DOAG

Deutsche ORACLE-Anwendergruppe e.V.

Konferenz – Nürnberg 2014

NoSQL Vortrag – Taiwan/Taipei 2014

Zusammen ein Team?

ORACLE UND HADOOP

Warum nun Big DATA

Was treibt uns an?

Neue Lösungen für alte Probleme?

Datenqualität und -Stabilität

Herausforderung

- Wenig feste Strukturen
- Hohe Fehlerwahrscheinlichkeit

- Daten ändern über die Jahre semantisch und syntaktisch ihre Bedeutung

- Nachvollziehbarkeit der Ergebnisse

Datenwachstum und Performance

Herausforderung

- Geringer Nutzinhalt
- Hohes Aufkommen

- Platzbedarf

- Performance

Warum Hadoop?

Ein neuer Standard oder nur ein neuer Hype?

File System und Batch Prozessing

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

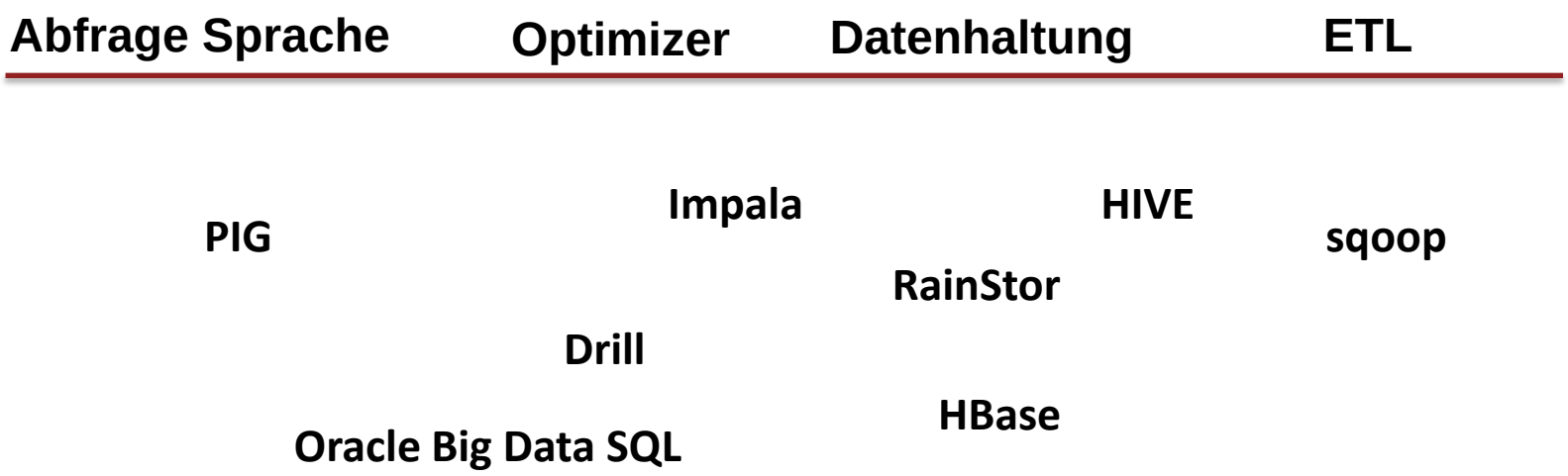
Alternative: Ceph Clusterfiles System

- Hadoop - MapReduce Framework
  - Framework für das MapReduce Pattern über vielen Knoten

Alternativen MapReduce Abfragen:
- Parallel SQL Engines wie Impala
- Oracle Big Data SQL

Qual der Wahl – Die Software um Hadoop

- Hadoop wird immer mehr ein Synonym für eine verteilte Datenhaltung
- Im Hadoop Ökosystem wird jede Komponente eines „großen“ RDBMS Systems über ein eigenes Produkt umgesetzt



Hadoop integrieren

Für was kann das jetzt alles nur verwendet werden?

Einsatzgebiet – Ergänzung zum Bestands DWH

- Einsatzgebiete
  - Sammeln von Log Informationen aus dem Betrieb
    - Wie Listener und Audit Logs der Datenbanken
  - Stream Processing von Maschinendaten
  - Archivieren von Historischen Daten
    - Wie Partitionen aus DWH Tabellen

Ziele:

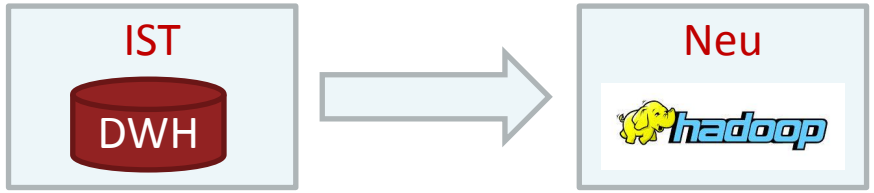
- Nur die zur Zeit wichtige Daten im Live System vorhalten
- Keine Daten verlieren, die noch wichtig werden könnten

Anforderung: Große Datenmengen aufbewahren und nur bei Bedarf SELTEN analysieren

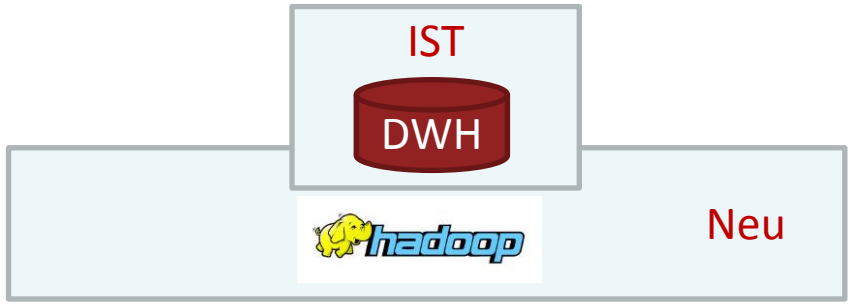
Die Herausforderung in der Praxis

- Integration in die bestehende IT Landschaft
  - Alle Kernprozesse sind bereits im DWH effizient abgebildet
  - Berichtwesen / Reporting bereits aufgesetzt

Migration



Integration



Aufwand:

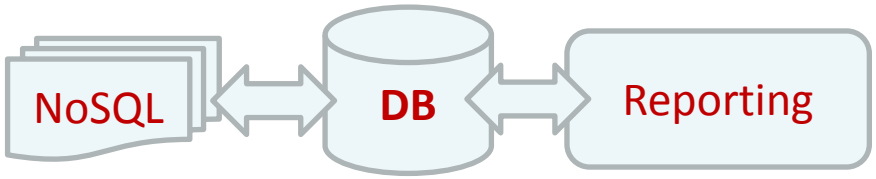
- **Implementierung**
- **Migration von Alt Daten**
- **Schulung von Mitarbeitern**

Motivation:

- Lizenz Kosten erzwingen bereits vor dem Erreichen technischer „Problemgrößen“ neue Lösungen
- Storage Kosten steigen überproportional bei Anforderungen an Performance / Verfügbarkeit / Volumen

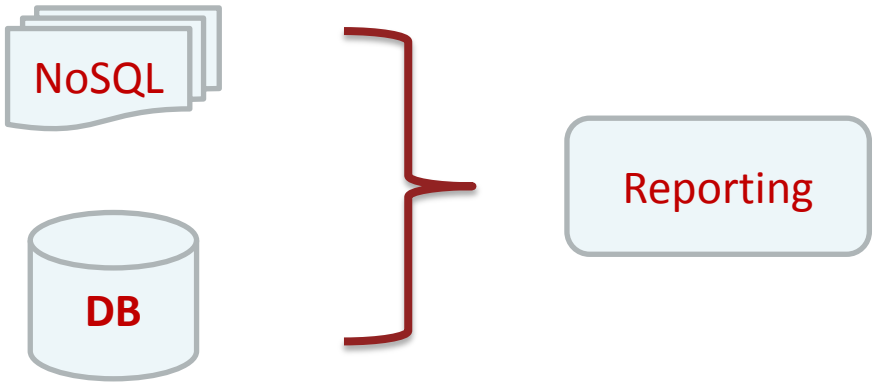
Wo setzt die Integration an?

- Zugriff auf die Daten über die Datenbank



Bestehende BI/Reporting Umgebung
Bestehende Zugriffsberechtigung
und Userverwaltung

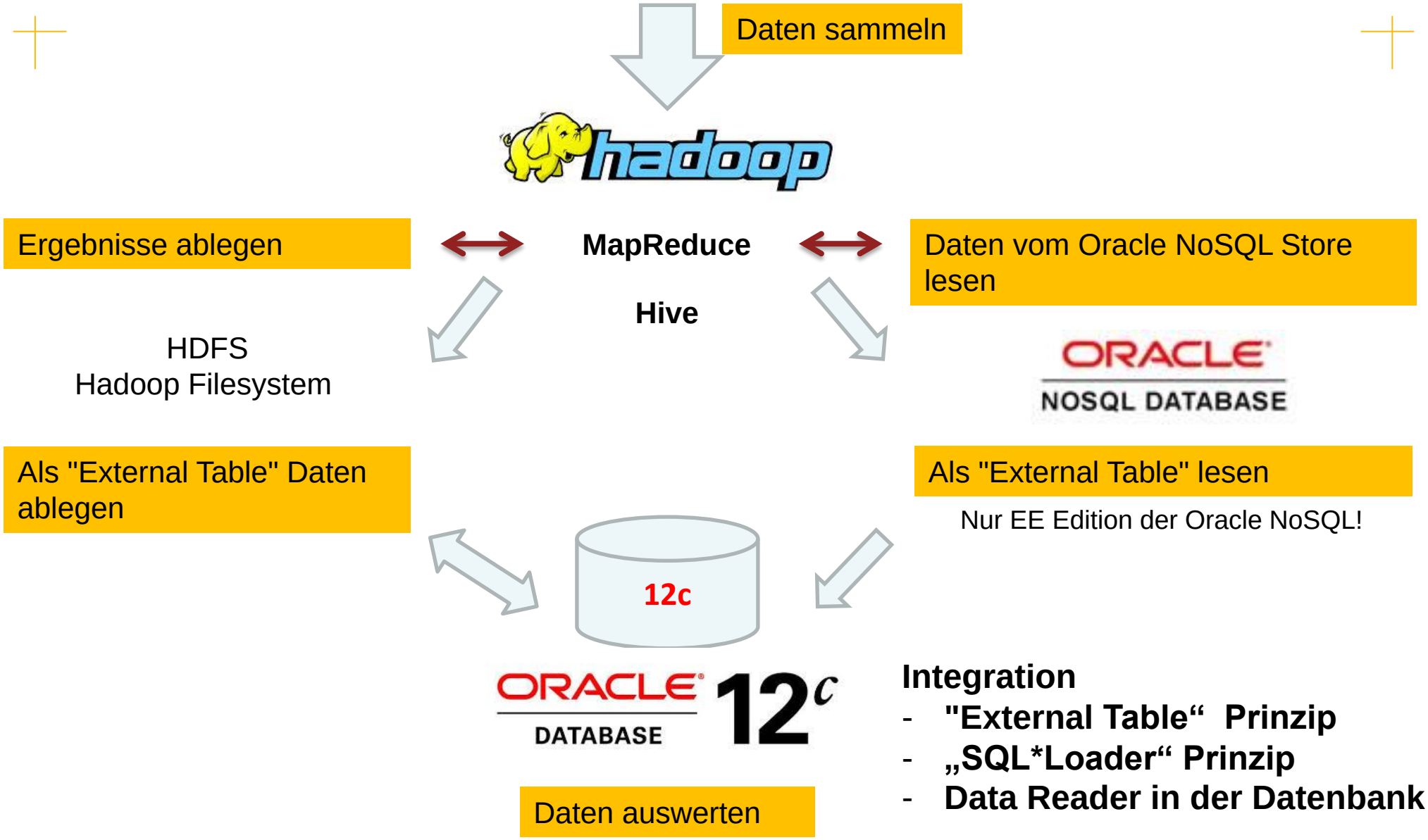
- Logische Integration über das Berichtswesen



Vollständige Umsetzung des NoSQL Konzepts

Wie wird der Zugriff auf die Daten kontrolliert und protokolliert – Stichwort Berechtigungskonzept

Integration der DB‘s in das Hadoop Ökosystem



Integration

- **"External Table“ Prinzip**
- **„SQL*Loader“ Prinzip**
- **Data Reader in der Datenbank**

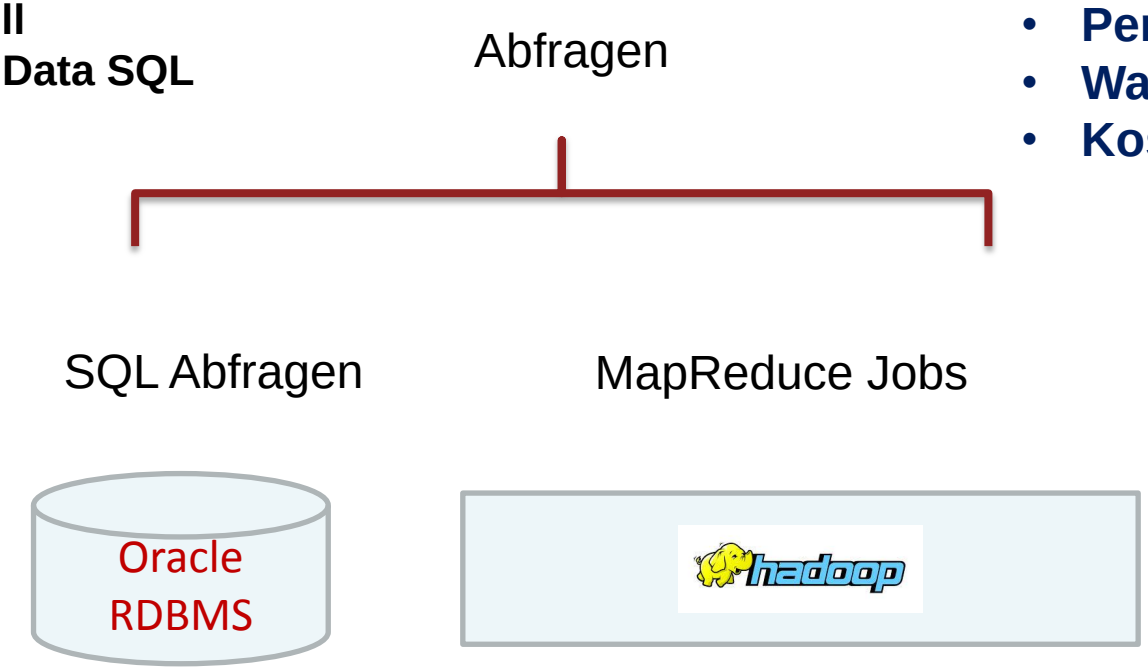
Umsetzung

Wie kann die Integration technisch gelingen?

Hadoop Sicht – Ziel – Eine zentrale Abfrage (1)

- Ziel: mit einer Abfrage ein Ergebnisse aus allen Quellen erhalten

Apache Drill
Oracle Big Data SQL



Auswahl Kriterien:

- **Performance**
- **Wartbarkeit/Pflege**
- **Kosten**

Kein direkter Master für die Daten

Hadoop Sicht – Ziel – Eine zentrale Abfrage (2)

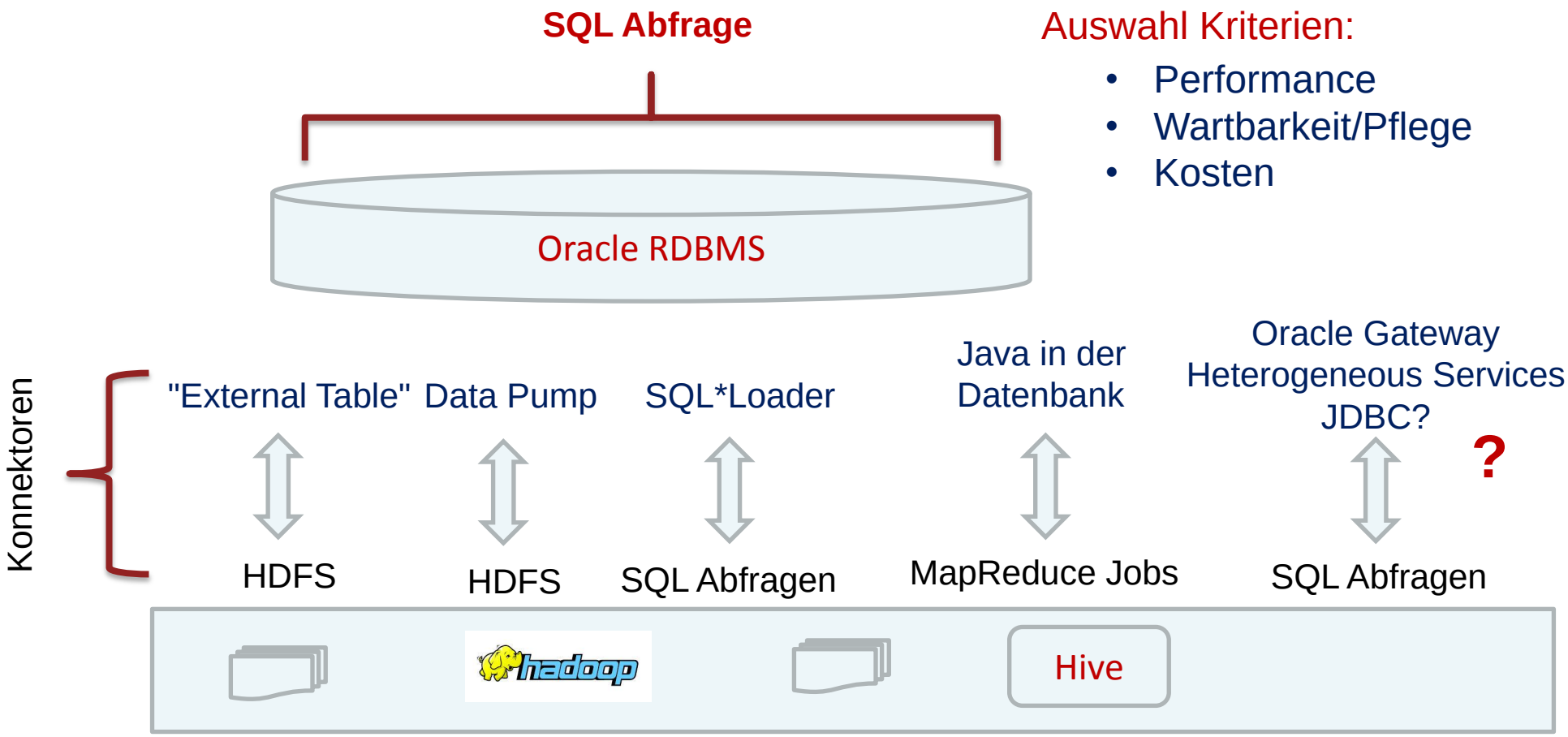
- Prinzip des umgekehrten ETL
  - Quelldaten erstmal einfach in Rohformat speichern
  - Die Quelldaten werden nicht aufbereitet!
    - Die Abfragen auf die Daten passen sich flexible an die Daten an, (soweit möglich)
  - Mehrere Quellen werden kombiniert

- Abfrage Systeme über mehrere Datenquellen
  - Apache Drill
    - SQL Meta Suche

  - Oracle Big Data SQL – ein neues Oracle Produkt

Datenbank Sicht – Ziel – eine SQL Abfrage

- Ziel: Eine SQL Abfrage aus der Datenbank



Auswahl Kriterien:

- Performance
- Wartbarkeit/Pflege
- Kosten

Der Master ist die Oracle Datenbank

Konnektoren ⇔ ORACLE RDBMS (1)

- "External Table" Prinzip – Daten verbleiben im HDFS
  - CSV Dateien / Hive Tabellen als Oracle Tabellen lesen

    - Eigene Skripte mit "PREPROCESSOR" für "External Table" einbinden
    - FUSE Mount Points im Filesystem für "traditionelle" Zugriffe einbinden
      - HDFS als "normales" Filesystem einbinden
    - Kostenpflichtige Lösung: Oracle SQL Connector for HDFS - OSCH
      - Ehemals Oracle Direct Connector for Hadoop Distributed File System
    - Direkt vom HDFS die Daten per Java Klassen aus der DB lesen

Konnektoren ⇔ ORACLE RDBMS (2)

- „External Table" Prinzip – Daten verbleiben im HDFS
  - Data Pump "External Table" als Oracle Tabellen lesen

    - Tabellen / Partitionen in das HDFS archivieren und in der DB löschen
    - Bei Bedarf die Daten wieder einbinden
    - FUSE Mount Point
      - HDFS als Filesystem für die Archivierung nützen
    - Als "External Table" einbinden
      - Oracle Direct Connector – "External Table" im Data Pump Format

Oracle "External Table" TYPE oracle_datapump können NICHT mit "impdb" als Import Quelle verwendet werden, da das Export File Format unterschiedlich ist!

Konnektoren ⇔ ORACLE RDBMS (3)

- SQL*Loader Prinzip – Daten werden in das jeweilige Ziel geladen
  - Oracle*Loader für Hadoop
    - Für das einlesen der Daten vom Hadoop Cluster
  - Quest (veraltet?)
    - Data Connector for Oracle and Hadoop (Erweiterung Sqoop)
  - Apache Sqoop(TM)
    - Open Source SQL*Loader für Hadoop – lesen UND schreiben

=> Apache Flume

Mehr für das zentrale Sammeln von Log Daten: Einsammeln, filtern und in das HDFS schreiben

Konnektoren ⇔ ORACLE RDBMS (4)

- Data Konnektoren über Java IN der Datenbank
  - Eigene Java Programm lesen Daten aus der DB von Hadoop
    - MapReduce Framework aus der Datenbank heraus aufrufen
    - Wie ein SOA Adapter oder ein klassisches REST Interface
  - MapReduce Jobs per Java mit Oracle Klassen mit 12c r2 in Vorbereitung

Wie kann das praktisch umgesetzt werden?

Wie fange ich da nun an?

Unser Ziel – Eine zentrale Log Ablage

- Alert.log
- listener.log
- DB Job Logs

- Audit$ Logs

- Statspack/AWR/ASH
- Trace Files
- SQL Text und Execution Pläne

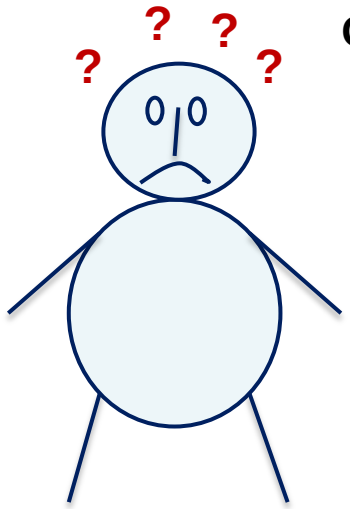
Die Aufgabe- Ein eigener Oracle Audit Vault (1)

- Zentrales Sammeln und Auswertung aller Security Log Daten alle Datenbanken im Unternehmen
  - Ausgewertet wird:
    - AUD$ Einträge
    - Listener.log
    - Alert.log
  - Über eine zentrale Oracle Datenbank sollen die Ergebnisse mit Apex ausgewertet werden

Anforderung: Autorisierung und Unveränderbarkeit der Daten sicherstellen

Qual der Wahl!

- Wo anfangen? – Welche Produkte wählen?



OSCH – Oracle SQL Connector for HDFS

Oracle BIG Data SQL

Hadoop

Apache Drill

Oracle*Loader für Hadoop - OLH

Hive

Sqoop

Flume

PIG

Wie verwalte und warte ich das Ganze?

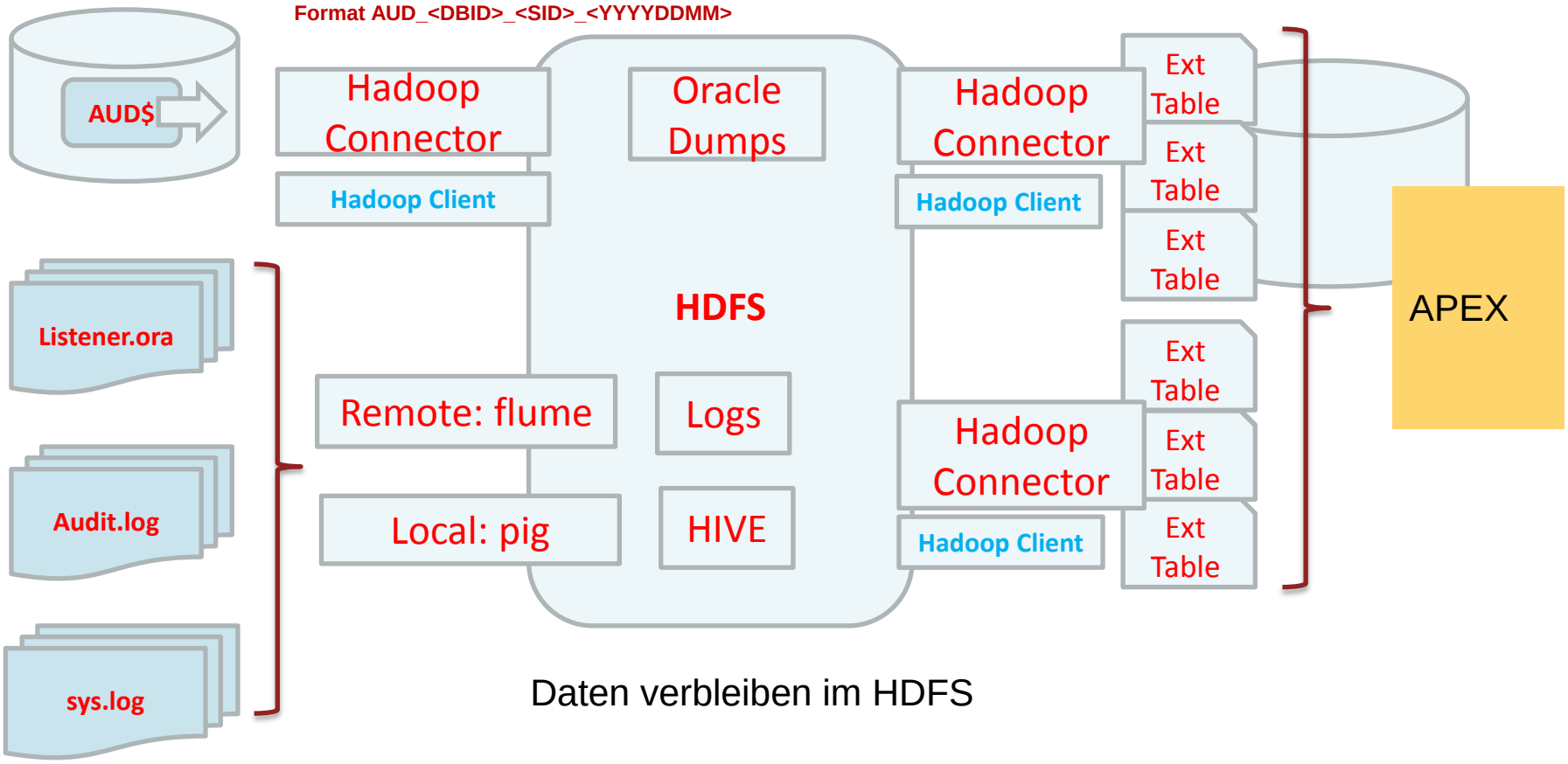
Was kann ich mir leisten?

Was muss ich am wenigsten installieren /konfigurieren?

Architektur Übersicht – Variante Oracle Adapter

Quellen pro Maschine

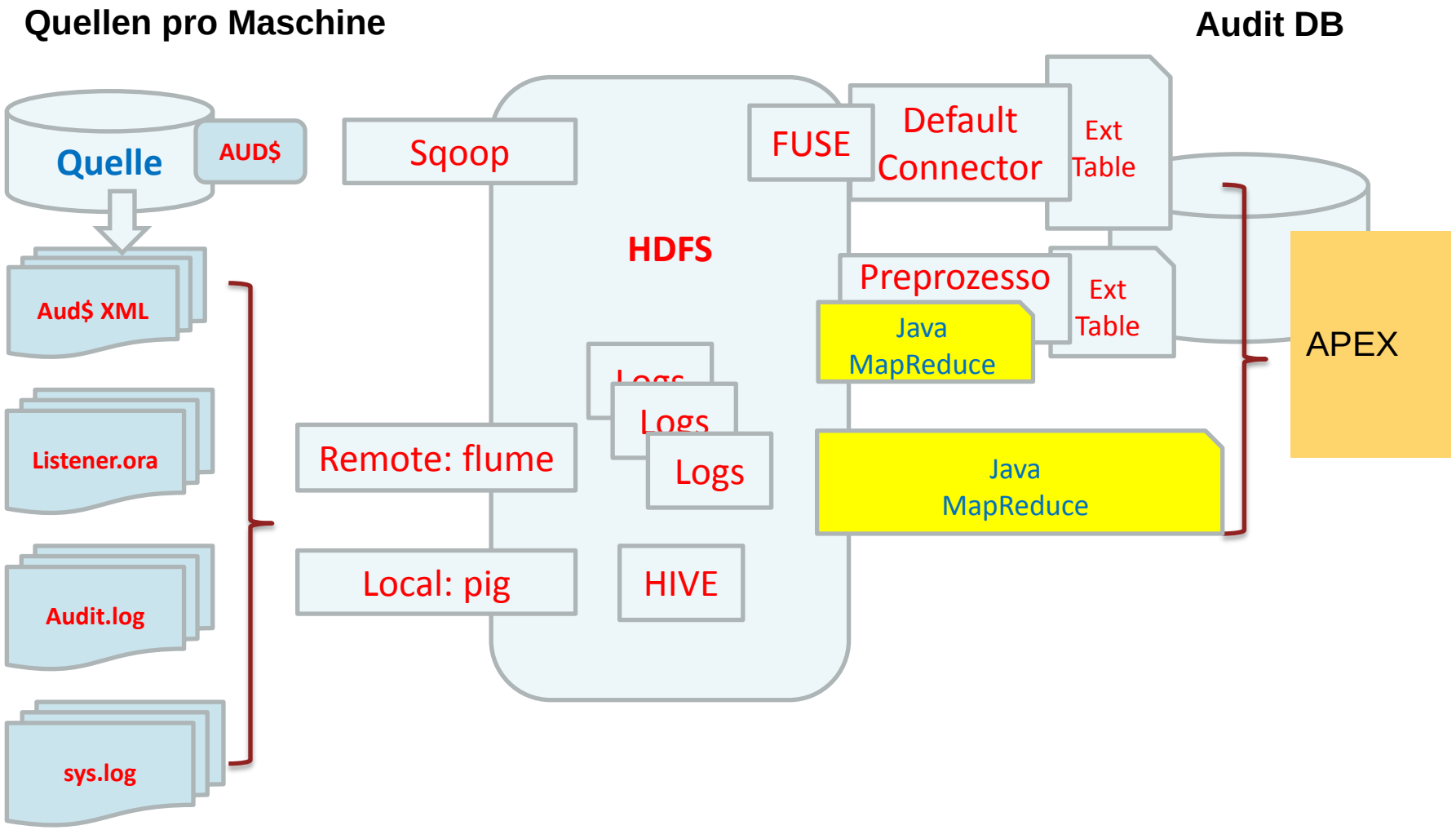
Audit DB



Daten verbleiben im HDFS

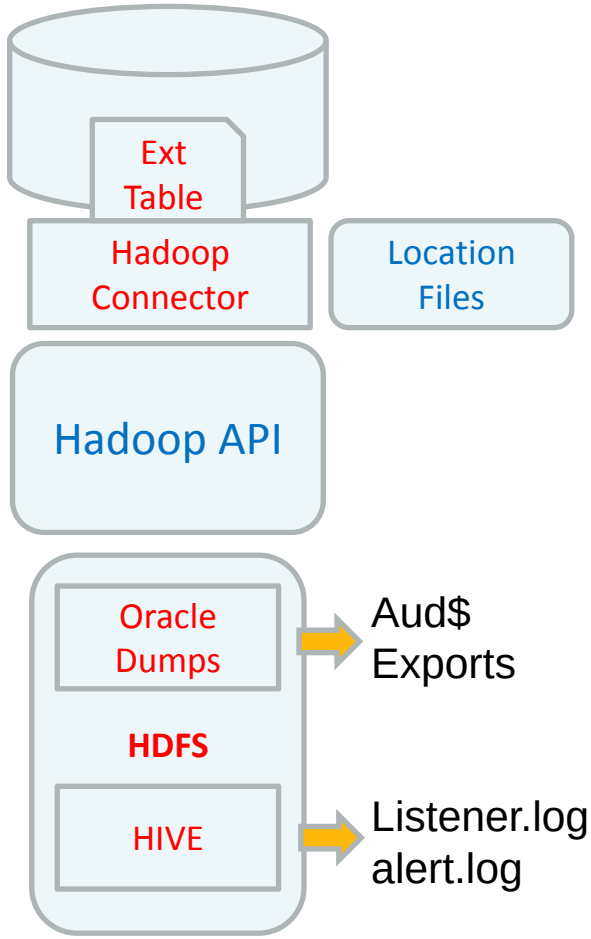
Zugriffsberechtigungen
werden in der Oracle DB gepflegt

Architektur Übersicht – Variante Open Source



Umsetzung – "External Table" Konnektoren

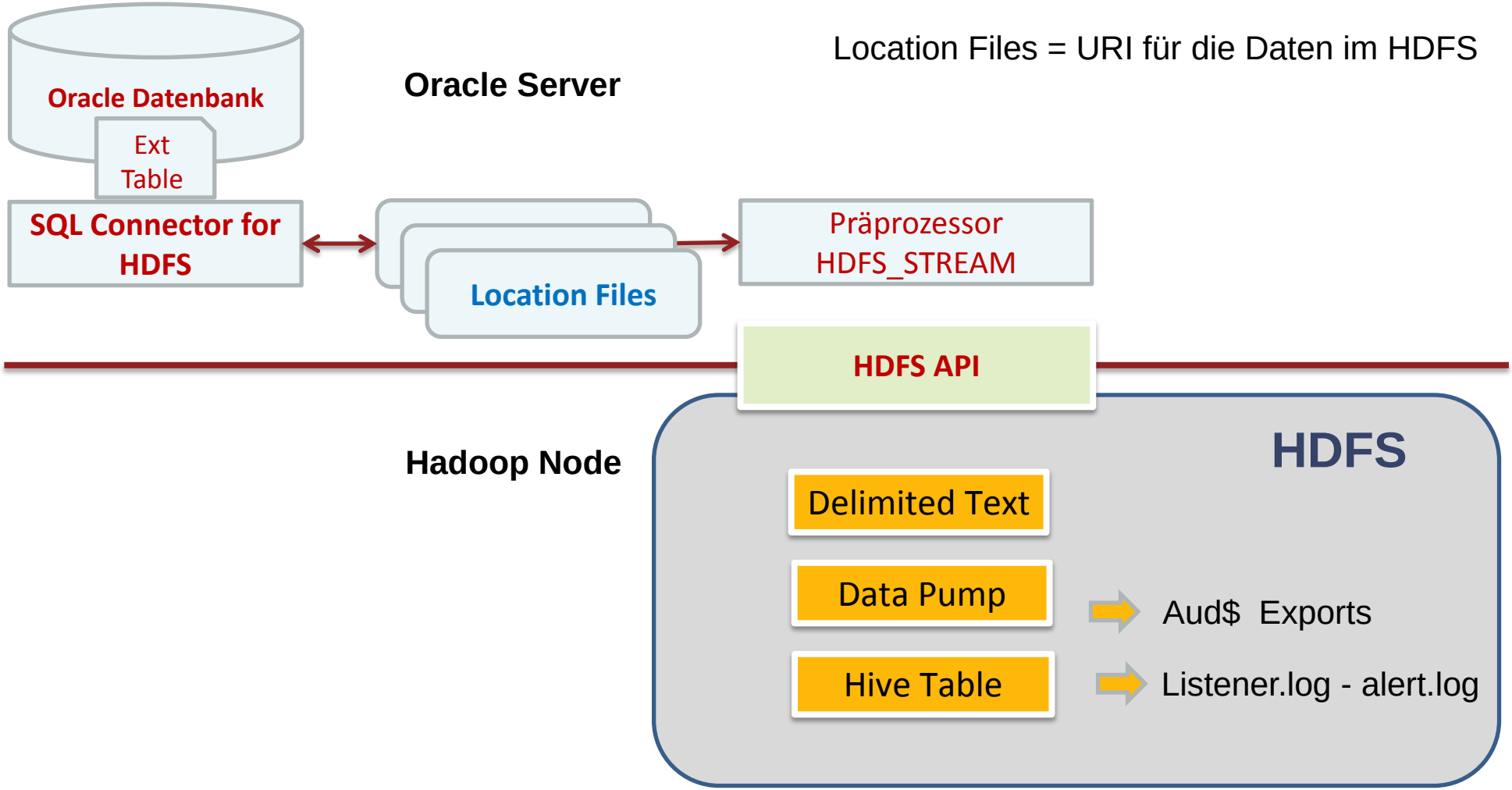
- **O**racle **S**QL **C**onnector for **H**DFS **OSCH**
  - Data Pump Table Exports möglich
    - Vorteil: Schnelles wiedereinlesen
    - Nachteil:
      - Für andere Tools nicht lesbar!
      - Evtl.. Datenbank-Versions abhängig!
      - Es können keine Daten angefügt werden!
  - Direktes Erzeugen von Hive Tabellen
    - Vorteil: auch andere Werkzeuge können mit den Daten arbeiten
  - CSV Dateien



Zur Zeit nur für Datenbanken unter Linux verfügbar!

OSCH - Oracle SQL Connector for HDFS

- SQL Abfrage über eine "External Table"



OSCH – Die Bedeutung der Location Files

- Location Files – Pointer auf die Daten im HDFS
  - Die Location Files enthalten die einzelnen Ziel Dateien auf dem HDFS
  - Werden beim „-createTable“ unter dem Verzeichnis des DIRECTORY Objekts anleget
  - Ändern die sich die Daten im HDFS können diese Dateien mit „-publish“ aktualisiert werden
  - Beispiel:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<locationFile>
    <header>
        <version>1.0</version>
        <fileName>osch-20140831075440-588-1</fileName>
        <createDate>2014-08-30T19:10:08</createDate>
        <publishDate>2014-08-31T07:54:40</publishDate>
        <productName>Oracle SQL Connector for HDFS Release 3.0.0 - Production</productName>
        <productVersion>3.0.0</productVersion>
    </header>
    <uri_list>
        <uri_list_item size="11723955"
compressionCodec="">HDFS://bigdatalite.localdomain:8020/tmp/auditGPI/part-m-00003</uri_list_item>
    </uri_list>
</locationFile>
```

Installation Oracle Connector for HDFS

- Hadoop Client auf DB Host aufsetzen
- Zip Archive auspacken und Umgebung anpassen
- Oracle DB User einrichten/konfigurieren
  - Recht read/write/execute auf das OSCH_BIN Directory für den User zusätzlich vergeben!
- Mit dem „external Command Tool“ die „External Table“ anlegen
- Abfragen

Oracle SQL Connector for Hadoop OSCH (1)

- „Oracle SQL Connector for Hadoop“ für HIVE verwenden
  - "External Table" erzeugen mit:

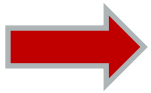
```
hadoop jar $OSCH_HOME/jlib/orahdfs.jar                                    \
oracle.hadoop.exttab.ExternalTable                                        \
-D oracle.hadoop.exttab.hive.tableName=ext_ora_audit_gpi                  \
-D oracle.hadoop.exttab.hive.databaseName=default                         \
-D oracle.hadoop.exttab.sourceType=hive                                   \
-D oracle.hadoop.exttab.tableName=ext_ora_audit_gpi_hive                  \
-D oracle.hadoop.connection.url=jdbc:oracle:thin:@//localhost:1521/orcl   \
-D oracle.hadoop.connection.user=scott                                    \
-D oracle.hadoop.exttab.defaultDirectory=HADOOP_EXT_TABS                  \
-createTable
```

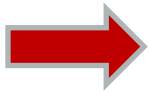
Auf der zentralen Datenbank
für die Auswertung der Log Daten

Oracle SQL Connector for Hadoop OSCH (2)

- Oracle SQL Connector for Hadoop
  - 4. "External Table" DDL abfragen mit „-getDDL“ / überprüfen:

```
ORGANIZATION EXTERNAL
   ( TYPE ORACLE_LOADER
     DEFAULT DIRECTORY "HADOOP_EXT_TABS"
     …
     PREPROCESSOR "OSCH_BIN_PATH":'hdfs_stream'
     ….
)
     )
     LOCATION
     ( 'osch-20140830073731-58-1',
       'osch-20140830073731-58-2',
       'osch-20140830073731-58-3',
       'osch-20140830073731-58-4'
     )
   )
```





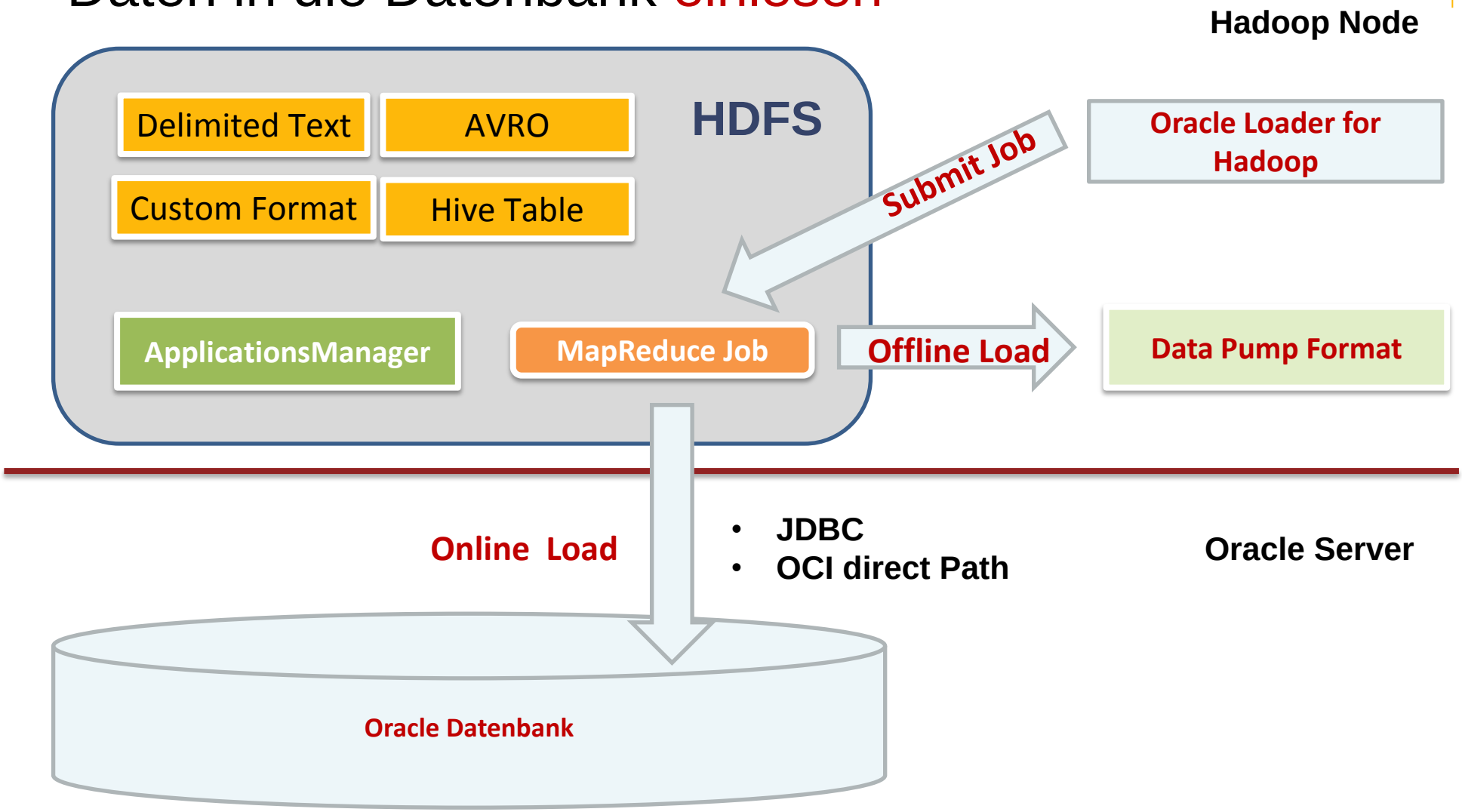
Location Files

Probleme über Probleme .....

- CLASSPATH muss die Jar Files von Hive enthalten
  - Bei Problem im Hadoop Kommando Script direkt anpassen
- HIVE Tabelle darf keine Spalten vom Typ „BINARY“ enthalten
- Problem mit Datumsformaten und Null Spalten
  - Export per Scoop => null wird 'null'
  - Die Generiert Anweisung=>
    - "TIMESTAMP" CHAR DATE_FORMAT DATE MASK 'YYYY-MM-DD' NULLIF "TIMESTAMP"=0X'**5C4E**', => \N
  - Benötigt wird aber ein =>
    - "TIMESTAMP" CHAR DATE_FORMAT DATE MASK 'YYYY-MM-DD' NULLIF "TIMESTAMP"=0X'**276E756C6C27**', => 'null'

Oracle Loader for Hadoop - OLH

- Daten in die Datenbank einlesen



Oracle Loader for Hadoop – OLH (1)

- Dateien im „Data Pump Export“ Format von HDFS Daten für den Import erzeugen
  - Online Mode – Daten werden direkt in die Datenbank geladen
  - Offline Mode – Daten werden zuerst aus dem Hadoop in Data Pump oder CSV Format erzeugt und können später traditionell eingebunden werden
- Ablauf:
  - Tabelle in der Oracle DB anlegen
  - Job Konfigurationsdatei für den MapReduce Job anlegen
  - Job Starten und Daten aus Hive / CSV Daten exportieren und in Richtung Oracle DB importieren

Nachteil:
An dem Data Pump Dump können keine weitern Daten angefügt werden
Dump müssen über select * und weitere "External Table" zusammen gefügt werden

Oracle Loader for Hadoop – OLH (2)

- Probleme:
  - Bei der Verwendung von Hive Tabellen müssen noch einige Jar Files einzeln angegeben werden!
  - Fehler:

```
Error: java.lang.ClassNotFoundException: org.apache.hadoop.hive.metastore.TableType at
.. Und weitere
```

  - Lösung

```
hadoop jar /u01/connectors/olh/jlib/oraloader.jar oracle.hadoop.loader.OraLoader \
-conf hdfs_job_config.xml                                                        \
-libjars /usr/lib/hive/lib/hive-metastore-0.12.0-cdh5.0.0.jar
,/usr/lib/hive/lib/hive-serde-0.12.0-cdh5.0.0.jar
,/usr/lib/hive/lib/libthrift-0.9.0.cloudera.2.jar
,/usr/lib/hive/lib/hive-common-0.12.0-cdh5.0.0.jar
```

Lizenz?

- Lizensierung Oracle Big Data Connectors
  - Pro Prozessor der Hadoop Clusters

    See: http://docs.oracle.com/cd/E53356_01/doc.30/e53065/bda.htm#BIGLI113

    Oracle Big Data Connectors must be licensed for all processors of a Hadoop cluster. When Oracle Big Data Connectors is installed on a single Hadoop cluster, it must be licensed on all processors. When Oracle Big Data Connectors is installed on multiple Hadoop clusters, it must be licensed on all processors of the clusters where the connectors are used.

  - Kosten:
    - Oracle Technology Global Price List July 22, 2014
    - 2000,- USA (Dollar) per Hadoop Cluster Processor

    See :http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

	Named User Plus	Software Update License & Support	Processor License	Software Update License & Support
Other Products				
Big Data Connectors	-	-	2,000	440.00

"External Table" per FUSE Projekt

- HDFS Filesystem per FUSE als Linux Filesystem einbinden
  - Mit Hilfe der FUSE Technology (Mountable HDFS - mount point for HDFS) kann auf das HDFS mit Standard Oracle Methoden wie auf ein normales Filesystem zugegriffen werden.

```
yum install hadoop-hdfs-fuse.x86_6

hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port>
```

  - Laut Oracle Doku langsamer als die Oracle Big Data Connectors (Faktor 5) und CPU Lastig (3' Fach höher)
  - https://wiki.apache.org/hadoop/MountableHDFS

"External Table" mit PREPROCESSOR

- Einbinden von Skripten über die PREPROCESSOR Anweisung
- Ein Shell Script wird aufgerufen und liefert Daten zurück
  - Die Dateiangaben im Location Part können als Parameter übergeben werden, damit kann der Job parametrisiert werden
  - Im Shell Script kann der Connect auf die Daten erfolgen
  - Standard Out Pipe wird mit der "External Table" verknüpft

Diese Lösung ist damit Lizenzkostenfrei!

Beispiel „external Table“ mit PREPROCESSOR

- Beispiel:

```
CREATE TABLE PREPREP_EXT(
 wert   VARCHAR2(2000)
 )
 ORGANIZATION EXTERNAL
 ( TYPE ORACLE_LOADER
   DEFAULT DIRECTORY DATA_LOAD
   ACCESS PARAMETERS
   ( RECORDS DELIMITED BY NEWLINE
     PREPROCESSOR EXT_COMMAND:'loadFromHadoop.sh'
     SKIP 0
     LOGFILE EXT_DATA:'data_load.log'
     BADFILE EXT_DATA:'data_load.bad'
     NODISCARDFILE
     FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
     MISSING FIELD VALUES ARE NULL
   )
   LOCATION (EXT_DATA:'hadoop_locator.dat')
 )
 REJECT LIMIT 0
 NOPARALLEL
 NOMONITORING
 ;
```

„External Table“ mit PREPROCESSOR (2)

- Im Script wird die entsprechende Logik aufgerufen
  - Lade die Daten aus dem Hadoop auf den DB Server
    - hdfs dfs –cat /user/auditdat/logfile.a..
  - Stelle die Daten der "External Table" zur Verfügung
    - Über Standard out wird verknüpft

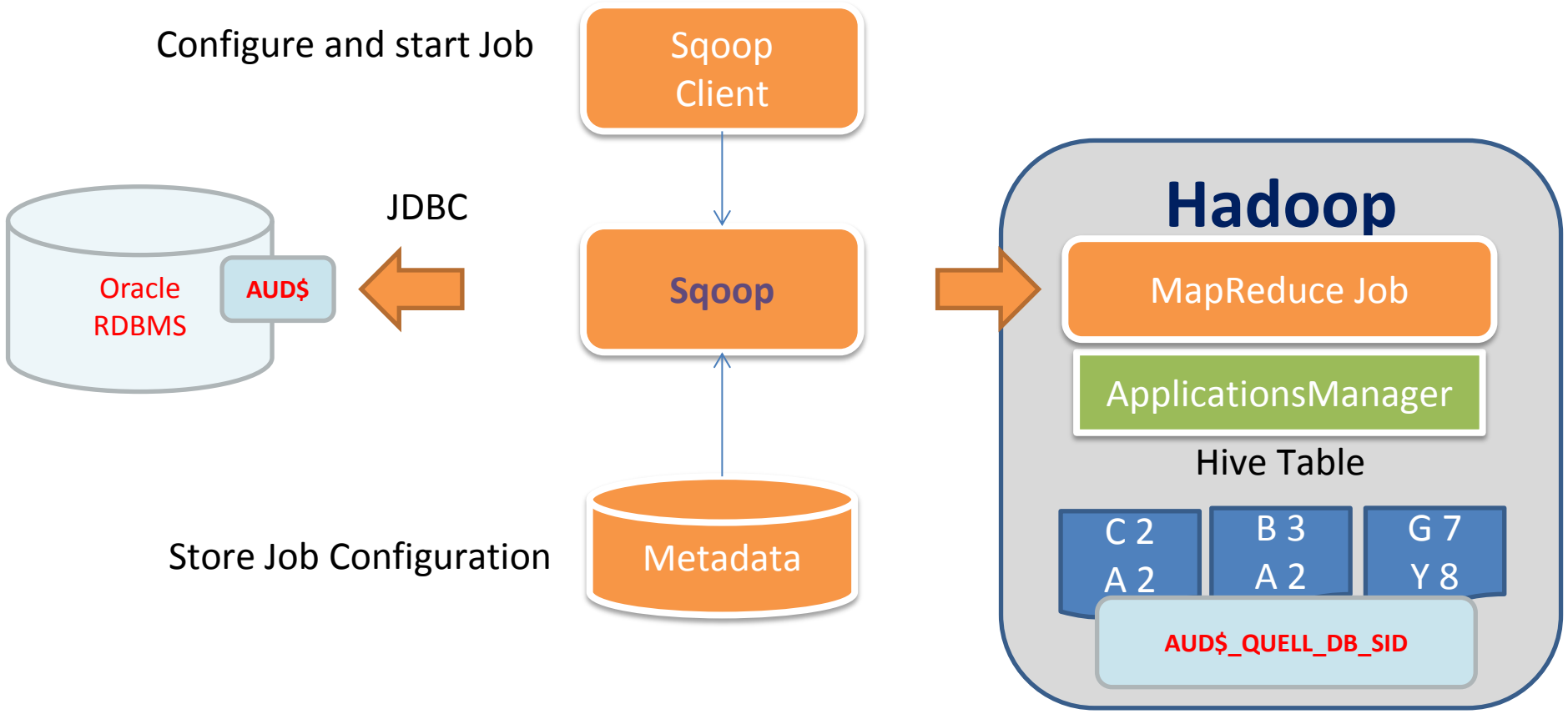
```
#!/bin/sh
#Laufzeitumgebung setzen
export PATH=$PATH:/bin:/sbin/:/usr/bin
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
export HADOOP_CONF_DIR=/etc/hadoop/conf.bigdatalite

# Übergabe Parameter Datei auslesen und Inhalt als Parameter verwerten
FILENAME=`/bin/cat $1`

#Mit HDFS Bordmitteln die Datei auswerten
/usr/lib/hadoop-hdfs/bin/hdfs dfs -cat $FILENAME
```

DEMO

- External Table Demo

Sqoop 2 einsetzen

- Sqoop 2 Data Loader – Eine Art SQL*Loader?



Daten aus der Quelldatenbank exportieren und in Hadoop ablegen

Sqoop 2 – Installation und Setup

- Software herunterladen und auspacken
  - JDBC Treiber für Oracle kopieren
- Connect zum Hadoop Cluster konfigurieren
  - Mindestens ein Hadoop Client muss installiert sein
- Server Prozess starten
- Daten transferieren

Flexibler Import mit Sqoop

- Daten aufteilen mit
  - Select <spalte> from <table> where <condition>
  - Eigene SQL Abfrage
- Parallele Abarbeitung
- Inkrementelles Laden über eine where Bedingung

Bei Test auf der Oracle BigData Appliance
„Exception in thread „main“ groovy.lang.MissingPropertyException: No such property: terminal for class: jline.Terminal“

Lösung:

```
unset CLASSPATH
export CLIENT_LIB=/usr/lib/sqoop2/client-lib

/usr/lib/sqoop2/bin/sqoop.sh client
```

Beispiel Sqoop 2

- Am Sqoop Server anmelden

```
$SQOOP_HOME/bin/sqoop.sh client
sqoop:000> set server --host nosqldb01 --port 12000 --webapp sqoop
```

- Connection Objekt anlegen
  - JDBC Zugriff auf die Oracle DB konfigurieren

```
sqoop:000> create connection --cid 1
```

- Job Objekt definieren
  - Wie soll der Job abgearbeitet werden

```
sqoop:000> create job --xid 1 --type import
```

- Job starten und Daten nach Hadoop importieren

```
sqoop:000> start job --jid 1
```

Sqoop ist nicht „SQL Injektion“ save!

- In der AUD$ hieß ein User 'TESTER
  - Beim Laden im Log:

```
SELECT * FROM SYSTEM.AUD$ WHERE ''TESTER' <= USERID AND
USERID < '+????'
```

```
014-08-30 16:46:51,537 INFO [main] org.apache.sqoop.connector.jdbc.GenericJdbcImportExtractor: Using query: SELECT * FROM SYSTEM.AUD$ WHERE ''TESTER' <= USERID AND USERID < '+????'
014-08-30 16:46:53,650 INFO [main] org.apache.sqoop.job.mr.SqoopMapper: Stopping progress service
014-08-30 16:46:53,650 INFO [main] org.apache.sqoop.job.mr.SqoopOutputFormatLoadExecutor: SqoopOutputFormatLoadExecutor::SqoopRecordWriter is about to be closed
014-08-30 16:46:53,659 INFO [OutputFormatLoader-consumer] org.apache.sqoop.job.mr.SqoopOutputFormatLoadExecutor: Loader has finished
014-08-30 16:46:53,660 INFO [main] org.apache.sqoop.job.mr.SqoopOutputFormatLoadExecutor: SqoopOutputFormatLoadExecutor::SqoopRecordWriter is closed
014-08-30 16:46:53,661 WARN [main] org.apache.hadoop.mapred.YarnChild: Exception running child : org.apache.sqoop.common.SqoopException: MAPRED_EXEC_0017:Error occurs during extrac
        at org.apache.sqoop.job.mr.SqoopMapper.run(SqoopMapper.java:101)
        at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:764)
        at org.apache.hadoop.mapred.MapTask.run(MapTask.java:340)
        at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:168)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:415)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1548)
        at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:163)
aused by: org.apache.sqoop.common.SqoopException: GENERIC_JDBC_CONNECTOR_0002:Unable to execute the SQL statement
        at org.apache.sqoop.connector.jdbc.GenericJdbcExecutor.executeQuery(GenericJdbcExecutor.java:63)
        at org.apache.sqoop.connector.jdbc.GenericJdbcImportExtractor.extract(GenericJdbcImportExtractor.java:50)
        at org.apache.sqoop.connector.jdbc.GenericJdbcImportExtractor.extract(GenericJdbcImportExtractor.java:31)
        at org.apache.sqoop.job.mr.SqoopMapper.run(SqoopMapper.java:96)
        ... 7 more
aused by: java.sql.SQLSyntaxErrorException: ORA-00920: invalid relational operator
```

Vergleich der SQL*Loader Werkzeuge

Oracle SQL*Loader - OLH

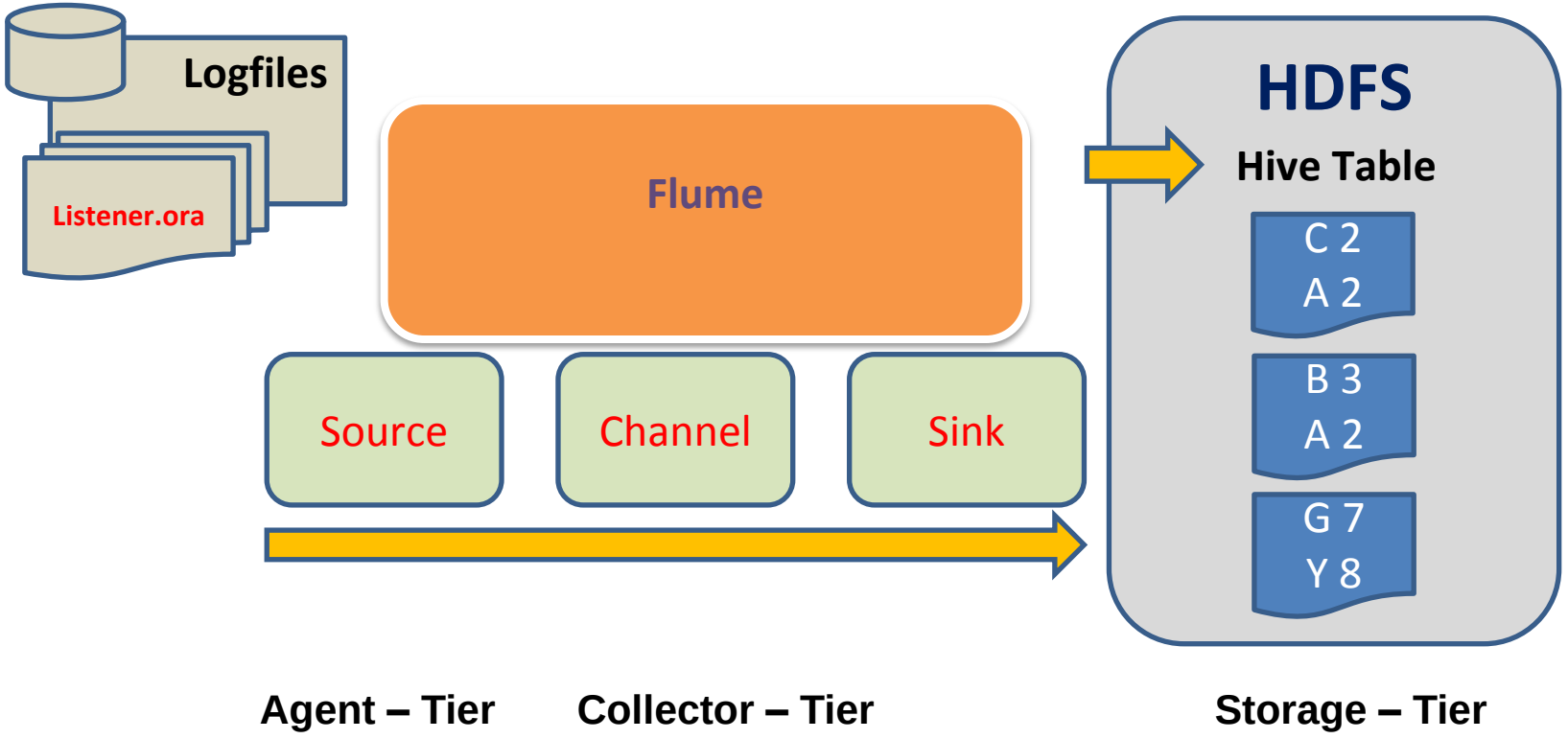
- Kostenpflichtig
- => Laden in die Oracle Datenbank
- Parallelisierung auf Oracle zugeschnitten

Apache Sqoop

- Open Source
- ⇔Laden von und nach Hadoop
- Version 1.4 für Hadoop 1
- Version 1.9 für Hadoop 2
- Parallel Verarbeitung ?

Apache Flume

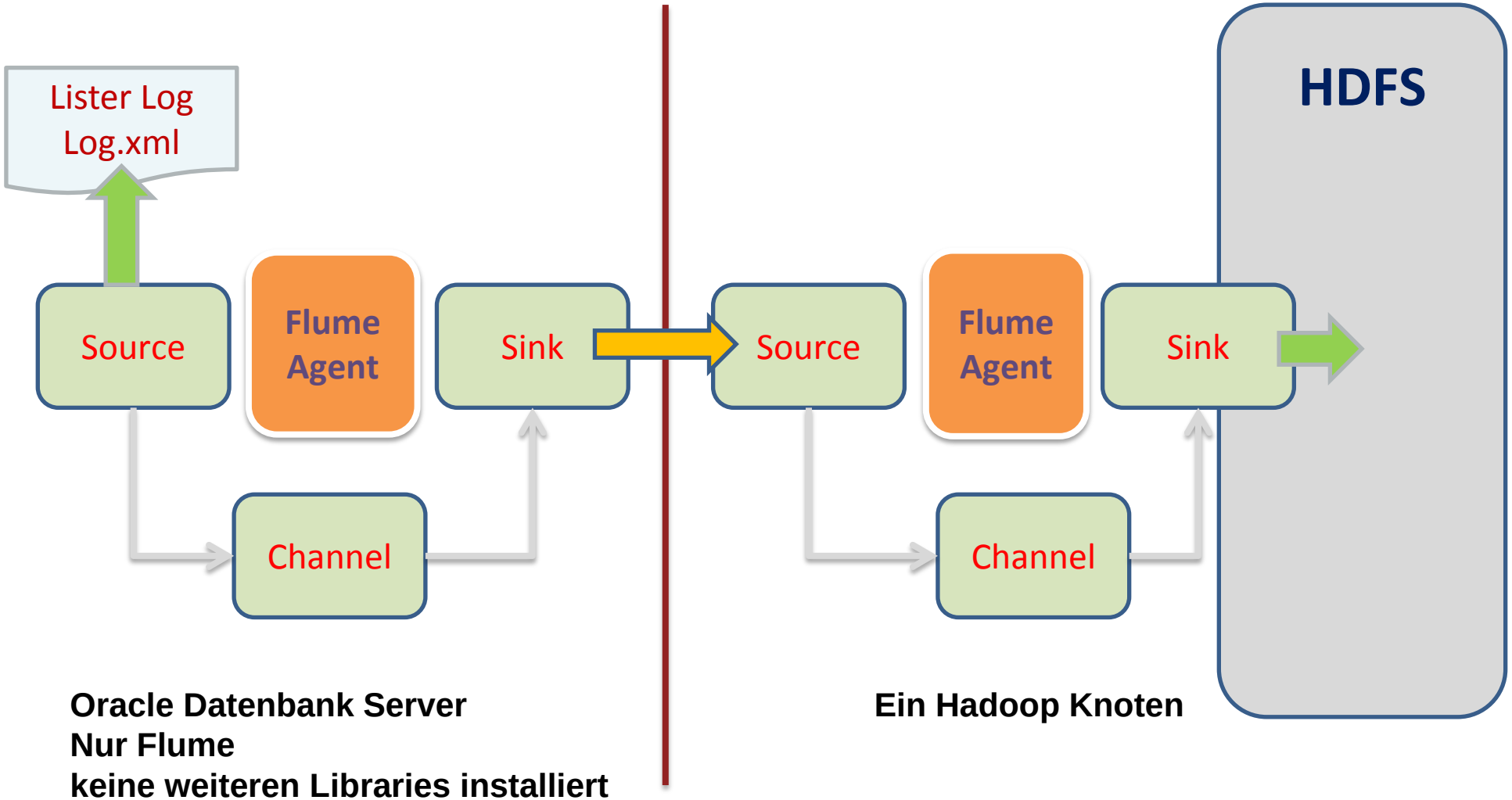
- Eine Art „syslog“ Demon
- Übernimmt den Transport der Log Daten
  - Log Daten in Hive ablegen



Flume sammelt und überträgt

Beispiel für den Einsatz mit dem Listener.log

- Ein Log Eintrag:



Flume Konfiguration

Listener Agent

```
vi /usr/lib/flume-ng/conf/agent-conf.properties

agent.sources = OraListener
agent.channels = memoryChannel
agent.sinks = avro-forward

# Channel type exec
agent.sources.OraListener.type = exec
agent.sources.OraListener.command = tail -F
/opt/oracle/diag/tnslsnr/oradb12c01/listener/alert/log.xml

# The channel can be defined as follows
agent.sources.OraListener.channels = memoryChannel
#sink
agent.sinks.avro-forward.type = avro
agent.sinks.avro-forward.hostname = 10.10.10.12
agent.sinks.avro-forward.port = 44444

#Specify the channel the sink should use agent.sinks.avro-
forward.channel = memoryChannel

# Each channels type is defined
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 100

# starten

export JAVA_HOME=/usr/java/jdk1.7.0_67
/usr/lib/flume-ng/bin/flume-ng agent -n agent -c
/usr/lib/flume-ng/conf -f /usr/lib/flume-ng/conf/agent-
conf.properties
```

HDFS Agent

```
vi /etc/flume-ng/conf/agent2-conf.properties

agent2.sources  = OraLogFiles
agent2.channels = memoryChannel
agent2.sinks    = hdfsSink

# Channel type avro
agent2.sources.OraLogFiles.type = avro
agent2.sources.OraLogFiles.bind = 10.10.10.12
agent2.sources.OraLogFiles.port = 44444


# The channel can be defined as follows
agent2.sources.OraLogFiles.channels = memoryChannel

# Each sinks type must be defined
agent2.sinks.hdfsSink.type = HDFS
agent2.sinks.hdfsSink.hdfs.path =
HDFS://bigdatalite.localdomain:8020/user/oracle/gpi
agent2.sinks.hdfsSink.hdfs.filePrefix = OraLogs-
agent2.sinks.hdfsSink.hdfs.rollCount = 1000
agent2.sinks.hdfsSink.hdfs.batchSize = 10

#Specify the channel the sink should use
agent2.sinks.hdfsSink.channel = memoryChannel

# Each channels type is defined
agent2.channels.memoryChannel.type = memory
agent2.channels.memoryChannel.capacity = 100

# starten
flume-ng agent -n agent2 -c conf -f /etc/flume-
ng/conf/agent2-conf.properties
```

Was ergibt sich daraus für unsere Aufgabe? (1)

- Bei Verwendung des Oracle Big Data Connectors für den Export:
  - Von jeder DB wird täglich die Aud$ exportiert
    - Format Dump AUD_<DBID>_<SID>_<YYYYMMDD>
    - Lokales PL/SQL Package (mit Hilfe einer Log Tabelle) führt den Export durch
  - Monatlich werden die Daten von der zentralen Datenbank in einen Monats Dump verdichtet
  - Nachteil:
    - Hadoop Client auf jeder Datenbank notwendig
    - Windows wird schwierig
    - Kosten
    - Dumps müssen ersten durch "Umwege" verdichtet werden

Was ergibt sich daraus für unsere Aufgabe? (2)

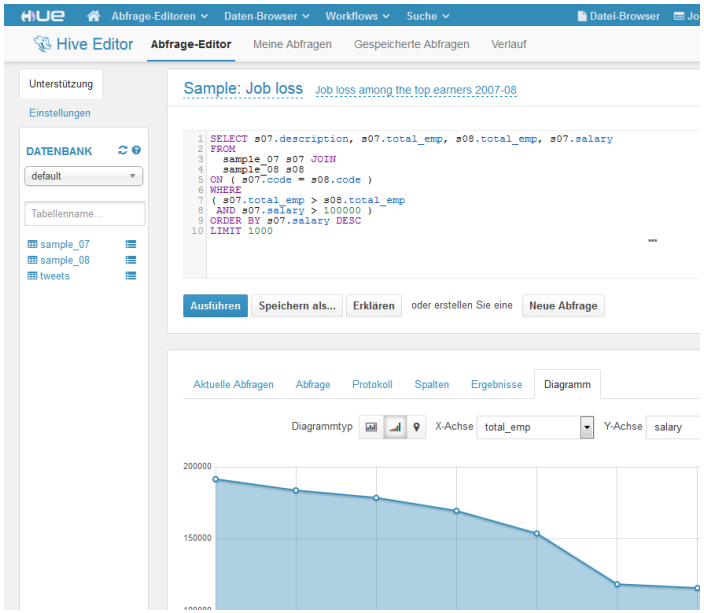
- Lösung mit Sqoop und Hive
  - Zentral vom einem Server
    - Tägliche wird die AUD$ Tabelle in eine Hive Tabelle exportiert
  - Vorteil:
    - Keine Installation auf den Quell Servern
  - Nachteil:
    - Geringere Performance

Die Daten auswerten

Was fange ich jetzt mit den Daten an?

Auswerten der Daten

- Hive Tabellen in Oracle einbinden
  - Mit dem OLH einfach - aber Lizenzkosten pflichtig
- Hive Tabellen per JDBC direkt abfragen
  - Open Source wie **Hue**:



Fragen zum Betrieb

Wie kann ich das über die Dauer der Jahre warten?

Ein Hadoop Cluster betreiben

- Backup Konzept?
  - Ausfallsicherheit und Daten Redundanz sind kein Schutz vor logischen Fehlern!
- Datenquellen entkoppeln
  - Wie lange können die Vorsystem ihre Daten halten?
    - Buffer Konzepte notwendig?
- Upgrade
  - Patch Konzept für Treiber und OS
    - Z.B. einzelne Knoten rollierend jeden Monat patchen
  - Software Upgrade?
    - Was tun bei einem Mayor Release?

Fazit

Lohnt sich der Aufwand?

Größtes Problem

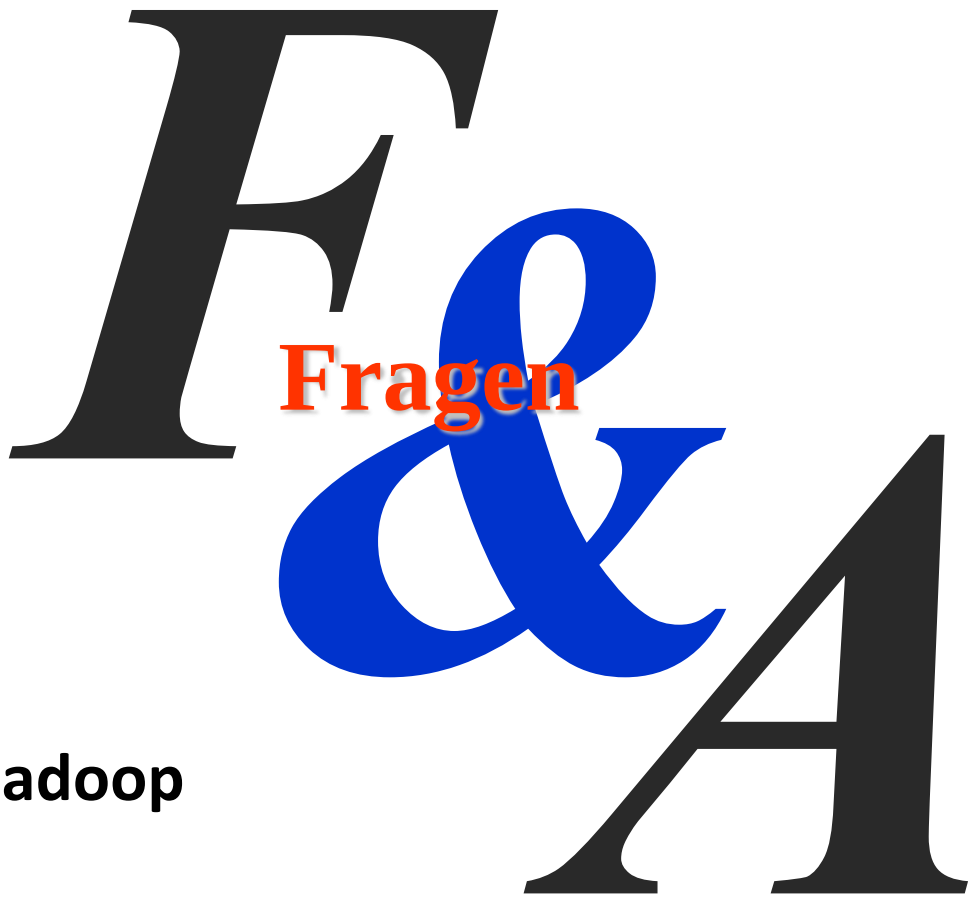
- Viele Updates und unübersichtliche Kompatibilitäten der ganze Werkzeugpalette
  - Was wird es auch in Zukunft noch geben?
- Mapping von Datentypen
  - Viele Detailprobleme
- Java Out of Memory Effekte beim Laden
  - Jobs bleiben einfach im Hadoop "hängen"

Fazit

- Ein Apache Hadoop Cluster bietet sich als eine gute Lösung für die Anforderung eines zentralen Audit Vaults für Log Daten an
- Die notwendigen Werkzeuge für die Umsetzung der Anforderung sind verfügbar
- Die Orchestrierung des Ganzen ist jedoch die große Herausforderung bei einem solchem Projekt
- Der damit verbunden Aufwand sollte nicht unterschätzt werden
- Der große Vorteil der Eigenentwicklung einer Audit Valut Lösung liegt darin, mit diesem Projekt einen guten Einstieg in diese neue Technologie ohne großes Business Risiko umsetzen zu können

Ihre Erfahrungen?

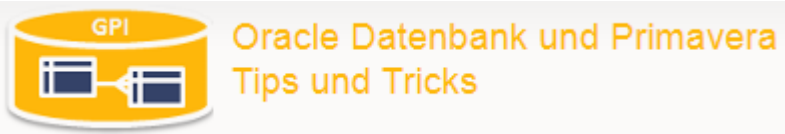
?



Oracle und Hadoop

Quellen

- Oracle – BIG Data
  - https://www.oracle.com/bigdata/products.html



- Mehr über das Thema und die Quellen im Detail siehe:
  - http://www.pipperr.de/dokuwiki/doku.php?id=nosql:hadoop_integration
  - http://www.pipperr.de/dokuwiki/doku.php?id=nosql_datenbank

Gunther Pippèrr - IT-Architekt - Berater

Background

Gunther Pippèrr arbeitet seit mehr als 16 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

Web

http://www.pipperr.de
http://orapowershell.codeplex.com

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, Loyalty Card Anwendungen
- CRM- und Ausschreibungsportal für großen Münchner Bauträger