Institut für Wirtschaftsinformatik

Fachhochschule
Wilhelmshaven
Oldenburg
Elsfleth

Jade Weser Region

Projektabschlussbericht

Modellierung und Integration von verteilten Datenbanken

Teilprojekt im Forschungsschwerpunkt

„Applikationen für massiv-parallele Rechnersysteme“

AGIP-FÖRDERPROJEKT, MWK
gefördert durch

VolkswagenStiftung

1 September 2009

Verfasst von: Prof. Alfred Wulff, Larissa Janssen

Inhalt

Zusammenfassung 3
Einleitung 3
Projektziele 3
Methodik und Vorgehensweise 3
Ergebnisse 4
Bewertung 7
Veröffentlichungen, Vorträge, Ausstellungen im Zusammenhang mit dem Projekt 7
Projektdokumente 7
Vorträge und Präsentationen 8
Grundlagen 9
Prognosen für Hardwaretechnologien 9
Homogene, (sehr) eng integrierte verteilte Datenbanksysteme 11
Parallele Datenbanksysteme (DBS) 12
Klassifikation der Architekturen von parallelen DBS 14
Abbildungsverfahren der DBS-Architekturen 14
Basisarchitekturen 17
Hierarchische Architekturen 18
Realisierungen von Oracle-RAC 21
Skalierbarkeit 21
Hochverfügbarkeit 22
Lastbalancierung 22
Interprozessorkommunikation 23
Cache-Kohärenzkontrolle 24
Synchronisation 26
Parallele Anfragenausführung 27
Einfache Administration 30
Kurze Antwortzeiten/hoher Durchsatz 31
Kosteneffektivität des RAC 31
Realisierung: MySQL Cluster 32
Skalierbarkeit 34
Hochverfügbarkeit 35
Lastbalancierung 37
Interprozessorkommunikation 38
Cache-Kohärenzkontrolle 41
Synchronisation 41
Parallele Anfragenausführung 43
Einfache Administration 45
Kurze Antwortzeiten/hoher Durchsatz 46
Kosteneffektivität im MySQL Cluster 46
Erprobung der Applikation FrontOffice 48
Vorstellung der Anwendung Office-Front 48
Testziele 49
Testfälle 49
Testumgebung Oracle RAC 51
Testumgebung: MySQL Cluster 55

RESÜMEE 56
LITERATURVERZEICHNIS 58

ZUSAMMENFASSUNG

EINLEITUNG

Leistungsfähige Datenbanksysteme, die immer größer werdende Datenmengen aufnehmen und schnell verarbeiten können, bilden die Grundlage heutiger moderner Anwendungs- und Informationssysteme. Verteilte und insbesondere parallele Datenbanksysteme spielen dabei eine besondere Rolle, da sie hohe Performanz gewährleisten, Hochverfügbarkeit sicherstellen und gute Skalierbarkeit aufweisen.

Im Rahmen dieses Teilprojektes lagen die Arbeitsschwerpunkt auf der Analyse der Einsatzmöglichkeiten verteilter Datenbankmanagementsysteme (DBMS), der Einrichtung einer Entwicklungs- und Testumgebung, der Realisierung konkreter Anwendungsfälle, der Erprobung verschiedener Verteilungstechniken und der Analyse kommerzieller und Open-Source DBMS. Die Projektziele, Vorgehensweise und erzielten Ergebnisse werden in diesem Kapitel kurz zusammengefasst und bewertet.

PROJEKTZIELE

Diese Arbeitsschwerpunkte dieses Teilprojektes bestanden insbesondere aus folgenden Aufgabenbereichen:

- Analyse der Einsatzmöglichkeiten verteilter DBMS
- Einrichtung einer Entwicklungs- und Testumgebung
- Realisierung konkreter Anwendungsfälle
- Entwicklung von Modellierungsverfahren
- Erprobung verschiedener Verteilungstechniken
- Verteilte, parallele SQL-Bearbeitung
- Analyse kommerzieller und Open-Source DBMS.

Die Arbeiten dieses Teilprojektes wurden im Januar 2006 vom IfW (Institut für Wirtschaftsinformatik) am FH-Standort Wilhelmshaven übernommen. Gegenüber der ursprünglichen Arbeitsplanung wurde eine Anpassung der Tätigkeitsschwerpunkte vorgenommen. Verstärktes Gewicht wurde auf die Entwicklung von Datenmodellen und Anwendungen aus realen Anwendungskontexten gelegt. Die Beispiele wurden dabei aus konkreten Fragestellungen oder Projekten von Unternehmen oder Institutionen übernommen. Dabei wurden bereits frühzeitig Kooperationen gesucht, um spätere Transferleistungen vorzubereiten. Etwas geringeres Gewicht wurde dagegen der Betrachtung von SQL-Ausführungsstrategien beigemessen, da die zugehörigen Mechanismen der DBMS über interne, kaum beeinflussbare DBMS-Prozesse geregelt werden. Die sehr proprietären Tuning-Techniken wurden ebenfalls mit geringerer Kapazität untersucht.

METHODIK UND VORGEHENSWEISE

Die Projektplanung wurde nach dem in der Softwareentwicklung etablierten Rational Unified Process (RUP) vorgenommen. Die Projektergebnisse konnten in diesem iterativen Entwicklungsprozess ständig überprüft und die erzielten Erfahrungen in die laufende Entwicklung einbezogen. Die zentralen Workflows (Anforderungsanalyse,

Entwurf und Entwicklung, Test und Deployment) wurden im Projekt mehrfach durchlaufen, wobei die Entwicklungsergebnisse ständig verfeinert, erweitert und verbessert wurden. Für dieses Teilprojekt wurden drei RUP-Iterationen gewählt:

1. Schwerpunkt Planung und Definition (01. 01. 2006 – 31.10.2006)
2. Schwerpunkt Ausarbeitung (01. 11. 2006 - 31. 10. 2007)
3. Schwerpunkt Konstruktion und Transfer (01.11.2007 – Projektende)

Die Arbeiten an diesem Teilprojekt wurden am IfW in einem Projektteam abgewickelt. Projektmitglieder waren M.Comp.Sc. Larissa Janssen (Softwareinstallation der Rechnersysteme, DBS-Installation, -Konfiguration und -Test), Dipl.-Wirtsch.Inf.(FH) Sascha Fankhänel (Hardwareinstallation), Prof. Alfred Wulff (Leitung, Modellierung, Anwendungsdesign) und studentische Mitglieder. Die Koordination mit der Gesamtprojektleitung und den übrigen Teilprojekten wurde über die Teilnahme an den Gesamtprojektmeetings des Forschungsschwerpunktes gewährleistet.

ERGEBNISSE

Bei der Einrichtung und Konfiguration von Datenbanksystemen werden die betroffenen Rechnersysteme häufig mehrere Tage ausschließlich für Administrationszwecke beansprucht. Zur vom Emder-Cluster unabhängigen Bearbeitung der Projektaufgaben wurde daher die Ausschreibung, Beschaffung und Inbetriebnahme eines Mini-Clustersystems (4x2 Nodes) für den Standort Wilhelmshaven durchgeführt. Alle Entwicklungen und Tests dieses Teilprojektes wurden darauf durchgeführt. Damit wurde ein Test- und Entwicklungscluster für verteilte Datenbankanwendungen aufgebaut, auf dem auch zukünftig insbesondere folgende Aufgaben wahrgenommen werden können:

- Abbildung verschiedener (max. 4) Remote-Standorte auf Clusterknoten
- Entwicklung und Erprobung verteilter Anwendungen
- Optimierungs- und Tuningmaßnahmen
- Benchmarks/Laufzeitmessungen
- Simulation verschiedener Betriebs-/ System- und Netzumgebungen.

Neben den unten beschriebenen Kooperationen innerhalb des Projektes wurden noch weitere Kontakte zu regional ansässigen Unternehmen aus Industrie und Wirtschaft aufgebaut.

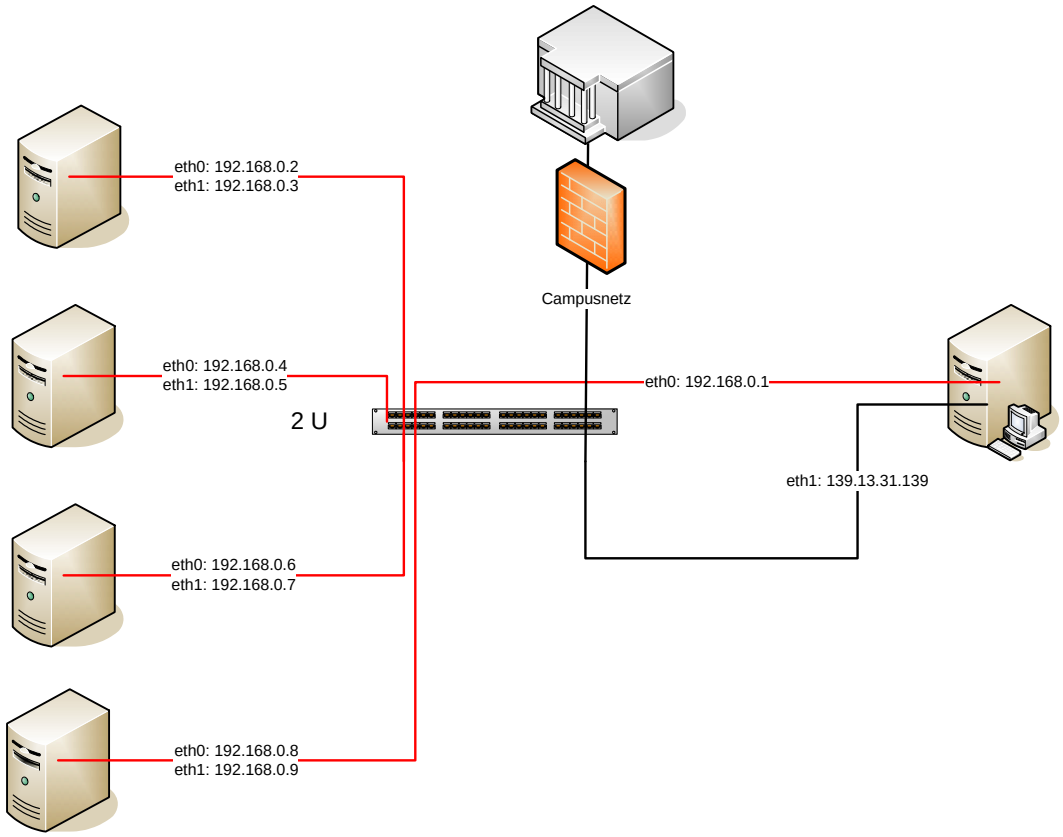


Abbildung 1: Clusterstruktur im IfW

Im Rahmen des Projektes wurde eine erste Diplomarbeit [Zyw06] mit dem Titel „Grundlagen und Design verteilter Datenbanksysteme für kommerzielle und Open-Source basierte Datenbankmanagementsysteme“ am IfW abgeschlossen. Neben einer Zusammenfassung des grundlegenden Wissens über verteilte Datenbanken wurden hier bekannte Partitionierungs- und Replikationstechniken dargestellt und verglichen sowie die Ansätze wichtiger DBMS-Produkte zur Unterstützung der Verteilungsaspekte herausgearbeitet.

Bei der Auswahl der zu untersuchenden DBMS wurden sowohl kommerzielle als auch Open-Source-basierte Produkte berücksichtigt. Aufgrund ihrer Verbreitung, Bedeutung und der unterstützten Verteilungsfunktionen wurden die DBMS ORACLE (kommerziell) und MySQL (Open-Source) als primäre Zielsysteme festgelegt. In der letzten Projektphase wurden noch Untersuchung zu den Verteilungsfeatures und Einsatzmöglichkeiten des Mircosoft SQL-Servers 2008 im Rahmen einer Diplomarbeit [Rem08] vorgenommen.

Bei der Erprobung und Prototyperstellung mussten in diesem Projekt umfangreiche Testdaten („Bulk-Data“) für unterschiedliche DB-Systemumgebungen bereitgestellt werden. Zu diesem Zweck wurden diverse Werkzeuge beschafft. Mit Hilfe dieser Datengeneratoren konnten Populationen von Testdatenbanken für unterschiedliche DBMS eingerichtet werden. Für die Durchführung von Performance-Messungen mussten ebenfalls einheitliche Test-Tools beschafft und erprobt werden, die eine automatische Lastgenerierung, die Simulation der Transaktionsverarbeitung

konkurrierender DB-Benutzer, gängige Standard-Benchmarks (z.B. TPC-C) und Stress-Tests für unterschiedliche DB-Plattformen unterstützen. Die Bewertung und ein Vergleich von Datenbanksystemen durch Leistungsmessung und Benchmarking wurden ausführlich im Rahmen einer weiteren Diplomarbeit [Bru08] vorgenommen.

Mit dem Deutschen Zentrum für Marine Biodiversitätsforschung am Senckenberg-Institut in Wilhelmshaven wurden Kontakte zur Auswertung umfangreicher Messdatenbestände aus Forschungsreisen und Langzeitstudien aufgenommen. In diesem Zusammenhang wurde eine Bachelorarbeit [Echt06] von Thomas Echterhagen mit dem Thema „Entwicklung eines Data Marts meeresbiologischer Daten und deren Anbindung an ein GIS-System auf Basis einer verteilten Datenbank in Kooperation mit dem Forschungsinstitut Senckenberg Wilhelmshaven“ angefertigt. Mit dem entwickelten Prototyp wurden grundlegende Abfragefunktionen auf Basis einer verteilten Datenbank (ORACLE) realisiert. Auch die Einbindung eines GIS konnte demonstriert werden. Abbildung 2 zeigt einen aus der Anwendung generierten Routenplot einer Forschungsreise des Forschungsschiffes Meteor aus dem Jahr 2005.

Abbildung 2: Routenplot einer Forschungsreise

Eine weitere Kooperation konnte mit dem FW-Systemhaus, Oldenburg durchgeführt werden. Ziel der Kooperation war die Untersuchung zur Fragestellung, ob ein von dem Unternehmen entwickeltes, kommerziell vertriebenes Softwaresystem zum Energiecontrolling auf eine verteilte Datenbankplattform migriert werden konnte. Dabei wurden verschiedene DBMS (MySQL Cluster, ORACLE RAC) berücksichtigt und Einsatzempfehlungen für bestimmte Systemkonfigurationen gegeben. Auf dem Wilhelmshavener Entwicklungscluster wurden dazu im Rahmen einer Diplomarbeit [Hol08] umfangreiche Last- und Stresstest durchgeführt und dokumentiert.

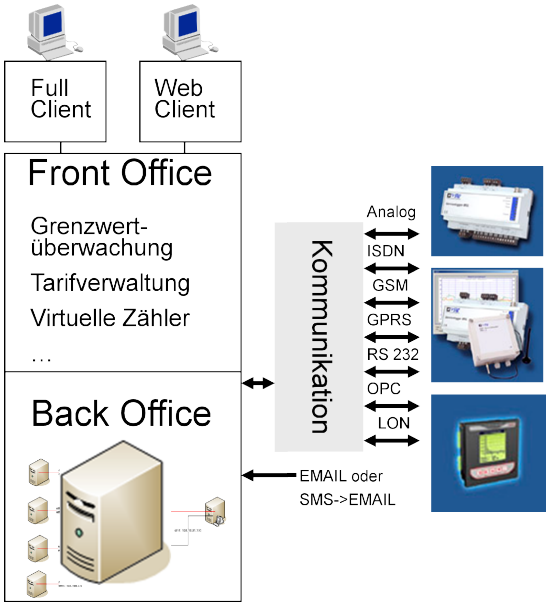


Abbildung 3: Anwendung Energiecontrolling mit Cluster-DBMS im Backend

Die Analyse kommerzieller und Open-Source-basierter, clusterfähiger DBMS wurde im Projekt breiter Raum eingeräumt. Hier wurden insbesondere die bekanntesten Vertreter beider Richtungen, MySQL Cluster und ORACLE RAC ausführlich untersucht und bewertet. Neben studentischen Projektarbeiten im Studienschwerpunkt „Moderne Datenbankbasierte Informationssysteme“ wurde in diesem Zusammenhang eine Bachelorarbeit [Kru07] angefertigt. Die besondere Rolle paralleler DBMS für Hochleistungsanforderungen untersuchte Larissa Janssen im Rahmen ihrer an der FernUniversität Hagen im Projektzeitraum angefertigten Masterarbeit [Jan07] zum Thema „Hochleistungs-Datenbanksysteme“. Die Masterarbeit wurde als Buch [Jan08] veröffentlicht.

BEWERTUNG

Die angestrebten Projektziele wurden erreicht. Bei den beteiligten Partnern konnte durch die Projektarbeiten wertvolles Beratungs- und Entwicklungs-Know-how über verteilte DBMS-Lösungen aufgebaut werden. Mit dem eingerichteten Test- und Entwicklungscluster am IfW steht eine Plattform bereit, die eine effiziente Entwicklung verteilter DBMS-Lösungen ermöglicht.

Die Hochschule konnte durch das Projekt seine Fachkompetenz im Bereich verteilter Softwareentwicklung und Datenbanksysteme weiter ausbauen und mit Präsentationen u.a. auf den Symposien „Clustercomputing“ in Wissenschafts- und Branchenkreisen dokumentieren. In Gesprächen und Kooperationen mit regionalen Unternehmen sind durch das Projekt wichtige Kontakte entstanden, die hohes Potential für zukünftige Vorhaben bergen. Die Drittmittelfähigkeit der Hochschule im Allgemeinen und des IfW im Besonderen ist damit nachdrücklich gesteigert worden.

Durch die Einbeziehung der Projektarbeiten in den Studienbetrieb des Standortes Wilhelmshaven (insbesondere Wirtschaftsinformatik) wurden aktuelle Entwicklungen aus dem Projektbereich „Verteilte DBMS“ an die Studierenden vermittelt. Die relativ große Anzahl an Diplom- und Bachelorarbeiten, die im Rahmen dieses Projektes angefertigt wurden, belegt den positiven Effekt des Vorhabens im Bereich des Studienbetriebs deutlich.

VERÖFFENTLICHUNGEN, VORTRÄGE, AUSSTELLUNGEN IM ZUSAMMENHANG MIT DEM PROJEKT

PROJEKTDOKUMENTE

[Rem09] Remmers, Ingo: Einsatzmöglichkeiten des MS SQL Server 2008 für verteilte Datenbanklösungen, Diplomarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2008

[Hol08] Holtmann, Andreas: Konzeption und Durchführung eines Last- und Stresstests für eine kommerzielle BDE-Software auf einem Rechnercluster, Diplomarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2008

[Bru08] Bruns, Andreas: Bewertung und Vergleich von Datenbanksystemen durch Leistungsmessung und Benchmarking, Diplomarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2008

[Jan07] Janssen, Larissa: Hochleistungs-Datenbanksysteme -Theorie und Praxis- , books-on-demand, 2008

[Jan07] Janssen, Larissa: Hochleistungs-Datenbanksysteme, Masterarbeit, Fernuniversität Hagen, 2007

[Kru07] Kruse, Johann: Entwicklung verteilter Informationssysteme auf der Basis von Open-Source basierten Datenbanksysteme, Bachelorarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2007

[Zyw06] Zyweck, Christian: Grundlagen und Design verteilter Datenbanksysteme für kommerzielle und Open-Source basierte Datenbankmanagementsysteme, Diplomarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2006

[Ech06] Echterhagen, Thomas: Entwicklung eines Data Marts meeresbiologischer Daten und deren Anbindung an ein GIS-System auf Basis einer verteilten Datenbank in Kooperation mit dem Forschungsinstitut Senckenberg Wilhelmshaven, Bachelorarbeit, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2006

[Wu06] Wulff, Alfred: Modellierung und Integration von verteilten Datenbanken, in Zwischenbericht zum Forschungsschwerpunkt „Applikationen für massiv parallele Rechnersysteme, Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven, 2006

VORTRÄGE UND PRÄSENTATIONEN

Autor(en)	Beschreibung	Datum
Wulff, Alfred Janssen, Larissa	Einsatz von Oracle RAC und MySQL Cluster im Rahmen eines Portierungsprojektes, Symposium „Cluster Computing“, 2009 Emden	Juni 2009
Benra, Juliane Wulff, Alfred	Der Forschungsschwerpunkt Applikationen für massiv parallele Rechencluster, Kolloquium für Informatik an Fachhochschulen, 2007 Elmshorn	März 2007
Wulff, Alfred Fankhänel, Sascha	Cluster Computing, Seminar des Instituts für Wirtschaftsinformatik, 2007 Wilhelmshaven	Mai 2007
Wulff, Alfred Janssen, Larissa	Anwendungen für verteilte Datenbanken, Symposium „Cluster Computing“, 2006 Emden	November 2006

GRUNDLAGEN

PROGNOSEN FÜR HARDWARETECHNOLOGIEN

Bei der Entwicklung moderner Hochleistungs-DBS sollten absehbare Fortschritte der Hardwaretechnologie berücksichtigt werden. Die Prognose der technischen Entwicklung ist relevant, da die Leistung und Kosteneffektivität moderner Hochleistungs-DBS oft durch die Hardwarekapazitäten eingedämmt werden. Unter *Kosteneffektivität* wird in diesem Bericht das Preis-/Leistungsverhältnis verstanden. Hohe Kosteneffektivität besitzt ein Datenbanksystem dann, wenn bei relativ geringem Gesamtpreis Hochleistung erbracht wird.

Seit Beginn der 90-er Jahre werden immer mehr auf Standard-Mikroprozessoren basierende Rechnersysteme im Bereich von Hochleistungs-DBS eingesetzt, deren Gesamtrechenleistung die Rechenleistung von Großrechnern übertreffen.[1] *Großrechner* (engl. *main frame*) sind für vielfältige Zwecke einsetzbare Computer der oberen Leistungs- und Preisklasse.[2] Ein *Rechnerverbund* auf Mikroprozessorenbasis besteht i.d.R. aus einer Gruppe von Standard-PC, die über ein Hochgeschwindigkeitsnetzwerk miteinander verbunden sind. Die Gesamtrechenleistung des Rechnerverbundes ist im Idealfall die Summe der Rechenleistungen der einzelnen Standard-PC. Durch die Parallelverarbeitung können sowohl geringe Kosten als auch enorme Leistungssteigerungen unmittelbar im Datenbankenbereich erzielt werden.[3]

Die stetig wachsende Leistung der Hardware soll sich bis zum Jahr 2047 fortsetzen.[4] Insbesondere werden in den nächsten Jahren weitere starke Fortschritte in Bereichen der Mikroprozessoren, Primär- und Sekundärspeicher sowie Netzwerktechnologie erwartet.

Hardwarelösungen von Herstellern wie IBM, Sun und HP stellen eine kostspielige Alternative zu den Rechnerverbünden mit Standardcomputern dar. Der Vorteil dieser kommerziellen Lösungen liegt darin, dass sie angepasste Speziallösungen für Hochleistungs-DBS anbieten. W. G. Spruth und E. Rahm haben im Jahr 2002 einen Bericht zu IBM Susplex-Cluster-Technologien veröffentlicht, in dem sie solche Implementierungsansätze genauer vorstellen.[5]

Im Weiteren werden Prognosen für die Entwicklung der folgenden Hardwaretechnologien besprochen:

- (a) Mikroprozessoren- und Speichertechnologien,
- (b) Netzwerktechnologie.

(a) Mikroprozessoren- und Speichertechnologien

[1] Vgl. Gelsinger, P.P. et al. (Microprocessors 1989) **zitiert nach** Rahm, E. (Transaktionssysteme 1993b), S. 56.

[2] Vgl. Claus, V., Schwill, A. (Fachlexikon 2003), S. 273 **vgl. dazu auch** Rahm, E. (Transaktionssysteme 1993b), S. 55.

[3] Vgl. Rahm, E. (Mehrrechner-DBS 1994), S. 314.

[4] Diese Prognosen stützen sich weitgehend auf Härder, Th. (DBMS 2002) sowie Rahm, E. (Transaktionssysteme 1993b), S. 55-58.

[5] Vgl. Spruth, W. G., Rahm, E. (Sysplex 2002).

Zurzeit steigt die Leistungswachstumsrate bei Mikroprozessoren jährlich um 50-100%, was vor allem auf die stark zunehmende Verbreitung der RISC [6] -Technologie zurückzuführen ist. Es wird vorausgesagt, dass diese Tendenz bis 2047 anhalten wird (siehe Abbildung 1.1).[7]

MIPS (*million instructions per second*, dt. „eine Million Operationen pro Sekunde“) ist eine Messeinheit für Rechengeschwindigkeit von Mikroprozessoren, die durch die Anzahl der pro Sekunde ausgeführten Operationen beschrieben wird. [8] Gängige Maßeinheitabkürzung für Mikroprozessoren ist als Beispiel: GOp/s = Giga Operationen pro Sekunde. Ein Mikroprozessor mit Geschwindigkeit 1 GOp/s führt also im Schnitt 1.000.000.000 (10^9) einfache Maschinenbefehle in der Sekunde aus.

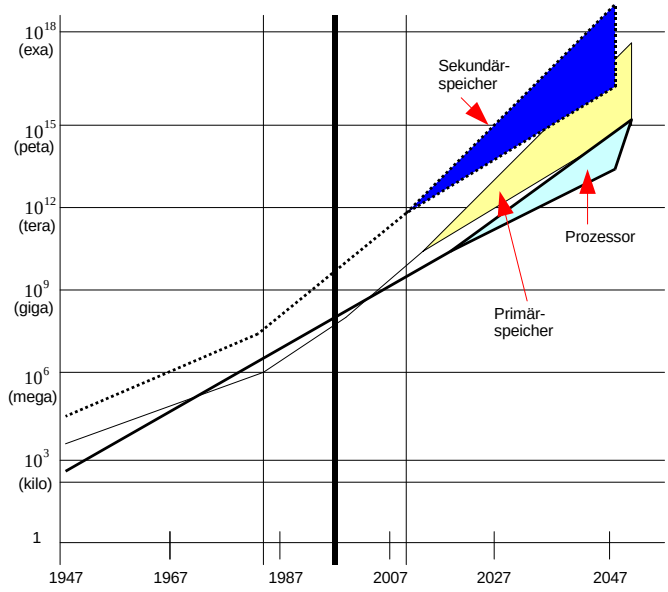


***Abbildung 1.1**: Voraussagen für Entwicklung von Computerhardware [Quelle: In Anlehnung an Härder, Th. (DBMS 2002)]*

Aus Abbildung 1.1 ist zu erkennen, dass bereits im Jahr 2017 ein Mikroprozessor die Geschwindigkeit (gemessen in MIPS) von mehr als 500 Giga-Operationen pro Sekunde (> 500 GOp/s) haben könnte. Im Jahr 2047 könnte die Geschwindigkeit von Mikroprozessoren zwischen Tera- und Peta-Op/s liegen. Derzeit liegt die maximale Geschwindigkeit bei 50 GOp/s.[9]

Eine ähnlich starke Entwicklung ist auch im Bereich des Primär- und Sekundärspeichers zu beobachten. Die Angaben der Speichergrößen in Abbildung 1.1 erfolgen in Bytes. Die Größe des Primärspeichers soll im Jahr 2027 im Terabyte- und im Jahr 2047 im Petabyte-Bereich liegen. Die Größe des Sekundärspeichers wächst ebenfalls sehr schnell und kann im Jahr 2047 über die Grenze des Exabyte-Bereiches hinausgehen.

---

[6] RISC: reduced instruction set computer, dt. „Computer mit verringertem Befehlsvorrat“. Vgl. Claus, V., Schwill, A. (Fachlexikon 2003), S. 566.
[7] Vgl. Härder, Th. (DBMS 2002), S. '1-6' - '1-10'.
[8] Vgl. Claus, V., Schwill, A. (Fachlexikon 2003), S. 411.
[9] Vgl. The Core 2 Duo Speed Tests, URLs: 'http://www.intel.com/ca/multiply/hub_core2duo_benchmarks_070307.pdf' vgl. dazu auch CPU Charts, 'http://www23.tomshardware.com/cpu.html?modelx=33&model1=430&model2=464&chart=158', Abrufe am 7.04.2007.

(b) Netzwerktechnologie

Darüber hinaus werden starke Fortschritte in der Netzwerk-Technologie erwartet (siehe Abbildung 1.2).

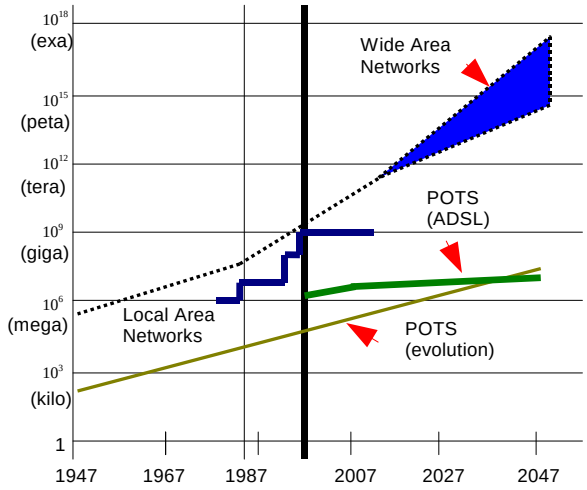


***Abbildung 1.2**: Voraussage für Entwicklung im Netzwerk-Bereich*
[Quelle: In Anlehnung an Härder, Th. (DBMS 2002)]

Laut Prognosen wird es in der Zukunft weltweite Netze (WAN) mit Geschwindigkeit bis zu mehreren Petabit/s geben. Schnellste Hochgeschwindigkeitsnetze für WAN haben zurzeit maximale Bandbreite von mehreren Gigabit/s, welche jedoch nur in wenigen Bereichen eingesetzt werden. Aus Kostengründen werden heute zumeist weltweite Netze mit Geschwindigkeiten von mehreren Mbit/s eingesetzt. Die Projektion für WAN-, LAN- und „plain old telephone service" (POTS)-Bandbreiten (in bits pro Sekunde) bis zum Jahr 2047 ist in Abbildung 1.2 zu sehen.

HOMOGENE, (SEHR) ENG INTEGRIERTE VERTEILTE DATENBANKSYSTEME

Homogene, eng integrierte verteilte Datenbanksysteme sind homogene, prä-integrierte verteilte Datenbanksysteme, die sich wie ein einziges DBMS verhalten. Ein solches System bietet nach außen ein Transaktionskonzept an. Es gibt nur ein globales Schema, in das die lokalen Schemata vollständig integriert sind.

Eine Sonderrolle dieser Kategorie nehmen die so genannten parallelen Datenbanksysteme ein. Es handelt sich hierbei um homogene, sehr eng integrierte verteilte Datenbanksysteme. Sie werden als sehr eng integriert bezeichnet, weil sie i.d.R. auf parallelen Rechnersystemen realisiert werden. Sie sind durch ein (sehr) hohes Potenzial zur Leistungssteigerung gekennzeichnet. Es gibt drei grundlegende Architekturen paralleler Datenbanksysteme:

- Shared-Everything-,
- Shared-Disks- und
- Shared-Nothing-Architekturen.

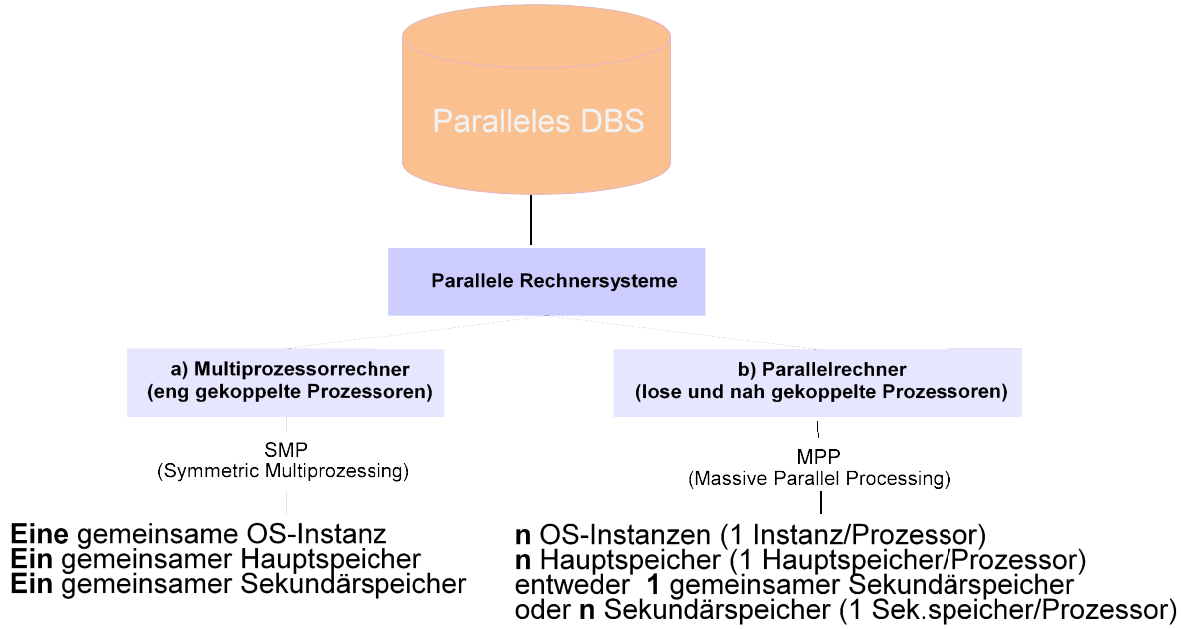
Nur die Shared-Nothing-Architektur stellt ein echtes verteiltes Datenbanksystem dar, da bei dieser Architektur Datenbanken über mehrere Knoten separat vorliegen. Im Fall der beiden anderen Architekturen wird eine Datenbank gemeinsam genutzt. Dennoch

zählen auch Shared-Everything- und Shared-Disks-Architekturen zu verteilten Datenbanksystemen:

> Shared-Disks-Systeme genügen der Definition eines verteilten Datenbanksystems und Shared-Everything-Systeme unterscheiden sich von den Systemen mit der Shared-Disks-Architektur hauptsächlich dadurch, dass Prozessoren nicht auf separate DB-Puffer sondern auf einen gemeinsamen DB-Puffer zugreifen. Da sich die Systeme ähnlich sind, weisen sie viele gleiche oder ähnliche Problemstellungen auf.

PARALLELE DATENBANKSYSTEME (DBS)

Parallele Datenbanksysteme basieren ein auf parallelen Rechnersystemen. Ein paralleles Rechnersystem kann verschiedene Architekturen in Bezug auf die Kopplung von Prozessoren mit Haupt- und mit Sekundärspeicher besitzen (siehe Abbildung 3.1).[10]



a) Multiprozessorrechner

Eng gekoppelte Prozessoren, auch *Symmetric Multiprocessing* (*SMP*) genannt, haben alle Zugriff zu den gemeinsam genutzten Hardwareressourcen wie Hauptspeicher und Sekundärspeicher. Es existiert für alle Prozessoren eine einzige Instanz des Betriebssystems. Dieser Ansatz gestattet eine effiziente Kommunikation zwischen Prozessoren über gemeinsame Hauptspeicherstrukturen. Die Erweiterbarkeit des Systems ist stark begrenzt, da mit wachsender Prozessoranzahl der Hauptspeicher oft zum Engpass wird. Es bestehen Verfügbarkeitsprobleme, da der Hauptspeicher nur eine geringe Fehlerisolation bietet und das Betriebssystem in einer Instanz vorliegt. Eine effektive Lastbalancierung wird i.d.R. durch das Betriebssystem unterstützt. Ein Computer mit eng gekoppelten Prozessoren wird als *Multiprozessorrechner* bezeichnet.

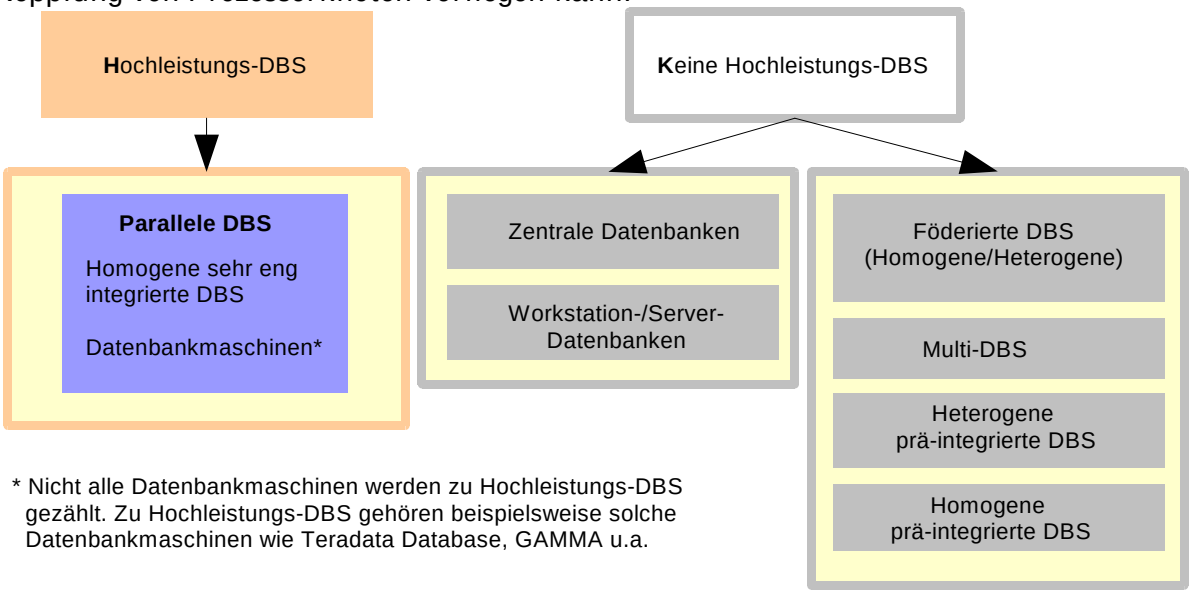
[10] Dieser Abschnitt stützt sich im Folgenden weitgehend auf Rahm, E. (Mehrrechner-DBS 1994), S. 30-34, 313-316.

b) Parallelrechner

Lose und nah gekoppelte *Prozessoren*, auch *Massive Parallel Prozessing* (*MPP*) genannt, haben ihre eigenen Hauptspeicher und Instanzen des Betriebssystems. Der Sekundärspeicher kann entweder durch alle Prozessorknoten gemeinsam genutzt werden oder jeder Prozessorknoten greift auf seinen eigenen Sekundärspeicher zu. Das Fehlen eines gemeinsamen Hauptspeichers erlaubt eine höhere Fehlerisolation und bessere Erweiterbarkeit. Die Interprozessorkommunikation und effektive Lastbalancierung sind jedoch wesentlich schwieriger als bei enger Kopplung. Die Interprozessorkommunikation geschieht per Nachrichtenaustausch. Beim Verteilen der Last muss neben der Vermeidung von Überlastungen auch eine Minimierung der Kommunikationsvorgänge unterstützt werden. Üblicherweise geht man bei losen und nah gekoppelten Prozessoren von *Parallelrechnern* aus, die aus hunderten oder tausenden von Prozessorknoten bestehen können.

Die nahe Kopplung unterscheidet sich von der losen Kopplung dadurch, dass bei der nahen Kopplung Prozessorknoten zusätzlich über einen gemeinsam genutzten Speicherbereich (z.B. Halbleiterspeicher) verfügen.[11] Mit der nahen Rechnerkopplung wird versucht, (starke) Leistungsverbesserungen des Systems zu erzielen. Ein gemeinsam genutzter Halbleiterspeicher kann als Beispiel für einen sehr schnellen Nachrichtenaustausch zwischen Prozessorknoten verwendet werden.

Parallelrechner können komplexe hierarchische Architekturen aufweisen.[12] Beispielsweise können einzelne Prozessorknoten des Parallelsystems als Multiprozessorrechner realisiert sein (d.h., dass innerhalb eines Prozessorknotens eine enge Prozessorkopplung besteht), während auf der Ebene des Parallelrechners eine lose Kopplung von Prozessorknoten vorliegen kann.



* Nicht alle Datenbankmaschinen werden zu Hochleistungs-DBS gezählt. Zu Hochleistungs-DBS gehören beispielsweise solche Datenbankmaschinen wie Teradata Database, GAMMA u.a.

Zu den Hochleistungs-DBS zählen also zurzeit homogene, sehr eng integrierte DBS und wenige Datenbankmaschinen wie beispielsweise Teradata Database, GAMMA u.a. Voraussichtlich können zukünftig wesentlich mehr Datenbankmaschinen als Hochleistungs-DBS auf Grund der Fortschritte in der Entwicklung der Hardwaretechnologien realisiert werden.[13] Diese Entwicklung ist bei Datenbankmaschinen jedoch anscheinend noch nicht eingetreten.

[11] Vgl. zu diesem Absatz Rahm, E. (Transaktionssysteme 1993b), S. 265-266.
[12] Dieser Absatz in Anlehnung an Sokolinsky, L.B. (Architectures 2004), S. 341.
[13] Vgl. Härder, T., Rahm, E. (DBS 1999), S. 542-543.

KLASSIFIKATION DER ARCHITEKTUREN VON PARALLELEN DBS

Es werden zuerst Abbildungsverfahren der Datenbanksystemarchitekturen auf Architekturen von Rechnersystemen vorgestellt. Es erfolgt die Beschreibung des Konzepts von virtuellen parallelen Datenbankmaschinen. Anschließend werden Basis- und hierarchische Architekturen paralleler Datenbanksysteme dargestellt.

ABBILDUNGSVERFAHREN DER DBS-ARCHITEKTUREN

In diesem Abschnitt werden ein direktes und ein indirektes Abbildungsverfahren der Architekturen von parallelen Datenbanksystemen auf Hardwarearchitekturen vorgestellt.

Direktes Abbildungsverfahren

Beim *direkten Abbildungsverfahren* wird die Architektur eines parallelen Datenbanksystems unmittelbar auf die gegebene Architektur des parallelen Rechnersystems abgebildet, so wie es in Abbildung 3.7 gezeigt ist.[14]

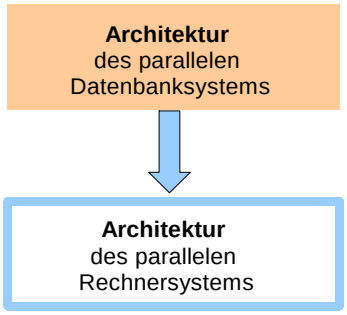


Abbildung 3.7:
Direkte Abbildung
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

In der gesichteten Literatur zu verteilten und parallelen Datenbanksystemen wird die Trennung der Rechner- und Datenbanksystemarchitekturen nicht hervorgehoben.[15] Es wird generell vorausgesetzt, dass die Architektur eines parallelen Datenbanksystems mit der gegebenen Hardwarearchitektur „verschmilzt“. Für ein Datenbanksystem bedeutet dies, dass es i.d.R. dieselbe Architektur besitzen muss wie das Rechnersystem, auf dem das Datenbanksystem aufgesetzt ist.

Eine direkte Abbildung der Architekturen paralleler Datenbanksysteme auf Architekturen paralleler Rechnersysteme erweist sich dann als schwierig, wenn parallele Rechnersysteme hierarchische Hardwareachitekturen besitzen. Sokolinsky versucht dieses Problem zu lösen, indem er eine indirekte Abbildung der Architekturen vornimmt. Dieses Abbildungsverfahren wird im Folgenden erläutert.

Indirektes Abbildungsverfahren nach Sokolinsky

Bei dem *indirekten Abbildungsverfahren* wird die Architektur eines parallelen Datenbanksystems auf die Architektur einer virtuellen parallelen Datenbankmaschine abgebildet, welche wiederum auf eine bestimmte Hardwarearchitektur des parallelen

[14] Dieser Abschnitt stützt sich weitgehend auf Sokolinsky, L.B. (Architectures 2004), S. 337-338.
[15] Vgl. Dadam, P. (Datenbanken 1996), KE 1 vgl. dazu auch Rahm, E. (Mehrrechner-DBS 1994), S. 314 vgl. dazu auch Ötzu, M. T., Valduriez, P. (DDBS 1999), S. 425-429.

Rechnersystems oder auf eine andere darunterliegende virtuelle parallele Datenbankmaschine projiziert wird. [16] Die vorgeschlagene mehrschichtige Architekturabbildung nach Sokolinsky ist in Abbildung 3.8 zu sehen.

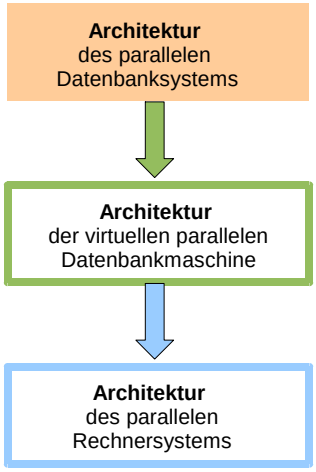


Abbildung 3.8:
Indirekte Abbildung
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

Dieses Verfahren ist also besonders bei der Abbildung von Architekturen paralleler Datenbanksysteme geeignet, die auf moderne hierarchische parallele Rechnerarchitekturen aufbauen.

Im Weiteren wird das Konzept der virtuellen parallelen Datenbankmaschine beschrieben, auf welchem das indirekte Abbildungsverfahren basiert.

Virtuelle parallele Datenbankmaschinen

Sokolinsky entwickelte das Konzept der parallelen virtuellen Datenbankmaschinen zur Abbildung von Architekturen paralleler Datenbanksysteme auf Hardwarearchitekturen. [17] Eine virtuelle parallele Datenbankmaschine stellt eine Abstraktion der Architektur des parallelen (hierarchischen) Rechnersystems dar. In der Praxis wird die Abstraktion mithilfe von speziell entworfenen Programmen des DBMS realisiert.

Eine virtuelle parallele Datenbankmaschine nach Sokolinsky besteht aus folgenden virtuellen Einheiten:

- virtuelle Prozessoren,
- virtuelle Hauptspeicherbereiche,
- virtuelle Platten und
- virtuelles Kommunikationsnetzwerk.

Virtueller Prozessor

Ein *virtueller Prozessor* (engl. *virtual processor*) ist eine virtuelle Einheit zur Erledigung von separaten Aufträgen des Datenbanksystems. Ein typisches Beispiel für einen

[16] Vgl. zu diesem Abschnitt Sokolinsky, L.B. (Architectures 2004), S. 338.
[17] Dieser Abschnitt basiert auf Sokolinsky, L.B. (Architectures 2004), S. 338-339.

Auftrag ist eine (Teil-)Anfrage. Ein virtueller Prozessor kann nur dann auf eine Relation zugreifen, wenn sie sich in einem für ihn zugänglichen virtuellen Hauptspeicherbereich (siehe unten) befindet.

Virtueller Hauptspeicherbereich

Ein *virtueller Hauptspeicherbereich* (engl. *virtual memory module*) ist eine virtuelle Einheit, die zur Pufferung von DB-Objekten eingesetzt wird. Der virtuelle Speicherbereich eines Datenbanksystems entspricht einem oder mehreren getrennten Bereich(en) der verfügbaren physischen Hauptspeicher eines oder mehrerer Rechnerknoten. Ein physischer Hauptspeicherbereich kann einem oder mehreren virtuellen Hauptspeichern zugeteilt werden.

Virtuelle Platte

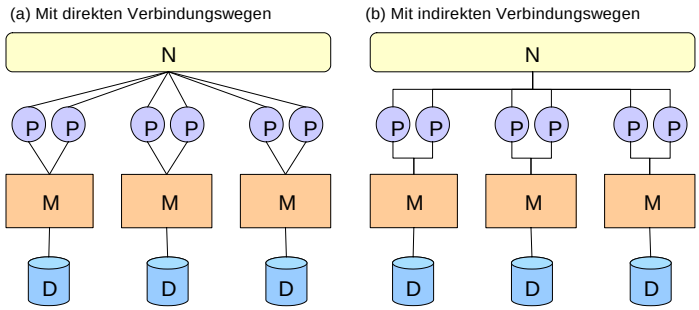
Eine *virtuelle Platte* (engl. *virtual disk*) ist eine virtuelle Einheit zur persistenten Speicherung von DB-Objekten. Die virtuelle Platte eines Datenbanksystems wird gewöhnlich entweder durch eine bzw. mehrere Festplatte(n) oder Plattenfelder realisiert.

Virtuelles Kommunikationsnetzwerk

Ein *virtuelles Kommunikationsnetzwerk* (engl. *virtual communication network*) ist eine virtuelle Einheit zur Übertragung von Daten von einem virtuellen Hauptspeicherbereich zu einem anderen. Der Datentransfer kann nur durch Kommunikationsaktivitäten der korrespondierenden virtuellen Prozesse geschehen. Generell wird davon ausgegangen, dass eine virtuelle parallele Datenbankmaschine (siehe unten) nicht mehr als ein virtuelles Kommunikationsnetzwerk besitzt. Falls ein Datenbanksystem über einen gemeinsam genutzten Hauptspeicherbereich verfügt, enthält dieses Datenbanksystem naturgemäß kein virtuelles Kommunikationsnetzwerk, da die Interprozessorkommunikation direkt über den Hauptspeicher erfolgt.

Virtuelle parallele (Datenbank-)maschine

Eine *virtuelle parallele (Datenbank-)Maschine* (engl. *virtual parallel (database) machine*) ist durch einen ungerichteten Graphen definiert, dessen Knoten unterschiedliche virtuelle Einheiten und dessen Kanten den möglichen Informationsfluss darstellen. Ein Beispiel für zwei änliche virtuelle parallele Maschinen mit separaten, direkten (a) und gemeinsam genutzen, indirekten (b) Kanten ist in Abbildung 3.9 gezeigt.



Legende: **P** – ein virtueller Prozessor, **M** – ein virtueller Hauptspeicher, **D** – eine virtuelle Platte, **N** – ein virtuelles Kommunikationsnetzwerk

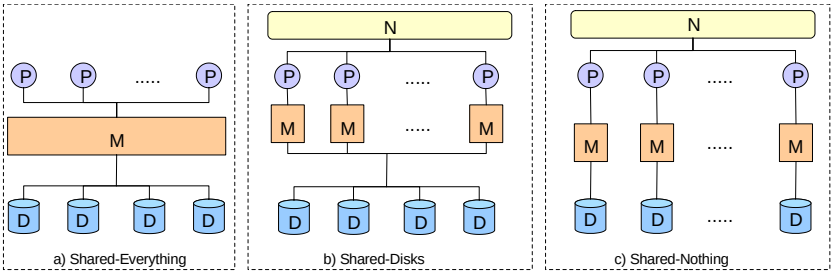
Abbildung 3.9: *Beispiel für eine virtuelle parallele Maschine*
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

Zwischen der gegebenen Architektur der Rechnersysteme und der gewünschten Architektur eines parallelen Datenbanksystems müssen gegebenenfalls Hardwarekomponenten virtuell zu einer einzigen logischen Komponente vereinigt werden. In Abhängigkeit von den Architekturen der beiden Systeme kann/können eine oder mehrere virtuelle Datenbankmaschine(n) entstehen. Jede virtuelle Datenbankmaschine repräsentiert eine Abstraktion der jeweiligen logisch zusammengefassten Hardwarekomponente. Eine virtuelle Maschine nutzt Funktionen und Dienste der virtuellen Maschine der darunterliegenden Schicht.

Basisarchitekturen

Abbildung 3.11 zeigt das meist verbreitete Verfahren zur Klassifizierung der parallelen Datenbanksysteme, welches von M. Stonebraker vorgeschlagen wurde.[18]

Nach Stonebraker´s Klassifikation können parallele Datenbanksysteme in folgende drei grundlegende Klassen, in Abhängigkeit von der gemeinsamen Nutzung der wichtigsten Hardwareressourcen wie Prozessoren, Hauptspeicher und Sekundärspeicher, aufgeteilt werden:[19]



Legende: **P** – Prozessor, **M** – Hauptspeicher, **D** – Sekundärspeicher, **N** – Netzwerk

Abbildung 3.11: *Klassifikation nach Stonebraker*
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

(a) *SE* (*Shared-Everything*) - Architektur, in der Hauptspeicher und Plattensubsystem von Prozessoren gemeinsam genutzt werden (siehe Abbildung 3.11 (a));

[18] Vgl. Stonebraker, M. (SN 1986).
[19] Vgl. zu dieser Auflistung (a-c) Stonebraker, M. (SN 1986) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 339-340.

(b) *SD* (*Shared-Disks*) - Architektur, in der nur das Plattensubsystem von Prozessoren gemeinsam genutzt wird (siehe Abbildung 3.11 (b));

(c) *SN* (*Shared-Nothing*) - Architektur, in der es keine gemeinsam genutzten Ressourcen gibt (siehe Abbildung 3.11 (c)).

Hierarchische Architekturen

Zu Beginn dieser Ausführungen werden zweischichtige hierarchische Architekturen (CE- und CD-Architekturen) als Erweiterung der grundlegenden Klassifikationsarten paralleler Datenbanksysteme nach Copeland und Keller vorgestellt. Danach wird die dreischichtige hierarchische CDN-Architektur nach Sokolinsky beschrieben.

Seit dem Einsatz hierarchischer paralleler Rechnersysteme werden oft darauf implementierte parallele Datenbanksysteme auch als *Cluster-Datenbanksysteme* bezeichnet. So verwendet Rahm in seinen jüngsten Veröffentlichungen die Bezeichnung Cluster- statt parallele Datenbanksysteme.[20]

Clustered-Everything-, Clustered-Disk-Architektur

Auf der Stonebraker-Klassifikation basieren viele wissenschaftliche Arbeiten, die der Analyse von Architekturen paralleler Datenbanksysteme gewidmet sind. Diese Klassifikation gilt seit einiger Zeit als überholt und nicht mehr adäquat.[21] Schon im Jahr 1996 nannten M.G. Norman et al. hierzu folgende Argumente:[22]

(1) Die Stonebraker-Klassifikation schließt nicht alle Variationen existierender Architekturen ein: Es gibt z.B. parallele Rechnersysteme, die Merkmale mehrerer Architekturen gleichzeitig besitzen;

(2) Ein Klassifikationsverfahren, das sich direkt auf (gemeinsam genutzte) Hardwareressourcen stützt, reicht bei der Abbildung von hierarchischen parallelen Systemen nicht aus.

Zur Beschreibung der modernen parallelen Datenbanksysteme schlagen G.P. Copeland&T. Keller die Einführung der zwei zusätzlichen Architekturen (siehe Abbildung 3.12) vor:[23]

(a) *CE* (*Clustered-Everything*) – Architektur mit mehreren SE-Clustern, die nach dem Shared-Nothing-Prinzip miteinander verbunden sind (siehe Abbildung 3.12 (a));

(b) *CD* (*Clustered-Disk*) – Architektur mit mehreren SD-Clustern, die nach dem Shared-Nothing-Prinzip miteinander verbunden sind (siehe Abbildung 3.12 (b)).

[20] Vgl. das Beispiel aus Spruth, W. G., Rahm, E. (Sysplex 2002), S. 18.
[21] Vgl. Sokolinsky, L.B. (Architectures 2004), S. 340.
[22] Vgl. zu dieser Aufzählung (1-2) Norman, M.G. et al. (SN 1996), S. 16-21 zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 337, 340.
[23] Vgl. zu dieser Auflistung (a-b) Copeland, G., Keller, T. (Comparison 1989), S. 98-109 zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 340.

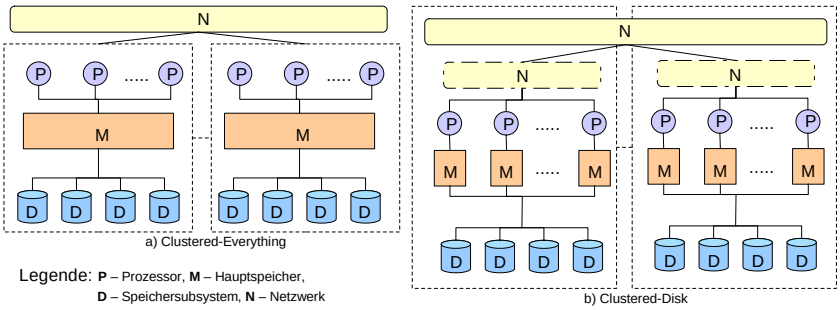


Abbildung 3.12: *Zweischichtige hierarchische Architekturen*
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

Die CE- und CD-Architekturen basieren auf hierarchischen Architekturen paralleler Rechnersysteme und werden ebenfalls als *hierarchische (geclusterte) DBS-Architekturen* bezeichnet.[24] Abbildung 3.12 (a) zeigt beispielsweise ein geclustertes Datenbanksystem, das aus zwei SE-Clustern besteht, wobei ein SE-Cluster mehr als einen Prozessor besitzt. Das geclusterte Datenbanksystem aus der Abbildung hat eine zweischichtige Architektur. Die erste Schicht enthält die SE-Architektur innerhalb der einzelnen Cluster und die zweite Schicht die SN-Architektur oberhalb der Cluster.

Hierarchische Datenbanksysteme besitzen Eigenschaften mehrerer Architekturen gleichzeitig: Innerhalb von Clustern selbst gelten die Eigenschaften der zugrunde liegenden Architektur (z.B. ein SD-Cluster besitzt die Eigenschaften der SD-Architektur) und außerhalb der Cluster wirken die Eingenschaften der übergeordneten Architektur. Die hierarchischen CE- und CD-Architekturen haben (wie bereits angegeben) in der obersten Schicht die SN-Architektur und weisen deshalb deren Merkmale auf.

Bei geclusterten parallelen Datenbanksystemen erfolgt die Interprozesssorkommunikation innerhalb eines Clusters je nach Architektur gewöhnlich über einen gemeinsam genutzten Hauptspeicher oder über ein lokales Hochgeschwindigkeitsnetzwerk. Die Kommunikation zwischen den einzelnen Clustern geschieht über ein globales, schnelles Netzwerk.[25]

Die zweischichtigen Hierarchien wurden bereits in mehreren Arbeiten untersucht und beschrieben.[26] Beispiele der modernen parallelen Rechnersysteme sind Systeme der russischen MBC-100/1000-Serie[27], das SP2-System[28] von IBM, computerbasierte Systeme auf der ServerNet-Technologie von Tandem[29] und andere.

Clustered-Disks-Nothing-Architektur

Die *CDN (Clustered-Disks-Nothing)-Architektur* ist eine dreischichtige hierarchische Architektur paralleler Datenbanksysteme, die in der unteren Schicht aus einem Satz von SD-Clustern und in der oberen Schicht aus einem Satz von SN-Clustern besteht. Die einzelnen SN-Cluster sind nach SN-Prinzip miteinander verbunden. Diese Architektur

[24] Vgl. Graefe, G. (Query 1993) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 340.
[25] Vgl. Sokolinsky, L.B. (Architectures 2004), S. 340.
[26] Vgl. Hua, K.A., Lee, C., Peir, J.-K. (Interconnecting 1991), S. 262-270 vgl. dazu auch Pramanik, S., Tout, W.R. (NUMA 1997) vgl. dazu auch Copeland, G., Keller, T. (Comparison 1989) vgl. dazu auch Bouganim, L. et al. (Loadbalancing 1996) vgl. dazu auch Ötzu, M. T., Valduriez, P. (DDBS 1999) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 341.
[27] Vgl. Korneev, V.V. (Parallelrechner 1999) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 341.
[28] Vgl. Shmidt, V. (SP2 1995) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 341.
[29] Vgl. Shnitman, V. (ServerNet 1996) zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 341.

basiert auf dem Vorgehen, welches Rahm in seiner Arbeit im Jahr 1992 vorgeschlagen hat.[30] Abbildung 3.13 zeigt ein Beispiel für die hierarchische CDN-Architektur paralleler Datenbanksysteme.

Mit dieser Architektur wurden Problematiken der Synchronisation von konkurrierenden Zugriffen und der Cache-Kohärenzkontrolle beseitigt, die in der unteren Schicht in dem Satz mit SD-Clustern bestehen.[31]

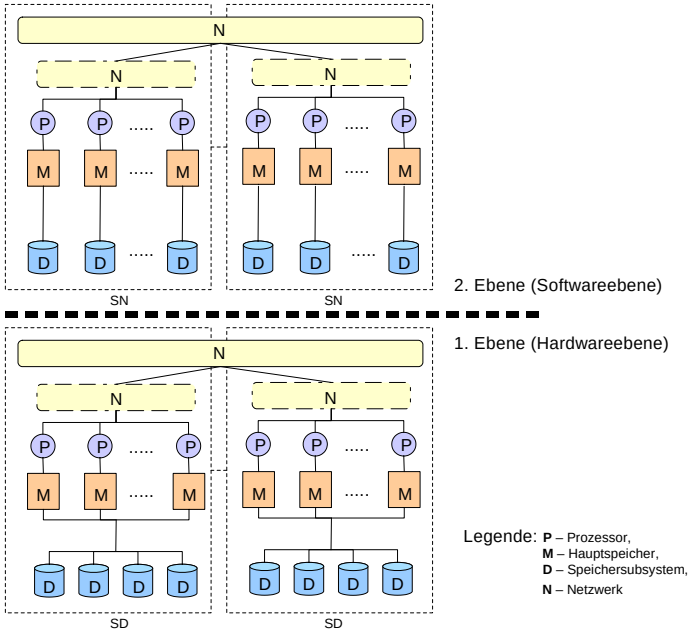


***Abbildung 3.13**: Hierarchische CDN-Architektur*
[Quelle: In Anlehnung an Sokolinsky, L.B. (Architectures 2004)]

Die geclusterte CE-Architektur steht in Konkurrenz zu der CDN-Architektur. Die Probleme der CE-Architektur sind hauptsächlich mit Hochverfügbarkeit, Synchronisation und Cache-Kohärenzkontrolle für CPUs verbunden und führen zu schlechteren Ergebnissen.

Die CDN-Architektur wurde von Sokolinsky in einem Forschungsprojekt für einen Prototyp des parallelen Datenbanksystems Omega eingesetzt, das auf einem parallelen Rechnersystem MBC-100/1000 basierte.[32] Sokolinsky führte verschiedene Erprobungen auf diesem Datenbanksystem durch und bestätigte die meisten Ergebnisse seiner Vergleichsanalyse.

Die Ermittlung der existierenden parallelen Datenbanksysteme hat ergeben, dass die Forschungsschwerpunkte im Bereich der Shared-Nothing-Architektur liegen. Im proprietären kommerziellen Bereich dagegen wird neben der Shared-Nothing- oft die Shared-Disks-Architektur implementiert. Es gibt im Vergleich zu anderen Alternativen relativ wenige freie und Open-Source-Produkte. Diese Lösungen existieren allerdingst für alle Basisarchitekturen. Hierarchische Architekturen haben im Vergleich zu Basisarchitekturen noch keine starke Verbreitung gefunden. Man kann davon ausgehen, dass die CE-Architektur in der Praxis am häufigsten vorkommt.

[30] Vgl. Rahm, E. (Framework 1992), S. 171–19 zitiert nach Sokolinsky, L.B. (Architectures 2004), S. 341.
[31] Vgl. Valduriez, P. (Parallel DBS 1993), S. 460–465.
[32] Vgl. Sokolinsky, L.B. (Architectures 2004), S. 344.

REALISIERUNGEN VON ORACLE-RAC

Oracle Real Application Clusters (kurz *RAC*) ist ein paralleles Datenbanksystem, welches laut der Oracle RAC-Dokumentation die Shared-Everything (SE)-Architektur besitzt.[33] In der Fachliteratur zu RAC wird dagegen häufig geschrieben, dass es sich um ein paralleles Datenbanksystem mit der Shared-Disk (SD)-Architektur handelt.[34] Beim Studieren der Architektur des RAC ist jedoch aufgefallen, dass die Angaben der SE- oder

SD-Architekturen nicht ganz zutreffend sind. Wenn das auf der Notation der virtuellen parallelen Datenbankmaschine basierte Abbildungsverfahren von Sokolinsky angewendet wird, dann entspricht die Architektur des RAC nicht der SE- oder SD-, sondern einer Shared-Disks-Everything (SDE)-Architektur (siehe Abbildung 4.1).

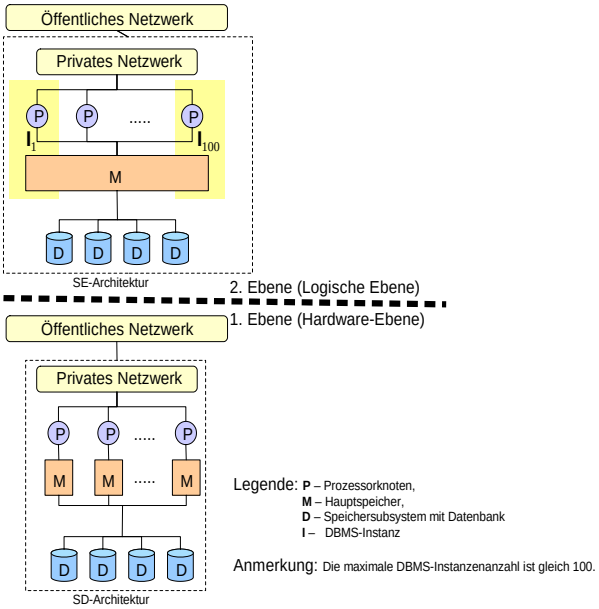


Abbildung 4.1: Die SDE-Architektur des RAC

In der SDE-Architektur besitzt die physische Ebene eine SD- und die logische Ebene eine SE-Architektur. Auf der logischen Ebene wird der virtuelle Hauptspeicher gemeinsam genutzt. Diese Vereinigung der Hauptspeicher wird im RAC mittels der Cache Fusion-Technologie realisiert.

SKALIERBARKEIT

Zurzeit werden in einer RAC-Umgebung bis zu 100 DBMS-Instanzen unterstützt. Dadurch ist die Skalierbarkeit im RAC zumindest theoretisch begrenzt.[35] Da die einzelnen DBMS-Instanzen ihre Daten nicht komplett im Hauptspeicher halten müssen, sind im RAC relativ große Datenmengen aus dieser Sicht unproblematisch.[36] Im RAC können alle Daten im Hauptspeicher gehalten werden, wenn die Größe des Hauptspeichers es zulässt. Die Daten werden jedoch i.d.R. nicht sofort, sondern erst beim Zugriff auf sie in den Hauptspeicher geladen.

[33] Vgl. Oracle (RAC-Dokumentation 2006).
[34] Vgl. Held., A. (RAC 2005), S. 209 vgl. dazu auch Lockemann, P.C., Dittrich, K.R. (DBMS 2004), S. 246.
[35] Vgl. Oracle (RAC-Dokumentation 2006), S. '1-5'.
[36] Vgl. Oracle (Dokumentation 2007).

Die Verknüpfung sehr vieler Rechner mit allen Platten ist wie bei allen auf SD basierten Architekturen aufwändig.

HOCHVERFÜGBARKEIT

RAC garantiert eine hohe Verfügbarkeit des Datenbanksystems. Der Ausfall von einzelnen DBMS-Instanzen wird für Benutzer mittels des so genannten Transparent Application
Failover (TAF) transparent gehalten. Je nach Konfiguration kann ferner eine auf einem Knoten gestartete Anfrage nach dem Ausfall des Knotens auf einem anderen Knoten fortgesetzt werden.[37] Insgesamt ist der Aufwand für den Wiederanlauf von ausgefallenen Knoten gering.

RAC-Konfiguration sieht die Nutzung der redundanten Hardware vor.[38] Beispielsweise können mehrere Netzwerkadapter für dieselbe Aufgabe eingesetzt werden. Fällt ein Netzwerkadapter aus, so wird die Kommunikation ohne Unterbrechung auf einem anderen Netzwerkadapter fortgesetzt. Des Weiteren können Disk-Arrays eingesetzt werden, die i.d.R. eine hohe Fehlertoleranz für persistente Daten ermöglichen.

Um den Ausfall des gesamten RAC in einem Katastrophenfall (z.B. Brand im Maschinenraum, Naturkatastrophen o.ä.) zu vermeiden, empfiehlt Oracle folgende Lösung: den Einsatz des RAC in Kombination mit einem zusätzlichen Standby-Datenbanksystem. Der Ort für das Standby-Datenbanksystem muss je nach vermuteter Gefahr entsprechend gewählt werden. In einer *Standby-Datenbank* wird eine transaktionskonsistente Kopie der aktiven Datenbank auf einem vollständig unabhängigen Rechnersystem gehalten. Die Standy-Datenbank ist dabei betriebsbereit, sodass beim Ausfall der aktiven Datenbank die Funktionalität an die Standby-Datenbank schneller übertragen wird.[39]

LASTBALANCIERUNG

Bei der SDE-Architektur des RAC bestehen i.Allg. gute Möglichkeiten der Lastbalancierung auf Basis der zugrunde liegenden SD-Architektur. So können DB-Operationen theoretisch auf jedem Knoten bearbeitet werden, da alle Knoten gleichermaßen Zugriff auf die gemeinsame Datenbank besitzen.

Darüber hinaus bietet Oracle zusätzliche Softwarelösungen für die Verteilung der ankommenden Lasten an. Es gibt in RAC zwei verschiedene Alternativen der Lastbalancierung. Die Lastbalancierung kann (a) auf der Seite der Benutzer (*clientseitig*) und/oder (b) auf der Seite des Real Application Clusters (*serverseitig*) vorgenommen werden.[40]

(a) Clientseitige Lastbalancierung

Bei der Verwendung der *clientseitigen* Lastbalancierung stellt ein Benutzer eine Verbindung zu einer DBMS-Instanz des Clusters nach dem so genannten Round-Robin-Prinzip her.[41] Beim *Round-Robin-Prinzip* von Oracle werden die Verbindungen zu einzelnen Knoten des Clusters per Zufall ausgewählt. Die Liste der Clusterknoten ist in

[37] Vgl. Oracle (Network 2005), S. '13-13'.
[38] Vgl. Oracle (RAC-Installation 2006), S. '1-4'.
[39] Vgl. Oracle (Data Guard 2006) vgl. dazu auch. Oracle (RAC-Dokumentation 2006), S. '2-9', '2-21'.
[40] Vgl. Held., A. (RAC 2005), S. 215-216.
[41] Vgl. zu diesem Absatz Held., A. (RAC 2005), S. 215-216.

einer Konfigurationsdatei auf der Seite des Benutzers abgelegt. Beim Herstellen einer Verbindung zu der RAC-Umgebung wird zuvor auf diese Datei zugegriffen und eine DBMS-Instanz ausgewählt.

Das Round-Robin-Prinzip stellt kein echtes Verfahren zur Lastbalancierung dar, denn bei Lastbalancierung geht es darum, die Anfrage- und Transaktionslast über die Knoten des Clusters gleichmäßig zu verteilen. Wird aber von einem Benutzer per Zufall gerade ein Knoten ausgewählt, welcher stark ausgelastet ist, so hat dieses Verfahren die ungleiche Lastverteilung noch mehr vergrößert. Deshalb kann dieses Verfahren nur als Ergänzung zu dem serverseitigen Verfahren angesehen werden. Es kann ausschließlich für die Balancierung der eigenen Anfragen sorgen. Statt alle Anfragen an denselben Listener zu senden, werden die Anfragen an verschiedene Listener verteilt und damit auf der Benutzerseite für eine aus Sicht des Benutzers ausgewogene Anfragenverteilung gesorgt. *Listener* ist ein Prozess, der auf Verbindungsanforderungen von Benutzerprozessen wartet.[42] Beim Eintreffen einer Verbindungsanforderung wird vom Listener i.d.R. eine für die Sitzungsdauer feste Verbindungsadresse zwischen dem Benutzer und der DBMS-Instanz festgelegt.

(b) Serverseitige Lastbalancierung

Bei der *serverseitigen* Lastbalancierung wird ein Benutzer in Abhängigkeit von der Auslastung der einzelnen Knoten des Clusters mit einer DBMS-Instanz verbunden.[43] Standardmäßig wird der Knoten mit der geringsten Gesamtauslastung ausgewählt. Zur Bewertung der Auslastung eines Knotens werden folgende Informationen herangezogen:

1. Informationen zur aktuellen Arbeitslast des Knotens,
2. Informationen zu der aktuellen Auslastung der DBMS-Instanz,
3. Informationen zur Arbeitslast von Dispatchern, im Fall der gemeinsamen Nutzung eines Serverprozesses. *Dispatcher* sind Prozesse, die es ermöglichen, dass ein Serverprozess durch mehrere Benutzer gemeinsam genutzt werden kann.[44] Das gilt für eine Multithread-Serverkonfiguration.

Alle Informationen zur Auslastung der einzelnen Knoten werden von Oracle Listener verwaltet. Oracle Listener sind zusätzlich für die Koordination der serverseitigen Balancierung zuständig. Ein Benutzer stellt zunächst die Verbindung zu einem der Listener her. Der Listener sorgt anschließend dafür, dass der Benutzer sich mit der DBMS-Instanz verbindet, die am wenigsten ausgelastet ist. Die serverseitige Lastbalancierung verursacht i.d.R. einen geringen zusätzlichen Kommunikationsaufwand.

INTERPROZESSORKOMMUNIKATION

Im RAC muss jeder Knoten mindestens zwei Netzwerkkarten besitzen.[45] Die IP-Adressen der Netzwerkkarten müssen verschiedenen Subnetzen angehören. Im RAC wird zwischen einem privaten und einem öffentlichen Netzwerk unterschieden. Die Kommunikation mit Benutzern bzw. Anwendungen geschieht über das *öffentliche* Netzwerk. Die interne Kommunikation zwischen den DBMS-Instanzen des Clusters verläuft über das *private* Netzwerk.

[42] Vgl. Oracle (Network 2005), S. '1-16'.
[43] Vgl. zu diesem Absatz Oracle (Network 2005), S. '3-7' - ' 3-8'.
[44] Vgl. Oracle (Network 2005), S. '3-8'.
[45] Dieser Abschnitt stützt sich weitgehend auf Oracle (RAC-Installation 2006), S. '1-3' - '1-4'.

Über das private Netzwerk werden hauptsächlicht DB-Blöcke versendet. *DB-Block* ist ein Synonym für eine DB-Seite. Die Bezeichung DB-Block wird bei der Oracle Beschreibung des RAC verwendet. Für den Versand der DB-Blöcke ist der Cache Fusion-Mechanismus zuständig, auf den später noch näher eingegangen wird. Die Bandbreite des privaten Netzwerkes hat eine unmittelbare Auswirkung auf die Leistung des RAC. Aus diesem Grund verlangt Oracle den Einsatz eines Hochgeschwindigkeitsnetzwerkes (ab einer Bandbreite von 1Gbit/s) für das private Netzwerk.[46] Für das öffentliche Netzwerk gelten dieselben Empfehlungen wie im Fall eines zentralen Datenbanksystems.[47]

Cache-Kohärenzkontrolle

Im Real Application Clusters werden physisch getrennte DB-Puffer wie ein gemeinsam genutzter DB-Puffer benutzt. Die Implementierung dieses Ansatzes geschieht mithilfe der so genannten *Cache Fusion-Technologie*.[48] Mit diesem Mechanismus wird versucht, den Problematiken wie Cache-Cohärenzkontrolle und Synchronisation der zugrunde liegenden SD-Architektur des Datenbanksystems entgegenzuwirken. Dadurch, dass ein DB-Block auf Basis der logischen SE-Architektur nicht von vielen Instanzen gleichzeitig geändert werden kann, entfällt der zusätzliche Aufwand zur Pufferinvalidierung (Cache-Kohärenzkontrolle).

Die Cache Fusion wird hauptsächlich mittels zweier Dienste (Global Cache Service und Global Enqueue Service) realisiert. Bei dem *Global Cache Service* (*GCS)* handelt es sich um einen globalen Dienst, welcher die Informationen zu benutzten DB-Blöcken aller DBMS-Instanzen des Clusters verwaltet und für den Transfer von DB-Blöcken zwischen den DBMS-Instanzen zuständig ist. Alle Informationen zu genutzten DB-Blöcken trägt dieser Dienst in dem globalen Ressourcenkatalog (siehe unten) ein.

In der RAC-Umgebung müssen diverse Bereiche der Datenbank durch globale Sperren (Enqueues) geschützt werden. Hierzu gehören beispielsweise Datenbankkataloge, also Bereiche des Hauptspeichers, die für die interne Verwaltung des DBMS gedacht sind (z.B. Library-Cache). Die Verwaltung von globalen Ressourcen im Sinne des sperrenden/exklusiven Zugriffs und die nachfolgende Freigabe sowie die Erkennung und entsprechende Auflösung von globalen Verklemmungen übernimmt der *Global Enqueue Service (GES)*.

Die GCS- und GES-Dienste werden selbst als verteilte Dienste realisiert. Jede lokale DBMS-Instanz besitzt vier zusätzliche Prozesse, die bei der Synchronisation der gleichzeitigen Zugriffe und Cache Fusion eingesetzt werden. Dazu gehören folgende Dienste:[49]

- LMS (Lock Manager Service, auch Global Cache Service Process genannt),
- LMD (Lock Manager Daemon, auch Global Enqueue Service Daemon genannt),
- LMON (Lock Monitor, auch Global Enqueue Service Monitor genannt),
- LCK (Lock Process, auch Instance Enqueue Process genannt).

[46] Vgl. Oracle (RAC-Dokumentation 2006), S. '1-2'.
[47] Vgl. Held., A. (RAC 2005), S. 223.
[48] Dieser Abschnitt stützt sich weitgehend auf Oracle (RAC-Dokumentation 2006), S.'1-6' sowie Lockemann, P.C., Dittrich, K.R. (DBMS 2004), S. 246-247.
[49] Vgl. Oracle (RAC-Dokumentation 2006), S. '1-7'.

Der LMS-Dienst ist für die lokale Sperrverwaltung verantwortlich. LMD und LMON überwachen und stellen globale Dienste wieder her. LCK ist für globale Sperranforderungen zuständig.[50]

Um zu vermeiden, dass *GCS*- und *GES*-Dienste ausfallen, werden weitere interne Mechanismen zur Sicherstellung der Verfügbarkeit benutzt, auf die in dieser allgemeinen Beschreibung jedoch nicht näher eingegangen wird.[51]

Weitere wichtige Konzepte der Cache Fusion-Technologie sind Past-Image und der globale Ressourcenkatalog, die im Folgenden näher erläutert werden.

Das Past-Image

Bevor ein geänderter DB-Block über das private Netzwerk von einem Knoten auf einen anderen übertragen wird, legt die sendende DBMS-Instanz immer eine lokale Kopie des geänderten DB-Blockes an. Der unmodifiziert vorgehaltene DB-Block wird als *Past-Image* bezeichnet.[52] Der GCS-Dienst verwaltet in dem globalen Ressourcenkatalog die Past-Images der DB-Blöcke für alle DBMS-Instanzen. Das Past-Image kann beispielsweise zu Wiederherstellungszwecken verwendet werden.

Der globale Ressourcenkatalog

Der globale Ressourcenkatalog (Global Ressource Directory, kurz GRD) spielt in der RAC-Umgebung eine wichtige Rolle als Erweiterung der globalen Sperrtabelle. Er enthält Informationen über gemeinsam genutzte Ressourcen in der RAC-Umgebung und ermöglicht die Lokalisierung eines aktuellen DB-Block-Images im Cluster.[53] Die GCS- und GES-Dienste stellen die Verwalter des globalen Ressourcenkataloges dar.[54]

Die Cache-Kohärenzkontrolle funktioniert im Groben wie folgt (siehe Abbildung 4.2 auf Seite 176):[55]

> Will eine DBMS-Instanz auf einen DB-Block zugreifen, so wird geprüft, ob der DB-Block bereits von einer der DBMS-Instanzen beansprucht wurde. Falls nicht, kann der Zugriff auf den DB-Block mit sehr geringem Kommunikationsaufwand erfolgen. Liegt der angeforderte DB-Block bereits in mindestens einem DB-Puffer der RAC-Umgebung vor, so wird zuerst geprüft, um welchen Zugriff (Read/Write) es sich handelt. Bei einem schreibenden Zugriff wird die aktuellste Version und bei einem lesenden Zugriff die gültige (konsistente) Version des DB-Blockes benötigt. Befindet sich die erforderliche Version des angeforderten DB-Blocks bereits im lokalen DB-Puffer, dann erfolgt keine Übertragung des DB-Blocks von einer entfernten zur lokalen DBMS-Instanz. Andernfalls muss der DB-Block zu der lokalen DBMS-Instanz übertragen werden. Beim Zugriff auf einen DB-Block wird ein entsprechender Sperrmodus in dem globalen Ressourcenkatalog durch den GCS-Dienst eingetragen. In allen Fällen ist ein Kommunikationsaufwand mit dem GCS-Dienst erforderlich.

Anhand der Beschreibung der Cache Fusion-Technologie können folgende Schlussfolgerungen gezogen werden: Durch die logische Vereinigung der DB-Puffer in

[50] Vgl. Lockemann, P.C., Dittrich, K.R. (DBMS 2004), S. 247.
[51] Vgl. Vallath, M. (RAC 2004), S. 128-139.
[52] Vgl. Held., A. (RAC 2005), S. 230.
[53] Vgl. Held., A. (RAC 2005), S. 227.
[54] Vgl. Oracle (RAC-Dokumentation 2006).
[55] In Anlehnung an Vallath, M. (RAC 2004), S. 183-198.

der RAC-Umgebung entfällt das Problem der Cache-Kohärenzkontrolle. Jedoch entsteht eine neue Aufgabe, die mit der Realisierung des einzelnen logischen DB-Puffers verbunden ist. Bei häufigen knotenübergreifenden Zugriffskonflikten (gleichzeitige Zugriffe auf dieselben Daten) erhöhen sich i.d.R. der Verwaltungsaufwand und die Anzahl der DB-Blöcke, die über das Netzwerk von einer DBMS-Instanz zu einer anderen übertragen werden müssen. Der Transport dieser DB-Blöcke kann im schlimmsten Fall zu einer Engpassbildung im Netzwerk führen. Die Effektivität der Skalierbarkeit in RAC wird deshalb voraussichtlich bei vielen knotenübergreifenden Zugriffskonflikten geringer. Da die Übertragungsgeschwindigkeit über das Netzwerk deutlich niedriger ist als die Übertragungsgeschwindigkeit innerhalb des Hauptspeichers, stellt das Netzwerk darüber hinaus generell einen Engpass dar.

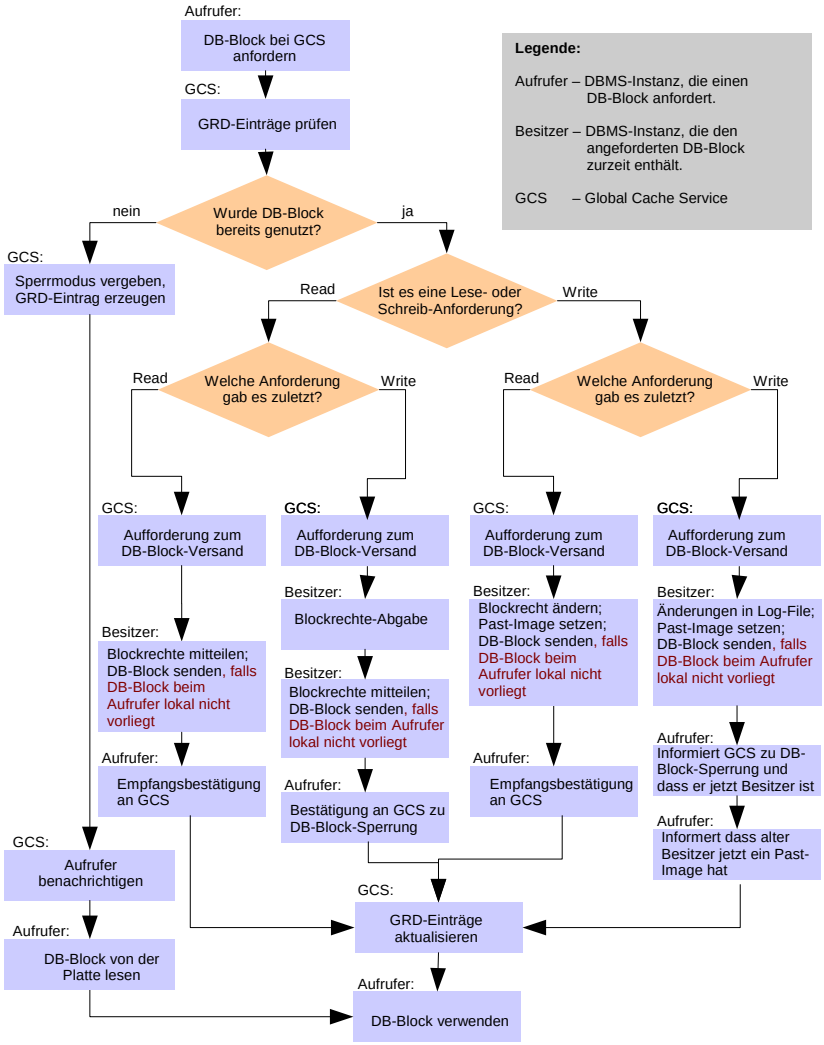


Abbildung 4.2: *Grober Ablauf der Cache Fusion*

SYNCHRONISATION

In einer RAC-Umgebung wird ein modifiziertes Mehrversionen-Sperrprotokoll eingesetzt. [56] Zur Erinnerung: *„Mehrversionen-Synchronisationsverfahren* (engl. *multiversion concurence control*) [Herv. nicht im Original] streben eine Reduzierung von

[56] Vgl. Lockemann, P.C., Dittrich, K.R. (DBMS 2004), S. 247.

Synchronisationskonflikten an, indem für geänderte Objekte zeitweilig mehrere Versionen geführt werden und Leser ggf. auf ältere Versionen zugreifen."[57]

Der GES-Dienst verwaltet das Sperren und Entsperren der globalen Ressourcen und ist für die Erkennung und Auflösung von Verklemmungen verantwortlich.

Die Sperrstruktur dieses Sperrmechanismus enthält insgesamt mehrere Sperrmodi.[58] Die Abbildung 4.3 repräsentiert die Sperrstruktur im RAC. Ein Sperrmodus wird mittels eines dreistelligen Codes beschrieben. Die erste Stelle beschreibt den Sperrmodus, die zweite die Sperr-Rolle und die dritte Stelle zeigt an, ob ein Past-Image für diese Sperre in der lokalen DBMS-Instanz vorliegt. Anhand dessen, welcher Sperrmodus für einen DB-Block gesetzt ist, können gleichzeitige Zugriffe von verschiedenen DBMS-Instanzen auf dieselben Daten koordiniert werden.

Mögliche Einträge für M (Modus):
N – Null (keine Sperre gesetzt. Das gilt für SELECT)
S – Shared (Sperre, die konkurrierende lesende Zugriffe durch andere Benutzer zulässt)
X – Exclusiv (Sperre, die bei Änderungen gesetzt wird und die keine konkurierende Zugriffe zulässt)

Mögliche Einträge für R (Rolle):
L – Lokaler Zugriff auf einen DB-Block (d.h. nur lokale DBMS-Instanz greift auf den DB-Block zu)
G – Globaler Zugriff auf einen DB-Block (d.h. mehrere DBMS-Instanzen greifen auf den DB-Block zu)

Mögliche Einträge für n (Past-Image-Markierung):
1 – DBMS-Instanz besitzt ein Past-Image
0 – DBMS-Instanz besitzt kein Past-Image oder besitzt das aktuelle Block-Image

Abbildung 4.3: *Sperrstruktur im RAC*
[Quelle: In Anlehnung an Vallath, M. (RAC 2004)]

Der größte Vorteil bei der Verwendung des modifizierten Mehrversionen-Sperrprotokolls besteht in der Reduzierung der Synchronisationskonflikte.[59] Nachteilig ist jedoch der erhöhte Aufwand und der zusätzliche Hauptspeicherbedarf, der zur Haltung von mehreren Versionen von DB-Blöcken erforderlich ist.[60] Oracle gibt an, dass ca. 10% des Hauptspeichers einer DBMS-Instanz für die Haltung von Versionen zusätzlich benötigt werden.[61] Da RAC auf der logischen Ebene die SDE-Architektur besitzt, ist die Verwaltung der Versionen offensichtlich insgesamt effizienter als bei der zugrunde liegenden SD-Architektur.

Parallele Anfragenausführung

Kurze Antwortzeiten und hoher Durchsatz hängen stark von der Möglichkeit der Parallelisierung auf der Datenbankebene ab. An dieser Stelle werden die Möglichkeiten der Parallelisierung, Partitionierung und weitere Ansätze zur Anfragenoptimierung in der RAC-Umgebung zusammengefasst.

[57] Rahm, E. (Mehrrechner-DBS 1994), S. 147.
[58] Vgl. zu diesem Absatz Vallath, M. (RAC 2004), S. 136-137.
[59] Vgl. Rahm, E. (Mehrrechner-DBS 1994), S. 147.
[60] Vgl. Rahm, E. (Mehrrechner-DBS 1994), S. 147.
[61] Vgl. Oracle (Metalink 2007).

Parallelisierung von Anfragen

In einer RAC-Umgebung wird die Intra- und Inter-Transaktionsparallelität unterstützt. Folgende Operationen können im Real Application Clusters parallelisiert werden: [62]

- elementare Lese- und Schreiboperationen,
- komplexe Anfragen mit z.B. rechenintensiven Sortier- und Vereinigungs-Operationen,
- das Erstellen von Indizes auf Relationen mit großen Datenmengen,
- bei Sicherungs- sowie Wiederherstellungsoperationen im Zuge einer Datensicherung bzw. –wiederherstellung, sowie andere lang laufende Operationen (z.B. Batch-Jobs).

Beim Parallelisieren einer DB-Operation wird diese in kleinere Teiloperationen zerlegt.[63] Diese Teiloperationen werden an einzelne Prozesse, die so genannten *Parallel Query Slaves* (*PQS*), verteilt. Die Ergebnisse der Teiloperationen gibt jeder Slave-Prozess anschließend an einen Koordinator, den so genannten *Parallel Query Coordinator* (*PQC*), zurück.

RAC unterstützt dynamische oder statische Parallelisierung[64] von Anfragen. Bei der Wahl der *dynamischen Parallelisierung* entscheidet der Anfragenoptimierungsprozess (Oracle Optimizer) selbst, ob eine Anfrage parallelisiert werden soll oder nicht. Darüber hinaus legt der Optimierungsprozess den anzuwendenden Parallelitätsgrad fest. Wird die parallele dynamische Optimierung nicht gewählt, so muss ein Parallelitätsgrad für das DBMS oder auf einer anderen Ebene eingestellt werden. Im Allgemeinen kann bei Oracle auf folgenden Ebenen der Parallelitätsgrad *statisch* (d.h. durch explizite Angabe des Parallelitätsgrades) angegeben werden:

- DB-Operationsebene

  In einer DB-Operation können im Bereich der Zusatzinformationen (Hints genannt) zu der DB-Operation die Angaben zum Parallelitätsgrad vorgenommen werden.

- DB-Objektebene

  Beispielsweise kann beim Erstellen einer Tabelle ein Parallelitätsgrad angegeben werden. Beim Ausführen von Anfragen auf diese Tabelle wird dann der für diese Tabellen bestimmte Parallelitätsgrad angewendet.

- Sitzungsebene

  Ein Parallelitätsgrad kann innerhalb einer Benutzersitzung eingeschaltet, abgeschaltet oder geändert werden.

- DBMS-Instanzebene

  Es kann ein Standard-Parallelitätsgrad für alle parallelen Operationen in einer DBMS-Instanz gesetzt werden.

---

[62] Dieser Abschnitt stützt sich weitgehend auf Oracle (Warehousing 2005), S. '25-24' - '25-35' sowie Held., A. (RAC 2005), S. 345-348.
[63] Vgl. zu diesem Absatz Vallath, M. (RAC 2004), S. 210.
[64] Vgl. Oracle (Reference 2005), siehe Initialisierungsparameter PARALLEL_AUTOMATIC_TUNING und PARALLEL_MAX_SERVERS.

Da grundsätzlich auf mehreren Ebenen verschiedene Parallelitätsgrade für ein DB-Objekt angegeben werden können, wird die Prioritätsreihenfolge für den zu wählenden Parallelitätsgrad festgelegt. Die höchste Priorität hat der Parallelitätsgrad auf der DB-Operationsebene. Wurde bei der DB-Operation kein Parallelitätsgrad angegeben, so gilt der für das DB-Objekt angegebene Parallelitätsgrad. Wenn auch der Parallelitätsgrad auf der DB-Objektebene nicht gesetzt wurde, so gilt der Standardwert des Parallelitätsgrades auf der DBMS-Instanzebene. Dieser Standardwert wird jedoch durch den Wert des Parallelitätsgrades auf der Sitzungsebene überschrieben, falls auf der Sitzungsebene dieser Wert gesetzt ist.

Der Parallelitätsgrad für eine einzelne DBMS-Instanz ist i.Allg. durch folgende Kennzahlen begrenzt:

- die Anzahl der CPUs auf dem Knoten,
- die Anzahl der Threads pro CPU,
- die Anzahl der Partitionen und die Verfügbarkeit des Parallel-Execution-Serverprozesses zur Ausführungszeit (im Fall der partitionierten Tabellen).

Wird der Wert für den Parallelitätsgrad zu niedrig eingestellt, so können Parallelisierungsmöglichkeiten nicht komplett ausgenutzt werden. Setzt man dagegen den Wert zu hoch, führt die Parallelisierung zur Verschlechterung der Antwortzeiten und der Minimierung des Durchsatzes.[65] Dies ist damit zu begründen, dass bei der Parallelisierung erheblicher Kommunikationsaufwand entsteht. So müssen sich die einzelnen Slave-Prozesse über die Vergabe von Teiloperationen mit dem Anfragenoptimierungsprozess verständigen und sich gegenseitig bezüglich des aktuellen Status benachrichtigen. [66] Die richtige Wahl bei der expliziten Angabe des Parallelitätsgrades ist damit bei der Anfragenoptimierung maßgebend.

Partitionierung

Oracle ermöglicht den transparenten Einsatz von horizontalen und horizontal abgeleiteten Partitionierungsformen von Tabellen und Indizes.[67] *Transparent* in diesem Kontext heißt, dass die Verteilung von Daten für Benutzer unsichtbar ist. Die Partitionierung steigert den Parallelverarbeitungsgrad bei großen Datenmengen. So können mehrere Teiloperationen gleichzeitig auf verschiedenen Partitionen innerhalb einer DBMS-Instanz und/oder knotenübergreifend ausgeführt werden. Die Ergebnisse der Teiloperationen werden dann an den Anfragenkoordinator der entsprechenden DBMS-Instanz aus der RAC-Umgebung zurückgeliefert.

RAC unterstützt folgende horizontale Partitionsstypen:

- RANGE-(Bereichs-),
- LIST-(Listen-) und
- HASH-(Zufalls-)Partitionierung.

[65] Vgl. Oracle (Reference 2005), Parameter PARALLEL_MAX_SERVERS.
[66] Vgl. Held., A. (RAC 2005), S. 342.
[67] Vgl. zu diesem Unterabschnitt Oracle (Administration 2005), S. '17-1' - '17-7'.

In der RAC-Umgebung können Tabellen und Indizes, die durch RANGE partitioniert wurden, noch weiter in Unterpartitionen aufteilen werden. Unterpartitionen können entweder die HASH- oder die LIST-Partitionierung verwenden.

Anfragenoptimierung

Da alle DBMS-Instanzen gleichzeitig auf alle Daten zugreifen können, ist die Unterstützung der verteilten Anfrage- und Transaktionsausführung nicht notwendig. Im Allgemeinen bietet Oracle mehrere unterschiedliche statistikbezogene Optimierungsverfahren für DB-Operationen an.[68] Bei der Ausführung von Anfragen wird der bestmögliche Ausführungspfad durch den Anfragenoptimierungsprozess errechnet und ausgeführt.

Wie man sieht, bietet RAC viele Optimierungsmöglichkeiten für (parallele) Anfragenausführung. Die fehlende physische Aufteilung der Datenbank, die im Fall der SN basierten Architektur typisch ist, ergibt für die SDE-Architektur auch erhöhte Freiheitsgrade zur Intra-Transaktionsparallelität.

Einfache Administration

Die Administration der RAC-Umgebung kann – auf Empfehlung von Oracle – mit dem Werkzeug Oracle-Enterprise-Manager vorgenommen werden.[69]

Der Oracle Enterprise Manager (OEM) existiert in zwei Produktvarianten:

- dem OEM Grid Control und
- dem OEM Database Control.

Die beiden Varianten unterscheiden sich in ihrer Konfiguration (dezentral bzw. zentral) sowie ihrem administrativen Wirkungsbereich.

OEM Grid Control eignet sich zur Verwaltung von RAC-Umgebungen, da es sowohl einzelne DBMS-Instanzen als auch Clusterumgebungen verwalten kann. Demgegenüber steht das so genannte OEM Database Control, welches nur ausgewählte DBMS-Instanzen verwaltet. Hierbei ist es jedoch unerheblich, ob es sich um eine RAC-Umgebung oder eine zentrale Datenbank handelt.

Das Hinzufügen oder das Entfernen von Knoten im RAC hat keine Auswirkung auf andere Knoten des Clusters. Diese Aufgabe ist jedoch mit relativ hohem Aufwand verbunden und bedarf des manuellen Eingriffs durch einen DB-Administrator.[70] Nach einer Rekonfiguration der RAC-Umgebung ist i.d.R. kein Neustart notwendig. Die Wiederaufnahme eines ausgefallenen Knotens erfolgt voll automatisch und ohne administrativen Eingriff.

Die Bestimmung der Datenverteilung über Knoten hinweg erfordert i.d.R. einen hohen administrativen Aufwand, welcher in der RAC-Umgebung bei der zugrunde liegenden SD-Architektur entfällt.

---

[68] Vgl. Oracle (Administration 2005).
[69] Vgl. zu diesem Abschnitt Oracle (OEM 2007), S. '1-1' - '1-9', '13-1' vgl. dazu auch Oracle (RAC-Dokumentation 2006), S. '9-2'.
[70] Vgl. Oracle (RAC-Dokumentation 2006), S. '10-1'-'10-25' vgl. dazu auch Vallath, M. (RAC 2004), S. 330-331.

KURZE ANTWORTZEITEN/HOHER DURCHSATZ

Das Erreichen hoher Antwortzeiten und eines hohen Durchsatzes ist bei sehr vielen knotenübergreifenden Zugriffskonflikten wahrscheinlich nicht möglich, da es zu einer häufigen Übertragung derselben DB-Blöcke zwischen den Knoten kommen könnte, wodurch der Kommunikations- und Verwaltungsaufwand voraussichtlich deutlich steigen würde.

KOSTENEFFEKTIVITÄT DES RAC

Für die Umsetzung eines Real Application Clusters werden folgende Hardwarekomponenten benötigt:

- mindestens zwei Knoten für die RAC-Umgebung;
- ein gemeinsam genutztes Plattensubsystem;
- ein privates Netzwerk für die interne Kommunikation im Cluster;
- ein öffentliches Netzwerk für die Kommunikation mit Benutzern oder Anwendungen.

In der RAC-Umgebung kann nur bei einer kleinen Knotenanzahl auf den Einsatz der Spezialhardware wegen ihrer relativ hohen Kosten verzichtet werden. Ab mehr als zwei Knoten wird als Plattensubsystem SAN oder NAS empfohlen.[71] *Storage Area Network* (*SAN*, dt. Speichernetzwerk) ist ein Netzwerk im Bereich der Datenverarbeitung zur Anbindung von Speichermedien wie z.B. Plattensubsystemen. [72] *Network Attached Storage* (*NAS*) bezeichnet an das lokale Netzwerk angeschlossene Speichereinheiten zur Erweiterung der Speicherkapazität.[73]

Für die interne Kommunikation der Knoten sollte im RAC bevorzugt ein Hochleistungsnetzwerk eingesetzt werden, das mit zunehmender Knotenanzahl skalieren kann. Dadurch können mögliche Engpässe im Bereich des Netzwerkes minimiert werden. Der Einsatz der Standardhardware, insbesondere weit verbreitete Mikroprozessoren für die CPUs, ist möglich.

Software- und Lizenzkosten fallen bei RAC je nach Konstellation des Datenbanksystems sehr unterschiedlich an. Nach der

E-Business Preisliste von Oracle betragen beispielsweise die 1-jährigen Lizenzkosten pro Prozessor ab 15.000 US-Dollar.[74] Die Supportkosten stellen in der Regel 22% der Gesamtlizenzkosten dar. Die Personal- und Schulungskosten kommen ebenfalls hinzu. Je nach Edition (z.B. Standard oder Enterprise) und zusätzlich gewählten Optionen (z.B. Partitionierung) können Lizenzkosten weiter steigen. Es sei angemerkt, dass bei Lizenzierung der Software für mehrere Jahre Rabatte gewährt werden.

In einer Studie hat International Data Corporation [75] (IDC) ermittelt, dass die Softwarekosten nur 15% der gesamten Einsatzkosten einer Oracle 8i Datenbank

[71] Vgl. Held., A. (RAC 2005), S. 216-217.
[72] Vgl. Troppens, U., Erkens, R. (Speichernetze 2002).
[73] Vgl. Troppens, U., Erkens, R. (Speichernetze 2002).
[74] Vgl. Oracle (Preiseliste 2006).
[75] IDC ist ein Anbieter in den Bereichen Marktbeobachtung und Beratung der IT- und Telekommunikationsindustrie. IDC analysiert und prognostiziert technologische und branchenbezogene

ausmachen – die Hardware nahm 17% in Anspruch, das Personal 21% und Schulungen 19%. 28% der Gesamtkosten eines Oracle-Datenbankeinsatzes werden der Systemausfallzeit beigemessen.[76] Wenn man diese Schätzung für RAC annehmen würde, so entfallen fast alle der oben genannten 28% für Systemausfallzeiten, da RAC ein hochverfügbares Datenbanksystem ist. Jedoch dürften sowohl die Personal- als auch die Schulungskosten höher als bei der zentralen Oracle 8i-Datenbank liegen, da komplexere und umfangreichere Administrationsaufgaben hinzukommen. Andererseits steht zu vermuten, dass sich die zusätzlichen Personal- und Schulungskosten für RAC durch geringere Ausfallkosten zum Teil oder ganz kompensieren lassen.

REALISIERUNG: MYSQL CLUSTER

In diesem Abschnitt werden Realisierungskonzepte und Techniken des MySQL Clusters vorgestellt. Die Beschreibung des MySQL Clusters basiert ab Version 5.1.[77] Da diese Betaversion wichtige Erweiterungen gegenüber der Version 5.0 enthält, wurde sie berücksichtigt. *Beta* des MySQL Clusters bedeutet, dass das Release funktionsseitig abgeschlossen ist und der gesamte neue Code getestet wurde.[78] Es werden keine wesentlichen neuen Funktionen und Merkmale mehr hinzugefügt. Es sollten auch keine kritischen Bugs mehr vorhanden sein. Eine Version wechselt vom Alpha- in den Betastatus, wenn mindestens einen Monat lang keine schweren Bugs mehr für die Alphaversion gemeldet wurden. Version 6.0 wird derzeit von MySQL AB für den betrieblichen Einsatz empfohlen.

MySQL Cluster ist ein paralleles Datenbanksystem mit der Shared-Nothing (SN)-Architektur.[79] Im MySQL Cluster werden drei Arten von Knoten unterschieden:

- Daten-Knoten

  Datenknoten speichern alle zu MySQL Cluster gehörenden Daten. Die Daten werden im Normalfall zwischen den Datenknoten des Clusters repliziert, um sicherzustellen, dass diese bei Ausfall eines oder mehrerer Knoten ununterbrochen verfügbar sind. Datenknoten verwalten außerdem Datenbanktransaktionen.

- Management-Knoten

  Ein Managementknoten ist für die Systemkonfiguration, die Knotenverwaltung und die Aufzeichnung der Aktivitäten im Cluster zuständig. Es können ein oder mehrere Managementknoten aus Verfügbarkeitsgründen gleichzeitig eingesetzt werden.

- SQL-Knoten

  Ein SQL-Knoten entspricht einem MySQL Datenbanksystem, das mit Datenknoten kommunizieren kann. Das MySQL Datenbanksystem erlaubt die Verwendung von Datenbank-Managementsystemen mit verschiedenen Konzepten: mit und ohne Durchführung von Transaktionen, mit und ohne

---

Trends sowie deren Potenziale und bietet ihren Kunden Unterstützung bei der Planung und Umsetzung ihrer Geschäfts-strategien. Vgl. Homepage: http://www.idc.com/germany/, Abruf am 22.03.2007.

[76] Vgl. MySQL (TCO 2005), S. 8-9.

[77] Vgl. MySQL (Handbuch 2007).

[78] Vgl. Homepage: http://www.mysql.com/products/database/cluster/, Abruf am 16.02.2007.

[79] Dieser Abschnitt stützt sich weitgehend auf MySQL (Handbuch 2007).

persistente Speicherung, mit und ohne den Einsatz von gespeicherten Prozeduren, mit synchroner oder asynchroner Replikation usw.

Der grobe Ablauf einer Benutzeranfrage ist wie folgt:

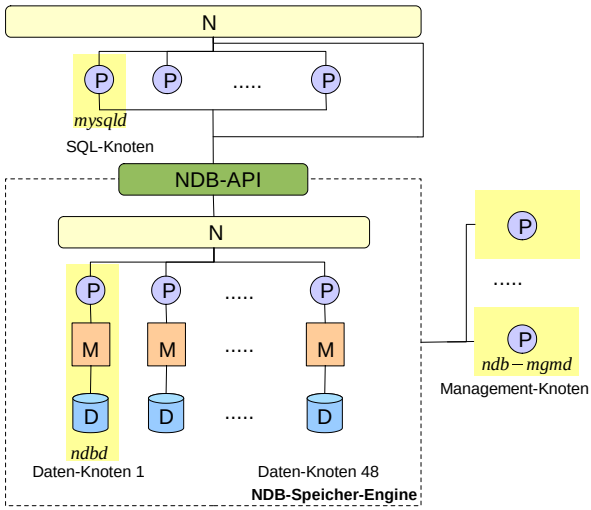
1. Eine Anfrage wird an einen SQL-Knoten gestellt.
2. Der SQL-Knoten leitet die Anfragen an die Datenknoten weiter.
3. Ein Datenknoten verarbeitet die Anfrage und sendet das Ergebnis an den SQL-Knoten zurück.
4. Der SQL-Knoten übergibt das Ergebnis an Benutzer.

Abbildung 4.4 auf Seite 191 zeigt die SN-Architektur des MySQL Clusters, bei der verschiedene Knotenarten auf getrennten Rechnern ablaufen. Diese Architektur ist durch die nachträgliche Integration der NDB-Datenbank (siehe unten) bedingt.

Auf jedem der Knoten des MySQL Clusters ist mindestens ein Prozess gestartet. Bei SQL-Knoten heißt der zuständige Prozess *mysqld*, bei Datenknoten *ndbd* und bei Verwaltungsknoten *ndb-mgmd*. Auf Rechnerknoten mit mehreren Prozessoren können mehrere MySQL Cluster-Prozesse gleichzeitig laufen. Beispielsweise auf einem Datenknoten mit 2 CPUs können zwei *ndbd*-Prozesse parallel ausgeführt werden. Es ist ebenfalls möglich, Prozesse verschiedener MySQL Cluster-Knotentypen auf einem Rechnerknoten mit mehreren CPUs einzusetzen. Zum Beispiel kann auf einem Rechner ein Prozess des SQL-Knotens (*mysqld*) und ein Prozess des Datenknotens (*ndbd*) gestartet sein.

Die *NDB* (*Network Database*)-*Speicher-Engine* (siehe Abbildung 4.4) ist eine unabhängige Komponente, die persistente Speicherung von Daten ermöglicht und für die Koordination aller Zugriffe auf Datenknoten in einem MySQL Cluster zuständig ist.[80] Anwendungen können direkt auf die NDB-Speicher-Engine über die NDB-API oder über einen MySQL-Knoten zugreifen.[81] Der Zugriff über einen MySQL-Knoten ist für Anwendungsprogrammierer wesentlich einfacher zu gestalten, da in diesem Fall Standard-SQL-Befehle verwendet werden können und das Erlernen der NDB-Spezialitäten nicht notwendig ist.

[80] Vgl. MySQL (Handbuch 2007), S. 985.
[81] Vgl. MySQL (5.1-Neuerungen 2006), S. 12.



Legende: **P** – Prozessor, **M** – Hauptspeicher (DB-Puffer), **D** – Sekundärspeicher mit Datenbank, **N** – Netzwerk,
mysqld - SQL-Knotenprozess, *ndbd* - Daten-Knotenprozess, *ndb – mgmd* - Management-Knotenprozess

Anmerkung: Die maximale Gesamtknotenanzahl ist gleich 63.

***Abbildung 4.4**: Die SN-Architektur des MySQL Clusters*

Die *NDB-API* ist eine multithread-fähige Schnittstelle zur Annahme aller ankommenden Datenanfragen.[82] Für jede Anfrage wird ein oder mehrere Threads gestartet. Auf die NDB-API ist nur ein sequenzieller Zugriff möglich, wodurch die Leistung des Clusters bei sehr vielen ankommenden Anfragen vermutlich eingeschränkt wird. Es sei angemerkt, dass Threads keine gemeinsame Nutzung derselben Objekte des NDB-Speicher-Engine haben. Das führt i.d.R. zu einem erhöhten Hauptspeicherverbrauch pro Thread.

SKALIERBARKEIT

MySQL Cluster kann durch das Hinzufügen zusätzlicher SQL- oder Datenknoten erweitert werden.[83] Der Skalierbarkeit im MySQL Cluster sind derzeit physische Grenzen gesetzt:

- MySQL Cluster unterstützt maximal 48 Datenknoten.[84] Die Gesamtzahl aller Knoten in einem MySQL Cluster beträgt ab Version 7.0 255. Das umfasst sämtliche SQL-, Daten- und Managementknoten.
- Des Weiteren ist die Skalierbarkeit durch die Größe des verfügbaren Hauptspeichers begrenzt.[85] Ab der Version 5.1 besteht zwar die Möglichkeit, dass nicht alle Daten einer Datenbank komplett im physischen Hauptspeicher gehalten werden, jedoch müssen mindestens alle Indizes einer Datenbank im Hauptspeicher liegen. Bei einem hohen Hauptspeicherbedarf für Indizes kann dieser schnell zum Engpass werden.

[82] Vgl. MySQL (NDB-API 2006), S. 1-13.
[83] Vgl. MySQL (5.1-Neuerungen 2006), S. 4.
[84] Vgl. zu diesem Absatz MySQL (Handbuch 2007), S. 983-1054.
[85] Vgl. zu diesem Absatz MySQL (5.1-Neuerungen 2006), S. 6-19 vgl. dazu auch MySQL (Handbuch 2007), S. 1057.

- Die Realisierung sehr großer Datenbanken wird dadurch erschwert, dass die Erweiterbarkeit der Hauptspeicherressource aufgrund technischer Schwierigkeiten bei der Speicheranbindung eingeschränkt ist.[86]

Wegen der strikten Grenzen für Datenknotenanzahl (max. 48 Datenknoten), und wegen der allgemeinen Skalierbarkeitseinschränkung für die Hauptspeicherressource, kann der MySQL Cluster theoretisch nur bis zu einer bestimmten Datenbankgröße bei hauptspeicherresidenten Tabellen eingesetzt werden. Große Datenbanken (zum Beispiel von mehreren Hundert TBytes) lassen sich zurzeit in diesem Fall wegen der erforderlichen Hauptspeichergröße zumindest auf Standardhardware nicht realisieren.

HOCHVERFÜGBARKEIT

MySQL Cluster unterstützt die Hochverfügbarkeit mithilfe der folgenden grundlegenden Konzepte:[87]

- Transparenz bei Knotenausfällen

  SQL-Knoten sind mit allen Datenknoten des Clusters gleichzeitig verbunden. Beim Ausfall eines Datenknotens benutzen SQL-Knoten einen anderen verfügbaren Datenknoten. Damit bleiben Knotenausfälle für Benutzer und Anwendungen transparent, aber nur dann, wenn gerade keine Transaktionen oder Anfragen auf den Knoten ausgeführt wurden. Fällt beispielsweise ein Datenknoten während einer Transaktionsausführung aus, dann wird die Transaktion zurückgesetzt und der SQL-Knoten darüber benachrichtigt.

- Redundante Datenspeicherung

  Daten werden i.d.R. über mehrere Datenknoten repliziert. Beim Ausfall eines Datenknotens verfügt mindestens ein anderer Datenknoten über dieselben Daten. Dies führt bei Datenknotenausfällen zu sehr niedrigen Ausfallzeiten (kleiner als eine Sekunde). Die Anzahl der redundant gespeicherten Kopien kann beliebig gewählt werden. Es ist sogar möglich, redundante Datenspeicherung ganz auszuschalten, was allerdingst wegen des möglichen Datenverlustes nicht zu empfehlen ist. Datenreplikation geschieht bei MySQL Cluster, wie gewöhnlich, transparent für Benutzer und Anwendungen. Die Konsistenzhaltung der redundanten Daten wird mittels eines synchronen Replikationsmechanismus nach dem ROWA-Prinzip gewährleistet.

  Zur Erinnerung: *ROWA* (*Read-One-Write-All*) – ist ein synchrones Replikationsverfahren, bei dem jede Kopie auf dem aktuellsten Stand ist. Bei jeder Änderungsoperation werden alle Kopien gleichzeitig geändert.[88] Die Dauer einer Änderungsoperation wird hierbei durch die langsamste Ausführung der Aktualisierung einer der Kopien bestimmt.

  Beim ROWA-Verfahren wird im MySQL Cluster ein modifiziertes 2-Phasen-Commit-Protokoll[89] eingesetzt.

[86] Vgl. zu diesem Absatz Rahm, E. (Transaktionssysteme 1993b), S. 84.
[87] Dieser Abschnitt stützt sich weitgehend auf Ronstrom, M., Thalmann, L. (Architektur 2004).
[88] Vgl. zu diesem Absatz Dadam, P. (Datenbanken 1996), S. 16.
[89] Siehe Dadam, P. (Datenbanken 1996), KE 5 S. 41.

Knoten jeglicher Art können aus der Clusterumgebung entfernt oder hinzugefügt werden, ohne dass es eine Auswirkung auf andere Knoten hat. Beim Hinzufügen oder Entfernen von Knoten ist jedoch ein Neustart des Clusters erforderlich.

Dieser Lösungsansatz hat einen Nachteil: Die Konsistenzhaltung der redundant gespeicherten Kopien, die für die Hochverfügbarkeit unverzichtbar ist, verursacht einen zusätzlichen Aufwand, welcher bei sehr vielen Änderungen an Daten und bei mehrfacher Redundanz zu Leistungseinbußen führen könnte. Die Anzahl der Kopien soll deshalb mit Sorgfalt gewählt werden.

Es existieren drei weitere Replikationskonfigurationen zur Unterstützung der Hochverfügbarkeit:[90]

- MySQL Cluster auf MySQL Cluster,
- MySQL-Server auf MySQL Cluster,
- MySQL Cluster auf MySQL-Server.

Diese Replikationskonfigurationen eignen sich bei einer räumlichen Verteilung von Datenbanksystemen mit dem Ziel der Erhöhung der Verfügbarkeit im Katastrophenfall. Die interessanteste Konfiguration von allen ist die „MySQL Cluster auf MySQL Cluster“-Replikation, die im Folgenden näher beschrieben wird.

Replikationsart: MySQL Cluster auf MySQL Cluster

Mit der Replikation „MySQL Cluster auf MySQL Cluster“ kann die höchste Hochverfügbarkeit des MySQL Clusters erreicht werden (siehe Abbildung 4.5). Dabei wird zwischen dem Master-MySQL Cluster und Slave-MySQL Cluster unterschieden. In dem *Master-MySQL Cluster* sind nicht nur Lese- sondern auch Schreibtransaktionen erlaubt. Der *Slave-MySQL Cluster* ist ausschließlich auf Lesetransaktionen beschränkt. Änderungen an Daten eines Master-MySQL Clusters werden mittels eines asynchronen Replikationsmechanismus zu dem Slave-MySQL Cluster übertragen.

Der Replikationsmechanismus besteht darin, dass alle Aktivitäten eines Master-MySQL Clusters in einem Binärprotokoll aufgezeichnet und in regelmäßigen Abständen auf dem Slave-MySQL Cluster ausgeführt werden. Diese Aufzeichnung wird von einem Thread, bezeichnet als *NDB-Binlog-Injector-Thread*, durchgeführt, welcher auf jedem MySQL-Knoten läuft und ein Binärprotokoll (auch Binlog genannt) erzeugt.[91] Der NDB-Binlog-Injector-Thread garantiert, dass sämtliche Änderungen im Cluster in der richtigen Reihenfolge in das Binärprotokoll des Slave-Clusters eingefügt werden. Auf diese Weise wird die Konsistenzhaltung von Lesekopien in dem Slave-MySQL Cluster gewährleistet. Da die Replikation asynchron verläuft, existiert eine aktuelle konsistente Kopie nur in dem Master-MySQL Cluster.

[90] Vgl. zu diesen Ausführungen MySQL (5.1-Neuerungen 2006), S. 10.
[91] Zum Einsatz des Binärprotokolls in MySQL siehe MySQL (Reference 2007), S. 375-378.

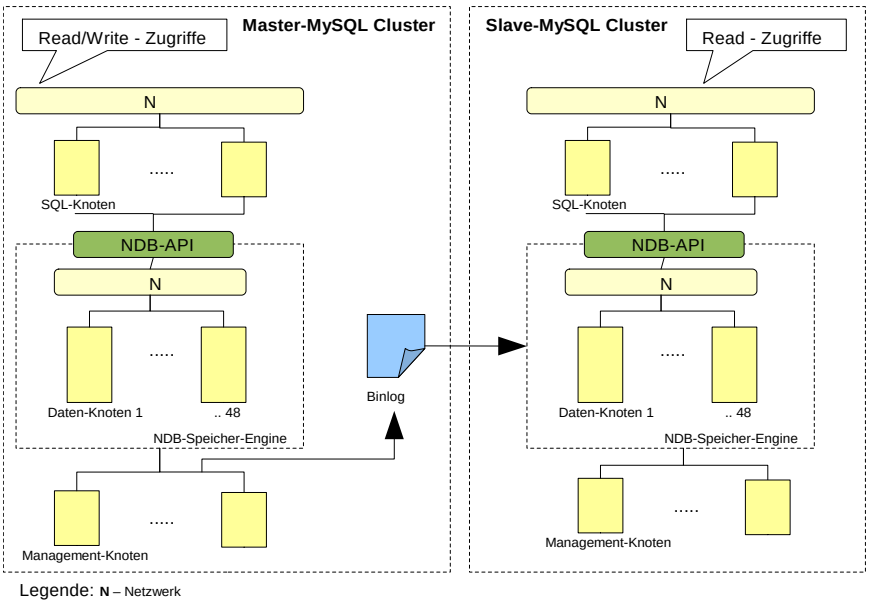


***Abbildung 4.5**: Die Replikation 'MySQL Cluster auf MySQL Cluster'*

LASTBALANCIERUNG

Im MySQL Cluster wird Lastbalancierung je nach Knotentyp unterschiedlich gestaltet.

Lastbalancierung von SQL-Knoten

Zu Verteilung der Lasten zwischen SQL-Knoten werden in der MySQL Cluster-Umgebung hardware- vor softwarebasierten Lösungen empfohlen.[92]

Beispiele möglicher Hardwarelösungen bieten die Firma Cisco[93] und die Corporation Big-IP[94] an.

Eine Softwarelösung für Lastbalancierung von MySQL-Knoten muss individuell für eine konkrete Anwendung und deren Ablaufumgebung gewählt werden.[95] Diese Methode ist i.d.R. weniger effizient als Lastbalancierung durch Datenbanksysteme selbst. Eine mögliche Softwarelösung wäre beispielsweise der Einsatz eines geclusterten JDBC (C-JDBC)-Treibers. *C-JDBC* oder sein Nachfolger *Sequoia* ist eine freie Software aus der Middlewareschicht, die einen transparenten Zugriff auf ein geclustertes Datenbanksystem mittels JDBC erlaubt.[96]

[92] Vgl. zu diesem Abschnitt Zawodny, J.D., Balling, D.J. (High Performance 2004), S. 171.
[93] Vgl. Homepage von Cisco: 'http://www.cisco.com/en/US/hmpgs/index.html', Abruf am 22.03.2007.
[94] Vgl. Homepage von F5: http://www.f5.com/, Abruf am 22.03.2007.
[95] Vgl. Zawodny, J.D., Balling, D.J. (High Performance 2004), S. 171.
[96] Vgl. Homepage von C-JDBC: http://c-jdbc.objectweb.org/, Abruf am 22.03.2007 vgl. dazu auch Homepage von Sequoia:
http://sequoia.continuent.org/HomePage, Abruf am 20.12.2007.

Lastbalancierung von Datenknoten

Lastbalancierung von Datenknoten ist standardmäßig über die Partitionierung der Daten vorgesehen.[97] Im MySQL Cluster wird zwischen primären und sekundären Partitionen unterschieden.[98] Bei einer Anfrage wird i.Allg. immer die Primärpartition verwendet. Nur wenn der Datenknoten mit der primären Partition nicht verfügbar ist, wird ein Knoten mit einer der sekundären Partitionen ausgewählt. Da alle Anfragen zuerst auf die Primärpartitionen zugreifen, muss die Aufteilung globaler Relationen in Partitionen mit großer Sorgfallt bestimmt werden.

Beim Einsatz des Replikationsmechanismus „MySQL Cluster auf MySQL Cluster" kann neben der Erhöhung von Systemverfügbarkeit zusätzlich Lastbalancierung verbessert werden: Wenn zum Beispiel Lesetransaktionen an den Slave-MySQL Cluster gerichtet werden, hat der Master-MySQL Cluster geringere Anfragenlast.

Lastbalancierung im MySQL Cluster kann wie in jeder SN-Architektur wegen möglicher ungleichmäßiger Datenverteilung ein ernsthaftes Problem werden.[99] Um eine effiziente Lastbalancierung in diesem Fall zu erreichen, ist gewöhnlich ein dauerhafter administrativer Aufwand für Datenreorganisation erforderlich.

INTERPROZESSORKOMMUNIKATION

Die Kommunikation im MySQL Cluster wird zurzeit mithilfe von drei verschiedenen Datentransportmechanismen, den so genannten *Transportern* (engl. *transports*), unterstützt:[100]

- TCP,
- SCI und
- den gemeinsam genutzten Speicherbereich.

Die oben genannten Transporter werden im Folgenden näher beschrieben.

TCP

Das verbindungsorientierte Transportprotokoll TCP (Transmission Control Protocol) wird in den meisten Fällen verwendet und ist mit Abstand der am besten getestete Transporter im MySQL Cluster. Bei Datenknoten wird ein Netzwerk mit der Bandbreite von mindestens 1 Gbit/s wegen der relativ hohen Kommunikationskosten empfohlen.[101] Der Einsatz des TCP/IP-Protokollstacks ist allerdings bei sehr großen Datenmengen nicht effizient genug und führt relativ schnell zu Leistungseinbußen.

[97] Vgl. MySQL (Handbuch 2007), S. 1067.
[98] Vgl. MySQL (Handbuch 2007), S. 986-988.
[99] Vgl. Rahm, E. (Mehrrechner-DBS 1994), S. 360-364.
[100] Diese Auflistung stützt sich weitgehend auf Ronstrom, M., Thalmann, L. (Architektur 2004), S. 6 sowie Davies, A., Fisk, H. (Clustering 2006), S. 145-146.
[101] Vgl. zu diesem Absatz MySQL (Handbuch 2007), S. 989 vgl. dazu auch Davies, A., Fisk, H. (Clustering 2006), S. 146.

SCI

SCI (Scalable Coherent Interface) ist ein ANSI/IEEE-Standard, welcher eine Netzwerktechnologie beschreibt, die einen sehr schnellen Datentransfer ermöglicht.[102] SCI definiert grundsätzlich nur unidirektionale Punkt-zu-Punkt-Verbindungen, wobei jeweils zwei solcher Verbindungen zu einem SCI-Link gehören. Die Geschwindigkeit der einzelnen Links wächst mit der Art der eingesetzten Technologie.

SCI-Techonologie stellt eine leistungsfähigere aber auch teure Alternative gegenüber einem Ethernet/Fast Ethernet oder Gigabit Ethernet dar. SCI wird von MySQL AB bei großen Datenbanken dringend empfohlen.[103] Diese Technologie erfordert spezielle Hardware und Treiber. Ihr Einsatz wurde bei MySQL Cluster mithilfe von Dolphin SCI Interconnect Adaptern der Corporation Dolphin[104] getestet. Tests, die von der MySQL AB durchgeführt wurden, haben gezeigt, dass die Leistungsfähigkeit beim Erweitern des MySQL Clusters mit Interconnect von Dolphin erheblich verbessert werden konnte.[105]

Gemeinsam genutzter Speicherbereich

Der Datenaustausch zwischen mehreren MySQL Cluster-Prozessen eines Rechnerknotens kann über einen gemeinsam genutzten Hauptspeicherbereich (engl. shared memory) realisiert werden, auf den die lokalen MySQL Cluster-Prozesse zugreifen können.[106] Diese Lösung wird bei MySQL AB als *SHM-Transport* bezeichnet.

Der Vorteil vom SHM liegt darin, dass ein schneller Datenaustausch über den Hauptspeicher möglich ist.[107] Zum Nachteil dieses Ansatzes zählt die Notwendigkeit der Synchronisation von gleichzeitigen Zugriffen auf einen gemeinsam genutzten Hauptspeicherbereich durch mehrere MySQL Cluster-Prozesse. Die Synchronisation dieser Zugriffe bedeutet für das System einen zusätzlichen Aufwand. Diese Alternative kann deshalb in Einzelfällen zum Leistungsverlust bei CPUs führen. SHM-Transport wird außerdem nicht von allen Betriebssytemen unterstützt und befindet sich noch in der Erprobungsphase.[108]

Nach der Beschreibung der MySQL Cluster-Transporter erfolgt die Erläuterung der Kommunikationsfehlerbehandlung im Cluster.

Behandlung von Fehlerfällen

Folgende Fehlerfälle werden im MySQL Cluster bei der Interprozessorkommunikation behandelt:

1. Verlust der Kommunikation,
2. Knoten reagiert nicht und
3. Netzwerkpartitionierung.

---

[102] Vgl. IEEE Standard für SCI, 'http://standards.ieee.org/reading/ieee/std_public/description/busarch/1596.3-1996_desc.html', Abruf am 14.12.2007.
[103] Vgl. MySQL (Handbuch 2007), S. 1027.
[104] Vgl. Homepage von der Corporation Dolphin: http://www.dolphinics.com/, Abruf am 22.03.2007.
[105] Vgl. MySQL (Handbuch 2007), S. 1052 vgl. dazu auch MySQL (Hochverfügbarkeit 2007), S. 19.
[106] Vgl. MySQL (Handbuch 2007), S. 1026 vgl. dazu auch Davies, A., Fisk, H. (Clustering 2006), S. 146-147.
[107] Vgl. Davies, A., Fisk, H. (Clustering 2006), S. 146-147.
[108] Vgl. MySQL (Handbuch 2007), S. 1004.

Die aufgelisteten Fehlerfälle werden anschließend näher beschrieben.

1. Fehlerfall: Verlust der Kommunikation

Generell kommunizieren im MySQL Cluster alle Datenknoten miteinander. Beim Verlust der Kommunikation eines Knotens zu einem anderen werden sofort alle verfügbaren Datenknoten darüber informiert und der „verlorene“ Datenknoten wird gemeinsam als unzugänglich erklärt. Es wird versucht, den unzugänglichen Datenknoten automatisch zu starten und ihn dem Cluster erneut hinzuzufügen. Falls diese Aktion gelungen ist, wird die Kommunikation dieses Knoten mit den anderen Datenknoten des Clusters wieder aufgenommen.

2. Fehlerfall: Knoten reagiert nicht

In diesem Fall existiert zwar die Verbindung zu dem Knoten, der Knoten reagiert aber nicht erwartungsgemäß. Ursache dafür kann ein physischer Fehler sein, zum Beispiel eine defekte Platte oder auch eine Überlastung des Prozessors. Hierbei wird das so genannte Heartbeat-Protokoll eingesetzt, um die Art des aufgetretenen Fehlers zu bestimmen.

Das *Heartbeat-Protokoll* ist organisiert wie ein logischer Kreis (siehe Abbildung 4.6). Jeder Datenknoten sendet kleine Nachrichten („Bist du am Leben?“) an den nächsten Datenknoten in dem Kreis. Nach dem Empfang einer Heartbeat-Nachricht wird im Regelfall eine Antwort versendet. Antwortet der Empfänger auf drei aufeinander folgende Nachrichten nicht, so wird der Datenknoten von den verbliebenen Knoten als unzugänglich erklärt und damit die Überprüfung des Knotenfehlers beendet.

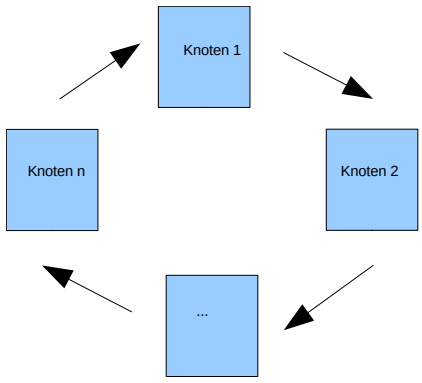


Abbildung 4.6: *Heartbeat-Protokoll*
[Quelle: In Anlehnung an Ronstrom, M., Thalmann, L. (Architektur 2004)]

3. Fehlerfall: Netzwerkpartitionierung

Eine *Netzwerkpartitionierung* entsteht dann, wenn das Rechnernetz durch Kommunikationsunterbrechungen temporär in isolierte Teilnetze zerfällt.[109] Eine Netzwerkpartitionierung muss behandelt werden, da Änderungen an Daten in mehreren Teilnetzen zu gravierenden Inkonsistenzen der Datenbank führen können. Im MySQL Cluster wird eine Netzwerkpartitionierung in vereinfachter Darstellung wie folgt berücksichtigt:

[109] Die Beschreibung der Fehlerfälle basiert weitgehend auf Ronstrom, M., Thalmann, L. (Architektur 2004), S. 6-8.

Wenn ein oder mehrere Datenknoten abstürzen, wird das Netzwerkpartitionierungsprotokoll eingesetzt. Es können zwei Fälle auftreten: Beide Teilnetze haben (a) unterschiedliche oder (b) gleiche Anzahl an Datenknoten. Die Behandlung der beiden Fälle ist wie folgt:

- Fall (a): Teilnetze haben unterschiedliche Knotenanzahl

  Wenn ein Teilnetz mehr als 50% der verbliebenen Knoten beinhaltet, dann liegt das so genannte „Die Mehrheit regiert“ (engl. „Majority Rules“)-Szenario vor. In diesem Fall zählen nur Knoten aus diesem Teilnetz zum Cluster.[110] Knoten aus dem anderen Teilnetz gehören also vorübergehend nicht zu dem MySQL Cluster und werden erst nach der Zusammenführung des Netzes zu dem Cluster hinzugefügt.

- Fall (b): Teilnetze haben gleiche Knotenanzahl

  Ein Koordinator (auch Arbitrator bei MySQL AB genannt) kommt dann zum Einsatz, wenn beide Teilnetze gleiche Anzahl an Knoten haben. In diesem Fall wird die Knotenmenge als MySQL Cluster betrachtet, zu welcher der Koordinator gehört. Alle anderen Knoten werden automatisch als unzugänglich erklärt und erst nach der Zusammenführung des Netzes zu dem Cluster hinzugefügt.

Aus der Beschreibung der Konzepte zur Interprozessorkommunikation im MySQL Cluster werden typische Probleme der SN-Architektur sichtbar. In der Clusterumgebung werden i.d.R. über das Netzwerk häufig sehr viele kleine aber auch große Datenmengen übertragen. Die große Anzahl von kleinen Nachrichten ist hauptsächlich durch die interne Kommunikation zwischen den Knoten verursacht. Die Übertragung von großen Datenmengen kann hauptsächlich durch verteilte Anfragenausführungen entstehen. Daraus lässt sich abschätzen, dass bei diesem Datenbanksystem bei sehr großer Anzahl von verteilten Anfragen oder Transaktionen das Netzwerk zum Engpass werden könnte.

Cache-Kohärenzkontrolle

Im MySQL Cluster existiert das Problem der Cache-Kohärenzkontrolle nicht, da es keine gemeinsame Nutzung von Resourcen gibt. Deshalb entfällt eine Realisierung dieser Anforderung komplett.

Synchronisation

Die Synchronisation von nebenläufigen verteilten Transaktionen im MySQL Cluster geschieht auf der NDB-Engine-Speicherebene und wird mithilfe des modifizierten 2-Phasen-Commit (2PC)-Protokolls unterstützt.[111] Der Einsatz dieses Protokolls erfordert generell zusätzlichen Kommunikationsaufwand. [112] Das modifizierte 2PC-Protokoll beinhaltet folgende Änderungen:

- Modifikationen an Daten durch bereits abgeschlossene Transaktionen werden im Hauptspeicher gehalten und nicht sofort auf die Festplatte geschrieben. Erst nach einem globalen Checkpoint werden Änderungen persistent gespeichert. Zur Erinnerung: „Ein *Checkpoint* ist eine Informationsmenge, die zu einem bestimmten Zeitpunkt vom System an das Datenbanksystem übergeben werden

[110] Vgl. MySQL (Handbuch 2007), S. 1063.
[111] Weitere Ausführungen basieren weitgehend auf MySQL (NDB-API 2006), S. 12-13.
[112] Vgl. Dadam, P. (Datenbanken 1996), KE 5 S 41-46.

kann. Ein Checkpoint enthält im einfachsten Fall die Liste der zu diesem Zeitpunkt aktiven Transaktionen."[113]

Das verzögerte Festschreiben der Änderungen auf die Festplatte begründet MySQL AB damit, dass geänderte Daten i.Allg. redundant auf anderen Knoten vorliegen und im Fehlerfall Kopien anderer Knoten zur Verfügung stehen. Es besteht allerdings die Gefahr des Änderungenverlustes, falls alle Kopien gleichzeitig ausfallen sollten. In diesem Fall gehen alle Änderungen bis zu dem letzten globalen Checkpoint verloren. Die Wahrscheinlichkeit des Ausfalls aller Kopien ist jedoch relativ gering.

- Im MySQL Cluster treten Blockierungen von Teilnehmern einer Transaktion beim Verlust des Transaktionskoordinators nicht auf, da Teilnehmer laufend Informationen zu allen Ausfällen von Datenknoten untereinander austauschen und den Ausfall des Transaktionskoordinators sehr schnell feststellen können.

Im MySQL Cluster gibt es keine Algorithmen zur Erkennung von Verklemmungen. Wenn sich eine verteilte Transaktion nach dem Ablauf einer festgelegten Zeit nicht gemeldet hat, so geht man davon aus, dass eventuell eine Verklemmung vorliegt.[114] In diesem Fall wird die verteilte Transaktion abgebrochen und anschließend erneut gestartet.

Da im MySQL Cluster in einer Transaktion alle redundant gespeicherten Daten synchron aktualisiert werden, gibt es im Cluster so gut wie keine rein lokalen Transaktionen, es sei denn, dass nicht alle (Teil-)Relationen Kopien aufweisen. Das ist zwar im MySQL Cluster erlaubt, jedoch wird in diesem Fall keine Hochverfügbarkeit des Clusters garantiert.

MySQL AB empfiehlt, um die Anzahl der abgebrochenen Transaktionen so gering wie möglich zu halten, die Anzahl der DB-Operationen pro Transaktion möglichst klein zu halten.[115] Diese Empfehlung kann grundsätzlich keine Lösung für das Problem sein, da viele aus mehreren Anweisungen bestehende Transaktionen eine logische Einheit bilden und in einer ununterbrochenen Abfolge nach dem Prinzip „Alles oder nichts" ausgeführt werden müssen.

Bei Transaktionen wird im MySQL Cluster nur die so genannte READ-COMMITTED-Isolationsebene unterstützt:[116] Beim konsistenten Lesen werden keine Sperren auf die benutzten Tabellen angefordert, sodass andere Benutzer diese Tabellen nach Belieben modifizieren können, während eine konsistente Leseoperation auf ihnen abläuft. Diese Transaktionsisolationsebene reicht jedoch nicht in allen Anwendungsfällen aus.

Bei der Deadlockbehandlung kann es zu folgenden Situationen kommen:

- Bei Verklemmungen von Transaktionen wird mindestens eine in den Deadlock verwickelte Transaktion abgebrochen und anschließend erneut gestartet. Ist dieselbe Transaktion erneut in einen Deadlock verwickelt, so muss ein Benutzer unter Umständen sehr lange auf die Durchführung seiner Transaktion warten. Das führt zur Minimierung des Durchsatzes und einen erhöhten Verbrauch von Systemressourcen.

[113] Schlageter G., Wilkes, W., Balzer, M.(DB1 2005), Glossar.
[114] Vgl. zu diesem Absatz MySQL (NDB-API 2006), S. 12-13.
[115] Vgl. MySQL (Handbuch 2007).
[116] Vgl. zu diesem Absatz MySQL (Handbuch 2007), S. 895.

- Langlaufende Transaktionen können besonders benachteiligt werden: Dadurch, dass Verklemmungen anhand der Ausnahmen zurzeitüberschreitung bestimmt werden, können Transaktionen voraussichtlich unfair als blockiert erkannt und zurückgesetzt werden.

Parallele Anfragenausführung

MySQL Cluster bietet zwei wichtige Alternativen zu Tabellentypen an, die eine direkte Auswirkung auf Antwortzeiten und den Durchsatz von DB-Anfragen haben. Die erste Alternative stellt hauptspeicherresidente und die zweite festplattenbasierte Tabellen dar.[117] Wie die Namen schon andeuten, liegen alle Daten einer hauptspeicherresidenten Tabelle vollständig im Hauptspeicher, während bei einer *festplattenbasierten Tabelle* nur Indizes von Tabellen im Hauptspeicher gehalten werden. Alle anderen Daten einer festplattenbasierten Tabelle befinden sich i.d.R. auf einer oder mehreren Festplatte(n). Da Zugriffszeiten bei Festplatten deutlich höher als beim Hauptspeicher sind, ist der Einsatz von festplattenbasierten Tabellen für Hochleistungs-DBS weniger empfehlenswert.

Derzeit müssen alle Datenknoten in der MySQL Clusterumgebung einen gleich großen Hauptspeicher besitzen.[118] Da naturgemäß Relationen einer Datenbank nicht gleich groß sind, führt es dazu, dass die Relationen in Partitionen nicht nach logisch zusammenhängenden Daten, sondern hauptsächlich nach verfügbarer Hauptspeicherressource aufgeteilt werden. Beim Ausführen von Anfragen müssen dadurch Zugriffe eventuell häufiger auf andere Datenknoten gemacht werden als es notwendig wäre.

Parallelisierung von Anfragen

MySQL Cluster unterstützt Parallelisierung von DB-Operationen über mehrere Datenknoten hinweg.[119] Ein Datenknotenprozess entscheidet, bei welchen Anfragen und bis zu welchem Parallelisierungsgrad DB-Operationen parallel ausgeführt werden. Muss beispielsweise ein vollständiger Tabellen-Scan (d.h. sequentielles Lesen der Tabelle) durchgeführt werden, dann wird eine DB-Operation in eine Menge von Teiloperationen zerlegt und zur parallelen Ausführung an mehrere Datenknotenprozesse im Cluster verteilt. Vorausgesetzt wird dabei, dass dieselben Daten auf mehreren Datenknoten vorliegen.

Des Weiteren sind parallele Zugriffe auf Indizes möglich, aber nur dann, wenn deren Werte über mehrere Datenknoten verteilt sind und zum Primärschlüssel der Relation gehören.

MySQL bietet zur Optimierung von Anfragen folgende zusätzliche Parallelisierungsmöglichkeiten an:[120]

- Datenknotenebene

  Es besteht die Möglichkeit, dass DB-Operationen auf den einzelnen Datenknoten ausgeführt werden und nur Zwischenergebnisse statt der ganzen (Teil-)Relationen über das Netzwerk übertragen werden. Dieser Parameter (auch

[117] Vgl. MySQL (Handbuch 2007), S. 1059 vgl. dazu auch MySQL (5.1-Neuerungen 2006), S. 6-10.
[118] Vgl. MySQL (Handbuch 2007), S. 1060.
[119] Vgl. zu diesem Absatz MySQL (Handbuch 2007), S. 1051.
[120] Weitere Ausführungen stützen sich weitgehend auf MySQL (Handbuch 2007), S. 1010-1013, 1056-1057.

Pushdown-Bedingung genannt) ist ab MySQL Cluster Version 5.1 standardmäßig voreingestellt.

- SQL-Knotenebene

  Alternativ können auch MySQL-Serverprozesse DB-Operationen parallel ausführen, wenn der entsprechende Parameter gesetzt ist.[121] So laufen Anfragen auf nicht indizierte Spalten unter Umständen deutlich schneller als zuvor, und zwar dann, wenn eine Anfrage über mehrere Datenknoten parallel bearbeitet werden kann.

- Clusterebene

  Es besteht die Möglichkeit, den Grad der Parallelität im Cluster einzustellen.[122] Jeder Transaktionskoordinator kann so viele parallele Teiloperationen starten, wie dieser Parameter erlaubt.

Partitionierung

MySQL Cluster unterstützt horizontale und horizontale abgeleitete Partitionierungen von Relationen. Vertikale Partitionierung kennt MySQL Cluster nicht. Die Aufteilung einer Relation in mehrere Partitionen bleibt für Benutzer und Anwendungen transparent.

Das gesamte Konzept der Datenverteilung im MySQL Cluster sieht wie folgt aus: Zunächst werden globale Relationen in Partitionen aufgeteilt.[123] Danach wird die Anzahl der redundant gespeicherten Kopien für die Partitionen der globalen Relationen festgelegt. Anschließend werden für alle (redundant zu haltenden) Partitionen Allokationsorte bestimmt. Datenknoten werden zusätzlich in Knotengruppen logisch zusammengefasst. Zu einer Knotengruppe gehören Datenknoten, welche dieselben Partitionen von Relationen enthalten.

MySQL Cluster unterstützt folgende horizontale Partitionsstypen:[124]

- RANGE-(Bereichs-),
- LIST-(Listen-),
- HASH-(Zufalls-) und
- KEY-(Schlüssel-)Partitionierung.

In MySQL 5.1 können Relationen, die durch RANGE oder LIST partitioniert wurden, noch weiter in Unterpartitionen aufgeteilt werden. Unterpartitionen können entweder HASH- oder KEY-Partitionen sein.[125]

---

[121] Siehe Option engine-condition-pushdown.
[122] Siehe Parameter MaxNoOfConcurrentScans. Der Standardwert von MaxNoOfConcurrentScans ist 256 und der Höchstwert beträgt 500.
[123] Vgl. MySQL (Handbuch 2007), S. 986-988.
[124] Vgl. zu diesen Ausführungen MySQL (Handbuch 2007), S. 1070-1096.
[125] Vgl. MySQL (5.1-Partitioning 2006), S. 6-19.

Nachteilig beim MySQL Cluster ist, dass partitionierte Relationen keine Fremdschlüssel erlauben.[126] Des Weiteren werden keine Indizes für den Datentyp TEXT unterstützt. Jedoch kann statt TEXT der Datentyp VARCHAR verwendet werden, der die Erstellung von Indizes erlaubt. Die vollständige Aufzählung aller Einschränkungen sind in der MySQL-Referenz zu finden.

Anfragenoptimierung

MySQL Cluster sorgt für optimale Ausführung von DB-Operationen.[127] Jedoch konnte im Rahmen der Recherche nicht ermittelt werden, welche Anfragenoptimierungsalgorithmen für parallele und verteilte Anfragenausführung tatsächlich verwendet werden.

Einfache Administration

Zur Verwaltung eines MySQL Clusters gehören diverse Aufgaben, darunter das Konfigurieren und das Starten von MySQL Clustern. Im Wesentlichen gibt es zwei Methoden, um einen MySQL Cluster zu verwalten:[128]

- Management-Clienttool: Ein *Management-Client* kann sich mit einem Managementknoten verbinden, um administrative Aufgaben auf dem Cluster durchzuführen. Man gibt Befehle in das Management-Clienttool ein, um unter anderem den Clusterstatus zu prüfen und einzelne Knoten anzuhalten, hoch- oder herunterzufahren. Das Tool verfügt zurzeit über zehn Grundbefehle, die zur Verwaltung des Clusters verwendet werden können.

  Da der Management-Client lediglich zu Verwaltungszwecken des Datenbanksystems eingesetzt wird, bleibt der MySQL Cluster unabhängig von dem Status bzw. der Verfügbarkeit des Management-Clients.
- Man prüft den Inhalt der Protokolldatei auf dem Management-Server des MySQL Clusters. Die Protokolldatei enthält Ereignisberichte, die von *ndbd*-Prozessen der Datenknoten generiert werden.

Da es keine graphischen Werkzeuge zur Administration des MySQL Clusters gibt, ist eine benutzerfreundliche Administration des MySQL Clusters nur eingeschränkt möglich. Alle Befehle auf der Management-Konsolenebene müssen, wie bereits erwähnt, manuell eingegeben werden. Da Administrationsaufgaben sich auf wenige Schritte beschränken, stellt das Fehlen der graphischen Hilfswerkzeuge kein großes Hindernis dar.

Das Hinzufügen oder das Entfernen von Management- und SQL-Knoten im MySQL Cluster erfolgt relativ einfach und hat keine Auswirkung auf andere Knoten des Clusters. Deutlich aufwändiger werden Datenknoten zum Cluster hinzugefügt. So müssen alle Datenknoten vorher gesichert, heruntergefahren und danach wiederhergestellt werden.[129] Das erfordert einen Neustart des Clusters. Nach dem Ausfall eines Datenknotens erfolgt die Wiederaufnahme des Knotens automatisch und ohne einen administrativen Eingriff. Da MySQL-Cluster die SN-Architektur besitzt, muss i.d.R. Datenneuverteilung beim Hinzufügen neuer Knoten durchgeführt werden. Das stellt jedoch eine relativ schwierige administrative Aufgabe dar.

---

[126] Vgl. zu diesem Absatz MySQL (Reference 2007), S. 1106-1110.
[127] Vgl. MySQL (Handbuch 2007).
[128] Vgl. zu diesen Ausführungen MySQL (Handbuch 2007), S. 1043-1047.
[129] Vgl. MySQL (Handbuch 2007), S. 1062.

Kurze Antwortzeiten/hoher Durchsatz

Datenhaltung im Hauptspeicher erlaubt generell nicht nur sehr kurze Antwortzeiten, sondern unterstützt auch einen linearen Antwortzeit-Speedup und Durchsatz-Scaleup bei Erhöhung der CPU-Kapazität. [130] Wegen der strikten Grenzen für die Datenknotenanzahl und wegen der allgemeinen Skalierbarkeitseinschränkung für die Hauptspeicherressource könnte der Einsatz des MySQL Clusters unter folgenden Umständen problematisch sein:

> Kurze Antwortzeiten/hoher Durchsatz sind bei sehr großen Datenmengen (z.B. 50 TByte) eher unwahrscheinlich: Reicht die Hauptspeicherressource nicht aus, so kann ein weiterer Betrieb nicht fortgesetzt werden, es sei denn, dass festplattenbasierte Tabellen verwendet werden. Solche Tabellen können allerdings die Leistung des Clusters stark einschränken, da sie permanente Zugriffe auf Festplatten erfordern.

Kosteneffektivität im MySQL Cluster

MySQL Cluster, der ein Minimum an Verfügbarkeit und Datensicherheit bietet, besteht aus vier Rechnerknoten, die eine feste Netzwerkadresse besitzen.[131] Interessanterweise gibt es sehr unterschiedliche Meinungen, was die Mindestanforderung für Hardware der Datenknoten eines MySQL Clusters betrifft (siehe Tabelle 4.1).

Hardware	**Empfehlungen aus dem MySQL-Handbuch**	**Empfehlungen von MySQL-Experten**[97]
Prozessor	1 x Intel-CPU	2 x Intel Xeon, Intel Inanium usw.
Hauptspeicher	256 MByte für eine Test-DB namens „World“	16 GByte
Sekundärspeicher	Standardfestplatte	4 x 36 GB SCSI (RAID-1-Kontroller)
Netzwerk	100 Mbit/s oder 1 Gbit/s (empfohlen)	< 8 Knoten – 1 Gbit/s Ethernet > 8 Knoten – SCI

Tabelle 4.1: *Empfehlungen zur Hardware für Datenknoten eines MySQL Clusters*

Da MySQL Cluster Doppellizenzierung anbietet, können bei diesem Datenbanksystem die Software- und Lizenzkosten entfallen oder im Vergleich zu vielen proprietären kommerziellen Lösungen relativ gering sein.[132] Wartungskosten fallen i.d.R. ebenfalls kaum ins Gewicht, nicht zuletzt dadurch, dass die Komplexität des Produktes relativ gering ist und die Wartungsarbeiten zumeist schnell durchgeführt werden können.

---

[130] Vgl. Rahm, E. (Transaktionssysteme 1993b), S. 83.
[131] Vgl. MySQL (Handbuch 2007), S. 988-990.
[132] Vgl. MySQL AB: Was ist MySQL?, URL: http://dev.mysql.com/doc/refman/5.1/de/what-is.html, Abruf am 05.04.2007.

MySQL AB veröffentlichte in Februar 2005 ein Dokument, in dem behauptet wird, dass mit dem lizenzierten Einsatz der MySQL-Datenbank Kosten bis zu 90% hätten eingespart werden können.[133] So zeigt eine Studie durch Meta Group[134] zum Vergleich von Datenbankkosten, dass MySQL deutlich günstiger als Produkte von anderen kommerziellen Anbietern sein kann (siehe Abbildung 4.7). Einige Statistiken dieses Dokumentes können zum Teil auch auf MySQL Cluster übertragen werden.

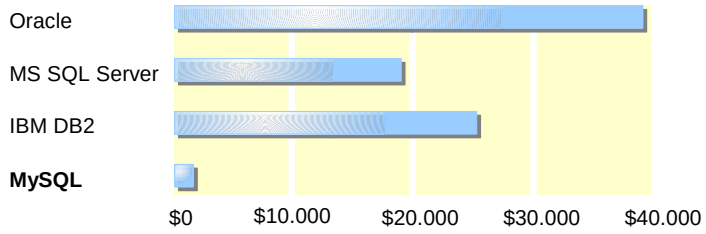


***Abbildung 4.7**: Vergleich der DBMS-Lizenzkosten pro Jahr*
[Quelle: In Anlehnung an MySQL (TCO 2005)]

Beim MySQL Cluster sind Lizenz- und Wartungskosten minimal, trotzdem ist die Kosteneffektivität aufgrund der geringen Skalierbarkeit des Datenbanksystems bei großen Datenmengen niedrig: Werden nur hauptspeicherresidente Tabellen eingesetzt, so hat man den Fall eines Hauptspeicher-Datenbanksystems. Hauptspeicher-Datenbanksysteme bieten zwar das beste E/A-Verhalten, jedoch weisen sie eine relativ geringe Kosteneffektivität aufgrund der hohen Hauptspeicherkosten auf.[135] Beim MySQL Cluster existiert noch eine weitere Einschränkung bezüglich der Datenknotenanzahl. Die geringen Gesamtbetriebskosten für MySQL Cluster wirken deshalb nur bis zu einer bestimmten DB-Größe.

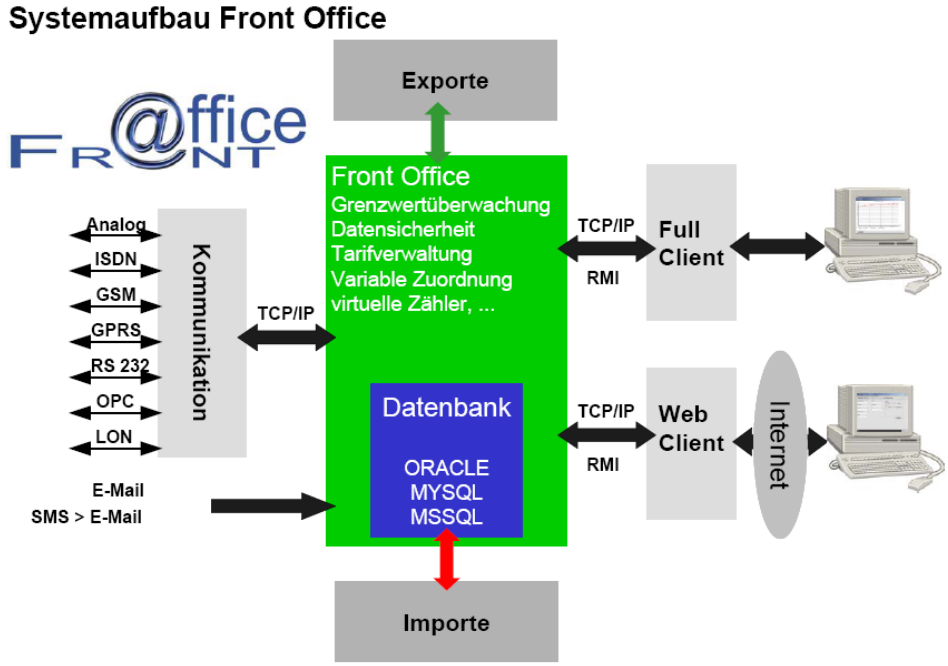
[133] Vgl. MySQL (TCO 2005).
[134] Meta Group Inc. war eine Marktforschungsorganisation und gehört jetzt zu Gartner Group Inc. Vgl. Homepage: http://www.gartner.com/, Abruf am 22.03.2007.
[135] Vgl. Rahm, E. (Transaktionssysteme 1993b), S. 83-87.

ERPROBUNG DER APPLIKATION FRONTOFFICE

VORSTELLUNG DER ANWENDUNG OFFICE-FRONT

Office-Front [136] ist ein für den Energiemarkt entwickeltes Produkt zur Erfassung, Verwaltung und Auswertung von Daten der verschiedenen Medien wie Strom, Gas oder Wasser.

Office-Front besitzt eine mehrschichtige Softwarearchitektur, wobei eine persistente Datenhaltung auf der Basis eines relationalen Datenbanksystems realisiert ist (siehe Abbildung 5.1). Zurzeit stehen für Office-Front drei relationale Datenbanksysteme zur Auswahl: Oracle Database 10g (RAC), MySQL (MySQL Cluster 5.1) und Microsoft SQL Server 2005.[137]



Das Datenbankschema von der Applikation FrontOffice enthält insgesamt 67 Tabellen. Das Schema hat folgende Merkmale:

Schemaeigenschaft	Anzahl	Anmerkung
Temporäre Tabellen	-	
TEXT, Bit- und Blob-Datentypen	-	
VARCHAR-Datentyp	203 Attribute	
AUTO_INCREMENT	4	Nur für Primärschlüssel
Anzahl der Attribute pro Tabelle	< 128	
Anzahl der Indizes	128	

[136] Das Produkt Office-Front wurde von der Firma SW-System GmbH in Oldenburg entwickelt. Vgl. Homepage der Firma SW-System GmbH: http://www.fw-systeme.de, Abruf am 22.03.2007.
[137] RAC und MySQL Cluster wurden bereits detailiert besprochen und bewertet.

Anzahl der referentiellen Integritäten	107	

Bei großen Energieversorgungsunternehmen werden i.d.R. sehr viele Geräte gleichzeitig und im Minuten- oder Stundentakt ausgelesen. Die Größe der Datenbank hängt von der Anzahl und der Frequenz der Ablesevorgänge der Geräte ab. Darüber hinaus ist von Bedeutung, wieviele Jahre Gerätedaten in der Datenbank gehalten werden. Je nach der Konstellation können also sehr große Datenmengen in der Datenbank gespeichert sein. Es besteht häufig die Notwendigkeit der Datenhaltung in einem Hochleistungs-DBS. Folgende Anforderungen an das eingesetzte Datenbankssystem ergeben sich für die Verwendung der Software FrontOffice:

- Hochverfügbarkeit
- Parallele Anfragenausführung
- Lastbalancierung
- kurze Antwortzeiten.

TESTZIELE

Für die Durchführung der Last- und Stresstests können spezifische Testziele gebildet werden:[138]

- Treten unter Last irgendwelche funktionalen Fehler der Software auf?
- Wie verhält sich die Anwendung bei großen Datenmengen?
- Werden die Tarifwechsel korrekt umgesetzt?
- Lässt sich eine Systemobergrenze ermitteln?
- Ist der Test-RAC-Cluster mit dem eingesetzten Datenbanksystem für die Anwendung geeignet?

TESTFÄLLE

Die Testfälle der Last- und Stresstests sollen einen möglichst realen Ablauf von Geschäftsfällen darstellen. Somit stellen sie reale Anwendungen zwischen dem Anwender, der Software, der Geräte und der Datenbank dar.

Testfallgruppe Simulationstool

Die erste Testfallgruppe beinhaltet die Anwendungsfälle, die mit Hilfe des Simulationstools realisiert werden. Diese Testfälle stellen die Ablesevorgänge der Energiedaten für die unterschiedlichen Kanäle dar. Es lassen sich vier konkrete Anwendungsfälle durch das Simulationstool beschreiben (siehe Abbildung 10).

[138] Dieser Abschnitt stützt sich im Folgenden weitgehend auf Holtmann, A. (Lasttests 2008).

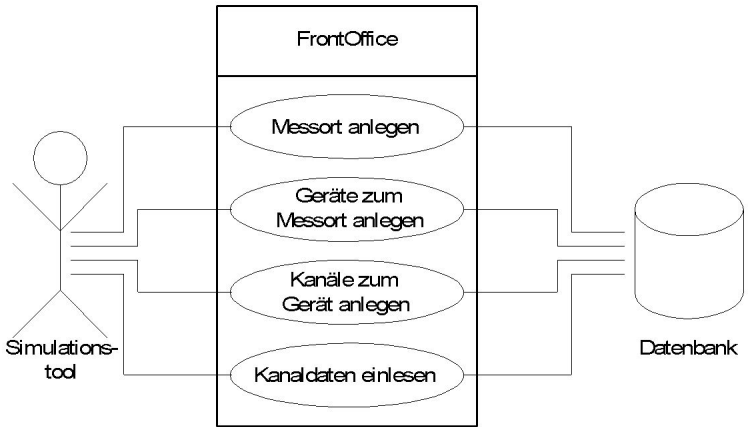


Abbildung 10: Anwendungsfälle im Simulationstool

Hier sind die Simulations-Eckdaten für Tests von FrontOffice. Die Lasterzeugung wurde mit dem entwickelten Lastsimulationstool für FrontOffice durchgeführt.

Simulations-Eckdaten	
Messorte	9
Geräte/Kanäle pro Messort	500/8
Datenbestand	10 Jahren
Frequenz der Geräteablesung	15 Min
Datenvolumen	180 GB
Datensätze (Energieverbrauchsdaten)	2 Mrd.

Testfallgruppe Funktionen

Die zweite Testfallgruppe beinhaltet Anwendungsfälle, welche Verwaltungsfunktionen der Software FrontOffice darstellen. Dabei werden die häufigsten durchgeführten und ressourcenintensivsten Geschäftsfälle im Anwendungssystem abgebildet (siehe Abbildung 11).

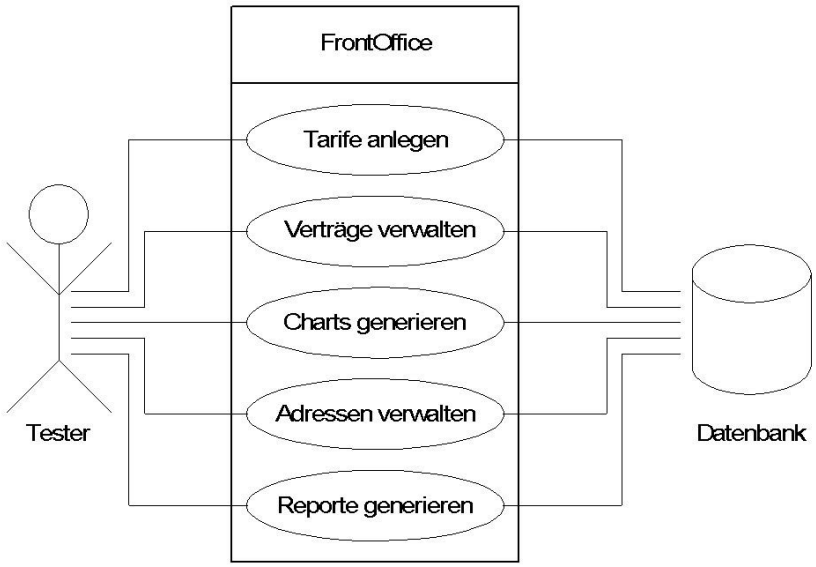
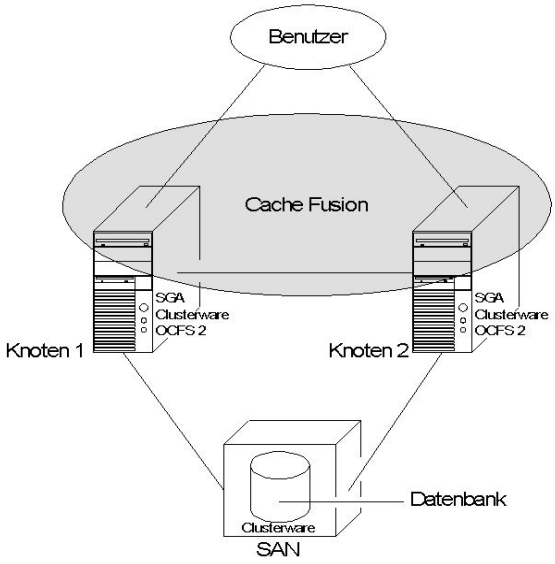


Abbildung 11: Geschäftsfälle im Anwendungssystem

TESTUMGEBUNG ORACLE RAC



Die Testumgebung vom Oracle RAC besteht aus zwei Knoten. Die beiden Clusterknoten sind direkt mit einem SAN-System verbunden. Die Anmeldung der Benutzer von FrontOffice kann auf jedem der Knoten durchgeführt werden. Hier sind die Eckdaten zu der Clusterknoten:

Hardware	
CPU	2 x Intel Xenon 3,2 GHz

RAM	4 GB (SGA 1,2 GB)
Privates Netzwerk	2 Gbit/s
Öffentliches Netzwerk	100 Mbit/s
OS	Red Had EL 4

Die Lizenzkosten inkl. Supportkosten für diese Clusterumgebung in Wilhelmshaven bei der Lizenzart ‚Prozessor' würde ca. 340.000 USD pro Jahr betragen.

Erprobung des Simulationstools

Simulation steigende Arbeitslast

Die steigende Arbeitslast konnte mit neun Lastgeneratoren realisiert werden. Dabei wurde keine Systemobergrenze ermittelt. Der RAC Clusters akzeptierte ohne Performanceverluste eine Aufnahme der generierten Daten. Eine Lastverteilung auf dem zweiten Knoten erfolgte ebenfalls nicht. Der benutzte Knoten zeigte einen Anstieg der Prozessorauslastung nur bis zu 60% während der Testphase an. Somit verfügte das System noch über genügend Leistung und eine Stresssituation trat nicht ein. Die Auslastung des benutzten Knotens sowie die Netzwerkauslastung sind in der nachfolgenden Grafik (siehe Abbildung 12) gekennzeichnet.

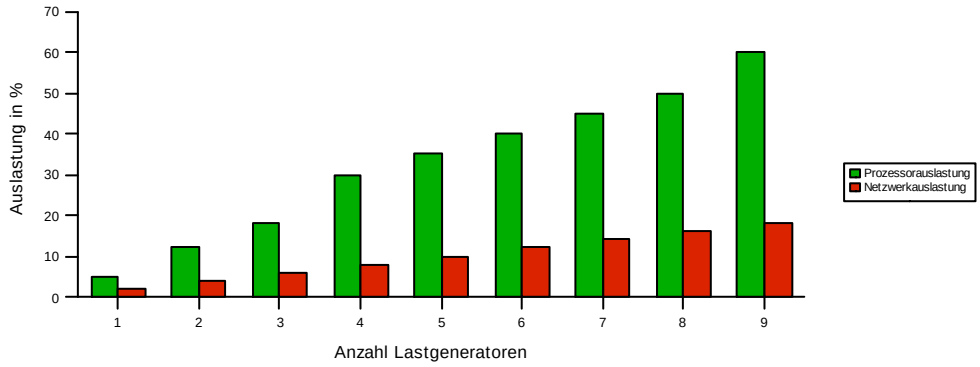


Abbildung 12: Prozessor- und Netzwerkauslastung des RAC Clusters

Simulation szenario-basierte Arbeitslast

Aufgrund der Realisierung der steigenden Arbeitslast, wurde ebenfalls diese Testphase ohne Probleme durchgeführt. Nachfolgende Grafik (siehe Abbildung 14) zeigt den durchschnittlich erzielten Durchsatz der gespeicherten Daten in der Datenbank, für eine jeweilige Lasterzeugung von ein bis neun Lastgeneratoren.

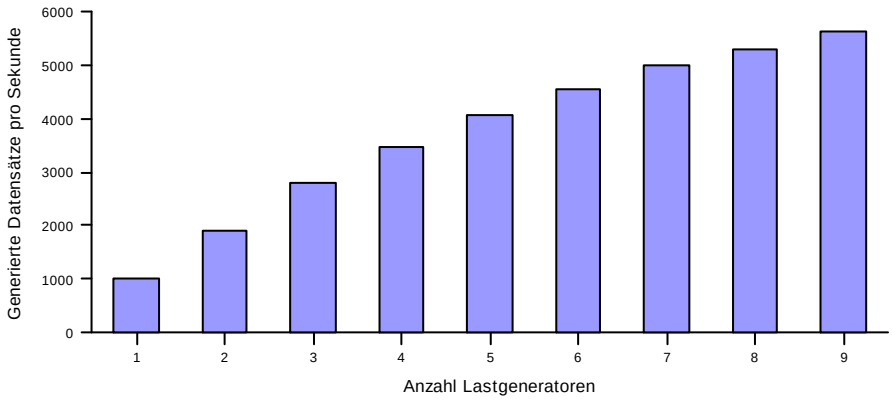


Abbildung 13: Durchsatzrate des Simulationstools auf dem Test-RAC-Cluster

Ergebnisse Testphase „Manuelle Testausführung“

Simulation szenario-basierte Arbeitslast

Es wurden Reporte und Auswertungen während allen neun Lastphasen ausgeführt. Dabei ließ sich kein Zeitunterschied zwischen den Ausführungen bei unterschiedlichen Lasten erkennen. Somit wird auch hier keine Stresssituation des Systems erreicht. Der erzeugte Durchsatz der Simulation des Testtools blieb konstant zur szenario-basierten Arbeitslast mit dem Simulationstool.

Langzeittest

Ein Langzeittest von über 100 Stunden konnte realisiert werden. Dabei wurde ein Datenbankbestand von zwei Milliarden Datensätzen erreicht. Dieser Datenbankbestand entspricht einer Festplattengröße von 180 GB. Die Ausführung von Reporten und Auswertungen wurde in diesem Langzeittest ebenfalls, ohne für den Tester erkennbaren erhöhten Zeitaufwand, realisiert. Während dieser Phase traten kleinere Fehler mit der Software auf. Ein Problem betraf den konkurrierenden Zugriff von FrontOffice auf die Datenbank. Da FrontOffice drei Verbindungen gleichzeitig zur Datenbank öffnet kam es zu Synchronisationsproblemen. Diese konnten jedoch durch ein Update der Software behoben werden. So war ein reibungsloser Verlauf der Tests weiterhin gegeben.

Die folgenden Tabellen zeigen die Antwortzeiten der Generierung von Charts und Reporte bei unterschiedlichen Datenbankbeständen.

Testfall: Charts generieren (Serverauslastung 60%)				
	1 Tag	1 Woche	1 Monat	1 Jahr
< 10 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek
< 20 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek
< 50 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek

< 100 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek
< 150 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek
< 180 GB	< 1 Sek	< 1 Sek	< 1 Sek	< 5 Sek

Tabelle 11: Durchschnittliche Antwortzeiten der Generierung von Charts bei unterschiedlichen Datenbankgrößen und einer Auslastung von neun Lastgeneratoren

Testfall: Reporte für ein Gerät mit 8 Kanälen[139] (Serverauslastung 60%)				
	1 Tag	1 Woche	1 Monat	1 Jahr
< 10 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek
< 20 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek
< 50 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek
< 100 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek
< 150 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek
< 180 GB	< 1 Sek	< 5 Sek	< 10 Sek	< 75 Sek

Tabelle 12: Durchschnittliche Antwortzeiten der Generierung von Reporten bei unterschiedlichen Datenbankgrößen und einer Auslastung von neun Lastgeneratoren

Nachfolgende Grafik (siehe Abbildung 13) zeigt die Ausführungszeiten für die resourcenintensivsten SQL-Anweisungen. Die Einfüge-Operationen weisen einen stabilen Verlauf auf. So konnte eine konstante Durchsatzrate während der gesamten Testdurchführung erreicht werden. Mit Zunahme der Geräte innerhalb der Datenbank, nimmt die Ausführungsdauer der SQL-Abfrage nach den Geräten linear zu. Diese bewegt sich jedoch in einem normalen Rahmen und weist auf keine Probleme mit der gegebenen Tabellenstruktur hin.

[139] Reporte werden als HTML-Dateien auf dem Ausführungsrechner gespeichert.

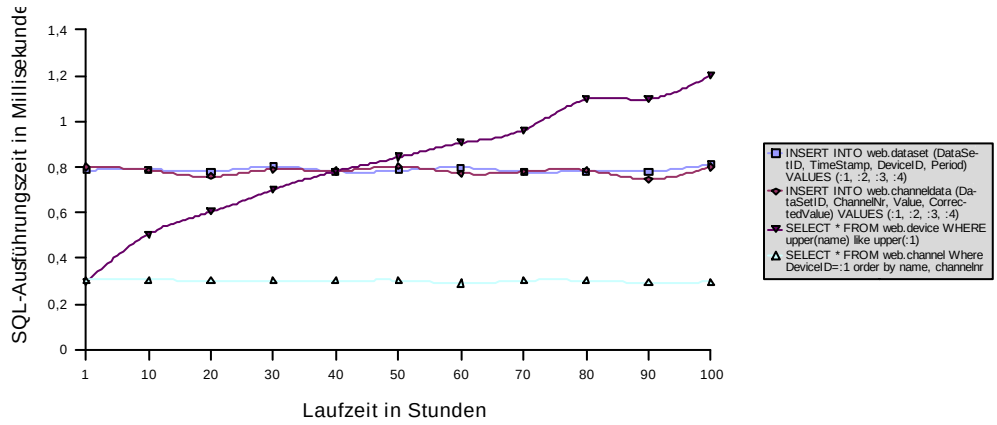


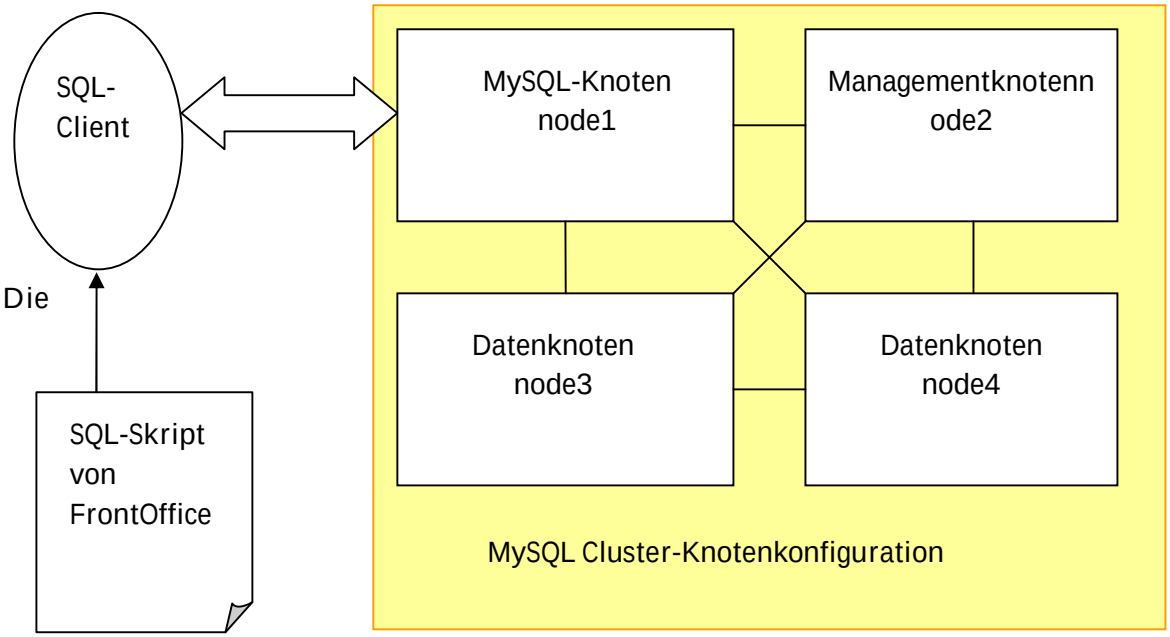
Abbildung 14: *Ausführungszeit der SQL-Anweisungen des Simulationstools bei konstanter Belastung von neun Lastgeneratoren*

TESTUMGEBUNG: MYSQL CLUSTER

Dieser Testbericht basiert auf der folgenden MySQL Cluster-Knotenkonfiguration:

- 2 Datenknoten
- 1 Managementknoten
- 1 MySQL-Knoten

SQL-Skripte-Installation:



Installation des FrontOffice-Schemas auf dem MySQL Cluster konnte nicht durchgeführt werden. Die Suche nach den Gründen für die aufgetretenen Fehlermeldungen war erfolglos. Laut der Aussage von MySQL AB enthält das Schema eventuell zu viele VARCHAR-Attribute. Diese Einschränkung ist jedoch in der MySQL Cluster-Referenz nicht notiert. Beim Ändern dieser Attribute in einen numerischen Datentyp (Die Umwandlung von VARCHAR-Datentypen in den numerischen Typ

SMALLINT) treten allerdings dieselben Fehlermeldungen auf. Die Firma MySQL AB hat sich zu unserem Installationsproblem nicht weiter geäußert.

Da die Installation der SQL-Skripte auf dem MySQL Cluster misslungen ist, konnten für die MySQL Cluster keine Last- und Stresstests durchgeführt werden.

RESÜMEE

Parallele Datenbanksysteme basieren i.Allg. auf parallelen Rechnersystemen. Solche Rechnersysteme weisen verschiedene Architekturen in Bezug auf die Kopplung der Prozessoren an den Hauptspeicher und Sekundärspeicher auf. Zunehmend besitzen moderne parallele Rechnersysteme hierarchische Architekturen. Parallele Datenbanksysteme können ebenfalls unterschiedliche Architekturen aufweisen. Die Eigenschaften und Merkmale von parallelen Datenbanksystemen hängen stark von der jeweiligen DBS-Architektur ab. Zu den Basisarchitekturen der Datenbanksysteme zählen die SE-, SD- und SN-Architektur. Moderne hierarchische (mehrschichtige) Architekturen paralleler Datenbanksysteme bestehen i.Allg. aus mehreren Clustern. Dabei besitzen einzelne Cluster zumeist eine der Basisarchitekturen und sind i.d.R. nach dem SN-Prinzip miteinander verbunden.

Die Abbildung der Architekturen paralleler Datenbanksysteme auf eine hierarchische Hardwarearchitektur kann flexibel nach dem Prinzip der virtuellen parallelen Datenbankmaschine durchgeführt werden. Dabei wird die Abbildung der DB-Architektur von der Hardwarearchitekturebene bis zur gewünschten Softwareebene (oder umgekehrt) stufenweise mithilfe von speziellen Programmen umgesetzt. Ein Beispiel für ein solches Datenbanksystem stellt die CDN-Architektur dar.

Die wichtigsten Optimierungsmöglichkeiten liegen in der Parallelität für Hardware- und Softwarekomponenten. Parallele Datenbanksysteme ermöglichen die Datenbankverarbeitung auf parallelen Rechnersystemen, sodass die Verarbeitungskapazität zahlreicher Prozessoren zur Leistungssteigerung genutzt werden kann. Die Datenbankparallelität ermöglicht eine Parallelisierung von Transaktionen bis hin zu einzelnen (Teil-)Operationen. Darüber hinaus erlauben Anfragenoptimierungsalgorithmen und die Lastbalancierung die Beschleunigung der (parallelen) Anfragenausführung. Durch die E/A-Parallelität können große Datenmengen parallel von mehreren Festplatten eingelesen werden, vorausgesetzt ist eine entsprechende Verteilung der Daten über mehrere Platten. Eine transparente Datenverteilung auf Hardwareebene ist beispielsweise mit Hilfe von Disk-Arrays ebenfalls möglich.

Zurzeit kann die Hochleistung der parallelen Datenbanksysteme durch eine nachträgliche Erhöhung der Prozessorkapazität i.d.R. nicht beliebig gesteigert werden. Die Leistung ist meistens durch zu hohen Verwaltungs- bzw. Interprozessorkommunikationsaufwand eingeschränkt, der je nach DBS-Architektur durch Datentransport, aufwändige Cache-Kohärenzkontrolle oder Synchronisation der gleichzeitigen Zugriffe verursacht wird.

Ein wichtiger Vorteil paralleler Datenbanksysteme liegt darin, dass beim Einsatz gebündelter leistungsfähiger Standardhardware (insbesondere Mikroprozessoren) eine effiziente Datenverarbeitung und damit eine hohe Kosteneffektivität erreicht werden können. Gleichzeitig kann die Verfügbarkeit des Datenbanksystems nach einem Ausfall

der einzelnen Rechner aus dem Rechnerverbund erhöht werden.[140] Untersuchungen haben gezeigt, dass die SN-Architektur die beste Leistung unter den Basisarchitekturen erbringen kann. Die hierarchische CDN-Architektur ermöglicht die beste Leistung der berücksichtigten parallelen Datenbanksysteme.

Es existieren in der Praxis parallele Hochleistungs-DBS mit unterschiedlichen Architekturen. Die meisten Datenbanksysteme aus der Praxis besitzen klassische Basisarchitekturen. Außer der CE-Architektur kommen hierarchische Systemarchitekturen noch relativ selten vor.

Es wurde ein proprietäres kommerzielles (Oracle RAC) und quelloffenes (MySQL Cluster) Produkt vorgestellt. Bei diesen parallelen Datenbanksystemen wurden folgende Probleme festgestellt:

1. Beide Datenbanksysteme besitzen strikte Grenzen für die Skalierbarkeit: Beim RAC ist die Erweiterbarkeit (zumindest theoretisch) durch die Anzahl von 100 DBMS-Instanzen begrenzt. MySQL Cluster lässt sich generell wegen der Skalierbarkeitsprobleme der Hauptspeicherressource und der begrenzten Datenknotenanzahl (max. 48 Knoten) deutlich schlechter erweitern.
2. Des Weiteren kann es in bestimmten Situationen bei den beiden Datenbanksystemen zu Engpässen kommen. Eine mögliche Engpasssituation im Oracle RAC wäre das Auftreten der vielen gleichzeitigen knotenübergreifenden Zugriffskonflikte. Die Engpasssituation mit der Hautspeicherressource könnte im MySQL Cluster insbesondere bei sehr großen Datenbanken entstehen.
3. Es ist zu vermuten, dass hohe Kosteneffektivität bei beiden Datenbanksystemen im Fall der Engpasssituationen (siehe Punkt 2) wahrscheinlich nicht geboten wird.

Die Firma Oracle hat SUN und MySQL AB übernommen. Die Zukunft des MySQL Clusters ist deshalb unklar. Es bleibt abzuwarten, welche Auswirkungen das auf die Entwicklung des MySQL Clusters haben werden.

[140] Vgl. Dadam, P. (Datenbanken 1996), KE 1 S. 14 vgl. dazu auch Rahm, E. (Mehrrechner-DBS 1994), S. 42.

LITERATURVERZEICHNIS

Claus, V., Schwill, A. (Fachlexikon 2003): *Ein Fachlexikon für Studium und Praxis.* Dudenverlag, 2003.

Härder, Th. (DBMS 2002): *Verteilte und Parallele Datenbanksysteme.* Universität Kaiserslautern, Fachbereich für Informatik, Homepage: 'http://wwwlgis.informatik.uni-kl.de/cms/index.php?id=7', Abruf am 01.11.2007.

Holtmann, A. (Lasttests 2008): Konzeption und Durchführung eines Last- und Stresstestes für eine kommerzielle BDE-Software auf einem Rechnercluster. Fachhochschule O/O/W, Studienort Wilhelmshaven (Prüfer Prof. A. Wulff), Diplomarbeit, 2008.

MySQL (TCO 2005): *Leitfaden zur Senkung der Datenbankkosten (TCO): Wie die Open-Source-Datenbank MySQL bis zu 90% der Kosten senkt.* Ein Business Whitepaper von MySQL, 22. Februar 2005, Homepage: 'http://www.mysql.de/why-mysql/white-papers/', Abruf am 31.10.2007.

MySQL (Handbuch 2007): *MySQL 5.1 Referenzhandbuch.* URL: 'http://dev.mysql.com/doc/', 2007-02-01 (revision: 398), Abruf am 17.02.2007.

MySQL (Hochverfügbarkeit 2007): *Hochverfügbarkeitslösungen von MySQL: Ein Überblick über die Hochverfügbarkeitslösungen von MySQL.* Ein technisches Whitepaper von MySQL, Januar 2007, URL: http://www.mysql.de/why-mysql/white-papers/, Abruf am 17.02.2007.

MySQL (NDB-API 2006): *MySQL NDB API DEVELOPERS' GUIDE: VERSION 2.0.* URL: http://dev.mysql.com/doc/, 10.11.2006 (revision: 3902), Abruf am 17.02.2007.

MySQL (5.1-Neuerungen 2006): *Die neuen Leistungsmerkmale von MySQL Cluster 5.1.* URL: http://www.mysql.de/why-mysql/white-papers/, September 2006, Abruf am 16.02.2007.

MySQL (5.1-Partitioning 2006): *Guide to MySQL 5.1 Partitioning.* URL: http://www.mysql.com/why-mysql/white-papers/, 2006-10-26 (revision: 3752), Abruf am 15.04.2007.

MySQL (Reference 2007): *MySQL 5.1 Reference Manual.* URL: 'http://dev.mysql.com/doc/', 2007-02-16 (revision: 4964), Abruf am 17.02.2007.

Oracle (Administration 2005): *Oracle Database: Administrator's Guide, 10g Release 2 (10.2).* URL: 'http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14231.pdf', Juni 2005, B14231-01, Abruf am 15.04.2007.

Oracle (Data Guard 2006): *Oracle Database Documentation Library, 10g Release 2 (10.2): Data Guard Concepts and Administration.* URL: http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14239.pdf, B14239-01, März 2006, Abruf am 17.04.2007.

Oracle (Dokumentation 2007): *Oracle Database Documentation Library 10g Release 2 (10.2).* Oracle Corporation, URL: http://www.oracle.com/pls/db102/homepage, Abruf am 12.02.2007.

Oracle (RAC-Dokumentation 2006): *Oracle Database: Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide 10g Release 2 (10.2).* URL: http://download-uk.oracle.com/docs/cd/B19306_01/rac.102/b14197.pdf, B14197-03, Januar 2006, Abruf am 04.04.2007.

Oracle (RAC-Installation 2006): *Oracle Database: Oracle Clusterware and Oracle Real Application Clusters Installation Guide, 10g Release 2 (10.2) for Linux.* URL: http://download-uk.oracle.com/docs/cd/B19306_01/install.102/b14203.pdf, B14203-08, September 2006, Abruf am 04.04.2007.

Oracle (Preiseliste 2006): *Oracle E-Business Global Price List.* Software Envestment Guide, 1.12.2006, URL: http://www.oracle.com/corporate/pricing/ePLext.PDF, Abruf am 01.11.2007.

Oracle (Metalink 2007): *Oracle Technologie Network.* Homepage: http://www.oracle.com/technology/support/metalink/index.html, Abruf am 01.11.2007.

Oracle (Network 2005): *Oracle Database: Net Services Administrator's Guide, 10g Release 2 (10.2).* URL: http://download-uk.oracle.com/docs/cd/B19306_01/network.102/b14212.pdf, B14212-02, Oktober 2005,

Abruf am 04.04.2007.

Oracle (OEM 2007): *Oracle Enterprise Manager: Concepts, 10g Release 3 (10.2.0.3).* URL: http://download-uk.oracle.com/docs/cd/B19306_01/em.102/b31949.pdf, B31949-01, January 2007, Abruf am 04.04.2007.

Oracle (RAC 10g R2): Oracle Corporation, Homepage: http://www.oracle.com/database/rac_home.html, Abruf am 16.02.2007.

Oracle (Reference 2005): *Oracle Database: Reference, 10g Release 2 (10.2).* URL: http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14237.pdf, B14237-02, November 2005, Abruf am 04.04.2007.

Oracle (Warehousing 2005): *Oracle Database: Data Warehousing Guide, 10g Release 2 (10.2).* URL: http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14223.pdf, B14223-02, December 2005, Abruf am 04.04.2007.

Rahm, E. (Transaktionssysteme 1993b): *Hochleistungs-Transaktionssysteme – Konzepte und Entwicklungen moderner Datenbankarchitekturen.* Vieweg-Verlag, 1993.

Ronstrom, M., Thalmann, L. (Architektur 2004): *Technical report: MySQL cluster architecture overview.* URL: http://www.mysql.de/why-mysql/white-papers/, 2004, Abruf am 13.03.2007.

Sokolinsky, L.B. (Architectures 2004): *Survey of Architectures of Parallel Database Systems.* Programming and Computer Software, Springer Netherlands, vol. 30, 6, November, 2004.

Vallath, M. (RAC 2004): *Oracle Real Application Clusters.* Library of Congress Cataloging-in-Publication Data, Elsevier Digital Press, printed in the United States of America, 2004.