DOAG

DOAG 2014 Datenbank

Nur ein Hype oder die Zukunft? Wird es ein miteinander geben?

ORACLE NOSQL - EINE ALTERNATIVE FÜR DIE TRADITIONELLE DATENBANK?

Grundlagen - Konzept

Was heißt hier ohne SQL?

Wieviel Struktur müssen meine Daten haben?

Für jedes Problem das richtige Werkzeug

- NoSQL
  - Mehr ein Denkansatz zur Datenverwaltung und Verarbeitung als eine spezielle Technologie
  - Grundkonzept
    - Verteilte ,skalierbare Lösungen (Scale-out Konzept)
    - **CAP** Theorem – **C**onsistency – **A**vailability - **P**artition Tolerance
    - **BASE - B**asically **A**vailible, **S**oft-state, **E**ventually
      versus
      ACID – Atomicity, **C**onsistency, **I**solation, **D**urability
  - Hochspezialisierte Lösungen für meist exakt ein großes Problem beim Handling von sehr großen, sehr gut verfügbaren oder sehr schnell benötigten Daten
    - Dutzende von Lösungsansätzen im Markt verfügbar

Brewer's CAP Theorem für verteilte Systeme

- Es ist nicht möglich alle Anforderungen zugleich zu erfüllen!

Die Oracle RDBMS Architektur erfüllt:

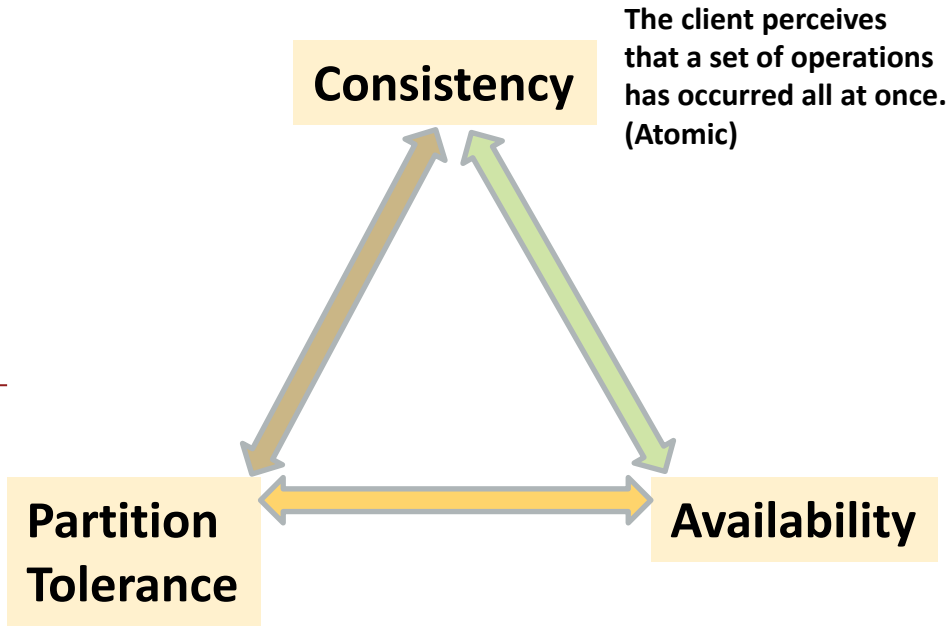
Consistency und **Availabitlity**

- Konsistenz und Verfügbarkeit sind garantiert

Die Oracle NoSQL DB Architektur kann :

Consistency und **Availabitlity**
Oder
Availabitlity und **Partition Tolerance**

- Ziel ist es ein verteiltes Systeme zu realisieren



Nur je zwei Eigenschaften können garantiert werden

See: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

NOSQL – Definition

- Eine reine NoSQL Datenbank ist:
  - Nicht relational ☑
  - Schema frei ☑
  - Meist eine distributed multi-node Architektur ☑
    - Ziel: eine Shared Nothing Cluster Lösung
  - Einfacher Replikations Mechanismus ☑
  - Eine einfache API (**Not only SQL = NoSQL**) ☑
  - Nicht unbedingt ein ACID Consistency Model ☑
  - Open Source ☑

Aber – der große Nachteil zur Zeit

=> Meist keine einfache Abfragesprache
=> Komplexe Joins nicht so einfach implementiert

Einordnung der Oracle NoSQL in die NoSQL Welt

- Key-Value
  - Ein Schlüssel zeigt einen auf einen Satz von Daten
    - Oracle ORACLE NOSQL DATABASE
- Document oriented
  - Alle Daten liegen in einem Dokument vor
    - Lotus Notes
- Graph
  - Vernetzte Strukturen
    - Neo4J

Merkmal		Beispiele
Dokumentenorientierte Datenbanken		Apache Jackrabbit, BaseX, CouchDB, eXist, Lotus Notes, MongoDB
Graphdatenbanken	Generisch	Neo4j, InfoGrid, HyperGraphDB, Core Data, DEX
	RDF-Zentriert	AllegroGraph, 4store, Virtuoso, andere
Verteilte ACID-Datenbanken		MySQL Cluster
Key-Value-Datenbanken	Festplattenspeicher	Chordless, Google BigTable, GT.M, InterSystems Caché
	Caches im RAM	Membase, memcached, Redis
	Eventually-consistente Speicher	Amazon Dynamo, Project Voldemort, Riak
	Sortierte Key-Value-Speicher	Berkeley DB, Memcachedb
Multivalue-Datenbanken		OpenQM, Rocket U2
Objektdatenbanken		Db4o, ZODB
Spaltenorientierte Datenbanken		Apache Cassandra, Google BigTable, Hbase, SimpleDB

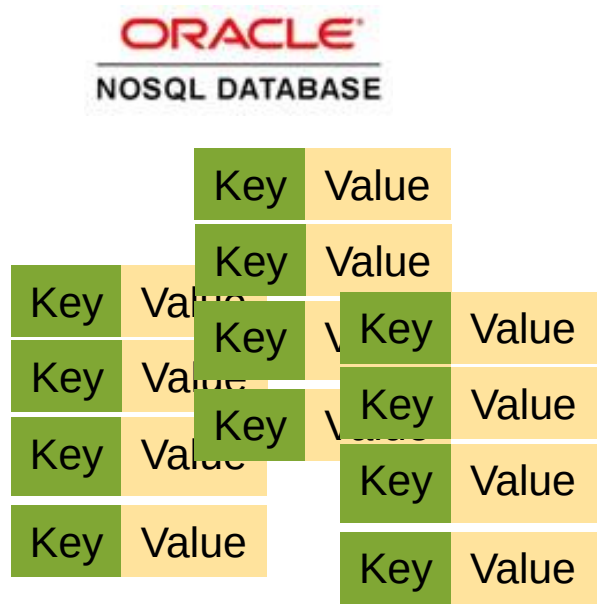
http://de.wikipedia.org/wiki/NoSQL

Idee Key-Value Store Database

- Oracle NoSQL – ein Key Value Store
  - Eine verteilte und skalierbare hoch verfügbare Datenbank
  - Consistent Hashing Ansatz

Client Software

Intelligenter Treiber
- Implementiert den Hash Mechanismus
- Kennt den Status des Stores
- Bei Ersten Connect wird die Partition Map gelesen

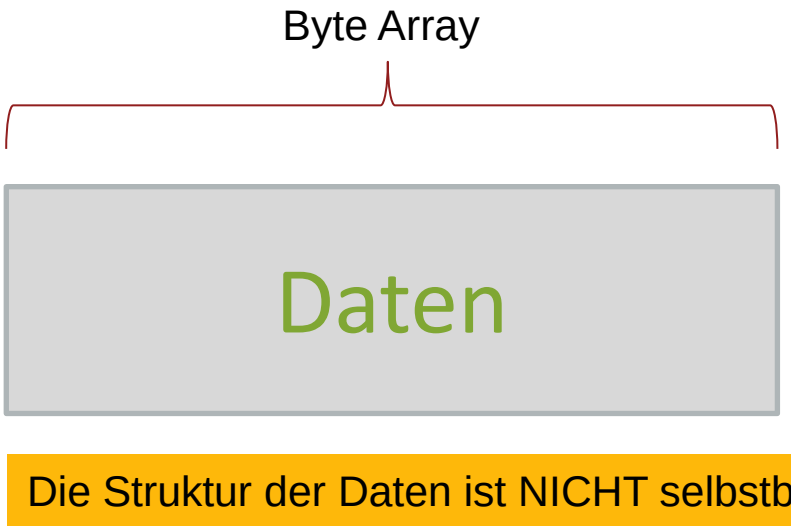


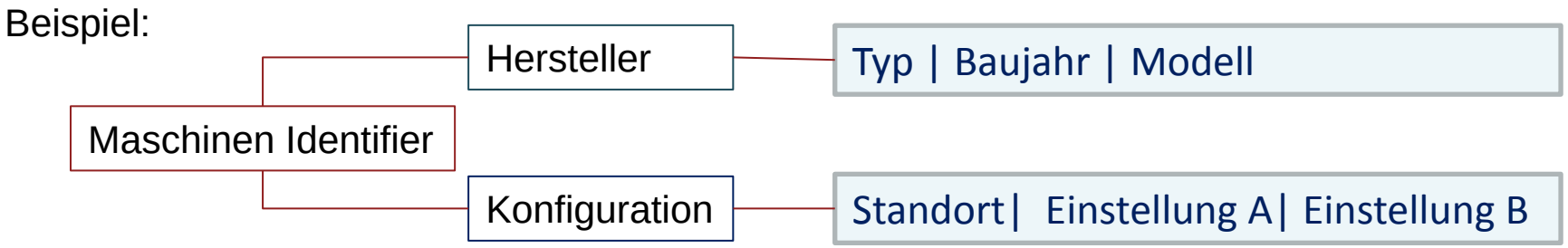
Datastore über verschiedene Nodes

Eine ACID Transaktion ist für alle Rekords unter einem Mayor Key als ein Single API Aufruf möglich.

Das Mayor –Minor Key Konzept

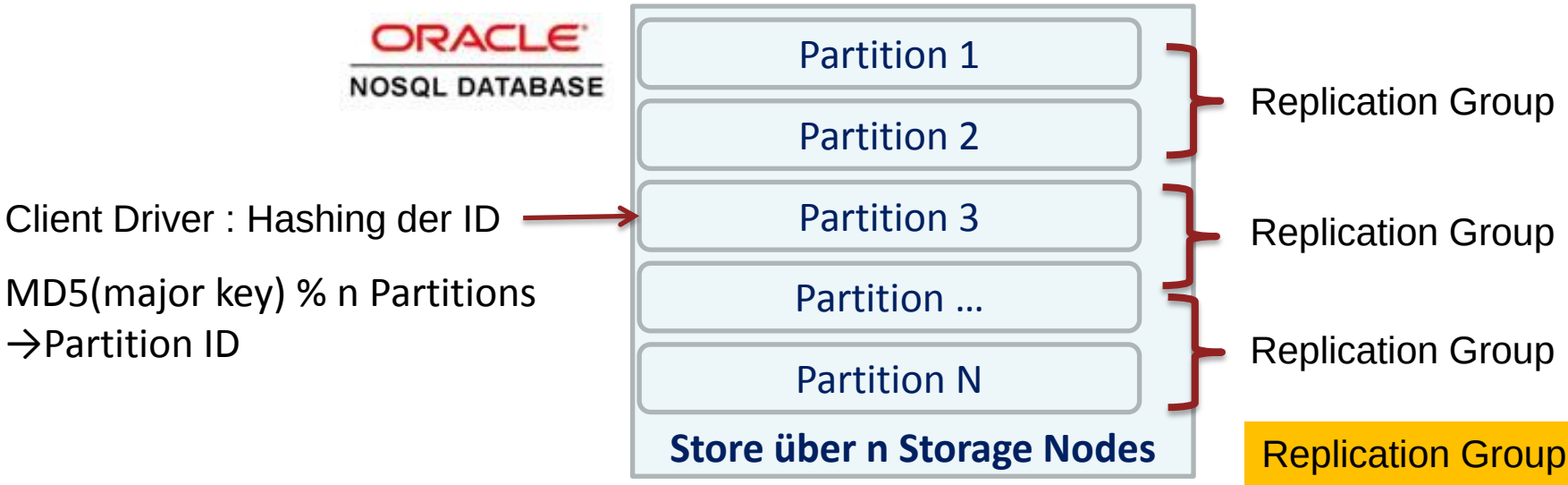
- Key – Zusammengesetzt aus zwei Komponenten





Implementierung

- Key = Hash Mayor Key => Partition Mapping
  - Beim Anlegen wird eine feste Anzahl von Partitionen über den Store angelegt
    - Jede Partition entspricht im Prinzip einer eigenen Berkeley DB „Datenbank“
    - Der Hash vom Key bestimmt die Partition
    - Die Partitionen werden über den Store redundant gehalten



Evolution von Key-Value über Avro zur Tabelle

- Release 1 – Key Value

  - Reines Key-Value Konzept
  - Kein „Data Dictionary“ – Gesamte Logik in der Applikation
- Release 2 – Avro

  - Schema Definition mit dem Apache Avro Standard
  - Eine Struktur Definition der Key-Value Daten wird der Applikation "bekannt" gegeben und in der DB hinterlegt
- Release 3 – Tabellen

  - Struktur Definitionen als Tabellen
    - Sekundärer Index auf Spalten der Tabelle möglich

Abfrage auf den Store

- Die Abfragen erfolgen über die Java API
  - Alternativ auch ein C API Verfügbar
  - Für Wartungsarbeiten steht eine Art SQL*plus zur Verfügung
- Der Key und NUR der Key kann abgefragt werden
  - Für die Suche in den Values muss der gesamte Datenbestand gelesen werden
  - Für einen Count wird über alle Keys einfach einer nach den anderen gezählt

Varianten / Versionen der Oracle NoSQL

- CC – Community Edition
  - Source Code liegt offen
    - Open Source mit einer GNU Affero General Public License
  - Support auf jährlicher Basis möglich
- EE – Enterprise Edition
  - Vollständiger Support
  - Oracle External Tables Unterstützung - Austausch von Daten mit „normaler“ Datenbank
  - Oracle Event Processing
  - RDF Adapter
  - SNMP Monitoring
  - Jena Adapter (Graph/SPARQL)
  - Ab Version 3 Security Optionen mit Oracle Wallet

Einsatz Szenarien

Für was kann das jetzt alles nun verwendet werden?

Einsatz Möglichkeiten

- Einsatzgebiete
  - Personalisierung von Webseiten
  - Authentifizierung
  - Stream Processing
    - Maschinendaten

Ziel:

Für EINEN Schlüssel Wert SCHNELL die Ergebnisse erhalten

Anforderung: Große Datenmengen mit geringer Latenz bereitstellen - Persistenter Cache

Praxis Beispiel: Web Applikation

- In einer Web Applikation gibt es pro Anwender eine sehr hohe Anzahl unterschiedlicher „Bonus Angebote, Gutscheine“, die individuell je nach Profile eines Anwender wöchentliche ermittelt werden
  - NoSQL Store hält über die UserID als Schlüssel eine Liste mit allen Metainformationen
    - Komplexe Joins über das DWH werden damit vermieden
    - Sofortige Anzeige auf dem SmartPhone oder im Web ohne Zeitverzögerung durch sehr niedrige Latenzen

Anforderung: Große Datenmengen mit geringer Latenz bereitstellen - Persistenter Cache

Welche Einsatz Strategie propagiert Oracle?

- Vorverarbeitung von Daten mit Oracle NoSQL und Hadoop - Klassische Auswertung mit der Oracle RDBMS Datenbank

ORACLE NOSQL DATABASE

hadoop

ORACLE DATABASE 12c

Ein Key-Value Store zum Sammeln unstrukturiert Daten?

Integration in das Hadoop Ökosystem

- Hadoop - Verteiltes Cluster File System
  - Container für verschiedene Datenbank Lösungen
    - Andere Datenbanken wie RainStore oder HIVE nützen Hadoop als Dateisystem

- Hadoop - MapAndReduce Framework
  - Framework für Abfragen

Integration in das Hadoop Ökosystem

Ergebnisse ablegen

Hadoop MapAndReduce

Daten aus dem NoSQL Store lesen

Pur Java

Hive

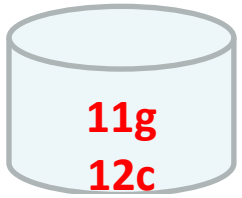
Hadoop Filesystem

ORACLE NOSQL DATABASE

Als External Table lesen

Als External Table lesen

Nur EE Edition der Oracle NoSQL!



ORACLE DATABASE 12c

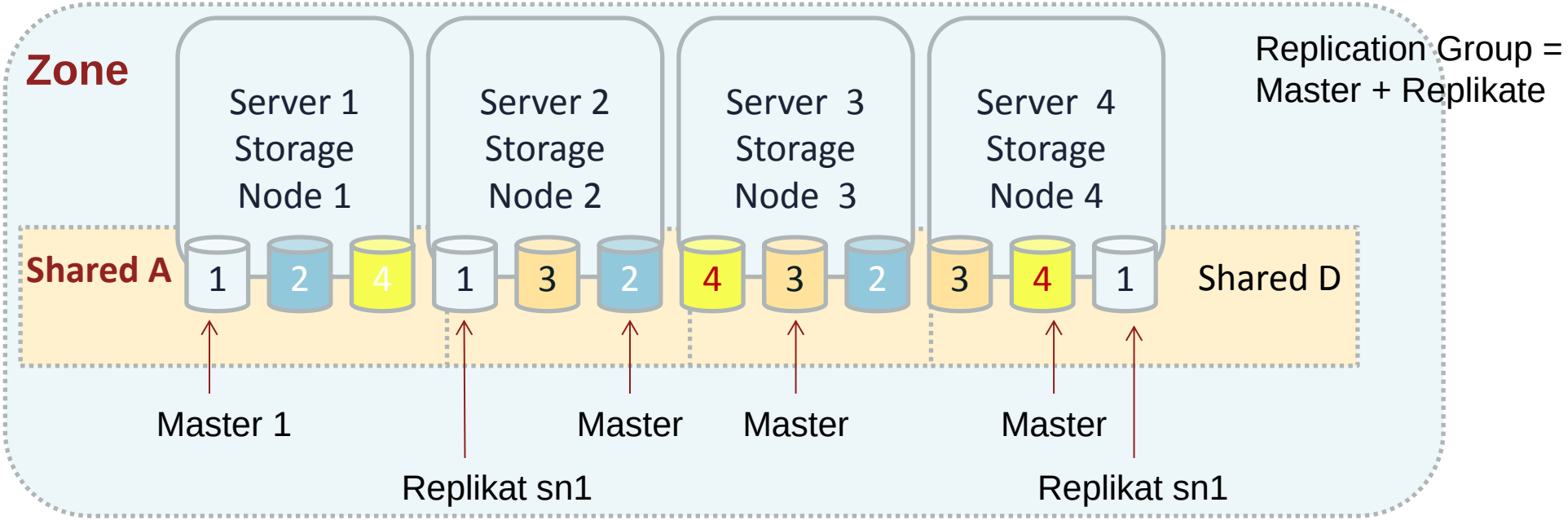
Die Architektur der Oracle NoSQL Lösung im Detail

Ein verteilter Key-Value Store

Der Store – Die Datenbank

- Store – der gesamte Speicherverbund über n Storage Nodes

  Beispiel für einen Store mit Replikation Faktor 3



Der **Replikationsfaktor** ist definiert als „Master + Anzahl der Replikate“.
In unsere Beispiel ein Master und zwei Replikate = Replikationsfaktor **3**
Zone = Eine Store node Umgebung – Kann auch repliziert werden

Übersicht über den Aufbau eines Stores

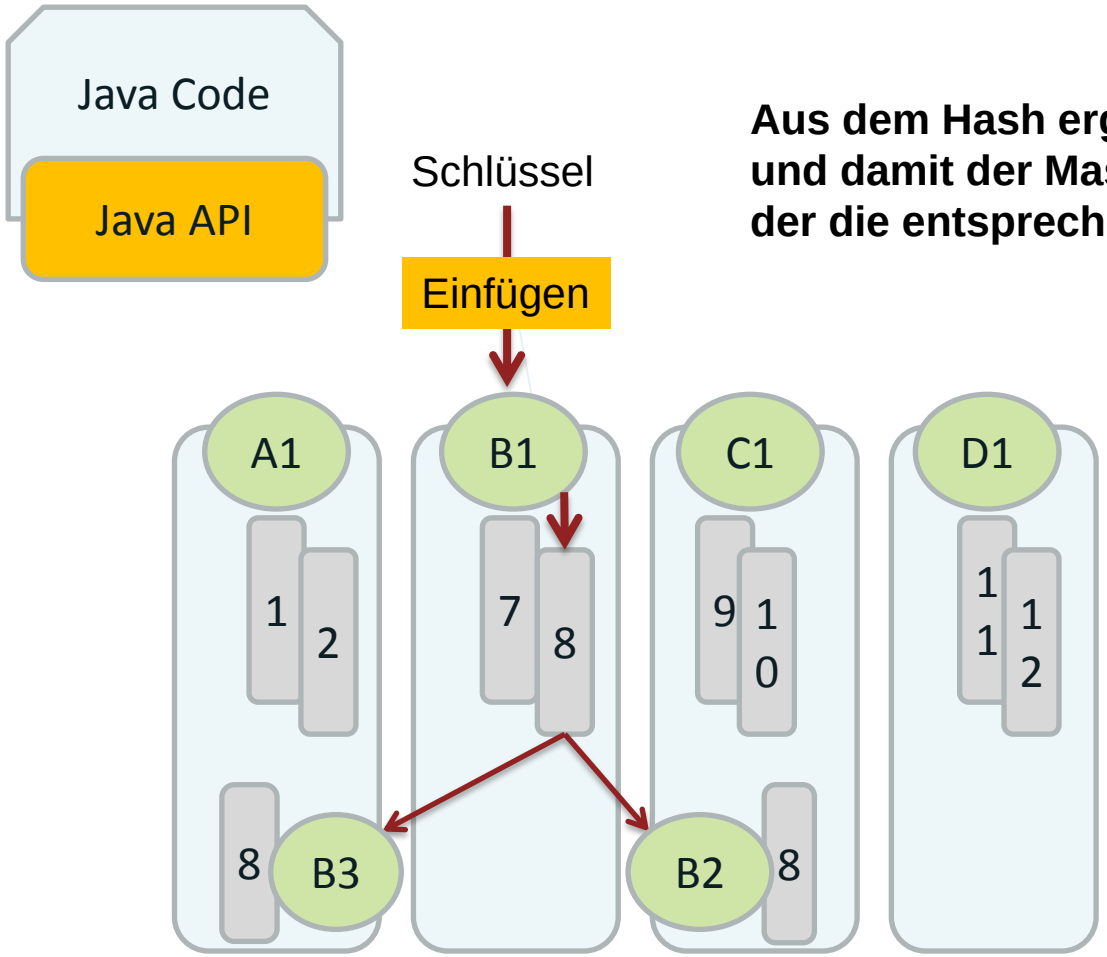
- Je nach eingestellten Replikationsfaktor werden die Daten über den Store mehrfach gespeichert
- Die Anzahl der eingestellten Partition wird über die Master Knoten verteilt
- Eine Topologie beschreibt den kompletten Store

- Beispiel – 4 Knoten - Replikationsfaktor 3 – 800 Partitionen
  - Ein Master pro Knoten + zwei Replikate pro Storage Node
  - Die 800 Partitionen werden über die 4 Master verteilt
    - 200 je Master
  - Die Daten werden über den Master Node eingefügt
    - Der Master Node verteilt die Daten an die Replikate
  - Die Daten können von jedem Knoten gelesen werden

Die Anzahl der Partition können nachträglich nicht mehr geändert werden!

Die Verteilung der Daten im Store

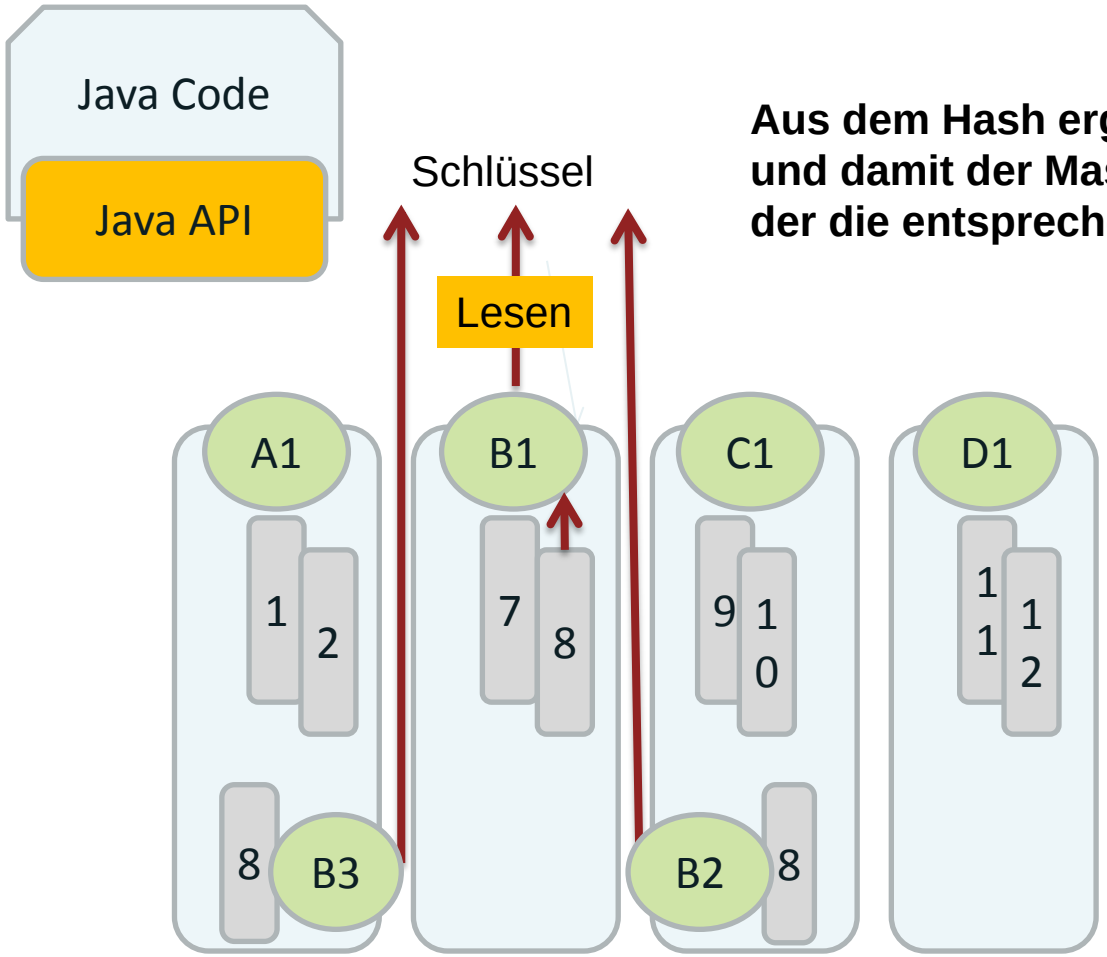
- Die Client API hashed jeden Schlüssel und verteilt die Schlüssel über die Anzahl der Partitionen



Aus dem Hash ergibt sich die Partition und damit der Master Knoten, der die entsprechende Partition hält.

Die Verteilung der Daten im Store

- Das Lesen kann von jeden Replikat erfolgen



Aus dem Hash ergibt sich die Partition und damit der Master Knoten, der die entsprechende Partition hält.

Konsistenz der Daten – Transaktionalität

Schreiben

- Innerhalb einer Session Transaktionalität beim Schreiben möglich
- Entwickler entscheidet wann ein Datensatz als geschrieben gilt

Durability

Sicher	Schnell
Auf allen Knoten auf Disk	Auf einem Konten im Cache

Performance

Lesen

- Entwickler entscheidet ob die gelesen Daten auch aktuell sein sollen

Consistency

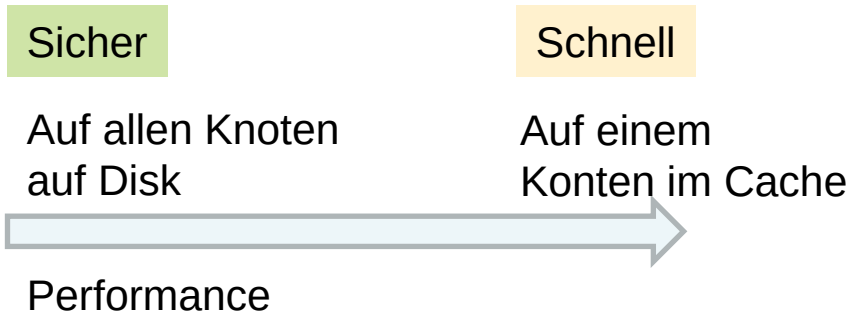
Sicher	Schnell
Aktuellste Version	Erstbeste Antwort

Performance

Die Schreib Konsistenz im Store - Durability

- Der Entwickler ist verantwortlich für die Konsistenz beim Schreiben!
- Wird beim Connect zum Store gesetzt
  - Klasse oracle.kv.Durability
    - Commit-Policy für den Master
    - Commit-Policy für die Replikate
    - Replica Acknowledgement-Policy

```
kvconfig.setDurability(
    new Durability(
        // master sync
        Durability.SyncPolicy.NO_SYNC
        // replicat sync
        ,Durability.SyncPolicy.NO_SYNC
        // replica acknowledge
        ,Durability.ReplicaAckPolicy.NONE
    )
)
```



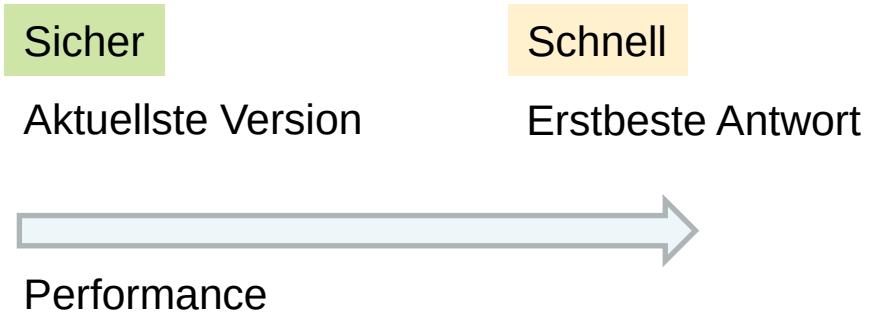
Durability Policies

- **Commit-Policy für Master und Replikate**
  - **SyncPolicy.SYNC**
    - warten bis der Storage Node die Transaktion in die Logdatei geschrieben und diese auf Platte synchronisiert hat
  - **SyncPolicy.WRITE_NO_SYNC**
    - warten bis der Storage Node die Transaktion in die Logdatei im Cache geschrieben hat (Nicht auf Platte geschrieben!)
  - **SyncPolicy.NO_SYNC**
    - Überhaupt nicht warten
- **Replica Acknowledgement**
  - **ReplicaAckPolicy.ALL**
    - Warten bis alle Replikate bestätigt haben, dass die Transaktion geschrieben wurde
  - **ReplicaAckPolicy.SIMPLE_MAJORITY**
    - warten bis die einfache Mehrheit der Replikate die Transaktion bestätigt hat
  - **ReplicaAckPolicy.NONE**
    - Nicht auf die Replikate warten

Die Lese Konsistenz im Store - Consistency

- Der Entwickler ist verantwortlich für die Lese Konsistenz!
- Jedem Key Objekt besitzt auch eine Versionsinformation
  - Eine Art Transaktion ID
- Verhalten wird beim Connect zum Store definiert
  - Lesen – Klasse oracle.kv.Consistency

```
kvconfig.setConsistency(Consistency.NONE_REQUIRED);
```



Consistency

- Consistency.ABSOLUTE
  - Es darf nur vom Master gelesen werden, damit ist sichergestellt, dass stets der aktuelle Inhalt gelesen wird
- Consistency.Time
  - Das Lesen von einer Replika ist erlaubt
  - Parameter legen fest, wie „veraltet der zurückgelieferte Inhalt des Keys sein kann
- Consistency.Version
  - Das Lesen von einer Replika ist erlaube
  - Versionsinformationen wird statt der Zeit verwendet, um zu erkennen ob der Datensatz veraltet ist
- Consistency.NONE_REQUIRED
  - jedes Ergebnis wird gelesen und darf beliebig veraltet sein

Interne Verwaltung

- Admin Service für die interne Verwaltung
  - Eigene Admin Datenbank
  - Eigener Port
- Der Admin Service sorgt für die Umsetzung des Master – Slave Konzepts
  - Bei einem Ausfall eines Knoten neuen Master bestimmen

Verfügbarkeit

- Was passiert beim Ausfall eines Storage Nodes?
  - Master evtl. nicht mehr verfügbar – es kann nichts mehr geschrieben werden?
  - Admin Prozess überwacht den gesamten Store
    - Besitzt eine eigene Admin Datenbank
    - Erkennt Fehler
    - Wählt aus den verbleibenden Replikaten den mit der höchsten LSN (log sequence number) zu einem neuen Master aus

Der Entwickler wählt aus:

- **Consistency** und **Availabitlity**

ODER

- **Availabitlity** und **Partition Tolerance**

Hat der Entwickler sich für **ReplicaAckPolicy.ALL** entschieden, kommt es aber zu einer Fehlermeldung falls nicht alle Node verfügbar sind!

Der Motor unter dem Ganzen (1)

ORACLE
BERKELEY DB

- Die Java Version der Berkeley DB
  - JE Version 5.0.83 für die NoSQL 2.2.18
  - JE Version 6.0.8 für die NoSQL 3.0.5
  - Jede Partition ist eine Datenbank
  - Berkeley DB Replikation für das Master Slave Konzept
  - Die DB Dateien lassen sich sogar mit dem Berkeley DB Klassen analysieren

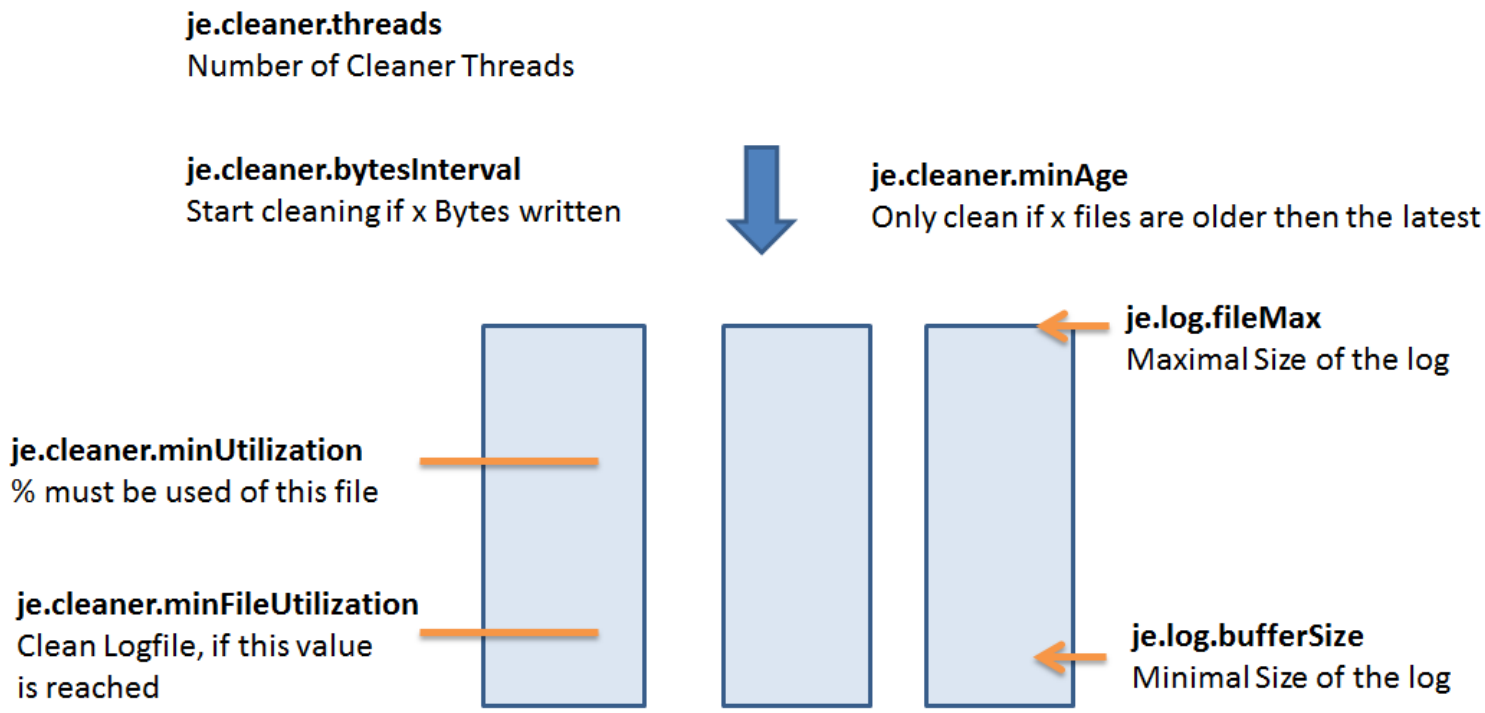
Version anzeigen lassen mit
java -classpath ".\lib\kvclient.jar;.\lib\kvstore.jar;.\lib\je.jar" com.sleepycat.je.util.DbVerify -**V**

Tipp:
Die original Berkeley DB Mechanismen können pro Storage Node konfiguriert werden
Siehe http://www.pipperr.de/dokuwiki/doku.php?id=nosql:log_file_verhalten_oracle_nosql_db_11gr2

Der Motor unter dem Ganzen (2)

ORACLE
BERKELEY DB

- Besonderheit der Berkeley DB:
  - Redo Log Einträge und Daten werden in den selben „Daten“ Dateien verwaltet
  - Hintergrund Jobs sorgen für das Cleaning der unnötigen Einträge



Beispiel für das Auslesen der Datendateien

- Mit dem Berkeley Klassen die DB Größe auslesen
  - Auf das richtige Einbinden der jar Files achten
    - export KVLIB=/opt/oracle/produkt/11.2.0/kv-2.1.8/lib
    - export KVCLASS=$KVLIB/kvclient.jar:$KVLIB/kvstore.jar:$KVLIB/je.jar
  - Pfad zu den Datendateien angeben
    - export JEENV=/opt/oracle/kvdata/GPIDB/sn1/rg1-rn1/env/
  - Auswerten mit:

```
java -classpath $KVCLASS com.sleepycat.je.util.DbSpace -h  $JEENV


File     Size (KB)   % Used
--------  ---------  ------
00000000     110808      21

 TOTALS      110808      21
(LN size correction factor: NaN)
```

Installation

Java entpacken, Verzeichnisse anlegen

fertig?

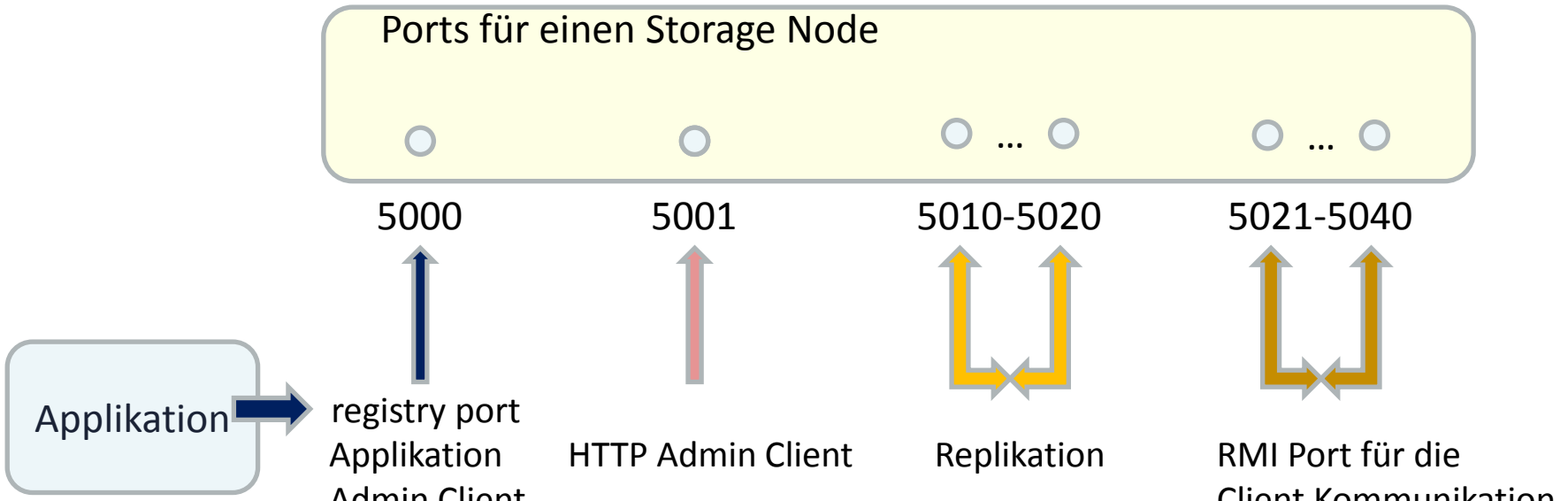
Planung einer NoSQL Installation - Netzwerk

- Netzwerk Infrastruktur
  - Physik
    - Hoch performante interne Kommunikation zwischen den Knoten notwendig
      - Auf geringe Latenzen achten
        - » Stichwort Infiband
        - » Eigenes 10Gbit Netz einplanen?
  - TCP Ports definieren
    - Für die Kommunikation unter den Storage Nodes intern
    - Für Kommunikation des Clients mit dem Master Nodes
      - Ein Port von Außen notwendig + Ein Port für die Admin Console + ServicePortRange für die RMI Verbindung
  - Wartbarkeit
    - SSH zwischen den Storage Nodes ermöglichen
      - Skript gesteuerte Wartung über mehrere Knoten ermöglichen

Durch die doch mit der Zeit hohe Anzahl von Server für Produktion / Staging / Test / Dev ist es ratsam vorab Standard zu definieren!

Planung einer NoSQL Installation - Ports

- Notwendige Ports - Netzwerk Infrastruktur



Parameter beim Anlegen mit "makebootconfig"

-port -admin -harange -servicerange

Parameter für das Ändern mit "change-policy -params "

haPortRange servicePortRange

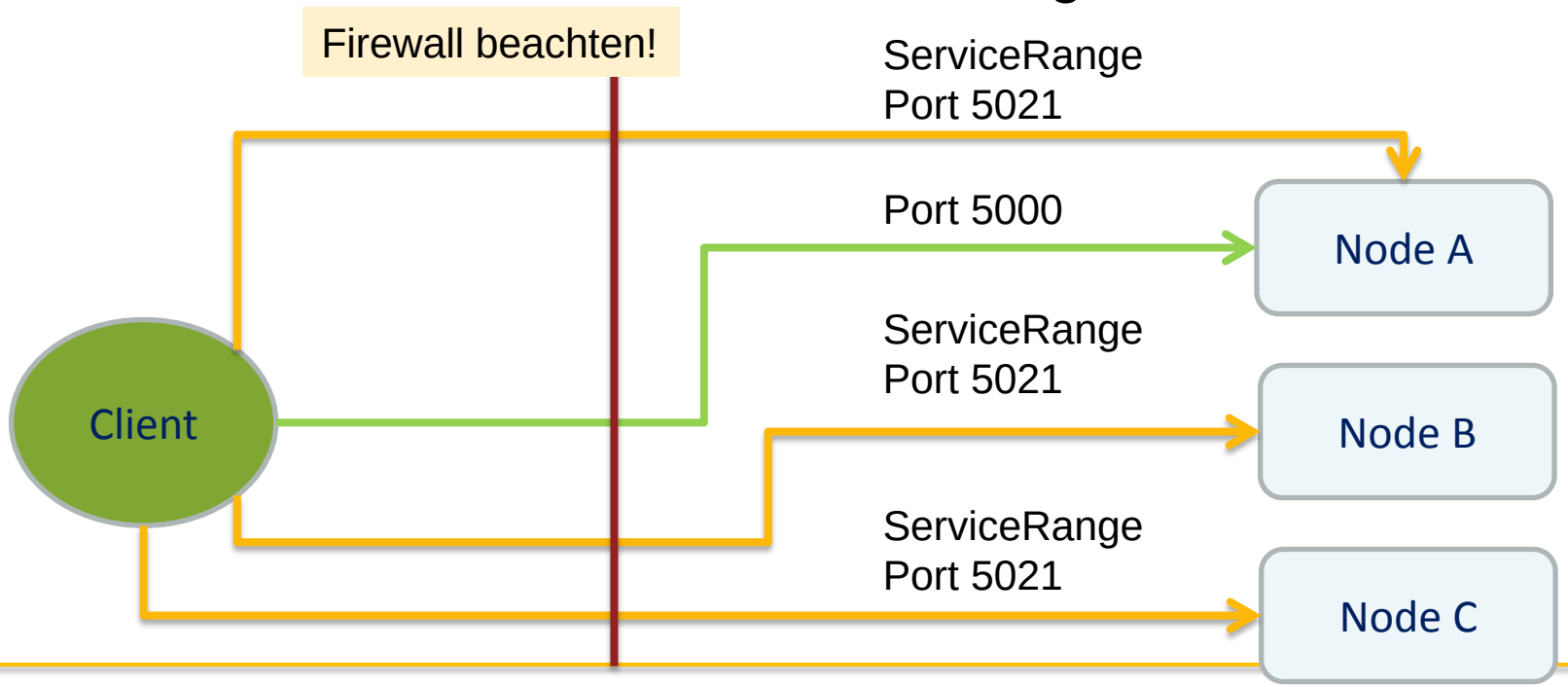
Oracle NoSQL Client Connect

- Zugriff vom Client auf den „registry port“
  - Zum Beispiel Port 5000 auf Knoten A

-port

- Abfragen werden über das Java RMI Protokoll auf den jeweiligen Node durchgeführt
  - Ausreichend Remote Ports notwendig

-servicerange



Planung einer NoSQL Installation - Hardware

- CPU und Speicher
  - Cache Größe kann von 2 bis n GB eingestellt werden
    - GC Effekte von Java im Auge behalten
      - Oracle NoSQL ist explicit auf sauber, kleine Objekte Größen optimiert um den Java Garbage Collecter nicht zu überfordern
- Storage Infrastruktur
  - Auf einem Store Node können auch mehrere Master und Replikate betrieben werden
    - Auf je eigene und performante Physik der einzelnen Datenbereiche (KVROOT) achten
      - Neben dem reinen Einfügen und Lesen von Nutzdaten werden die DB Dateien von „Cleaner“ Prozessen bei Bedarf bereinigt

Planung einer NoSQL Installation - Speicherplatz

- Speicherplatz
  - Berechnung nicht ganz trivial
    - Die Berkeley DB speichert ALLE Aktionen, Daten wie Redo Information in der selben Datei
    - Werden bestimmte Thresholds erreicht, werden diese Datendateien optimiert
    - Löschen von Daten vergrößert den Store im ersten Schritt!

Faustregel:
Pro Master/ Replikat => Anzahl Transaktionen (Insert/Update/Delete) * (Netto Daten Größe + Overhead)

=> Füllgrad der Datendateien wird per Default auf 40% optimiert

Installation

- Vorbereitung auf jeden Knoten
  - Java JDK installieren und Store User anlegen
  - SSH Verbindung einrichten
  - KVROOT anlegen / einbinden (Die Datenablage!)
- Oracle NoSQL installieren
  - Code auspacken – KVHOME anlegen
  - Store Node Basis Verzeichnis (KVROOT) Boot Konfiguration anlegen
  - Admin Node starten
  - Plan für den neuen Store definieren
    - Die Datenbank Topologie definieren
  - Plan ausrollen
  - Testen

Per Script: Aufwand : 5min

Mit dem NoSQL Store arbeiten

Wartung und Betrieb

Verwaltung (1) – Kommando Zeile

- Kv - Für die Store Administration

```
-- -- Command java -jar /opt/oracle/produkt/11.2.0/kv-2.1.57/lib/kvstore.jar runadmin -port 5000 -host nosqldb01
Redirecting to master at rmi://nosqldb03:5000
kv-> ping
Pinging components of store GPIDB based upon topology sequence #516
Time: 2014-06-02 15:12:14 UTC
GPIDB comprises 500 partitions and 3 Storage Nodes
Storage Node [sn1] on nosqldb01:5000    Datacenter: DCGPIDB [1]    Status: RUNNING   Ver: 12cR1.2.1.57 2014-01-10 06:21:46 UTC  Build id: 4d323d1b5495
        Rep Node [rg1-rn1]      Status: RUNNING,REPLICA at sequence number: 1,937,521 haPort: 5011
        Rep Node [rg2-rn1]      Status: RUNNING,MASTER at sequence number: 1,949,809 haPort: 5012
        Rep Node [rg3-rn1]      Status: RUNNING,REPLICA at sequence number: 1,934,227 haPort: 5013
Storage Node [sn2] on nosqldb02:5000    Datacenter: DCGPIDB [1]    Status: RUNNING   Ver: 12cR1.2.1.57 2014-01-10 06:21:46 UTC  Build id: 4d323d1b5495
        Rep Node [rg1-rn2]      Status: RUNNING,REPLICA at sequence number: 1,937,521 haPort: 5011
        Rep Node [rg2-rn2]      Status: RUNNING,REPLICA at sequence number: 1,949,809 haPort: 5012
        Rep Node [rg3-rn2]      Status: RUNNING,MASTER at sequence number: 1,934,227 haPort: 5013
Storage Node [sn3] on nosqldb03:5000    Datacenter: DCGPIDB [1]    Status: RUNNING   Ver: 12cR1.2.1.57 2014-01-10 06:21:46 UTC  Build id: 4d323d1b5495
        Rep Node [rg1-rn3]      Status: RUNNING,MASTER at sequence number: 1,937,521 haPort: 5011
        Rep Node [rg2-rn3]      Status: RUNNING,REPLICA at sequence number: 1,949,809 haPort: 5012
        Rep Node [rg3-rn3]      Status: RUNNING,REPLICA at sequence number: 1,934,227 haPort: 5013
```

- kvshell - Abfragen/Einfügen von Daten

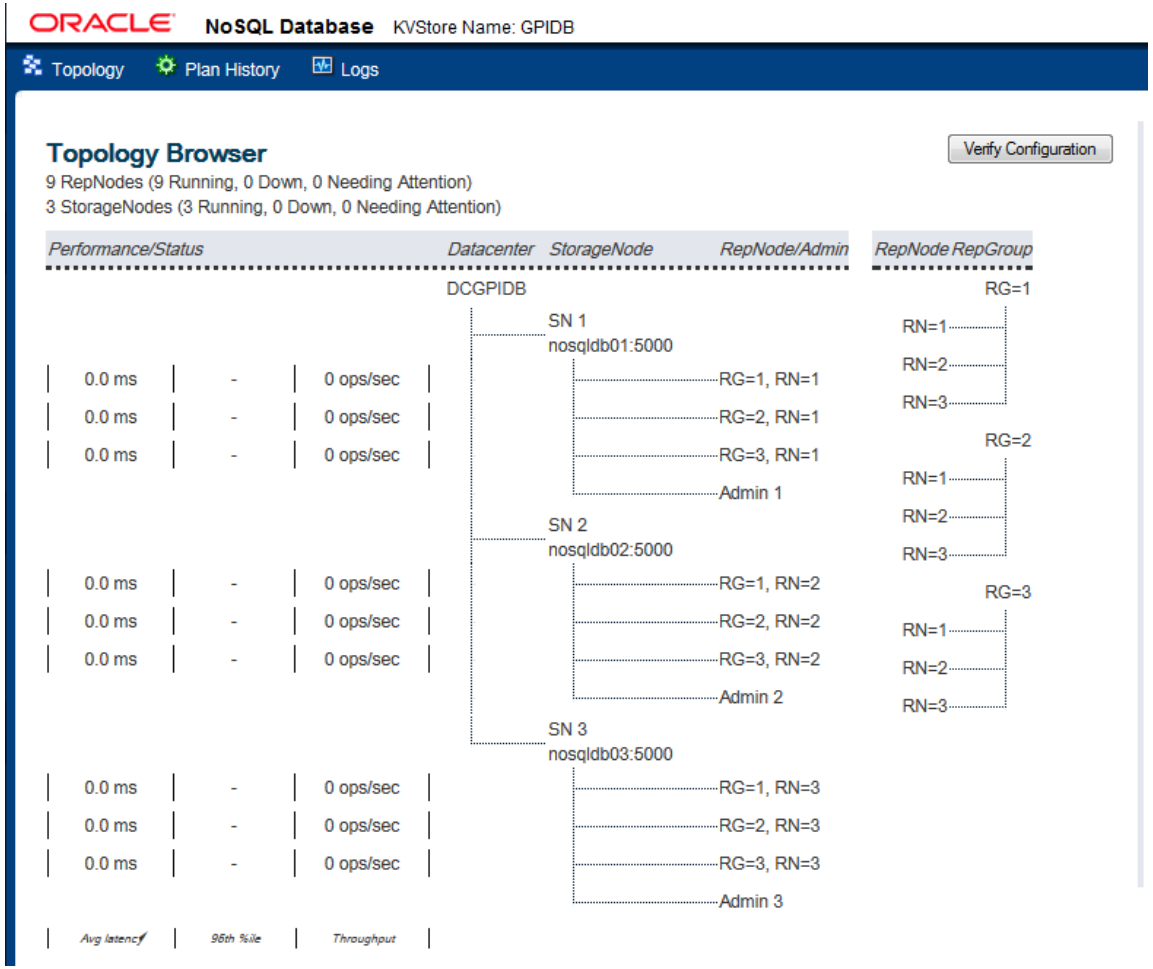
```
-- -- Command java -jar /opt/oracle/produkt/11.2.0/kv-2.1.57/lib/kvcli.jar -host nosqldb01 -port 5000 -store GPIDB
kvshell-> aggregate -count
count: 1010204
```

```
kvshell-> put -key "/Gunther/name" -value "Pipperr"
Operation successful, record inserted.

kvshell-> get -key "/Gunther/name"
Pipperr
```

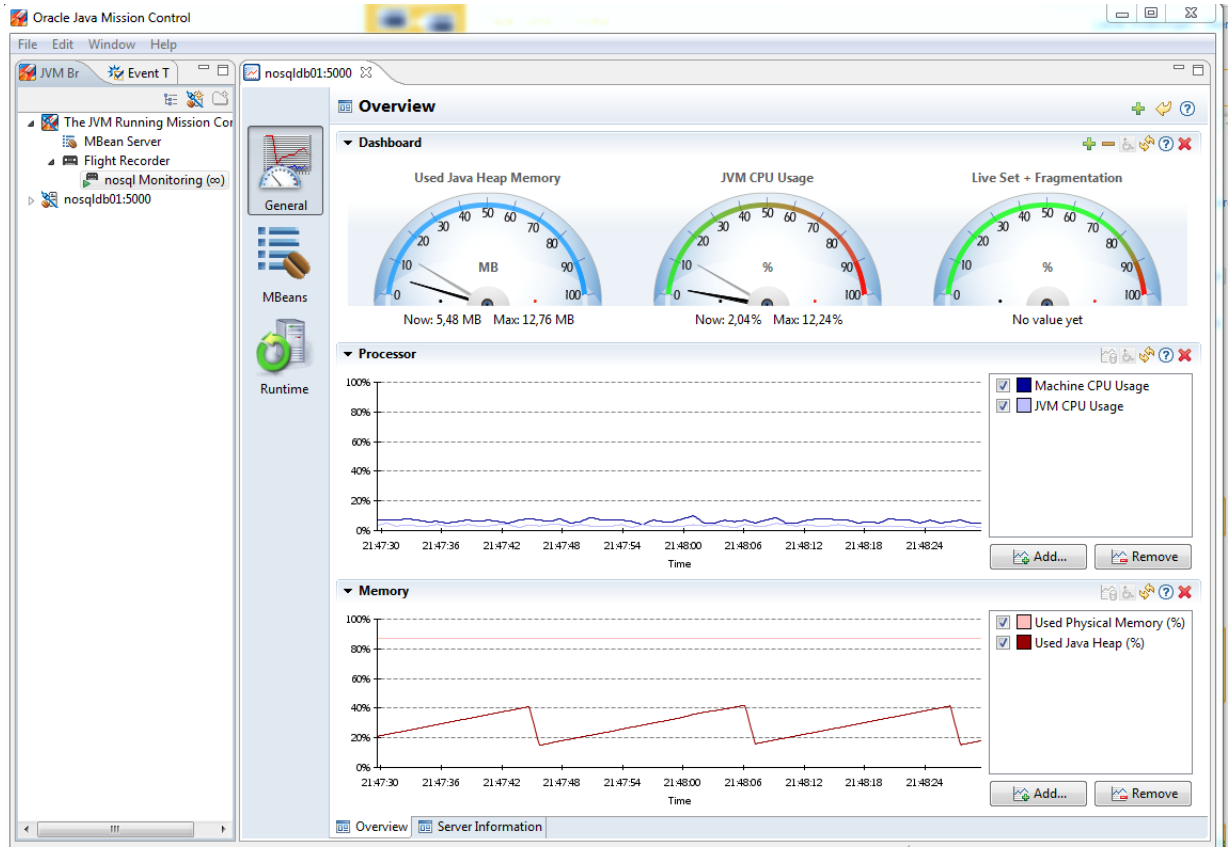
Verwaltung (2) – Webinterface

– Weboberfläche wird durch einen mit gelieferten jetty Server zur Verfügung gestellt: http://nosqldb03:5005/



JMX Oberfläche

- JMC starten – Java Mission Control
  - JAVA_HOME/bin/jmc.exe
  - Über den „registry port“ anmelden



Ausfallsicherheit

Verfügbarkeit und Backup

Backup per Snapshot möglich

- Backup:
  - Ein kompletter Store kann per Snapshot konsistent gesichert werden
    - Ausreichend Speicherplatz einplanen
- Restore
  - Ein Store kann komplett zurück gesetzt werden
    - Store Snapshot erzeugen
    - Store stoppen
    - Snapshot zur Verfügung stellen (Links im Filesystem z.B.)
    - Store starten, Storage Nodes lesen die Daten, legen den die Daten wieder an und LÖSCHEN den Snapshot!
  - Eine Art Import
    - Die erzeugten Daten können in einen andern Store wieder eingelesen werden

Performance

I/O und Netzwerk

Performance Überlegungen

- Lesen
  - Netzwerk
    - alle Datensätze müssen über das Netz gelesen werden!
- Schreiben
  - I/O
    - Hintergrund Prozesse für die Datendatei Optimierung benötigen genügende Reserven

Ein Einfaches Count(*) MUSS alle Daten lesen!

ALLE Daten werden auf dem Client verarbeitet!

Sicherheit?

Oracle NoSQL und Daten Sicherheit?

Sicherheit?

- In Version 2 ?
  - Darum kümmert sich der Netzwerk Admin und der Entwickler

- Ab Version 3
  - SSL Verschlüsselung
  - Password Schutz
    - Wallet Verwendung EE Feature

In Java den Store ansprechen

Abfragen werden mit Java erstellen

Verbindung zum Store in Java aufbauen

- Connection zu Store herstellen
  - Store Eigenschaften setzen

```java
// Node liste
String[] kvhosts = {"nosqldb01:5000","nosqldb02:5000","nosqldb03:5000"}

// Config erzeugen
KVStoreConfig kvconfig = new KVStoreConfig("GPIDB", kvhosts);

// Consistency definieren
kvconfig.setConsistency(Consistency.NONE_REQUIRED);

// Durabiltiy definieren
kvconfig.setDurability(Durability.COMMIT_NO_SYNC);

// Store Zugriff öffnen
KVStore kstore = KVStoreFactory.getStore(kvconfig);


// Mit dem Store arbeiten
// ----------
 ....
// ----------


// Store wieder schließen
kstore.close();
```

Mit Java Daten einfügen und auslesen

- Entwickler spricht die DB Objekte direkt über den Key an
- Daten anlegen

```
// Key mit Minor und Mayer  Komponente anlegen
Key k = Key.createKey("MAIN_KEY", "SLAVE_KEY");
// Wert als Byte Array anlegen
byte[] b = ("Wert").getBytes();
// Wert in den Store schreiben
kvstore.put(k, Value.createValue(b));
```

Schreiben

- Daten abfragen

```
// Key erzeugen
Key datakey = Key.createKey("ABCDEFG");

// Daten aus dem Store mit den Key wieder lesen
ValueVersion vv = kvstore.get(datakey);
// Daten auslesen
Value vdata = vv.getValue();

// Nutzdaten wieder herstellen
String data = new String(vdata.getValue());
// Daten ausgeben
System.out.println(data);
```

Lesen

Es steht keine SQL Syntax zur Verfügung

Fazit

Lohnt sich der Aufwand?

Warum die Oracle NoSQL wählen? (1)

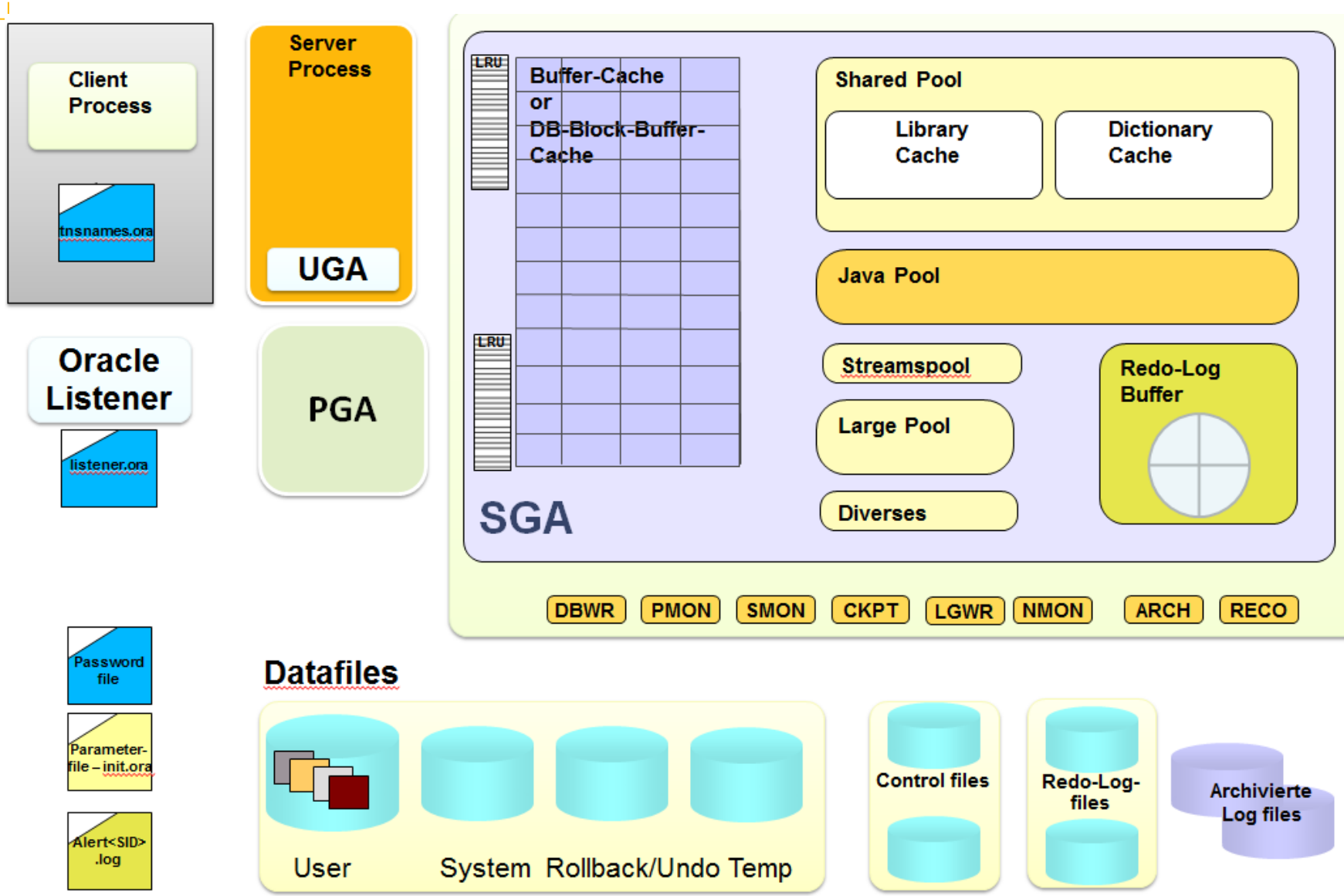
- Was ist für eine strategische Entwicklung wirklich wichtig?
  - Wartbarkeit
    - Ist das Produkt bedienbar / administrierbar?
  - Langfristige Strategie des Herstellers
    - Kann ein Produktzyklus von 4-5 Jahren eingehalten werden?
  - Release und Update Problematik
    - Läuft die Software auch Java 8 und höher? Was ist mit Linux 7?

Warum die Oracle NoSQL wählen? (2)

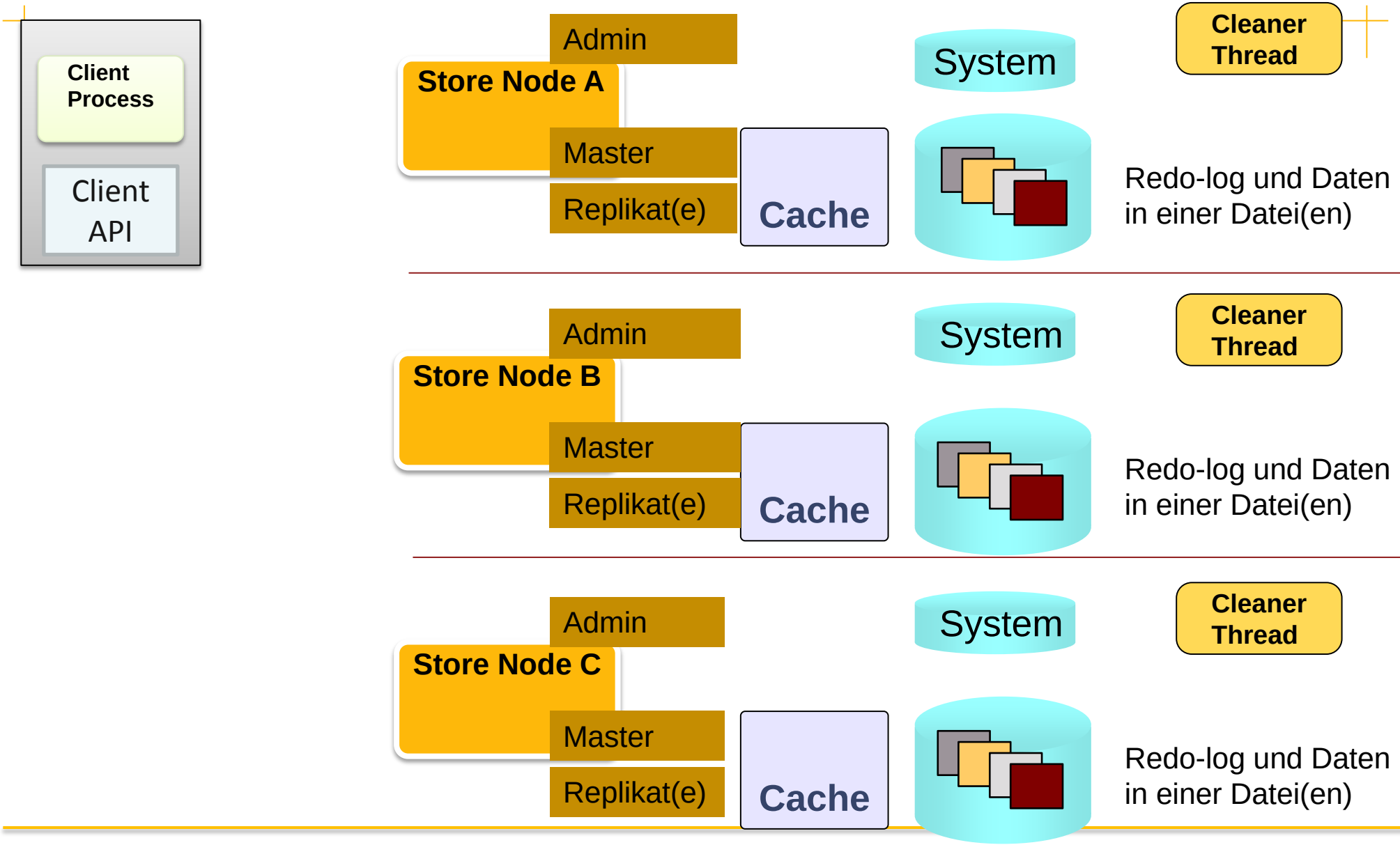
- Was ist für eine strategische Entwicklung wirklich wichtig?
  - Passt das Grundkonzept zu meiner technischen Anforderung?
    - Ändern sich meine Anforderungen mit der Zeit?
  - Kosten

Oracle verspricht dies einzuhalten
– Version 3 in Planung
– Basis Software Berkeley DB seit Jahren stabil im Einsatz

Die Architektur im direkten Vergleich (1)



Die Architektur im direkten Vergleich (2)



Der direkte Vergleich Oracle RDBMS ⇔NoSQL

Oracle RDBMS	Oracle NoSQL
▪ ~35 Jahre Entwicklung	▪ ~2-3 Jahre
▪ Mächtige SQL API	▪ Einfache Java API
▪ Joins	▪ Join nur über die Applikation
▪ Constraints	▪ ?
▪ Viele Features	▪ Reine Datenhaltung
▪ OLTP Einsatz	▪ Lese Operationen überwiegen
▪ Shared Anything Ansatz	▪ Shared Nothing Cluster
▪ Generalisierte All zweck Waffe	▪ Spezialisierte Nischen Lösung

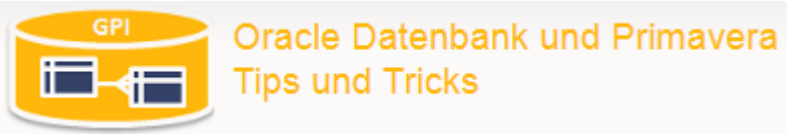
NO:SQL

Oracle NoSQL

Quellen

- Oracle - ORACLE DOJO NR 2
  - Siehe http://www.oracle.com/webfolder/technetwork/de/community/dojo/index.html

- Mehr über das Thema siehe auch:



  - http://www.pipperr.de/dokuwiki/doku.php?id=nosql_datenbank

Gunther Pippèrr - IT-Architekt - Berater

Background

Gunther Pippèrr arbeitet seit mehr als 15 Jahre intensiv mit den Produkten der Firma Oracle im Bereich Datenbanken/Applikationsserver und Dokumenten-Management.

Herr Pippèrr hat sich tiefes Wissen über den Aufbau komplexer IT Architektur aneignen können und hat dieses in der Praxis erfolgreich umgesetzt.

Herr Pippèrr hat eine Abschluss als Dipl. Ing. Technische Informatik (FH) an der FH Weingarten.

Functional Expertise

- IT System Architekt
- Technische Projektleitung
- Design und Implementierung von Datenbank Anwendungen
- Entwurf und Umsetzung von IT Infrastrukturen zum Datenmanagement

Industry Expertise

- High-Tech
- Real Estate
- Utility
- Communications

Selected Experience

- Datenbank Architekt für ein Projekt zur Massendatenverarbeitung in der Telekommunikation
- Architekt und technische Projektverantwortung für ein Smart Metering Portal für das Erfassen von Energiezählerdaten und Asset Management
- Architekt und Projektleitung , Datenbank Design und Umsetzung für die Auftragsverwaltung mit Steuerung von externen Mitarbeitern für den Sprachdienstleister von deutschen Technologiekonzern
- Architekt und technische Projektverantwortung für IT Infrastrukturprojekte, z.B.:
  - Zentrale Datenhaltung für Münchner Hotelgruppe mit über 25 Hotels weltweit,
  - Redundante Cluster Datenbank Infrastrukturen für diverse größere Web Anwendungen wie Fondplattform und Versicherungsportale
- CRM- und Ausschreibungsportal für großen Münchner Bauträger