Fehlertoleranz

- Einführung
- Begriffsbildung
- Techniken der Fehlertoleranz
  - Statisch
  - Dynamisch
- Quantitative Bewertung
- Systeme in der Praxis

Siehe: http://en.wikipedia.org/wiki/Fault_tolerance

Fehlertoleranz
Die zehn wichtigsten Fragen

- Wozu Fehlertoleranz?
- Warum sind Parallelrechner in diesem Bereich wichtige Architekturtypen
- Welche Kenngrößen werden verwendet?
- Welche Fehlerkategorien gibt es?
- Was ist statische Redundanz?
- Welches sind hierbei die typischen Konzepte?
- Was ist dynamische Redundanz?
- Welche Durchführungsphasen finden wir hierbei?
- Wie sieht die qualitative Bewertung eines Systems ohne Reparatur aus?
- Welche Umsetzung gibt es für Linux-Systeme?

Warum als Thema Fehlertoleranz?

Vorlesungsstunde hätte auch „Ausfallsicherheit" oder „Hochverfügbarkeit" heißen können

„Fehlertoleranz" ist der allgemeine Mechanismus, mit dem Ausfallsicherheit und Hochverfügbarkeit erzielt wird

Siehe: http://en.wikipedia.org/wiki/High-availability_cluster

Motivation

Wozu Fehlertoleranzmechanismen?

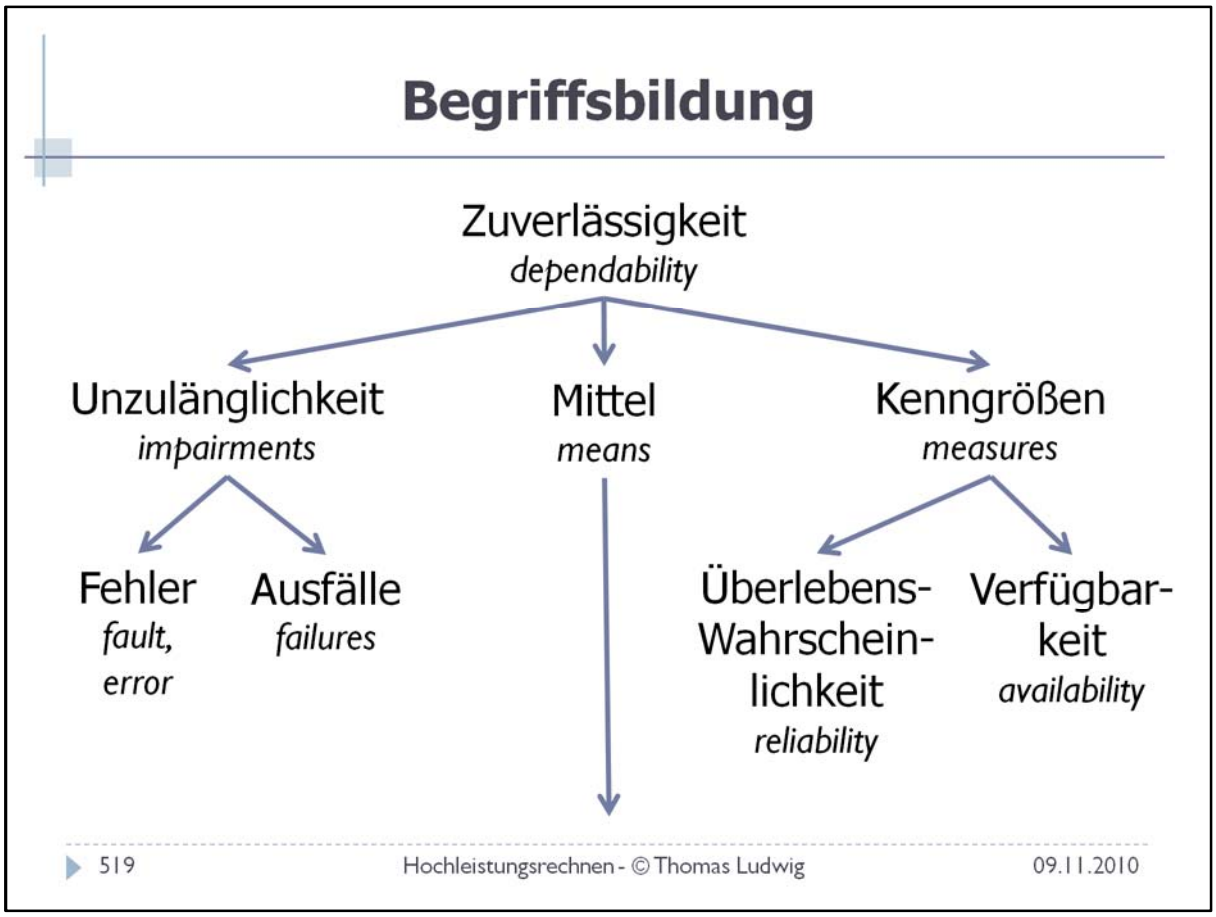
- Ausfälle im System werden verdeckt und das System läuft mit kurzer Unterbrechung oder mit verminderter Leistung weiter
- Schutz vor Verlust von Menschenleben und/oder Sachwerten

Warum auf Parallelrechnern / in Clustern?

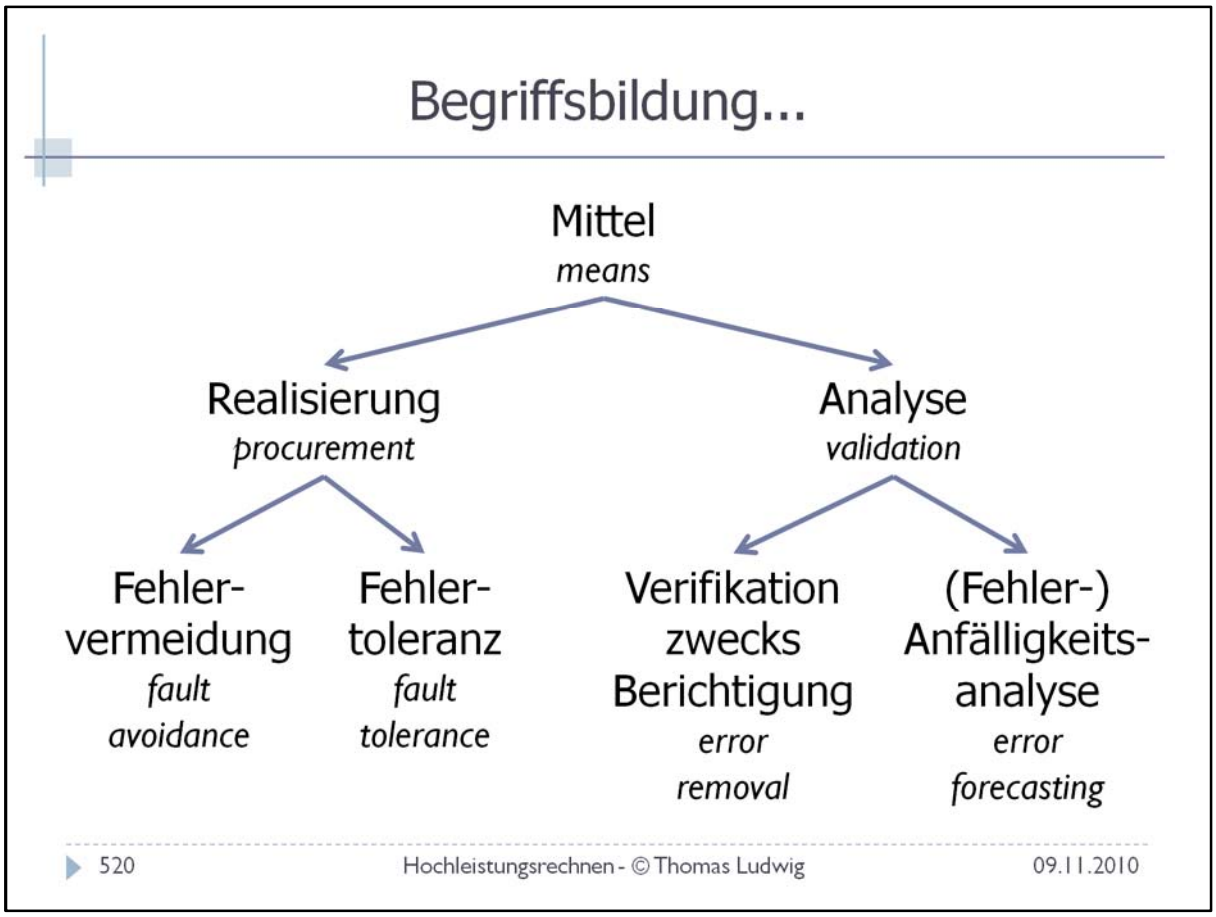
- Redundante Hardware-Strukturen eignen sich besonders gut zur Fehlererkennung und dynamischen Fehlerverdeckung

Warum wirklik auf Parallelrechnern / in Clustern?

- Redundante Hardware-Strukturen fallen ständig aus ☹



Begriffsbildung nach Belli, Echtle, Görke.



Begriffsbildung nach Belli, Echtle, Görke.

Zuverlässigkeit

Definition

Fähigkeit eines Systems, die spezifizierte Anwendungsfunktion während eines Einsatzzeitraumes zu erbringen

Kenngrößen

- Überlebenswahrscheinlichkeit
- Verfügbarkeit

Zuverlässigkeit...

Überlebenswahrscheinlichkeit

- Wahrscheinlichkeit R(t) dafür, daß das System im Zeitintervall [0;t] fehlerfrei bleibt, wenn es in t=0 intakt war

  [System ohne Reparatur]

Verfügbarkeit

- Wahrscheinlichkeit A(t) dafür, daß das System zum Zeitpunkt t intakt ist

  [System mit Reparatur]
- A(t)=MTBF / MTBF+MTTR

  MTBF: mean time between failures

  MTTR: mean time to repair

Siehe:
- http://en.wikipedia.org/wiki/Reliability_%28engineering%29
- http://en.wikipedia.org/wiki/Availability

Beachten Sie, daß auch bei 99% Verfügbarkeit ein System untauglich sein kann, wenn die MTBF nicht viel höher als die durchschnittliche Joblänge ist. Die hohe Verfügbarkeit kann von einer sehr kleinen MTTR herrühren.

Wichtige Kennzahlen der Verfügbarkeit

Man spricht z.B. von den fünf Neunen, gemeint ist 0,99999 % Verfügbarkeit

Was bedeutet das als Ausfallzeit im Jahr?

Verfügbarkeit	Ausfallzeit pro Jahr
0,9	876 Stunden
0,99	87 Stunden
0,999	9 Stunden
0,9999	52 Minuten
0,99999	5 Minuten

Unzulänglichkeiten

Beeinträchtigung der Zuverlässigkeit

- Fehler

  Unerwünschter (nicht die Spezifikation erfüllender) Zustand des Systems

  (unterscheide noch Fehlerursache/-zustand)

- Ausfall

  Ist das Ereignis, daß eine benötigte Funktion nicht erbracht wird

Fehler

Einteilung nach ihrem Entstehen im Lebenszyklus des Systems

- Entwurfsfehler
- Herstellungsfehler
- Betriebsfehler

Einteilung nach der Zeitdauer

- Permanent
- Transient
- Intermittierend (pseudotransient)

Fehler...

Entwurfsfehler

Vor Inbetriebnahme des Systems

- Spezifikationsfehler
- Implementierungsfehler
- Dokumentierungsfehler

Herstellungsfehler

System entspricht nicht der entworfenen Implementierung

- Z.B. fehlerhafte Materialen oder Werkzeuge

Fehler...

Betriebsfehler

In der Nutzungsphase nach Inbetriebnahme

- Zufällige physikalische Fehler
- Verschleißfehler
- Störungsbedingte Fehler
- Bedienungsfehler
- Wartungsfehler
- Absichtliche Fehler (z.B. Sabotage)

Softwarefehler meist Entwurfs- oder Herstellungsfehler

Fehlermodell

Aufgrund der Vielzahl möglicher Fehler:

- Einschränkung auf bestimmte Gruppen
- Fehlermodell nennt betroffene Subsysteme und beschreibt Fehlverhalten

Z.B. Fehlermodell für Rechner-Cluster

- Nur permanente Fehler in Knoten, Leitung, Switches. Zusätzlich evtl. Prozessoren, Speichermodule, Festplatten
- Fehlerzustände: intakt, defekt
- Systemzustand dargestellt durch Vektor

Mittel zur Realisierung von Zuverlässigkeit

Ergänzen sich gegenseitig:

- Fehlervermeidung
- Fehlertoleranz

Fehlervermeidung (Fehlerintoleranz)

- Sorgfältige Konstruktion
- Ausführliche Tests

Nicht vermeidbare Fehler versuchen zu tolerieren

Fehlertoleranz

Einsatz von Redundanz, um im Fehlerfall weiterarbeiten zu können

- Redundanz
  - HW-Redundanz (zusätzliche Bauteile)
  - SW-Redundanz (zusätzliche Programme)
  - Zeit-Redundanz (zusätzlicher Zeitaufwand)
- Aktivierung der Redundanz
  - Statische Redundanz (funktionsbeteiligte Redundanz)
  - Dynamische Redundanz (Reserveredundanz)

Siehe: http://en.wikipedia.org/wiki/Redundancy_%28engineering%29

Fehlertoleranz...

Statische Redundanz

- Ressourcen ständig in Betrieb
- Im Fehlerfall direkter Zugriff auf diese Einheiten
- Z.B. Hamming Codes / Triple Modular Redundancy

Dynamische Redundanz

- Aktivierung erst im Fehlerfall
- Bis zu diesem Zeitpunkt:
  - Ungenutzte Redundanz (Standby-Systeme)
  - Fremdgenutzte Redundanz (mit Verdrängung)
  - Gegenseitig nutzbare Redundanz (Fail-Soft-Systeme)

Fehlertoleranztechniken

Tolerierung verschiedener Fehlerarten

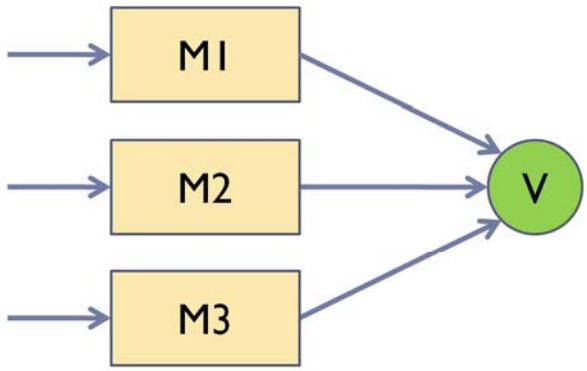
- HW-Fehler (siehe folgende Folien)
- SW-Fehler
  N-Versionen-Programmierung (gut für Mehrprozessorsysteme)

Betrachtungsebene

- Hier: Prozessoren, Speicher, Verbindungen
- Tiefere Ebenen auch möglich: z.B. fehlerkorrigierende Codes

Statische Redundanz

Wichtigste Technik: Triple Modular Redundancy (TMR)



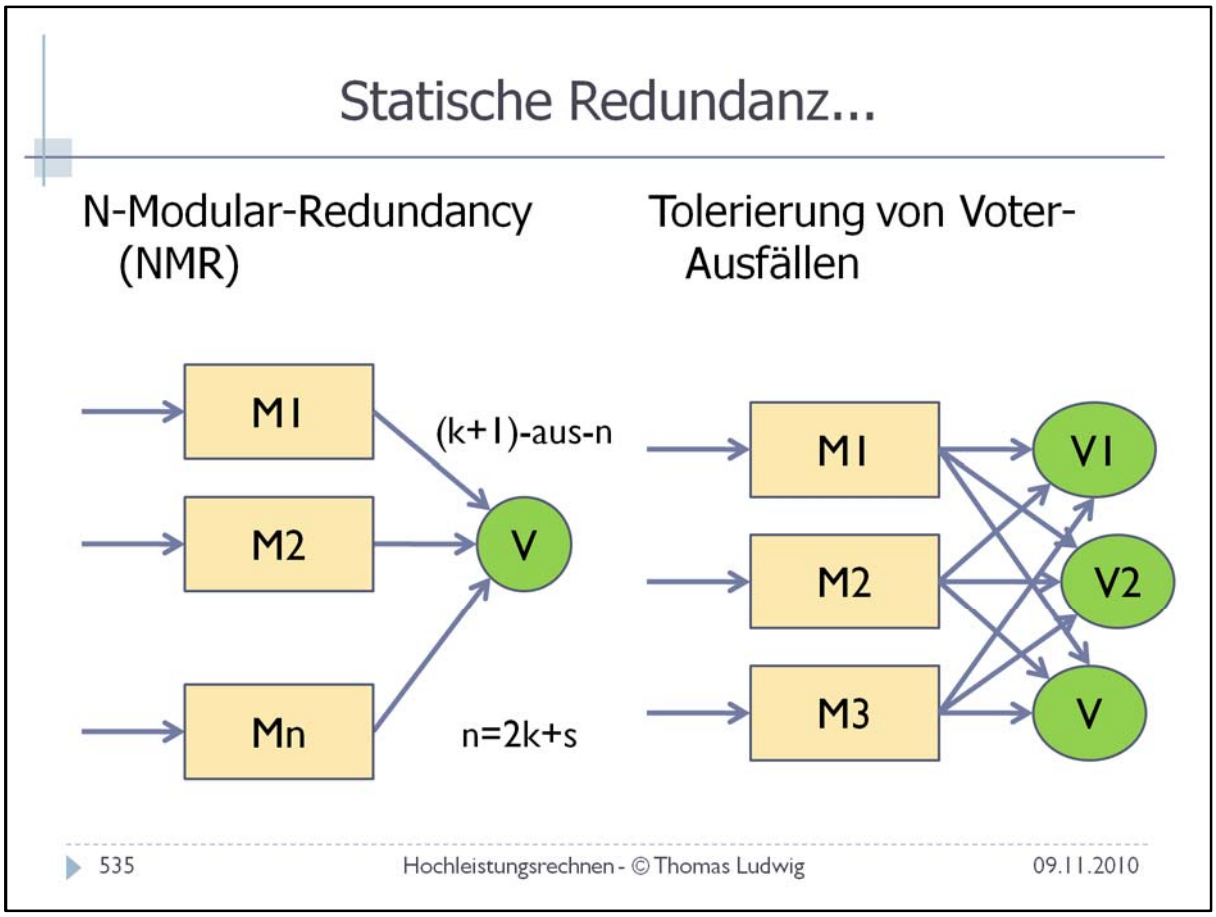
3 identische Module
1 Voter

Siehe: http://en.wikipedia.org/wiki/Triple_modular_redundancy

Statische Redundanz...

Konzept TMR

- Drei identische Module führen nebenläufig gleiche Funktion aus (kann HW oder SW sein)
- Voter fällt 2-aus-3-Mehrheitsentscheid
- Erster Ausfall kann toleriert werden
- Zweiter Ausfall kann noch erkannt werden
- Voter kann per Hardware oder Software realisiert werden


Statische Redundanz...
N-Modular-Redundancy (NMR)
Tolerierung von Voter-Ausfällen
M1
M2
Mn
(k+1)-aus-n
V
n=2k+s
M1
M2
M3
V1
V2
V
535
Hochleistungsrechnen - © Thomas Ludwig
09.11.2010

BILD

Statische Redundanz...

Vorteile TMR/NMR

- Toleriert permanente und transiente Fehler
- Fehlerbehebung kostet keine Zeit

Nachteile

- Hoher Aufwand in HW oder SW begrenzt Einsatzbereich

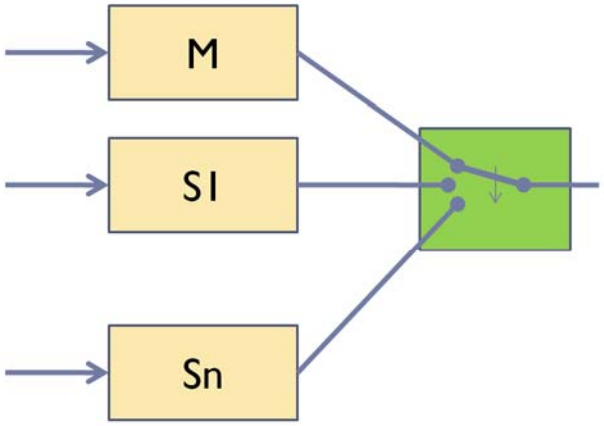
Dynamische Redundanz

Standby-Systeme

Zusätzlich zu den Systemkomponenten noch Reservekomponenten

Nachteile:

- Kein verzögerungsfreies Umschalten
- Ungenutzte Ressourcen (cold/hot standby)



Dynamische Redundanz...

Fail-Soft-Systeme (Graceful Degradation)

- Mehrfach vorhandene Subsysteme werden als Redundanzen genutzt (Eigenredundanz)
- Defekte Systemkomponenten werden deaktiviert
- Mit verminderter Leistung kann die Aufgabe weiterbearbeitet werden

- Ausgangsbedingungen und Verfahren durch ein Rechner-Cluster optimal abgedeckt

Dynamische Redundanz...

Vorgehensweise bei beiden Verfahren

- Fehlererkennung und –lokalisierung (Fehlerdiagnose)
- Rekonfiguration
- Fehlerbehandlung (Recovery)

Dynamische Redundanz...

(1) Fehlerdiagnose

- Aufgabe: Erkennen defekter Knoten
- Ebene: Komponenten- und Systemebene
- SRU (Smallest Replacable Unit)

  Ebene der Rekonfiguration

  Keine Lokalisation auf Komponentenebene
- Fehlererkennung mit Einprozessormethoden

  Fehlererkennende Codes, Selbsttestprogramme
- Zur Rekonfiguration noch Diagnose auf Systemebene

Dynamische Redundanz...

- Systemdiagnose
  - Fremddiagnose
    - Separater Prozessor für Diagnose (und i.a. auch für Rekonfiguration und Recovery)
    - Nachteile
      - Diagnose- und Wartungsprozessor als perfekt zuverlässig angenommen
      - Zusätzliche Prozessortypen (Heterogenität)
  - Eigendiagnose
    - Zentral: Testrunde in bestimmten zeitlichen Abständen
      Koordinatoreinheit ermittelt defekte Komponenten
      Problem: Wahl des Koordinatorknotens
    - Dezentral: Alle intakten Einheiten haben Diagnosebild
      Problem: Konsistenz der Einzelbilder

Dynamische Redundanz...

(2) Rekonfiguration

- Bildet aus den intakten Einheiten ein lauffähiges System
- Ausführung
  - Externer Wartungsprozessor
  - Zentraler Koordinator (unkritisch)
  - Dezentral
- Schwierigkeit
  - Suche einer neuen effizienten Abbildung der SW-Komponenten auf die HW-Komponenten

Dynamische Redundanz...

(3) Fehlerbehebung

- Versetzt das System in einen korrekten Zustand
- Setzt die Anwendung mit korrekten Daten wieder auf (Wiederanlauf)
- Methoden
  - Rückwärtsfehlerbehebung
  - Vorwärtsfehlerbehebung

Dynamische Redundanz...

- Rückwärtsfehlerbehebung
  - System zurücksetzen in früheren konsistenten Zustand (*rollback*)
  - Rücksetzpunkte (*recovery points*)
  - Abspeicherung in Haupt- oder Hintergrundspeicher
  - Zeitintervalle durch analytische Methoden
- Vorwärtsfehlerbehebung
  - Zusätzliche Operationen bringen das System in einen korrekten Zustand
  - Problem: Genaue Kenntnis der Anwendung notwendig

Dynamische Redundanz...

Bewertung

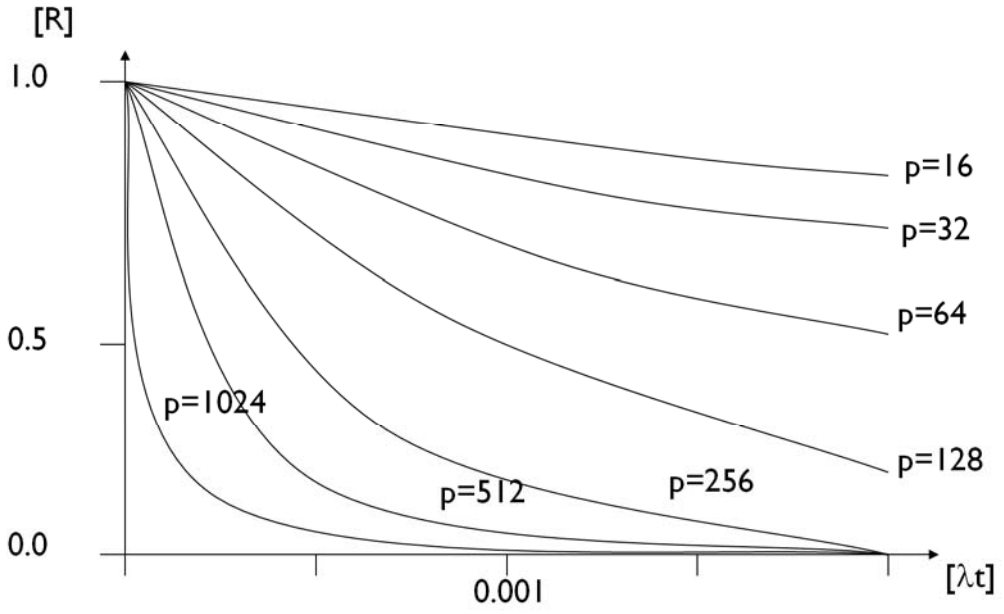
- Bei dynamischer Fehlertoleranz kein unterbrechungsfreier Betrieb
  - Für „normale“ Anwendungen akzeptabel, nicht jedoch bei Echtzeitanwendungen
- Je nach Variante können aber alle Komponenten produktiv genutzt werden

Quantitative Bewertung

Überlebenswahrscheinlichkeit R(t)
Exponentiell verteilte Lebensdauer $R(t)=e^{-\lambda t}$
MTTF (mean time to failure) = $1/\lambda$

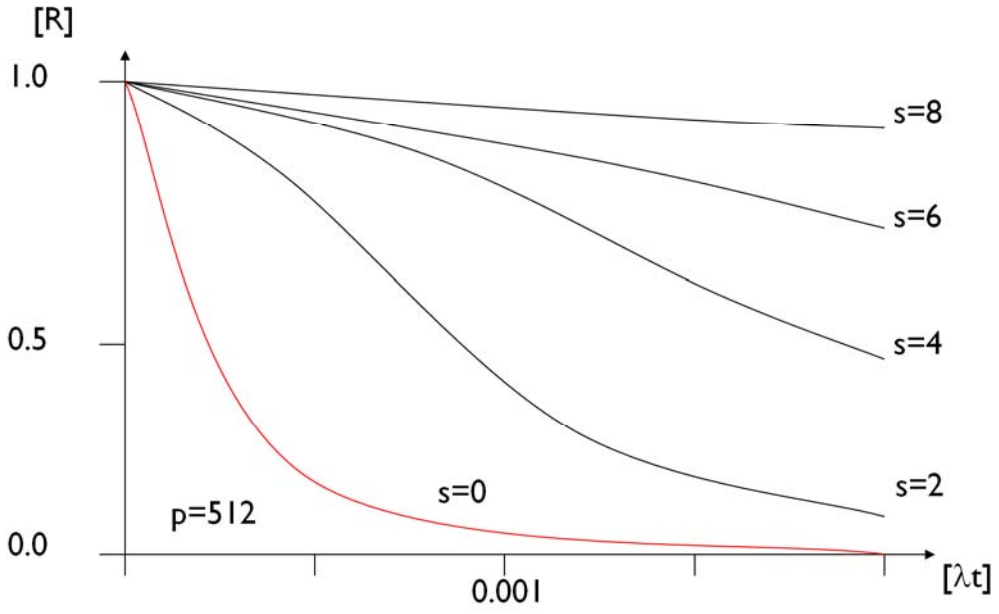
Zunächst das System ohne Fehlertoleranz:
$R(t)p=e^{-p\lambda t}$
Anzahl p PMU´s (*processor memory unit*)

Quantitative Bewertung...



547 Hochleistungsrechnen - © Thomas Ludwig 09.11.2010

Quantitative Bewertung...



Quantitative Bewertung...

System kann bis zu s ausgefallene PMU´s tolerieren:

$R_{(p-s) \text{ aus } p}(t) = \sum_{i=0} c^i \, (1-R(t))^i R(t)^{p-i}$

c=P(Fehler wird behandelt | Fehler tritt auf)

(*coverage factor* - Überdeckungsfaktor)

Systeme in der Praxis

- Parallelrechner und Hochleistungsrechnen
  - Nahezu keine Realisierungen
  - Gründe
    - Systeme zu aufwendig
    - Nutzen zu gering, da Schäden zu gering
- Parallelrechner und kommerzielle Anwendungen
  - High-Availability-Computing
    - Hochverfügbare Systeme
    - Normalerweise keine parallelen Programme
    - Bereich Datenbanken, Systemsteuerungen u.ä.

Systeme in der Praxis...

- High-Availability (HA) Linux Project
  - Definition von Verfahren, um Parallelrechner ausfallsicher zu machen
  - Cluster hier: Menge von HW-Komponenten, die als wechselseitige Redundanz dienen
- Linux High Availability HOWTO
- Viele kommerzielle HA-Cluster

Siehe: http://www.linux-ha.org/

Fehlertoleranz
Zusammenfassung

- Fehlertoleranz ist ein wichtiges Thema bei allen Systemen, die aus sehr vielen Einzelkomponenten bestehen
- Wir unterscheiden im allgemeinen Systeme mit und ohne Reparatur
- Wir unterscheiden verschiedene Klassen von Fehlern
- Ein Fehlermodell beschreibt das System analytisch
- Redundanz ist notwendig, um Fehler tolerierbar zu machen
- Wir unterscheiden statische und dynamische Redundanzverfahren
- Statische Redundanz: Triple-Modular-Redundancy-Verfahren
- Dynamische Redundanz: Fail-Soft-Verfahren
- In der Praxis Anwendungen nur im Bereich des sogenannten High-Availability-Computing

Lastausgleich

- Einführung und Problemstellung
- Die grundsätzliche Lösung
- Interessante Fragestellungen
- Die Lastbewertung
- Die Lastverschiebung

Allgemeine Betrachtung:
http://en.wikipedia.org/wiki/Load_balancing_%28computing%29

Lastausgleich
Die acht wichtigsten Fragen

- Was ist Lastungleichheit
- Was charakterisiert das Problem?
- Wie sieht die grundsätzliche Lösung aus?
- Welche interessanten Fragestellungen gibt es?
- Wie kann man Lastausgleich integrieren?
- Wie werden Lasten bewertet?
- Wie werden Lasten verschoben?
- Wie arbeiten Systeme mit Lastausgleich?

Was ist Lastausgleich?

Besser sollte man zuerst fragen:

Was ist Lastungleichheit?

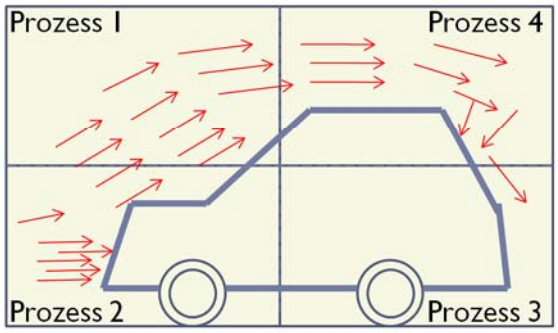
- Kritische Situation: die parallele Anwendung nutzt nicht alle Ressourcen zu jeder Zeit

  Folge: Speedup-Kurve bekommt Knick
- Grund: Rechenlast ist unausgeglichen

  Möglicherweise zu Programmstart ausgeglichen, dynamischer Effekt zur Laufzeit
- Folgen: längere Programmlaufzeit, geringere Effizienz der Parallelisierung

Beispiel für Lastungleichheit

- Gleichverteilung der Volumen zu Beginn:
  - alles ausgeglichen
- Danach Verlagerung der Teilchen:
  - Lastungleichheit zwischen den Volumenteilen



Warum ist das interessant?

Praktische Gründe

- Reduziere Programmlaufzeit
- Erhöhe Effizienz der Rechnernutzung

Forschungsgegenstände

- NP-harte Optimierungsprobleme
  Globales Scheduling (was wird wann wo berechnet?)
- Anspruchsvolle Fragestellungen aus dem Bereich der Betriebssystemtechnik

Das Problem

Lastungleichheit variiert dynamisch

- Typisches Verhalten vieler paralleler und verteilter Anwendungen
- Statischer Ausgleich (bei Programmstart) nicht möglich, da Lastungleichheit daten- und zeitabhängig
- Lastungleichheit verringert den Systemdurchsatz

Ressourcen von Interesse

- Hauptsächlich der Prozessor
- Seltener: Speicher, Netzwerk, Platten

Die grundsätzliche Lösung

Regelkreis der Lastverwaltung

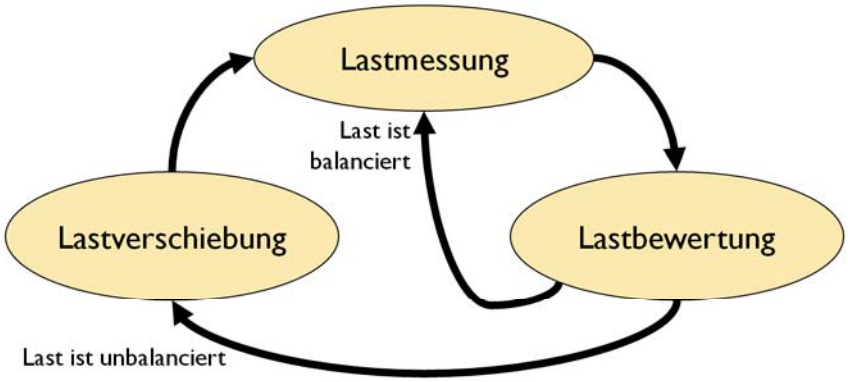


BILD verbessern

Die grundsätzliche Lösung...

Es gelten die üblichen Regelkreischarakteristika

- Schnelle Reaktion
- Genaues Nachregeln
- Keine Überreaktion
- Kein Oszillation
- Keine Belastung des geregelten Systems

Technische Umsetzung der Anforderungen sehr komplex

Teilweise sich widersprechende Anforderungen

Interessensbereiche

- Integrationstyp
  - Anwendungs- oder Systemintegration
- Lastmessung
  - Schnell, genau, umfassend, beeinflussungsarm
- Lastbewertung
  - Schnell, korrekt
  - Die Zukunft aus der Vergangenheit vorhersagen
- Lastverschiebung
  - Muss mehr bringen als kosten

Integrationstyp

Anwendungsintegriert

- In den Code der Anwendung integriert
- Überschaubare Implementierungskomplexität
- Wiederholter Implementierungsaufwand
- Meist nicht für andere Programme direkt übernehmbar
- Optimal an eine Anwendung angepasst
- Arbeitet mit allen Betriebssystemen zusammen (auch mit lastbalancierten)

Integrationstyp...

Systemintegriert

- In das Betriebssystem integriert oder mit ihm eng verbunden
- Hohe Implementierungskomplexität
- Einmaliger Implementierungsaufwand
- Optimal an das System angepasst, aber an keine spezifische Anwendung
- Arbeitet mit allen Anwendungen zusammen (auch mit lastbalancierten)
- Bei uns untersucht für Eingabe/Ausgabe

Lastmessung

Dynamisches Messen verschiedener Kenndaten

- Zunächst meist nur knotenbezogen
- Zur Verfeinerung auch prozessbezogen
- Meist sehr beeinflussungsarm

Mechanismen

- Online-Werkzeuge wie z.B. Dyninst/Paradyn

Lastbewertung

Einfach

- Einzelwerte: Prozessorleerlaufzeit, Speichernutzung
- Bei uns jetzt auch: Nutzung des E/A-Systems

Komplex

- Systemkenndaten, Anwendungskenndaten
- Auswertung mit neuronalen Netzen
- Global konsistenter Blick
- Metawissen (z.B. vergangene Entscheidungen)

Problem: alle Daten beschreiben Vergangenheit

Lastbewertung...

Konkretes System

- Zentrale Bewertungskomponente
- Erfasse alle knotenbezogenen Daten
- Bestimme einen überlasteten und einen unterlasteten Knoten
- Messe prozessbezogene Daten auf dem überlasteten Knoten
- Identifiziere geeigneten Kandidaten zur Verschiebung

Lastbewertung...

Probleme

- Zentrale Komponente in größeren Systemen nicht mehr praktikabel
- Daten müssen über ein Intervall erfasst werden, das nicht zu kurz sein darf
- Gleichzeitig aber schnelle Reaktion erwünscht
- Situationsabhängige Verfeinerung der Messungen
- Stabil gegen Oszillationen

Lastverschiebung

Verschiebeobjekte

- Einfacher: Anwendungsebene (feingranular)
  Datenpakete, Anfrage
- Komplex: Systemebene (grobgranular)
  Prozesse, Objekte, Dateien

Allgemeine Probleme

- Bezug zu anderen Objekten muss gesichert sein
- Verschiebung muss Gewinn bringen

Lastverschiebung...

Beispiel: Prozessverschiebung

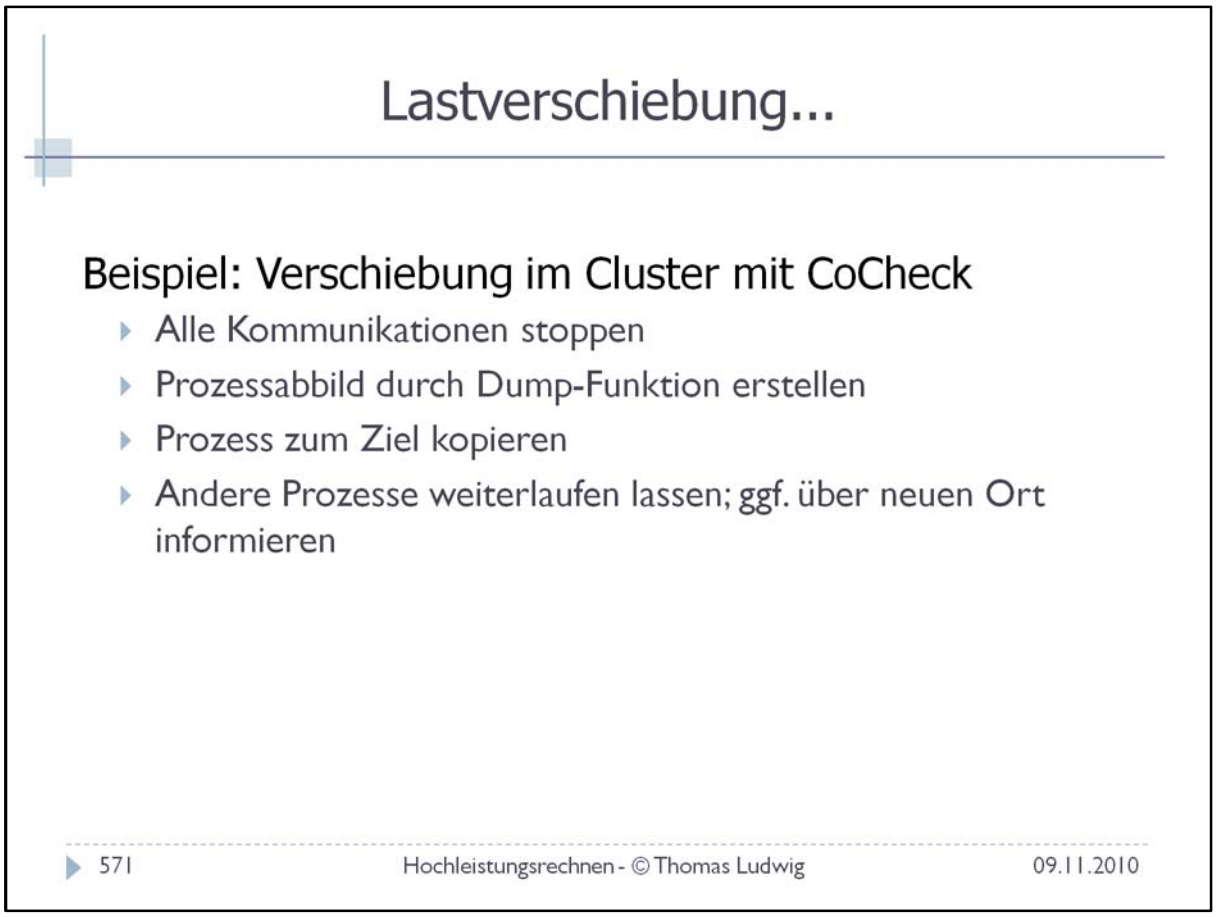
- Wie stoppt man einen laufenden Prozess?
- Wie verschiebt man ihn?
- Was verschiebt man konkret?
- Wie behandelt man offene Dateien?
- Wie behandelt man Signale?
- Was geschieht während der Verschiebung?
- Was passiert mit Nachrichten nach der Verschiebung?
- Wie finden sich die Kommunikationspartner nach der Verschiebung wieder?

Prozessmigration z.B. mit OpenMOSIX
Siehe: http://en.wikipedia.org/wiki/OpenMosix

Lastverschiebung...

Beispiel: Intel Hypercube Parallelrechner

- Verschiebung mittels Paging-Mechanismus über das Netzwerk
- Nur aktive Code-Seite verlagern; restliche Seiten werden durch auftretende Seitenfehler geholt
- Nachrichten an den Prozess bei den Sendern zwischengespeichert
- Quellknoten der Verschiebung gibt neuen Ort an Sender bekannt; diese korrigieren ihre Tabellen



CoCheck – Consistent Checkpointing System

Lastverschiebung...

Beispielproblem: Offene Dateien

- Mitprotokollieren ***aller*** Systemaufrufe durch zwischengeschaltete Bibliothek (wie MPI-Profiling-Bibliothek)
- Damit weiß man immer, wo der Prozess in einer Datei gerade liest
- Nach der Verschiebung Dateien wieder öffnen und die vermerkte Stelle wieder anfahren

Zur Lage der Werkzeuge

Benötigte Zusatzwerkzeuge

- Visualisierung der Messungen und Entscheidungen des Lastausgleichers

Situation der aktuellen Werkzeuge

- Keines kann mit Prozessmigration umgehen
- Beispielproblem: Verschiebung eines Prozesses, für den ein Haltepunkt definiert ist
- Beispielproblem: Verschiebung eines Prozesses, dessen Code dynamisch instrumentiert ist

Und weil das ganze so schwer ist...

... gibt es kaum Systeme mit dynamischem Lastausgleich

- Aber es gibt Tonnen von Literatur, bei der Simulationen des Verhaltens eines solchen Systems analysiert werden
- Und es gibt die initiale Lastplazierung, die meist banal ist, aber auch als Lastausgleich verkauft wird

Aber es gibt viele Anwendungen, bei denen die Programmierer mühevoll einen Lastausgleich eingebaut haben

Lastausgleich
Zusammenfassung

- Lastungleichheit ist die ungenügende Nutzung wichtiger Ressourcen
- Lastungleichheit variiert dynamisch und lässt sich deshalb nicht durch einen statischen Ansatz kontrollieren
- Die Lastverwaltung folgt dem Prinzip des Regelkreises
- Die Lastverwaltung kann in die Anwendung oder in das System integriert werden
- Eine effiziente Lastbewertung unterliegt vielen Einzelfragestellungen
- Die Lastverschiebung ist die technisch anspruchsvollste Komponente
- Aufgrund der Komplexität gibt es kaum Realisierungen auf Betriebssystemebene
- Häufig wird aber Lastausgleich in die Anwendung eingebaut

Grid- und Cloud-Computing

- Motivation für Grid-Computing
- Grid-Computing
- Anwendungsbereiche
- Benutzung / Dienste
- Projekte

- Motivation für Cloud-Computing
- Varianten
- Cloud-Computing und HPC

Grid- und Cloud-Computing
Die zehn wichtigsten Fragen

- Welche prinzipiellen Probleme gibt es bei der Nutzung von Clustern und Parallelrechnern?
- Wie faßt man solche Architekturen zu größeren Einheiten zusammen?
- Was bezeichnete man als Internetcomputing und Metacomputing?
- Was ist die Grundidee des Grid-Computing?
- Welche Fragestellungen finden wir hier?
- Welche Klassen von Anwendungen gibt es hier?
- Was ist die Grundidee des Cloud-Computing?
- Welche Dienste werden angeboten?
- Welche Vor- und Nachteile erkennen wir hier?
- Wie tauglich ist Cloud-Computing für HPC?

Motivation für Grid-Computing

Ausgangssituation

- Viele Cluster und Parallelrechner in verteilten Rechenzentren in der ganzen Welt
- Viel Knowhow in unterschiedlichen Gruppen verteilt

Aber:

- Ressourcen sind oft nicht da, wo die Benutzer sind
- Das Wissen ist nicht da, wo der Benutzer ist

Motivation für Grid-Computing...

Situation bei Clustern und Parallelrechnern

- Verschiedene Hardware-Architekturen
- Verschiedene Hersteller
- Verschiedene Betriebssysteme und Basis-SW
- Verschiedene Benutzungsmodelle (Stapelbetrieb, interaktiv)
- Verschiedene Formen der Systemverwaltung
- Verschiedene Formen der Datenverwaltung
- Verschiedene Sicherheitsmechanismen

Motivation für Grid-Computing...

Konsequenzen für den Benutzer

- Arbeitet nur mit vertrautem Rechner
- Benutzt nur Rechner mit passenden Ressourcen
- Vermeidet neue Rechner

Konsequenzen für den Systembetreiber

- Teure Ressourcen nicht optimal genutzt
- Probleme nicht optimal gelöst
- Lösbare Probleme zum Teil gar nicht gelöst

Motivation für Grid-Computing...

Neue Benutzungsmethodik

- Wir betrachten das Cluster, den Parallelrechner oder das Rechenzentrum als „einzelnen“ Rechner
- Zusammenfassen mehrerer solcher „Rechner“ zu einem neuen „Rechnersystem“
- Programme werden auf diesem „Rechnersystem“ ausgeführt

„Rechnersystem“

- Früher vernetzte Einzelrechner
- Heute vernetzte Hochleistungsrechner

Motivation für Grid-Computing...

Nutzungsmethode der neuen „Rechnersysteme" in Analogie zu früher

- Zuweisung eines Jobs zu einer freien Ressource
  Einzeljob nicht parallel bzgl. neuen Rechnersystems
  Bis ca. 1999 genannt: Internetcomputing
  Frühes Beispiel: seti@home
- Parallelisierung eines Jobs über mehrere Ressourcen
  Paralleler Job bzgl. neuen Rechnersystems
  Bis ca. 1999 genannt: Metacomputing

Motivation für Grid-Computing...

Nutzungsmethode...

- Hardware- und Software-Konzepte
  Vernetzung, Sicherheit, Ein-/Ausgabe usw.
- Programmkonstruktion
  Individuelle Jobs für Einzelsysteme oder Verbundsysteme
  (aber intern alles parallelisiert)
- Nutzungsmethodik
  Stapelbetrieb und/oder interaktiver Betrieb

Motivation für Grid-Computing...

Alles schon bekannt, aber jetzt eine Abstraktionsstufe höher

- Ressourcenverwaltung, Internetcomputing

  Die einzelne Anwendung ist parallelisiert und läuft an einem auswählbaren Ort

  Anbindung über Web-Frontend
- Metacomputing

  Die Einzelanwendung ist parallelisiert für verschiedene parallele Umgebungen und läuft parallel an verschiedenen Orten

Neuer Begriff seit 1998: GRID-Computing

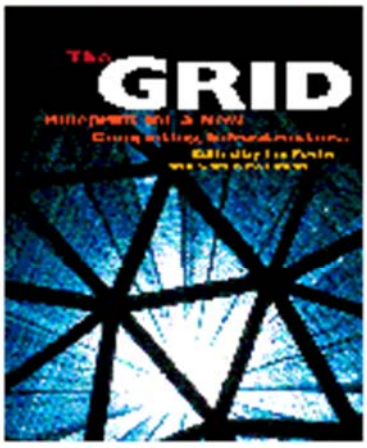
Eine wichtige Ressource

The Grid:
Blueprint for a New Computing Infrastructure

I. Foster, C. Kesselman (Eds), Morgan Kaufmann, 1999

- Available July 1998; ISBN 1-55860-475-8
- 22 chapters by expert authors including Andrew Chien, Jack Dongarra, Tom DeFanti, Andrew Grimshaw, Roch Guerin, Ken Kennedy, Paul Messina, Cliff Neuman, Jon Postel, Larry Smarr, Rick Stevens, and many others

"A source book for the history of the future" -- Vint Cerf



http://www.mkp.com/grids

Grid-Computing

Idee

„Rechenleistung an jedem Ort“

Analogie zum „*power grid*“, dem Stromnetz

Einspeisung an verschiedene Stellen, Nutzung an beliebigen anderen Stellen

- Begriff des „Grid“ mittlerweile sehr umfassend
- Kein Konsens über Anwendung und Konzepte

Grid-Computing...

Warum kommt das gerade jetzt auf?

- Zugriff auf Ressourcen erwünscht, die lokal nicht vorhanden sind; Replikation zu teuer
- Vernetzung immer leistungsfähiger und gleichzeitig immer billiger
- Erfahrung mit dem parallelen Rechnen brachte Erkenntnis zum Thema Verteilung von Programmen

Grid-Computing...

Fragestellungen

- Auswahl der Rechnerressourcen
- Auswahl der Parallelisierung
- Auswahl der Sprachkonzepte
- Sicherheitsaspekte
- Lastausgleich
- Fehlertoleranz

Insgesamt nichts neues!

Jetzt alles in großen Dimensionen betrachtet

Grid-Computing...

Die Ziele

- Weltweite Nutzung freier Rechenleistung
- Aggregation hoher Rechenleistung
- Einfacher Zugriff auf Rechenleistung
- Bewältigung großer Aufgabenstellungen
- Kollaboration zwischen Orten

Grid-Computing...

Voraussetzungen

- Hochgeschwindigkeitsvernetzung
  Gigabit-Vernetzung, ATM-Vernetzung
  Satellitenkopplung
- Geeignete Protokolle auf verschiedenen Ebenen

Problem

- Kommunikationslatenz
  Beim Grid-Computing deutlich höher, d.h. nur grobgranulare Parallelisierung gewinnbringend

Anwendungsbereiche

Vier wichtige Gebiete

- *Desktop Supercomputing*

  Zugang zu hoher Rechenleistung von überall aus
- *Smart Instruments*

  Verbindet Radioteleskope, Kernbeschleuniger, Tomographen usw. mit geeigneten Hochleistungsrechnern
- *Collaborative Environments*

  Verbindet Benutzungsumgebungen miteinander und mit Supercomputern zu Simulationsläufen usw.
- *Distributed Supercomputing*

  Erzielt Summe der Leistungen der Einzelrechner

Umgebungscharakteristika

- Wachsende Systemgrößen
  Jedoch meist nur noch Teilsysteme verwendet
- Heterogenität auf allen Ebenen
  HW, Betriebssysteme, Sprachen, Compiler usw.
- Wechselnde Strukturen
  Sowohl des Rechnersystems als auch des Programmsystems
- Dynamisches Verhalten
  Gemeinsam genutzte Ressourcen schwer kontrollierbar
- Verschiedene Administrationsprogramme
  Authentifizierung, Autorisierung usw.

Benutzung / Dienste

Das Grid ist nicht nur eine Ansammlung von Ressourcen sondern auch von Diensten

- Scheduling
- Ressourcen-Verwaltung
- Sicherungspunkt-Verwaltung
- Sicherheit und Abrechnung
- Weitere Dienste...

Benutzung / Dienste...

Scheduler

- Job-Scheduler
  - Optimierung auf Job-Durchsatz
- Ressourcen-Scheduler
  - Optimierung auf beste Ressourcen-Nutzung
- Anwendungs-Scheduler
  - Optimierung auf z.B. minimale Ausführungszeit

Fragestellungen

- Die üblichen:
  Was soll wann wo ausgeführt werden?

Benutzung / Dienste...

Im Grid heterogene Leistungs-Charakteristik

- Software, Hardware, Vernetzung
- Mitbenutzung durch andere Nutzer
- Keine zentrale Kontrolle über alle Ressourcen möglich
- Verfügbare Ressourcen sind dynamisch
- Leistung der Anwendung ≠ Leistung des Systems

Scheduling hier maximal schwer

- Nur heuristische Verfahren sinnvoll

Benutzung / Dienste...

Sicherheit und Abrechnung

- Bekannte Mechanismen auf höherer Ebene
  - Kerberos
  - Public-Key-Systeme
  - Zertifikate
  - Firewalls
- Zusätzliche Probleme
  - Sicherheitsvereinbarungen zwischen Organisationen
  - Verteilter Zugang zum System
  - Verteilte Abrechnung

Infrastrukturen

Wichtigste Infrastruktur

- Globus Toolkit (Argonne National Laboratory)

Globus-Alliance: www.globus.org

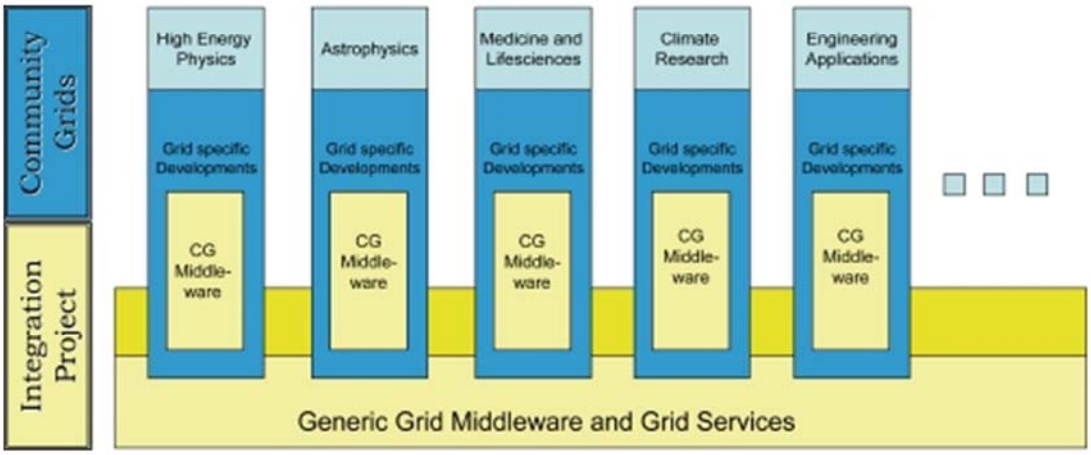
Das Konzept

- Globus stellt ein Toolkit zur Verfügung, das Basismechanismen bereitstellt (Kommunikation, Autorisierung usw.)
- Hierauf lassen sich höhere Dienste des Grid-Computing abstützen (Programmierwerkzeuge, Scheduler usw.)
- Langfristziel: AWARE (*Adaptive Wide Area Resource Environment*)

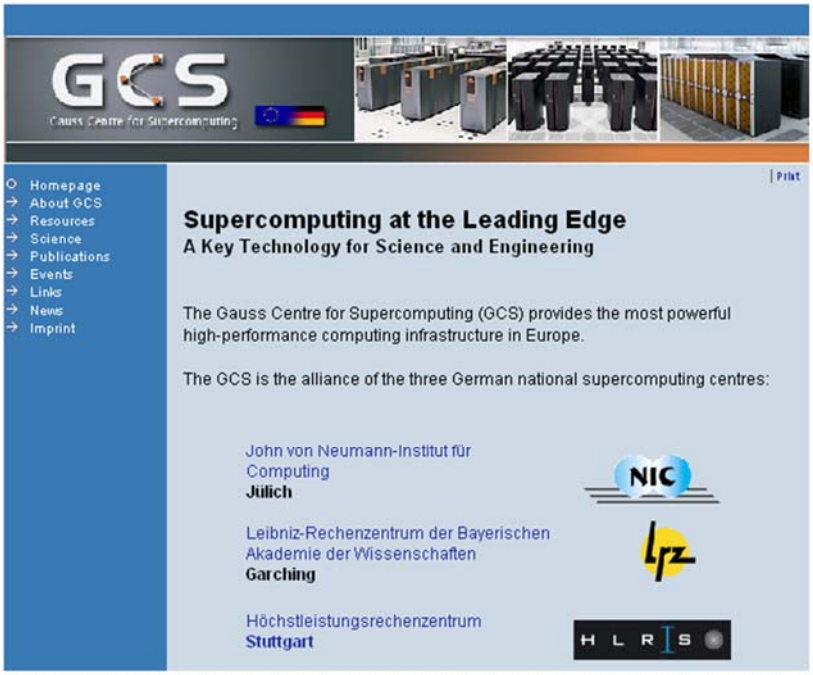
Deutsche Grid-Initiative (D-Grid)

Deutsche Grid-Initiative (D-Grid)…

Integration Project + Community Grids



Gauss Centre for Supercomputing



Siehe: http://www.prace-project.eu/

Nicht notwendigerweise Grid-Computing, aber vermutlich eng damit verbunden.

Motivation für Cloud-Computing

Weltweit existieren sehr viele Rechner- und Speichersysteme und auf ihnen viele Softwaresysteme

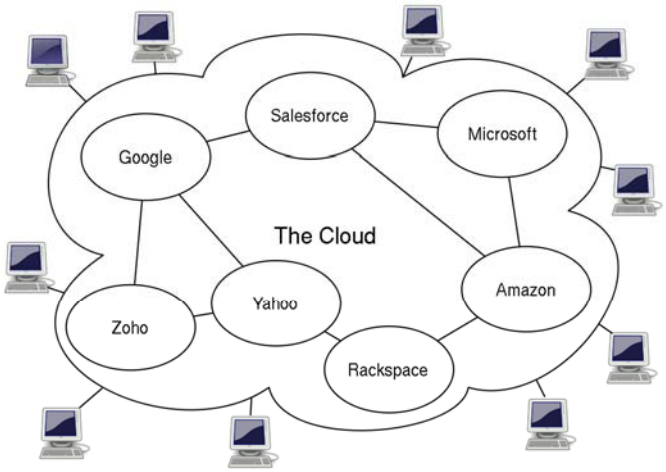
Nutzer an beliebigen Orten könnten vielleicht über eine Vernetzung auf diese Ressourcen zugreifen

Hätte für Anbieter und Nutzer einige Vorteile

Cloud-Computing

„Cloud-Computing" = Rechnen in der Wolke

Wolke? Abgeleitet vom Wolkensymbol für das Internet



603 Hochleistungsrechnen - © Thomas Ludwig 09.11.2010

Quelle: http://commons.wikimedia.org/wiki/File:Cloud_computing.svg

Charakter des Cloud-Computing

Konzepte und Systeme werden über das Netz zur Verfügung gestellt

Bezahlung je nach Nutzung

- Infrastructure as a Service (IaaS)
  - Rechenkapazität, Speicherkapazität bereitstellen
- Software as a Service (SaaS)
  - Nutzung fertiger Programmpakete
- Platform as a Service (PaaS)
  - Programmierumgebungen bereitstellen

Vorteile / Nachteile

Vorteile

- Aus Nutzersicht
  - Keine Investitionskosten, Wartungskosten usw.
  - Flexibler Einkauf von Diensten
  - Billige Abwicklung von Lastspitzen
- Aus Betreibersicht
  - Effizientes Anbieten von Diensten
  - Effizientes Verwalten der angebotenen Dienste

Nachteile

- Aus Nutzersicht
  - Datensicherheit
  - Anbieterbindung

Abgrenzungen

Grid-Computing

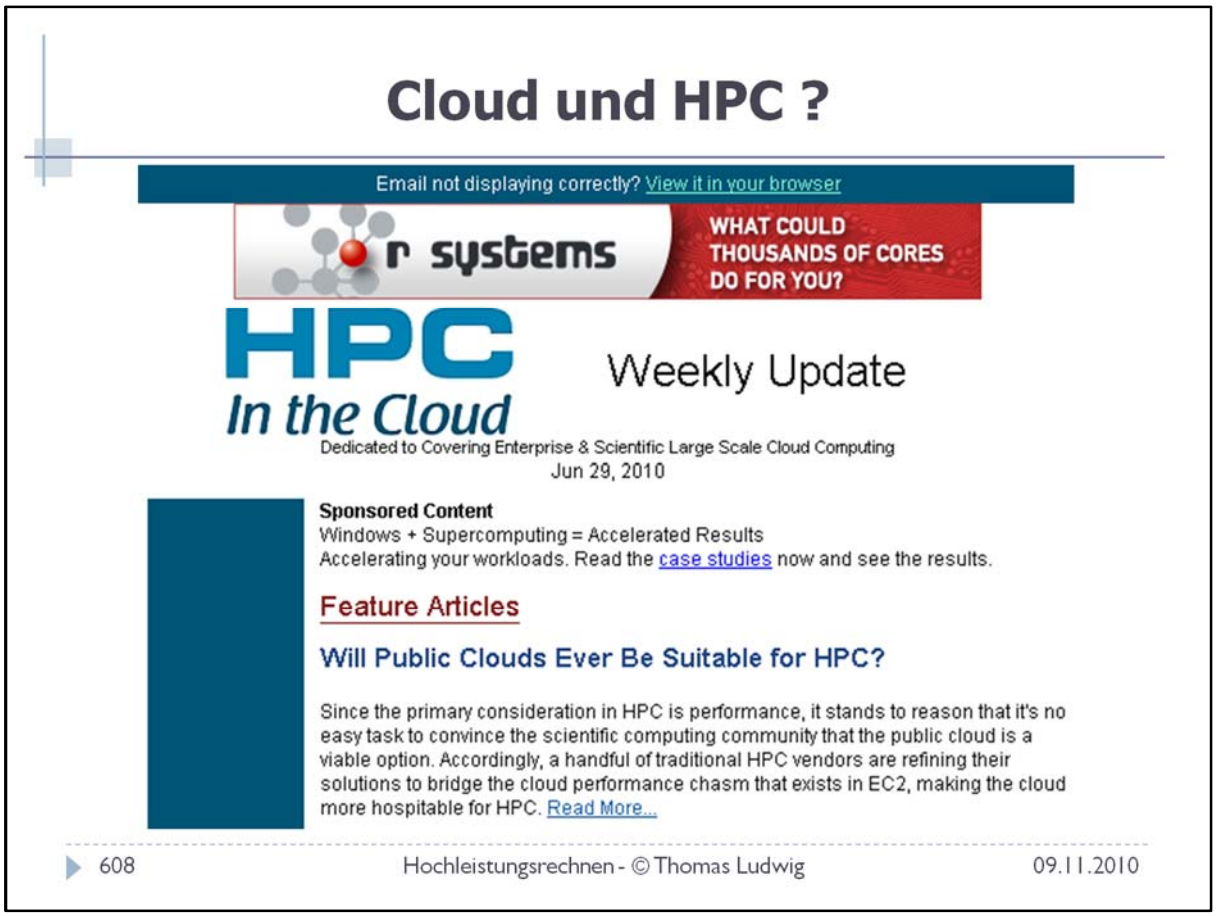
- Gemeinschaftliche Nutzung der gemeinsamen Ressourcen, meist im Bereich des Hochleistungsrechnens
  - Cloud hat wenige einzelne Anbieter und die sind zentral gesteuert

Peer-to-Peer-Computing

- P2P: Verteilung von Rechenlast auf viele Rechner
- Cloud: Verlagerung von Rechenlast auf andere Rechner

Cloud-Dienstanbieter

Cloud und HPC ?



Email not displaying correctly? View it in your browser

r systems

WHAT COULD THOUSANDS OF CORES DO FOR YOU?

HPC In the Cloud

Weekly Update

Dedicated to Covering Enterprise & Scientific Large Scale Cloud Computing

Jun 29, 2010

Sponsored Content

Windows + Supercomputing = Accelerated Results

Accelerating your workloads. Read the case studies now and see the results.

Feature Articles

Will Public Clouds Ever Be Suitable for HPC?

Since the primary consideration in HPC is performance, it stands to reason that it's no easy task to convince the scientific computing community that the public cloud is a viable option. Accordingly, a handful of traditional HPC vendors are refining their solutions to bridge the cloud performance chasm that exists in EC2, making the cloud more hospitable for HPC. Read More...

Grid- und Cloud-Computing
Zusammenfassung

- Die heute weltweit verfügbare Rechenleistung soll beim Grid-Computing an verschiedenen Orten nutzbar sein
- Cluster, Parallelrechner, Rechenzentren werde zu einem Grid zusammengefaßt
- Wir kennen verschiedene Nutzungsmethoden
- Die Fragestellungen bleiben dieselben, allerdings auf einem anderen Abstraktionsniveau
- Grid-Computing: Rechenleistung aus der Steckdose
- Globus ist eine Infrastruktur in Form einer Sammlung von Diensten
- Die großen Supercomputing-Zentren in Europa formieren sich zu einer Grid-Infrastruktur
- Cloud-Computing stellt verschiedene Dienste bereit
- Cloud-Computing zur Zeit (noch) nicht gut für HPC geeignet