# Verteilte Systeme 2012: Zusammenfassung



## 1 Systemarchitekturen nach Flynn

### 1.1 SISD (Von-Neumann-Architektur)

Single Instruction Single Data
ein Instruktionsstrom und ein Datenstrom
sequentielle Uniprozessorarchitektur, ggf. aber interne Parallelität durch Pipelining oder intelligente I/O-Kanäle

### 1.2 SIMD

Single Instruction Multiple Data
ein Instruktionsstrom, mehrere Datenströme
Beispiele: Vektorprozessoren, Grafikkarten
Ausblenden einzelner Prozessoren durch Tagging möglich ← Prozessoren können simultan verschiedene Operationen ausführen

### 1.3 MIMD

Multiple Instruction Multiple Data
Systeme mit

- gemeinsamem Speicher („shared memory“, SMP)
- verteiltem Speicher („distributed memory“, DMS)

Prozessoren werden über Verbindungsnetzwerk gekoppelt, jeder wird über einen unabhängigen Instruktionsstrom gesteuert
Prozessoren arbeiten autonom und haben über Verbindungsnetzwerk Zugriff auf die Daten der anderen Prozessoren

### 1.4 MISD

Multiple Instruction Single Data
Eher nicht verwendet, als Beispiel aber FPGA

## 2 SPMD - Single Program, Multiple Data

**nicht Teil der Flynn-Taxomonie**
Grundprinzip: Alle (MIMD-)Prozessoren arbeiten auf Kopien desselben Programmcodes, aber mit unterschiedlichen Daten und ggf. in unterschiedlichen Modulen.
Vorteile

- leichte Programmentwicklung, -debugging, -wartung
- leichtere Synchronisation als „echte“ MIMD-Programmierung
- gröbere Parallelitätsgranularität als SIMD

# 3 Verteilte Systeme allgemein

## 3.1 Definition

Menge miteinander verbundener, autonomer Computer, die dem Nutzer wie ein einzelnes kohärentes System erscheinen.

- „Computer“: Prozessoren/Prozesse
- „autonom“ : jeder Knoten hat private Kontrolle (kein SIMD)
- „miteinander verbunden“ : Informationsaustausch ist möglich

Typen:

- Computer in WANs - Internet, Intranet
- Computer im LAN - Hausnetz einer Universität
- kooperierende Prozesse/Threads - Prozesse und Threads auf einer Maschine

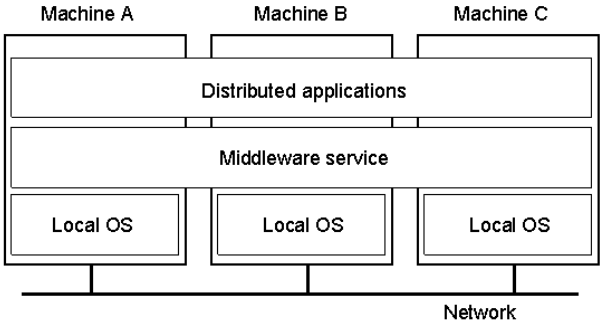
## 3.2 Vorteile/Motivation

- Informationsaustausch
- Zuverlässigkeit durch Replikation
- Ressourcensharing (Drucker, Festplattenspeicher, Rechenleistung)
- Leistungssteigerung durch Parallelisierung
- Vereinfachung des Systemdesigns durch Entkopplung/Spezialisierung

## 3.3 Anforderungen

- Transparenz - Verteilung bleibt dem Benutzer verborgen
- Offenheit - Austausch und Erweiterbarkeit (von Komponenten)
- Skalierbarkeit - gleich gute Leistung unabhängig von der Nutzeranzahl
  - Größe - mehr Nutzer und Ressourcen
  - geographische Verteilung
  - administrativ - über Organisationsgrenzen hinweg administrierbar

Realisierung von Transparenz durch Middleware:




# 4 Parallele Programmierung

## 4.1 Betriebssystemsicht

- Multicomputerbetriebssystem: erbringt die Systemdienste verteilt und transparent
  - präsentiert dem Nutzer ein kohärentes System
  - hat vollständige Kontrolle über Knoten und deren Ressourcen
  - kann keine heterogenen Systeme verwalten
- Netzwerk-Betriebssystem: erlaubt, Dienste entfernt zu nutzen, bietet aber keine Transparenz
  - verteilte Rechner mit autonomen Betriebssystemen und eigener Ressourcenverwaltung
  - eingebaute Netzwerkfunktionalität
  - skalierbar und offen

Middleware:

- bietet Abstraktionen für Netzwerkprogrammierung
- Event Handling und Filtering
- Auffinden von Ressourcen für mobiles Computing
- Unterstützen von Datenströmen