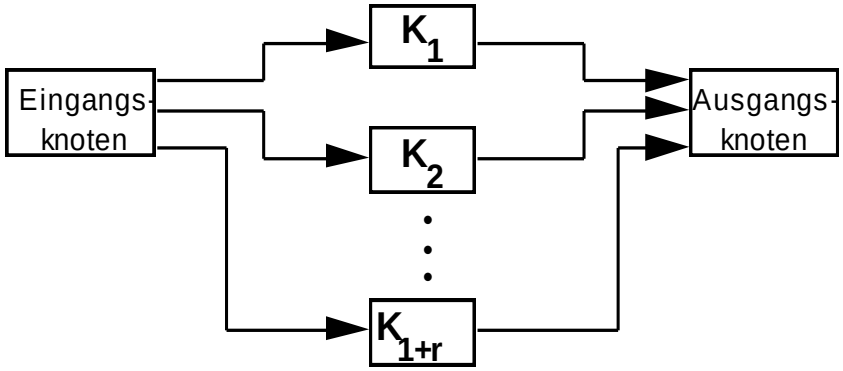
Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
  - Nichtredundantes Einfachsystem: S1=K1
  - Bei konstanter Ausfallrate beschreibt man die Zeitabhängigkeit der Funktionswahrscheinlichkeit φ (S_1, t) durch eine Exponentialverteilung
    - mit $z(t)=\lambda$, $\varphi(S_1, t) = e^{-\lambda \cdot t}$.
  - Beispiel:
    - Funktionswahrscheinlichkeit φ (S_1, t) mit $\lambda=10^{-5/h}$



Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Parallelsystem (Einfachsystem mit ungenutzter oder fremdgenutzter Redundanz)



Systemfunktion

$$S_{1+r} = K_1 \vee \ldots \vee K_{1+r}$$

Funktionswahrscheinlichkeit

$$\varphi(S_{1+r}, t) = 1 - \prod_{i=1}^{1+r} (1 - \varphi(K_i, t))$$

gleiche konstante Ausfallrate λ

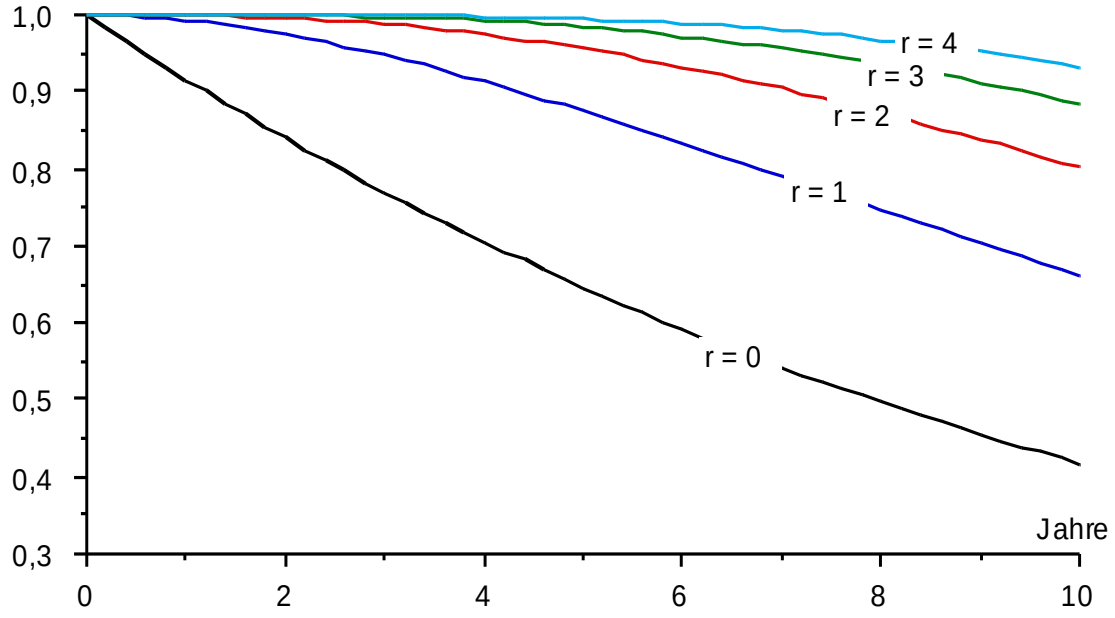
$$\varphi(S_{1+r}, t) = 1 - (1 - e^{-\lambda \cdot t})^{1+r}$$

Zuverlässigkeitsverbesserung

$$\Phi_{S1 \to S1+r} = (1 - e^{-\lambda \cdot t})^{-r}.$$

Zuverlässigkeit und Fehlertoleranz

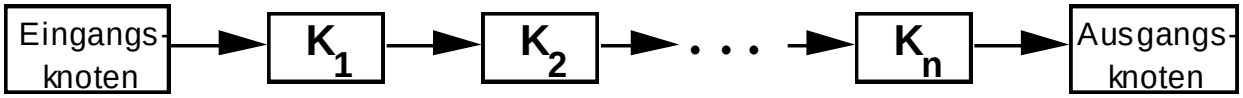
- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Funktionswahrscheinlichkeit für Parallelsystem



Annahme einer Komponentenausfallrate von $\lambda = 10^{-5}/\mathrm{h}$

Zuverlässigkeit und Fehlertoleranz

- **Verbesserung der Zuverlässigkeit durch Redundanz**
- Seriensystem

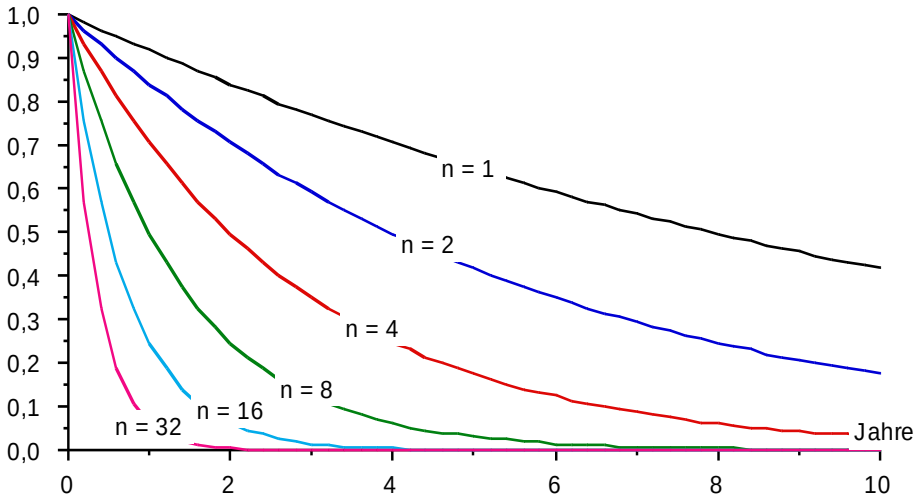


Seriensystem $S_n = K_1 \wedge \ldots \wedge K_n$

Zuverlässigkeit $\varphi(S_n, t) = \prod_{i=1}^{n} \varphi(K_i, t)$

Funktionswahrschein-lichkeit $\varphi(S_n, t)$

für $\lambda = 10^{-5}/h$



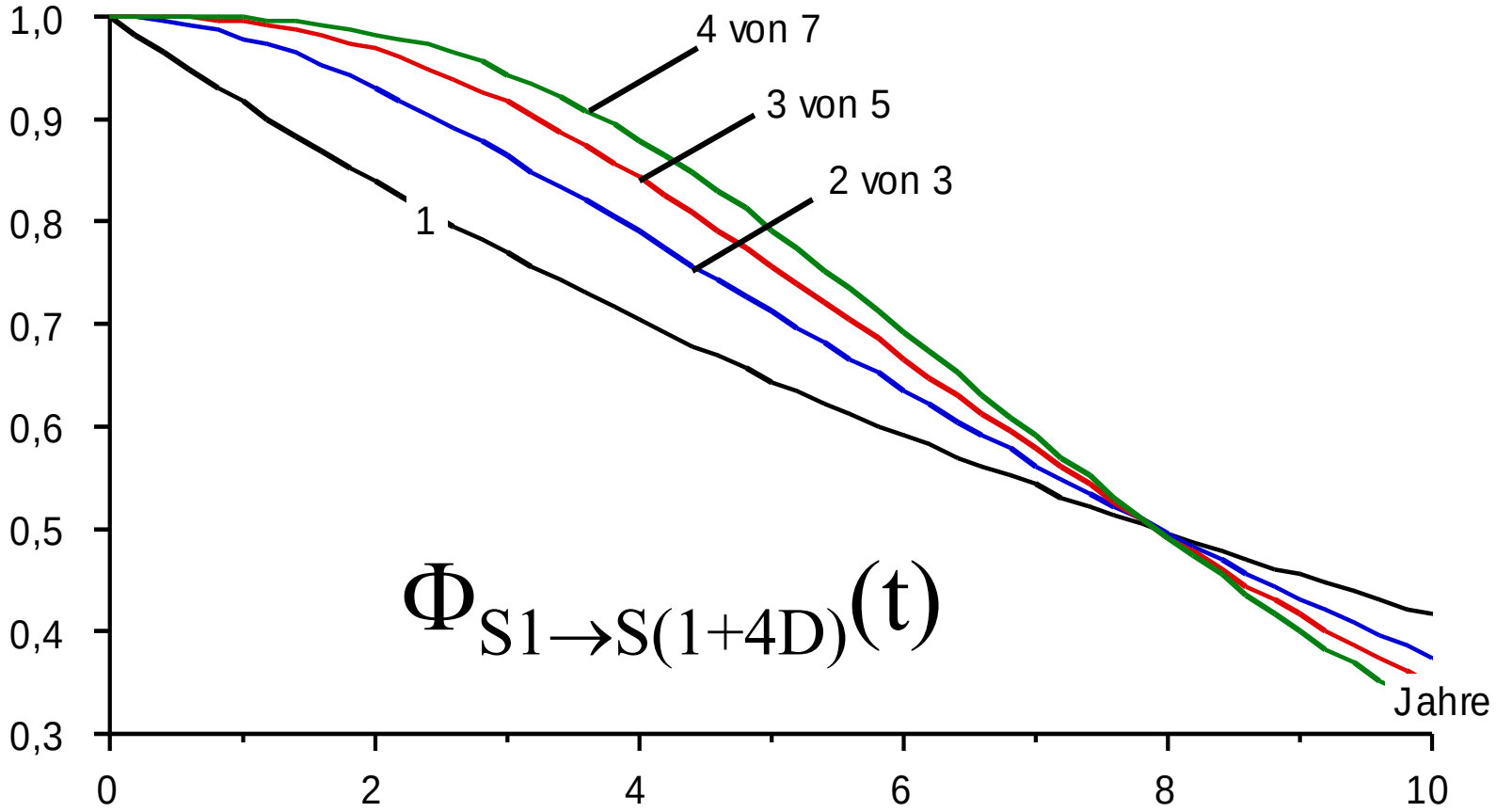
Zuverlässigkeit und Fehlertoleranz

- **Statisch redundantes System**
  - Ist die Fehlererfassung zu gering oder verbieten sich wiederholte Berechnungen wegen den geforderten maximalen Antwortzeiten, so kann statische Redundanz eingesetzt werden.
  - Dabei führen mehrere Komponenten die gleiche Berechnung aus, um anschließend die errechneten Ergebnisse zu vergleichen und ein mehrheitliches auszuwählen.
  - Bis zu f fehlerhafte Komponenten können überstimmt werden, wenn mindestens n=f+1 fehlerfreie, insgesamt also m=2·f+1 Komponenten vorhanden sind.

$$S_{m\ von\ m} = \bigvee_{1 \le i_1 < \ldots < i_n \le m} K_{i_1} \wedge \ldots \wedge K_{i_n}$$

Zuverlässigkeit und Fehlertoleranz

- **Statisch redundantes System**



Zuverlässigkeit und Fehlertoleranz

- **Statisch redundantes System**