

2. Klassifikation von Mehrrechner-DBS

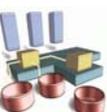
- Merkmale für PDBS/VDBS
 - *räumliche Verteilung*: ortsverteilt oder lokal
 - *Rechnerkopplung*: enge, lose oder nahe Kopplung
 - *Externspeicheranbindung*: partitioniert oder gemeinsam ('shared')
- PDBS-Architekturen
 - *Scale-Up vs. Scale-Out*
 - *Shared-Nothing vs. Shared-Disk*
- Weitere Klassifikationsmerkmale
 - funktionale Spezialisierung vs. funktionale Gleichstellung
 - integrierte vs. heterogene/föderierte MRDBS
- Systemansätze zur Datenintegration
- Grobbewertung von MRDBS-Alternativen



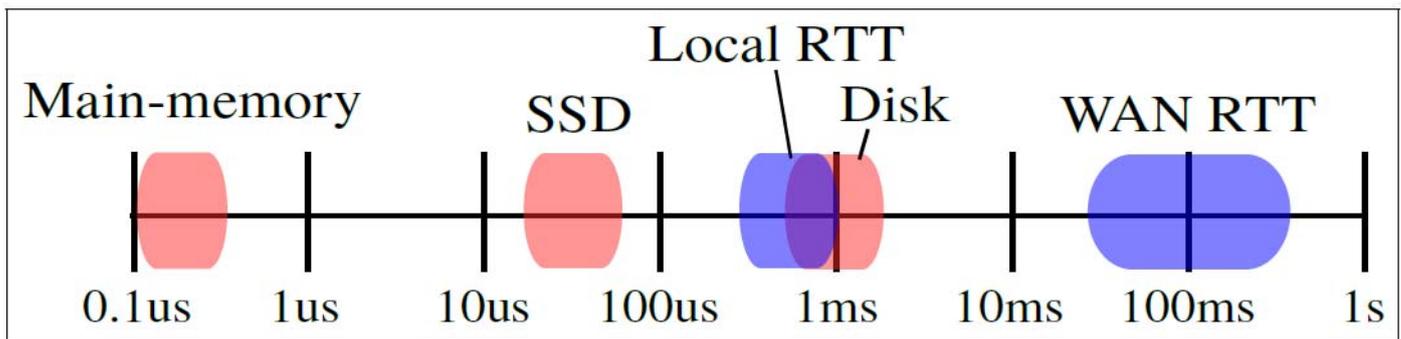
Räumliche Verteilung

- **ortsverteilt**:
 - unterstützt dezentrale Organisationsstrukturen
 - unterstützt Katastrophen-Recovery (replizierte DB an entfernten Knoten)
 - relativ langsame Kommunikation
 - Signallaufzeiten > 100 ms
 - aufwändige Protokolle (> 10.000 Instruktionen pro Send/Receive)
- **lokal**:
 - schnelle Rechnerkopplung (gemeinsame Speicher bzw. Hochgeschwindigkeitsnetz)
 - effektive dynamische Lastverteilung möglich
 - bessere Voraussetzungen für Intra-Query-Parallelität
 - einfachere Administration

	Durchmesser	Latenzzeit	Bandbreite (Mbit/s)		Übertragung 10 KB	
			1990	2010	1990	2010
Cluster	20 m	1 μ s	1000	10000	0,1 ms	0,01 ms
LAN	1 km	10 μ s	10	1000	10 ms	0,1 ms
WAN	10.000 km	100 ms	0,05	100	1.700 ms	101 ms



Vergleich Latenzzeiten



F. Nawab et al: A system infrastructure for strongly consistent transactions on globally-replicated data. IEEE Bull. TCDE, 2017

RTT: Round-trip time



Enge Rechnerkopplung (tightly coupled systems)

■ Eigenschaften

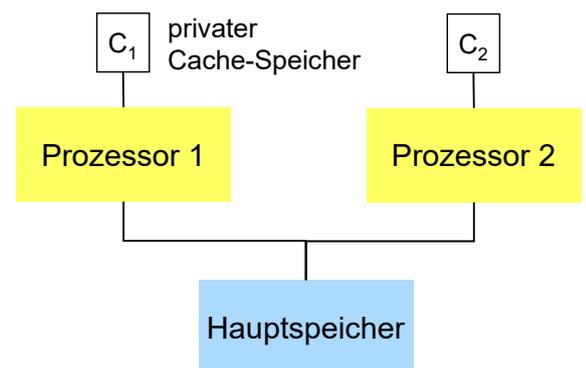
- gemeinsamer Hauptspeicher
- 1 Kopie von Software-Komponenten (Betriebssystem, DBMS)
- HW-Cache pro Prozessor

■ Vorteile

- weit verbreitet
- effiziente Kommunikation über Hauptspeicher
- Lastbalancierung durch Betriebssystem
- Single System Image

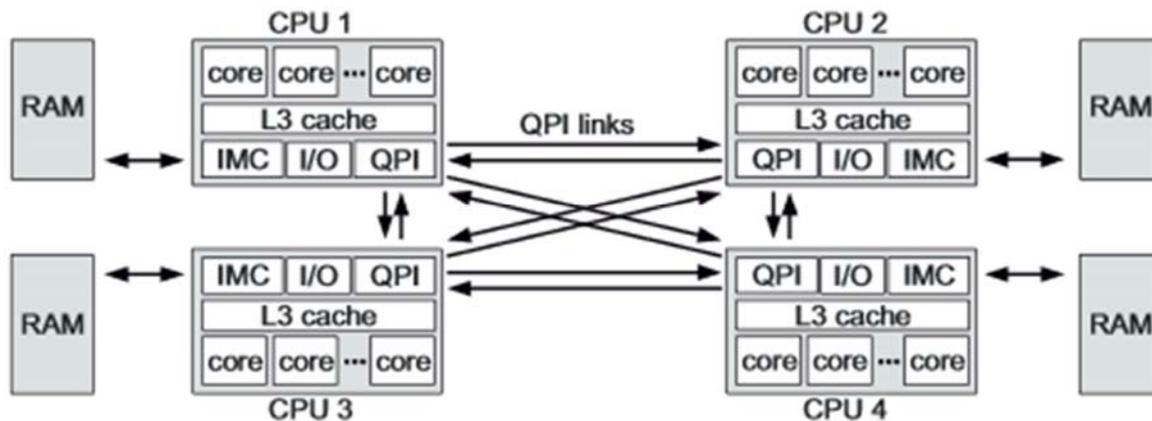
■ Nachteile

- mangelnde Fehlerisolation
- oft begrenzte Erweiterbarkeit und Skalierbarkeit
- Cache-Kohärenz



NUMA-Architekturen

- Non-Uniform Memory Access (NUMA) bei hoher Prozessorzahl
- Beispiel Intel



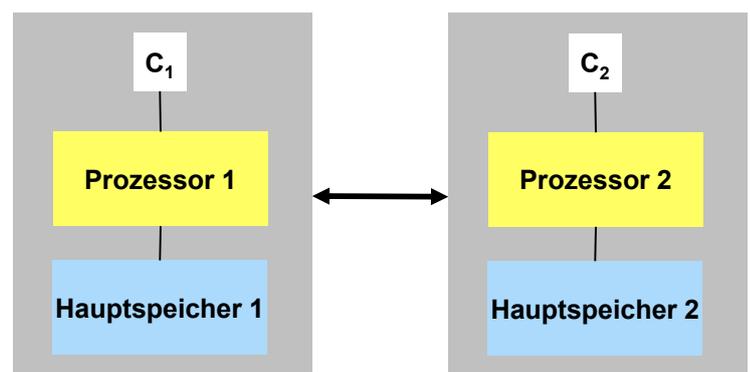
- Memory-Zugriff über Links wesentlich langsamer
 - Datenpartitionierung und -lokalität innerhalb eines NUMA-Servers verbessert Leistung für DB-Anwendungen



Lose Rechnerkopplung (loosely coupled systems)

■ Eigenschaften

- n selbständige Rechner (pro Knoten eigener Hauptspeicher, eigene Software-Kopien)
- Kommunikation über Nachrichtenaustausch



■ Vorteile:

- höhere Fehlerisolation/Verfügbarkeit
- bessere Erweiterbarkeit

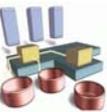
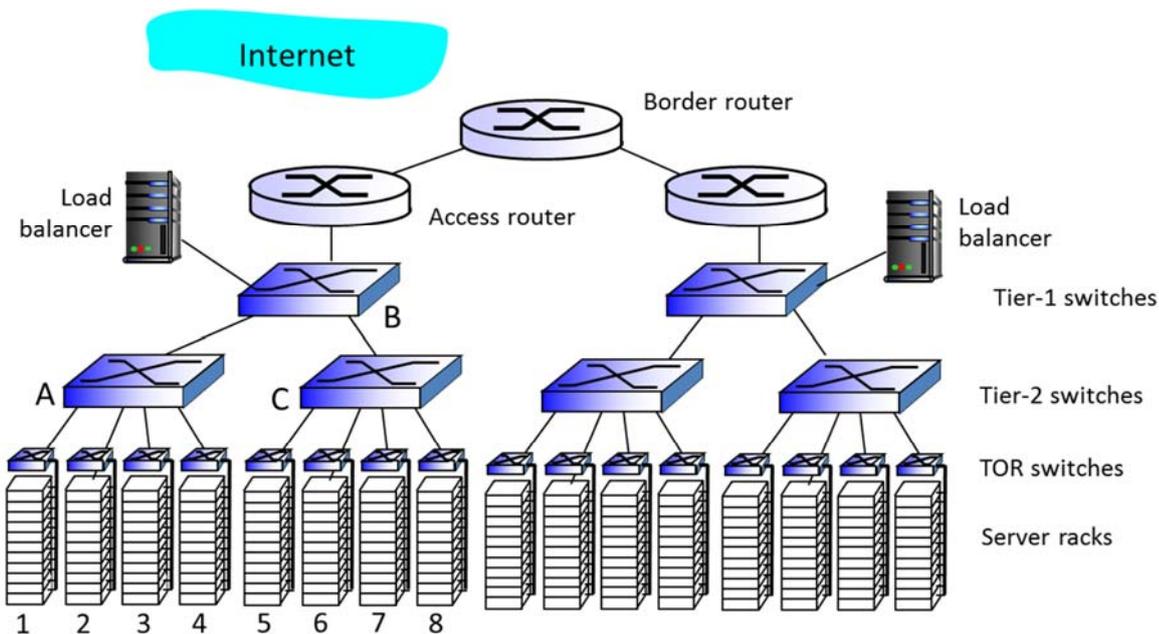
■ Nachteile:

- Nachrichtenaustausch aufwendig (Kommunikations-Overhead)
- kein 'single system image'



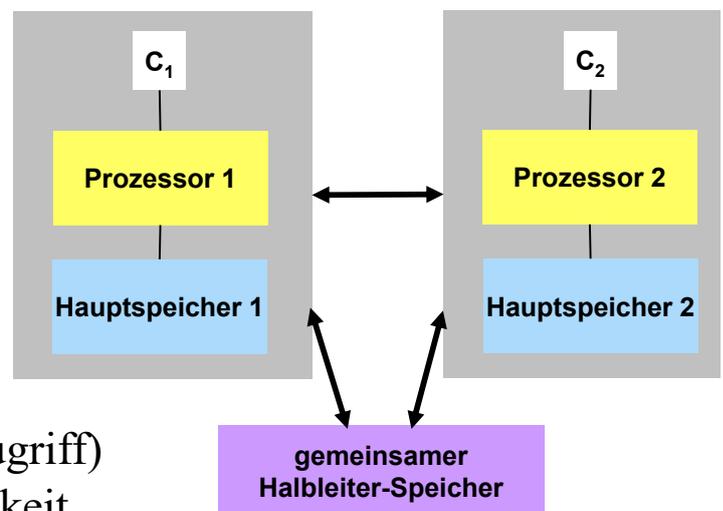
Lose Kopplung im Data Center

- mehrstufige Kopplung mit Server-Racks
 - Tausende von Rechnern
 - schnellere Kommunikation innerhalb Rack



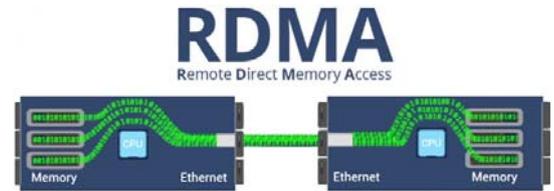
Nahe Rechnerkopplung (closely coupled systems)

- Kompromiß zwischen enger und loser Kopplung
 - effizientere Kommunikation als mit loser Kopplung unter Beibehaltung einer ausreichenden Fehlerisolation und Erweiterbarkeit
- Merkmale
 - n selbständige Rechnerknoten
 - gemeinsame Halbleiter-Speicherbereiche
 - lokale Rechneranordnung
- Speichereigenschaften
 - schneller, synchroner Zugriff (kein Prozesswechsel während Zugriff)
 - i.a. keine Instruktionsadressierbarkeit
 - ggf. nichtflüchtig
- Unterstützung z.B in IBM z/OS-Mainframes (Sysplex)



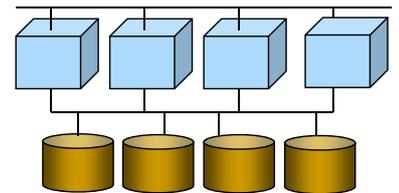
Remote Direct Memory Access (RDMA)

- direkter Zugriff auf Hauptspeicher anderer Rechner im Cluster
 - Abwicklung über Netzwerkkomponenten, ohne Involvierung des Betriebssystems und von CPUs
 - „Zero-Copy-Networking“
 - Unterstützung für InfiniBand, Ethernet (RDMA over Converged Ethernet)
- wesentlich schnellerer Datenaustausch als mit loser Kopplung
 - nur noch ca 10-mal langsamer als Hauptspeicherzugriff
- Alternative zu naher Kopplung



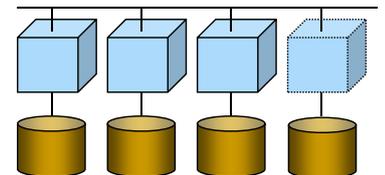
Externspeicheranbindung

- gemeinsam:* jeder Prozessor kann alle Externspeicher / Daten direkt erreichen
- Rechneranordnung: lokal
 - Rechnerkopplung: eng, lose oder nahe
 - hohes Potenzial zur Lastbalancierung



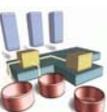
- partitioniert:* Externspeicher sind primär nur je einem Knoten zugeordnet

- Rechneranordnung: lokal oder ortsverteilt
- Rechnerkopplung: i. a. lose
- verteilte Transaktionsausführung, um auf entfernte Daten zuzugreifen

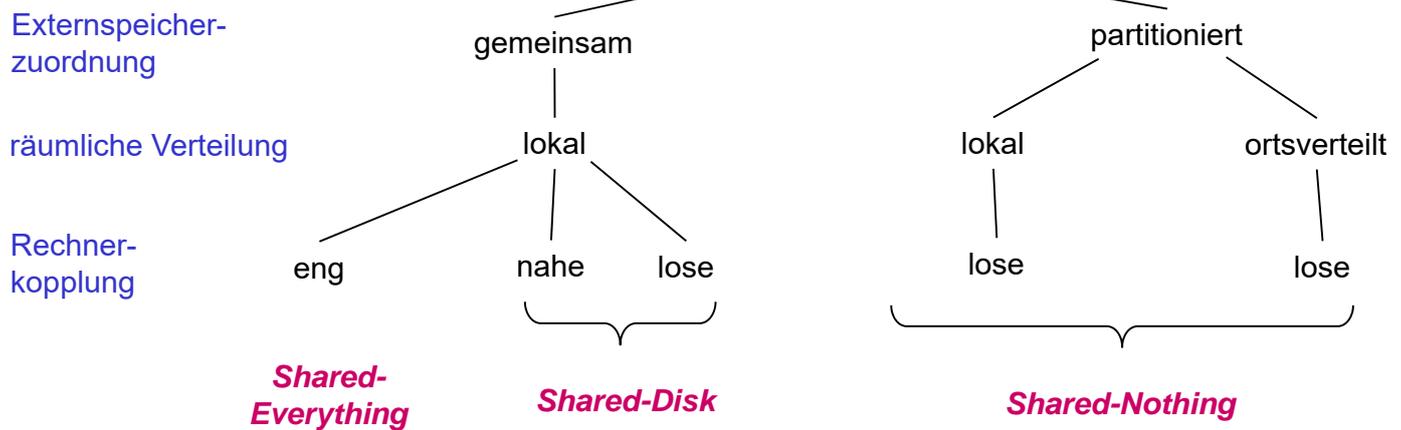


verteilte/parallele **In-Memory-Datenbanksysteme** ?

- enge Kopplung: gemeinsam genutzte In-Memory-DB
- lose Kopplung: partitionierte In-Memory-DB



Shared -*-Systeme



- **Parallele DBS:** lokal verteilte Shared -*-Systeme
- **Verteilte DBS:** ortsverteilte DBS
- **hybride Ansätze**
 - eng gekoppelte Knoten in lose gekoppelten Systemen
 - ortsverteilte Data Center mit PDBS pro Center



3 Stufen der Verteilung

1. **Scale-Up:** mehrere Prozessoren innerhalb von 1 Knoten (Shared Everything)
 - sehr effiziente Kommunikation (über Hauptspeicher)
 - direkter Zugriff auf gesamte Datenbank für alle DBMS-Instanzen; zentrale Datenstrukturen (Sperrtabelle, DB-Puffer, etc.)
 - einfache DB-Administration
 - von allen DBS-Herstellern unterstützt (Microsoft, Oracle, IBM ...)
 - Leistungsfähigkeit reicht für Mehrzahl von Datenbanken
 - begrenzte Erweiterbarkeit und Verfügbarkeit
 - relativ hohe Kosten im High-End-Bereich
2. **Scale-Out:** SN/SD/Hybrid-Cluster
 - hohe Skalierbarkeit durch unabhängige Rechnerknoten (kein gemeinsamer Hauptspeicher, lokale Software)
3. (geo-)verteilte Replikation (für SE, SD oder SN)



Skalierbarkeit: Scale-Up vs. Scale-Out

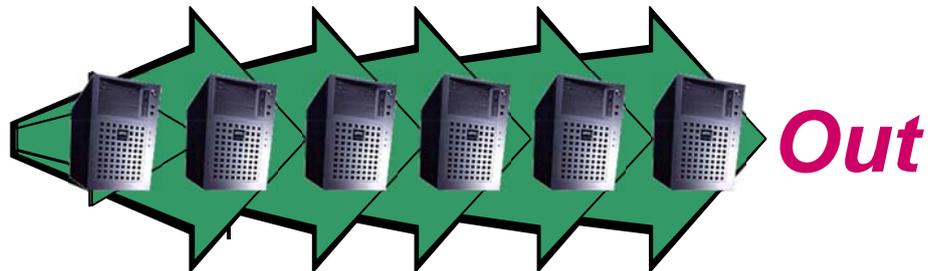


“Scale Up”

- schnellere Mehrprozessorknoten
- Shared Everything

“Scale Out”

- N unabhängige Rechner (z.B. Commodity-Server)
- Hinzufügen neuer Server nach Bedarf
- Shared Nothing oder Shared Disk (Cluster)



Scale-Out

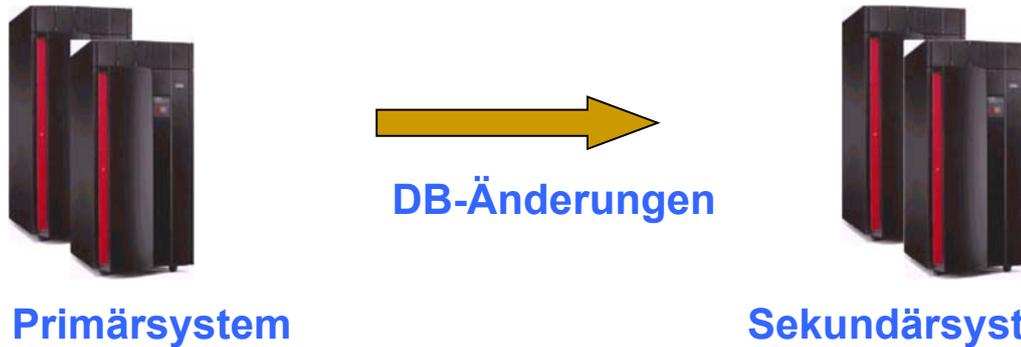
- sehr viele preiswerte Standard-Knoten (Blades)
 - geringer Administrationsaufwand
 - leichte Erweiterbarkeit
 - ausreichend für gut partitionierbare Lasten aus einfachen Operationen
 - Bsp.: Hadoop-Cluster, Data Center von Google / Amazon etc.

VS.

- moderate Zahl von High-End-Servern



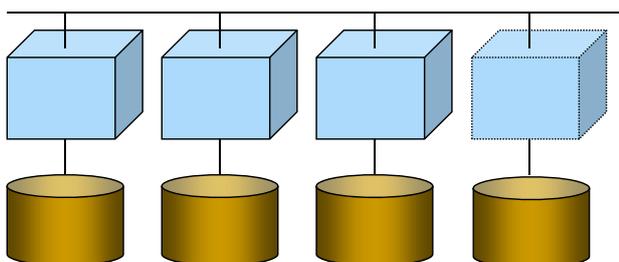
DB-Replikation für Hochverfügbarkeit



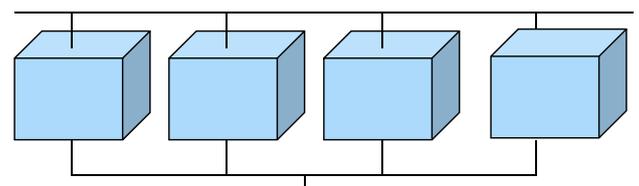
- komplette DB-Kopie an entferntem System (Mirroring, Geo-Replikation)
- fortlaufende Übertragung aller Änderungen aus Primärsystem (z.B. Log-Transfer) und Anwendung auf Kopie
- Schutz auch gegenüber Katastrophen
- anwendbar für alle PDBS-Architekturen im Primärsystem: SE, SN, SD und Kombinationen



Shared Nothing (SN) vs. Shared Disk (SD)



SN



SD

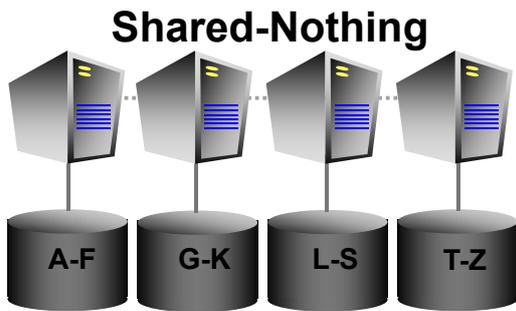
- ❖ Teradata
- ❖ IBM DB2 (Windows, Unix)
- ❖ MS Parallel Data Warehouse
- ❖ viele NoSQL-Systeme

- ❖ Oracle (RAC, Exadata)
- ❖ IBM DB2 z/OS, DB2 PureScale
- ❖ SAP Sybase

- SN auf SD-Hardware realisierbar und umgekehrt

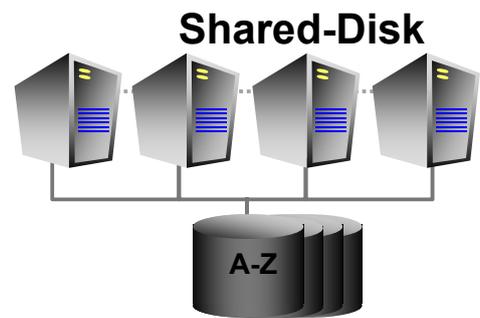


SN vs. SD: Leistungsfähigkeit



statische Datenpartitionierung
bestimmt Ausführungsort von
DB-Operationen und damit
Kommunikationsaufwand

geringe Möglichkeiten zur
dynamischen Lastbalancierung



hohe Flexibilität zur Parallelisierung
und Lastbalancierung

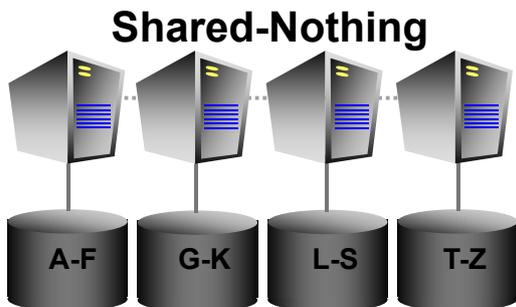
- Erreichbarkeit aller Daten von
jedem Knoten

hoher Aufwand für Synchronisation
und Kohärenzkontrolle

- Lokalität ermöglicht Einsparungen
- nahe Kopplung reduziert Overhead



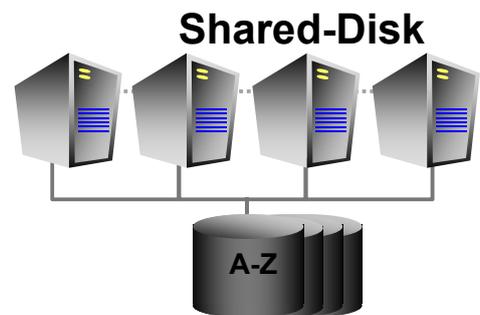
SN vs. SD: Erweiterbarkeit



Hinzufügen neuer Knoten
hardwareseitig einfach

- viele Knoten sind möglich

neuer Rechner erfordert
physische Neuauflteilung der
Datenbank (N -> N+1)

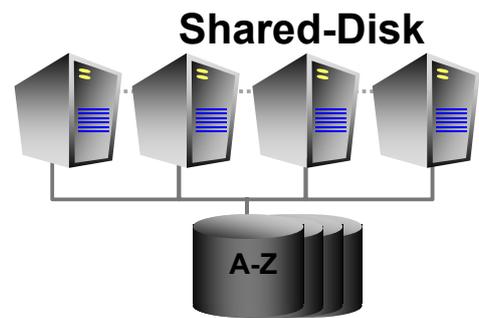
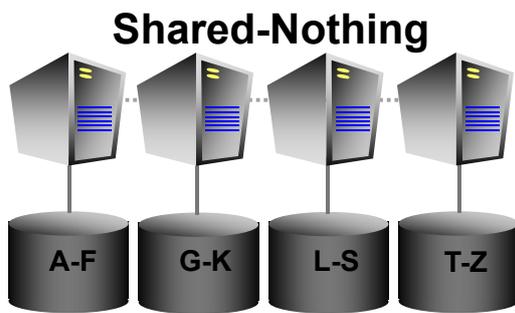


keine physische (Neu-)
Aufteilung der DB bei
neuem Rechner

direkte Plattenanbindung
begrenzt Rechneranzahl



SN vs. SD: Recovery

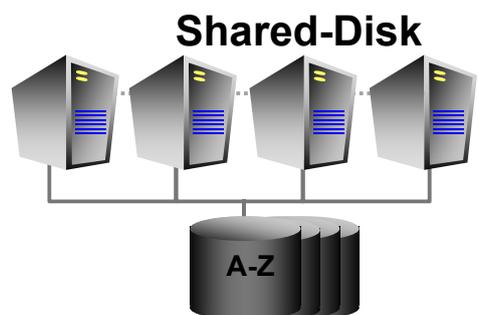
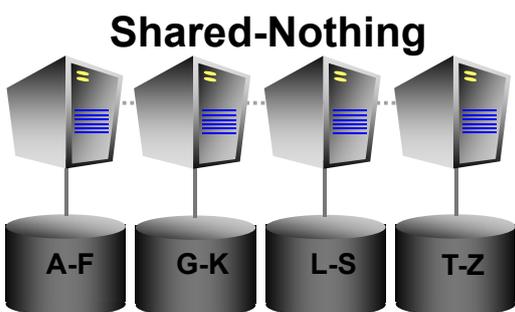


Übernahme/Recovery der betroffenen Partition durch anderen Rechner vorzusehen (ggf. Überlastungsgefahr)

gesamte DB bleibt nach Rechnerausfall erreichbar
komplexe Crash-Recovery
Erstellung einer globalen Log-Datei



SN vs. SD: Technische Probleme



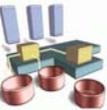
- DB-Partitionierung bzgl. Rechner
- verteilte und parallele Anfrageverarbeitung
- verteiltes Commit-Protokoll
- globale Deadlock-Behandlung
- Replikationskontrolle
- Katastrophen-Recovery ...

- DB-Partitionierung bzgl. Externspeicher
- parallele Anfrageverarbeitung
- globale Synchronisation
- Kohärenzkontrolle
- globaler Log, Crash-Recovery
- Lastbalancierung
- Katastrophen-Recovery ...



2. Klassifikation von Mehrrechner-DBS

- Merkmale für PDBS/VDBS
 - *räumliche Verteilung*: ortsverteilt oder lokal
 - *Rechnerkopplung*: enge, lose oder nahe Kopplung
 - *Externspeicheranbindung*: partitioniert oder gemeinsam ('shared')
- PDBS-Architekturen
 - *Scale-Up vs. Scale-Out*
 - *Shared-Nothing vs. Shared-Disk*
- Weitere Klassifikationsmerkmale
 - funktionale Spezialisierung vs. funktionale Gleichstellung
 - integrierte vs. heterogene/föderierte MRDBS
- Systemansätze zur Datenintegration
- Grobbewertung von MRDBS-Alternativen

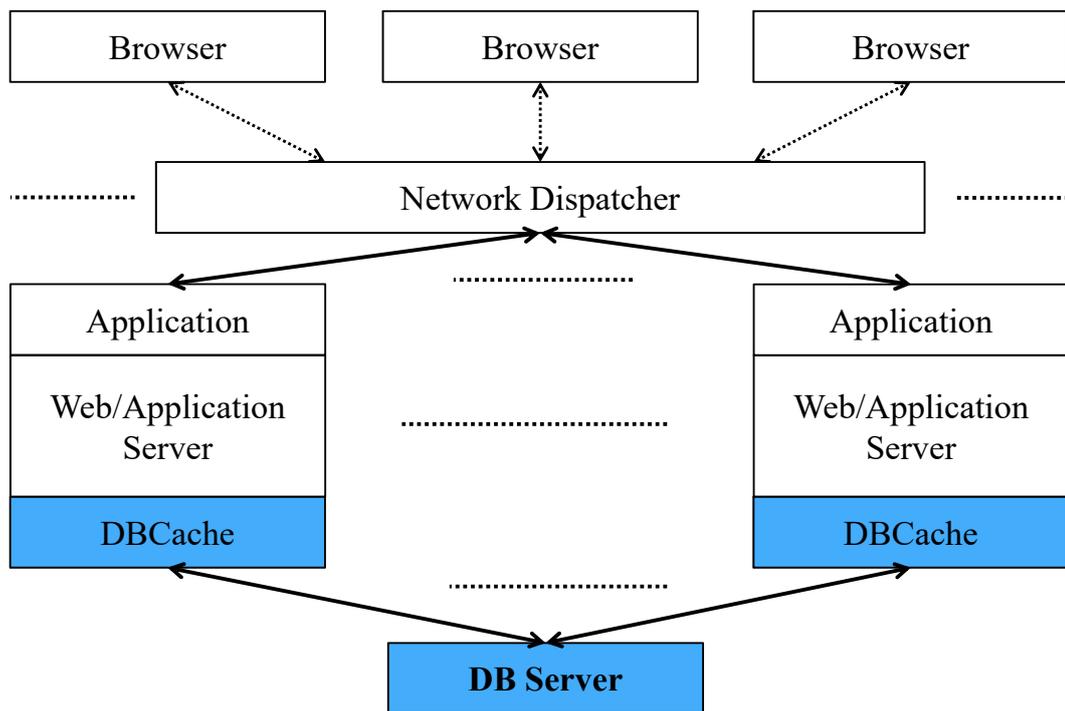


Verteilung der Funktionalität

- funktionale Gleichstellung („horizontale Verteilung“)
 - jeder Knoten besitzt gleiche DBS-Funktionalität (z.B. vollständiges DBMS pro Knoten)
 - Replikation der Funktionen
- funktionale Spezialisierung: Partitionierung von Funktionen
 - Beispiele:
 - *DB-Maschinen* mit Spezialprozessoren für bestimmte DB-Funktionen (Join-Prozessor, Sortier-Prozessor, etc.)
 - intelligente Platten-Kontroller z.B. für Selektion/Projektion (**Oracle Exadata**)
 - Optimierung von DB-Funktionen durch programmierbare/konfigurierbare GPUs und FPGA (Bsp. IBM Netezza)
 - Web-Informationssysteme (Multi-Tier-Architekturen) mit DB-Server und DB-Verarbeitung auf Applikations-Server
 - Spezialisierung erschwert Lastbalancierung, Erweiterbarkeit und Fehlertoleranz
- hybride Partitionierung + Replikation von DBS-Funktionen



Web-Informationssysteme mit Mid-Tier Caching



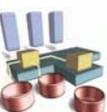
Integrierte vs. heterogene/föderierte MRDBS

■ integrierte Mehrrechner-DBS

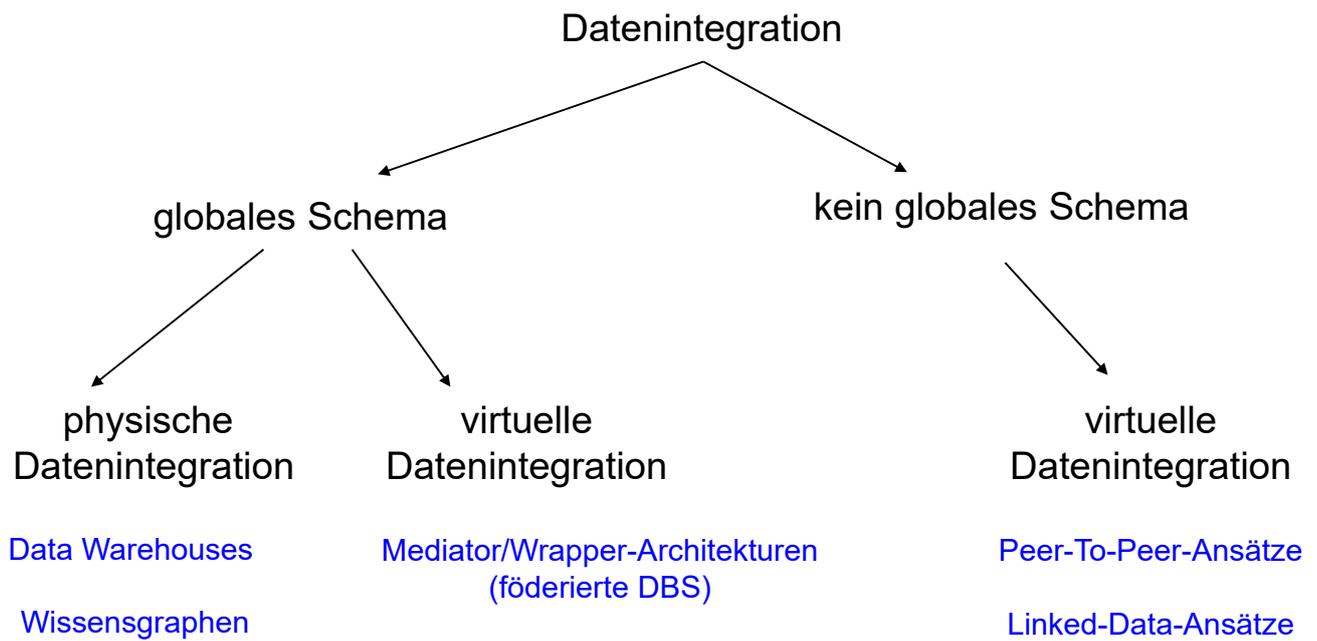
- 1 logische Datenbank: DB-Zugriff wie im zentralen Fall (Verteiltransparenz für AP)
- **Top-Down-Ansatz** zur Verteilung einer DB
- homogenes MRDBS (z. B. identische DBMS-Instanzen)
- geringe Autonomie für beteiligte DBMS
- Beispiele: Verteilte DBS, Parallele DBS

■ heterogene / föderierte Mehrrechner-DBS

- **Bottom-Up-artige Kopplung** existierender Datenbanken
- weitgehend unabhängige DBMS mit eigenen DB-Schemata
- partielle Zulassung externer Zugriffe (Kooperation)
- Heterogenität bei Datenmodellen und Transaktionsverwaltung möglich
- große Probleme mit semantischer Heterogenität
- Verteilungstransparenz i.a. nur bedingt erreichbar
- verschiedene Alternativen zur **Datenintegration**

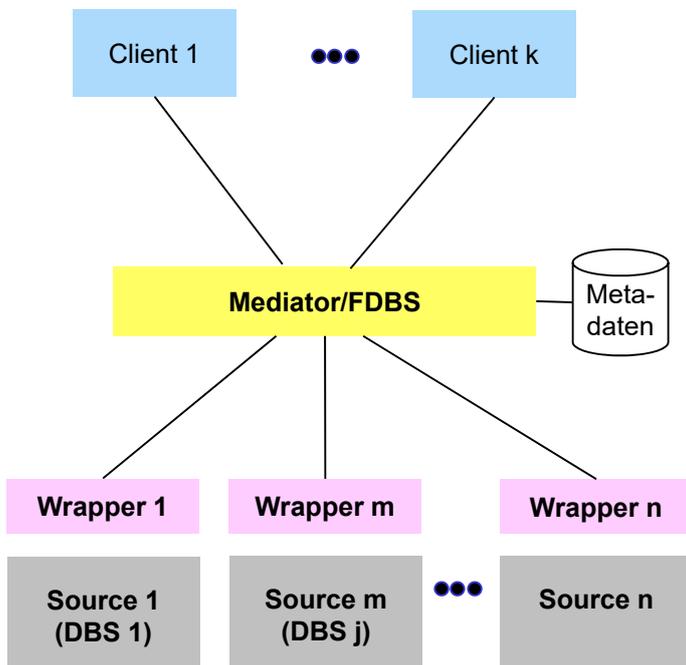


Heterogene DBS

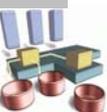
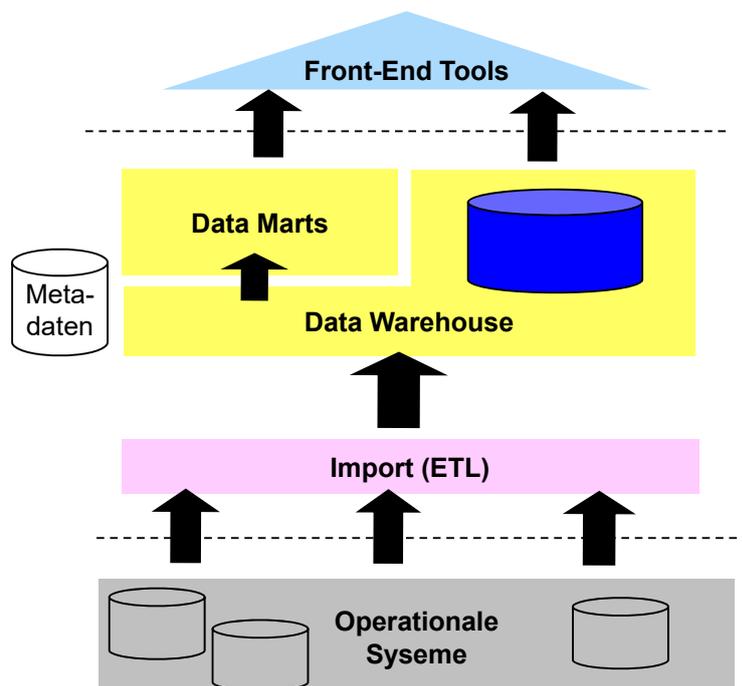


Grundlegende Alternativen zur Datenintegration

Virtuelle Integration
(Mediator/Wrapper-Architekturen,
föderierte DBS)

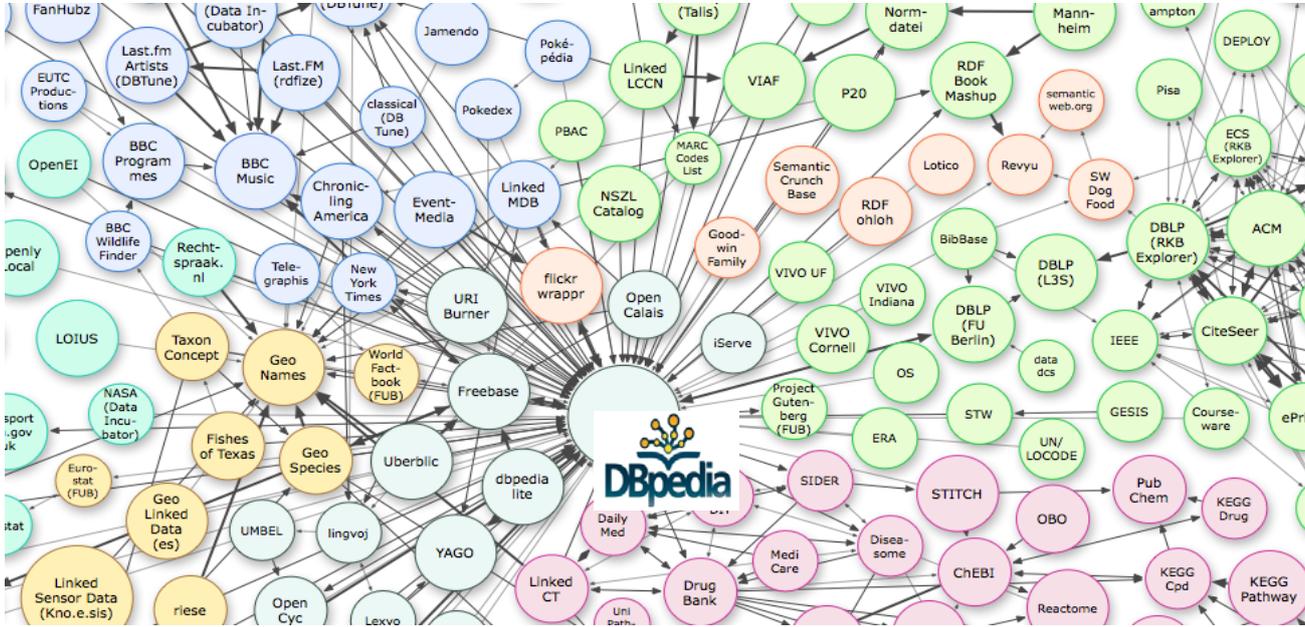


Physische (Vor-) Integration
(Data Warehousing)



Linked Open Data

- vernetzte RDF-basierte Datenquellen
- Links (Mappings) zwischen Instanzen und Ontologie-Konzepten ermöglichen Datenintegration



Wissensgraphen

- einheitliche Verwaltung und semantische Kategorisierung von Entitäten und ihren Beziehungen

- Beispiele: DBpedia, Yago, Wikidata
- Google KG, MS Satori
- Amazon Product KG ...



- Informationen stammen oft aus anderen Wissensquellen (Wikipedia, Wordnet etc.), Webseiten ...

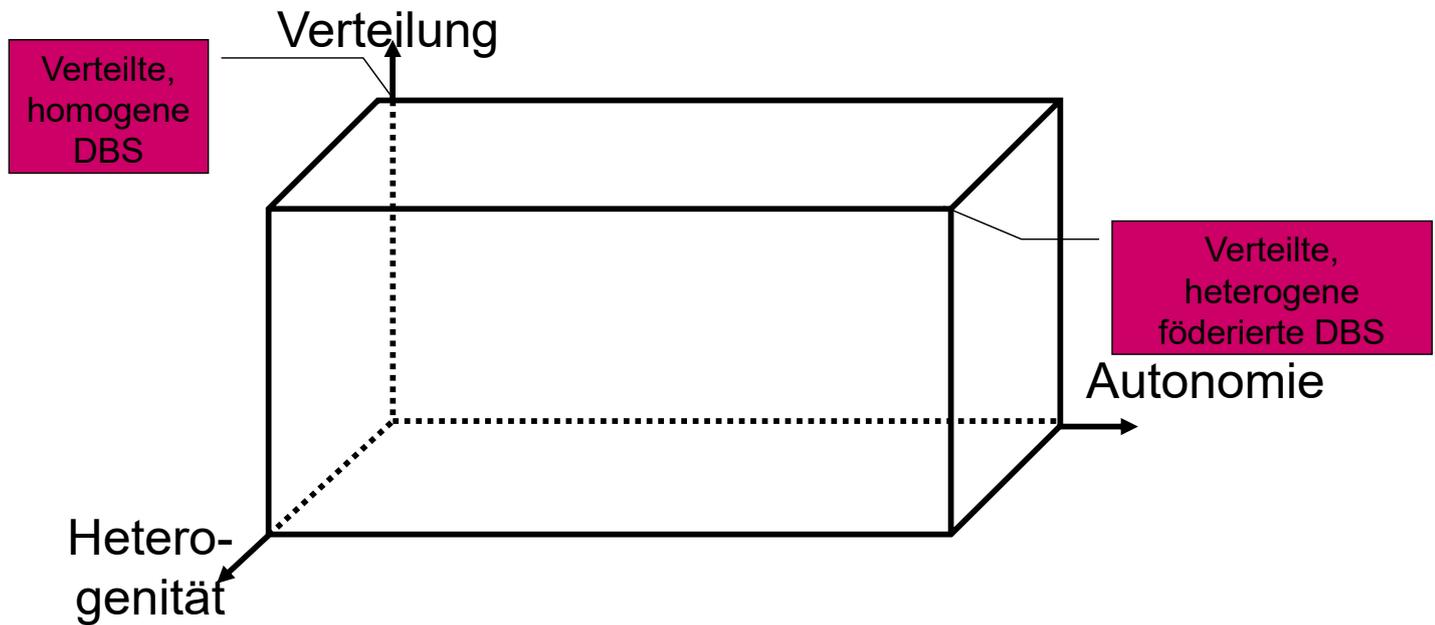
- domänenspezifisch oder übergreifend / global

- viele Nutzungsmöglichkeiten

- Anreicherung von Daten
- Verbesserung von Suchergebnissen
- Generierung von (Produkt-) Empfehlungen
- Gewinnung von Trainingsdaten für Machine Learning ...



Klassifikation nach Özsu/Valduriez



Grobbewertung von Mehrrechner-DBS

	Parallele DBS (SD, SN)	Verteilte DBS	Föderierte DBS
hohe Transaktionsraten	++	o/+	o
Intra-Query-Parallelität	++	o/+	-/o
Skalierbarkeit	+ / ++	o/+	o
Verfügbarkeit	+	+	-
Verteilungstransparenz	++	+	o
geographische Verteilung	-	+	+
Knotenautonomie	-	o	+
DBS-Heterogenität	-	-	+
Administration	o	-	-/--

- Big Data Architekturen nutzen v.a. Shared Nothing
 - Nutzung von Hadoop-Cluster mit Analyse-Frameworks / SQL-on-Hadoop
 - NoSQL auf Shared-Nothing-Cluster
 - Parallele DBS / NewSQL z.B. mit In-Memory-Datenbanken



Zusammenfassung

- vielfältige Anforderungen an Mehrrechner-DBS führen zu verschiedenen Architekturtypen:
 - Parallele DBS, Verteilte DBS, föderierte DBS, Data Warehouses ...
- Klassifikationsmerkmale
 - räumliche Verteilung, Rechnerkopplung, Externspeicheranbindung, integrierte/homogene vs. föderierte / heterogene DBS, funktionale Spezialisierung vs. Gleichstellung
- Parallele DBS:
 - Ziele: hohe Leistung/Parallelisierung, hohe Verfügbarkeit, Skalierbarkeit
 - lokale Rechneranordnung
 - Hauptansätze: Shared-Everything, Shared-Disk, Shared-Nothing
- Verteilte DBS: ortsverteilte, integrierte MRDBS (globales Schema)
- PDBS-Vergleich
 - Skalierbarkeit: Scale-Up (SE) einfacher umsetzbar als Scale-Out
 - DB-Mirroring für Hochverfügbarkeit
 - Scale-Out: Shared-Disk vs. Shared-Nothing



Zusammenfassung (2)

- Mehrrechner-DBS mit funktionaler Spezialisierung
 - z.B. Multi-Tier-Architekturen
 - Nutzung von Spezial-Hardware („Datenbank-Maschinen“) problematisch bezüglich Flexibilität und Erweiterbarkeit
 - Revival: Nutzung von GPUs / FPGAs
- virtuelle Datenintegration
 - föderierte DBS / Linked Data
 - Bewahrung einer relativ hohen Knotenautonomie
- physische Integration heterogener Datenbanken
 - populärste Datenintegrationslösung (Data Warehouses, Big Data)
 - Data Warehouse kann durch zentrales DBS oder PDBS verwaltet werden

