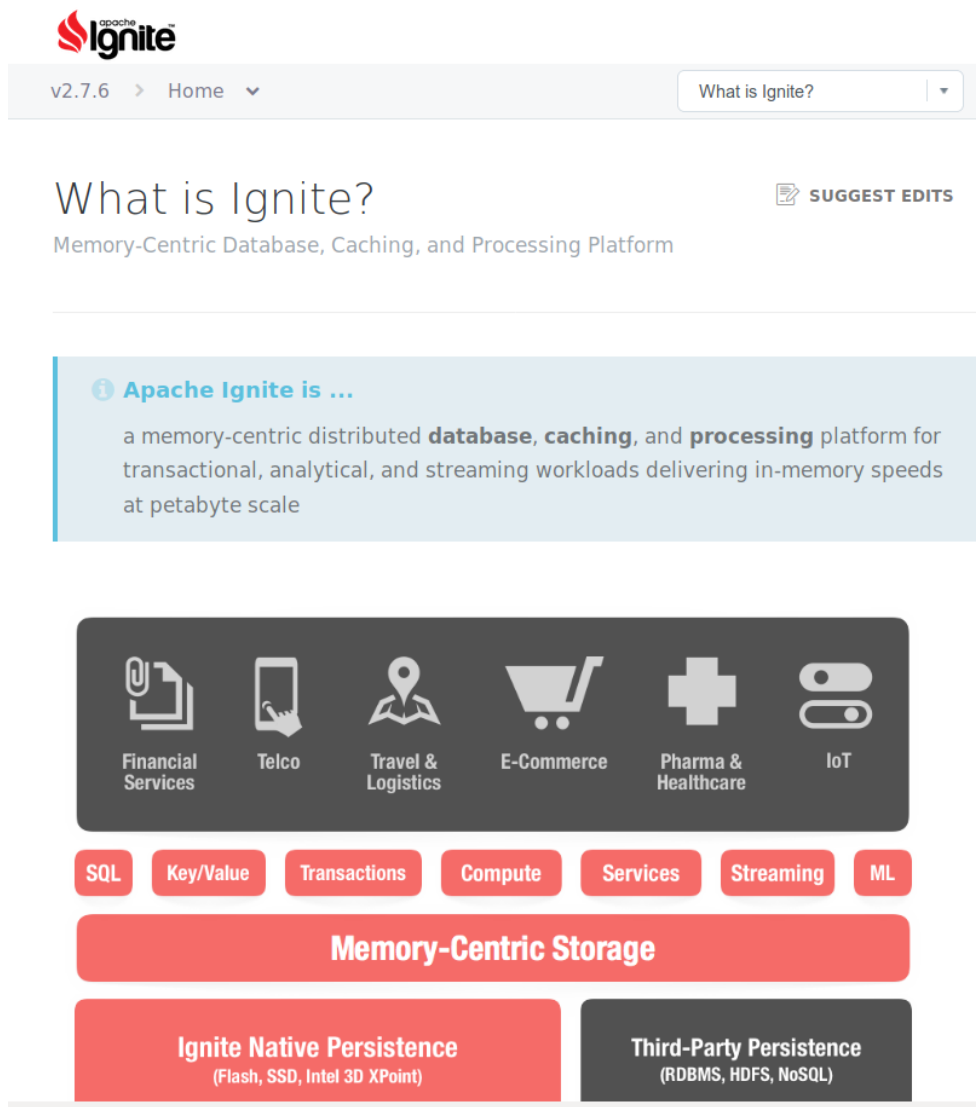


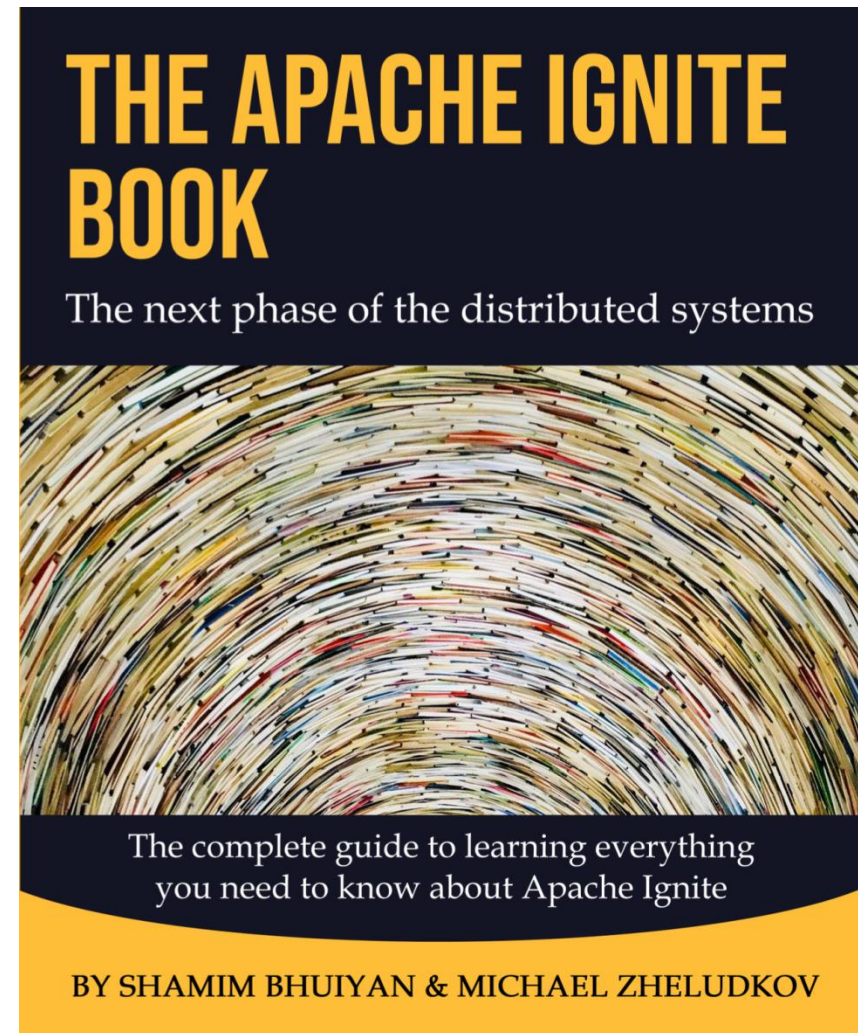
# Apache Ignite

Eine Key-Value-Datenbank

- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
  - LPGL Lizenzmodell
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
  - <https://apacheignite.readme.io/docs/what-is-ignite>



The screenshot shows the Apache Ignite website. At the top, there's a navigation bar with the Apache Ignite logo, version 'v2.7.6', a 'Home' link, and a search bar containing 'What is Ignite?'. Below the navigation bar, the main heading is 'What is Ignite?' with a 'SUGGEST EDITS' link. Underneath, it describes Ignite as a 'Memory-Centric Database, Caching, and Processing Platform'. A light blue box contains the text: 'Apache Ignite is ... a memory-centric distributed **database**, **caching**, and **processing** platform for transactional, analytical, and streaming workloads delivering in-memory speeds at petabyte scale'. Below this, there's a dark grey bar with six icons representing different industries: Financial Services, Telco, Travel & Logistics, E-Commerce, Pharma & Healthcare, and IoT. Underneath this bar is a row of red buttons labeled: SQL, Key/Value, Transactions, Compute, Services, Streaming, and ML. Below the buttons is a large red bar with the text 'Memory-Centric Storage'. At the bottom, there are two boxes: 'Ignite Native Persistence (Flash, SSD, Intel 3D XPoint)' and 'Third-Party Persistence (RDBMS, HDFS, NoSQL)'.



© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

**Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.**

NoSQL: Eine Kurzeinführung	6
Apache Ignite: Eine Übersicht	41
Einsatzbereiche von Apache Ignite	57
Das memory Grid	69
Anwendungen	89
Produktionsumgebung	102

# 1

## **NOSQL: EINE KURZEINFÜHRUNG**

1.1

## **BIG & FAST DATA**

- Definitionsversuche:
  - Physikalisch
    - Terabyte
  - System
    - Übersteigt den Bedarf eines normalen Speichermediums
  - Technisch
    - Datenverarbeitung beginnt, auf Grund der Größe Probleme zu bereiten
- Allerdings:
  - Keine wirklich eindeutige Definition erkennbar
  - Grenzen können sich verschieben



- Völlig verschiedene Bereiche:
  - DNS-Server für Internet
  - Indizierung von Web Seiten für Suchmaschinen
    - Google
  - Monitoring von Seitenzugriffen, aber auch Metrik-Informationen in großen Server-Systemen
  - Content Management und Waren-Systeme
    - Amazon
  - Social Media
    - Facebook

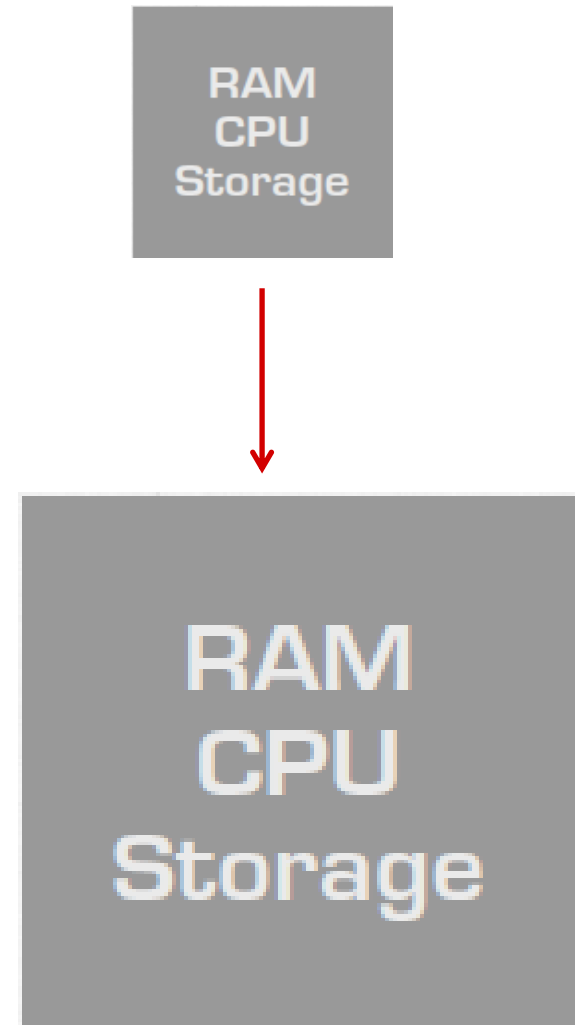
1.2

## **CLUSTERING**

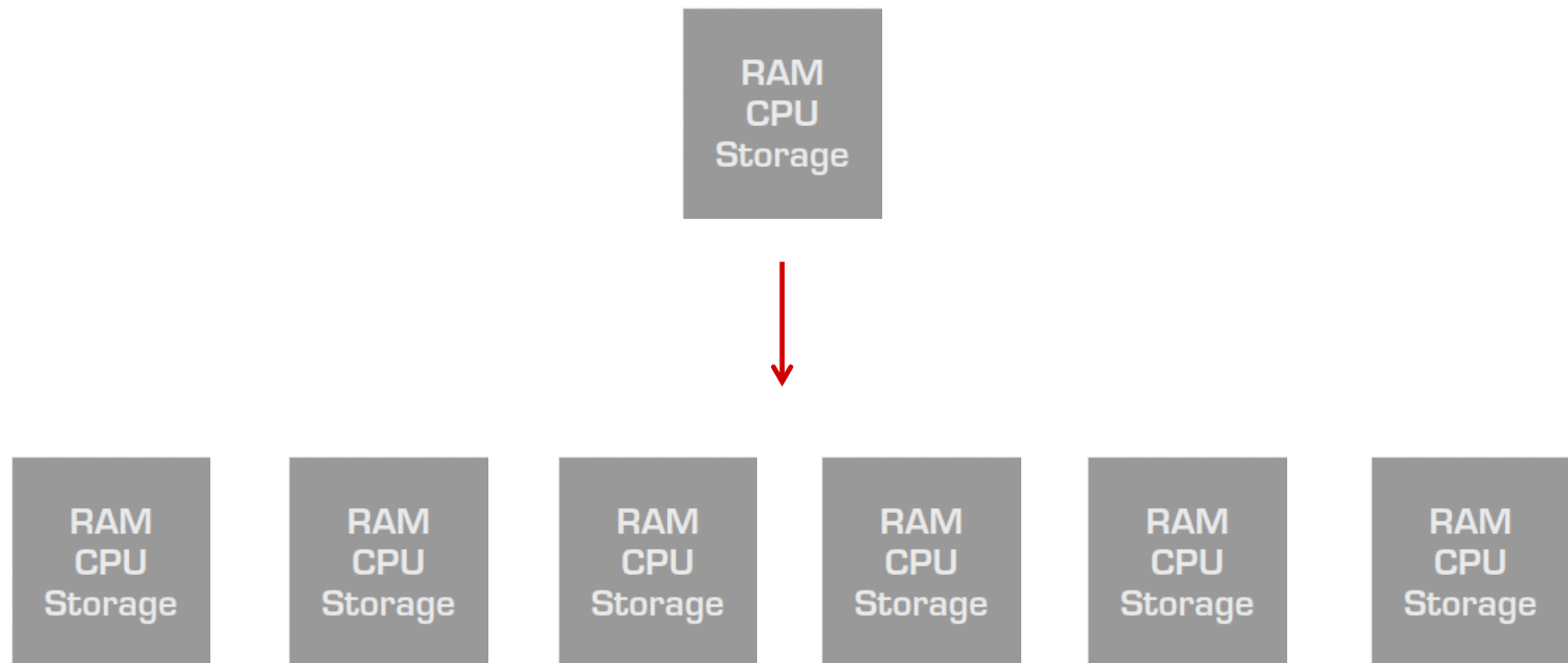
- Bei einem Cluster werden die Anwendungen auf identischen Servern betrieben
  - Betriebssystem
  - Laufzeitumgebungen
  - jeweils identische Versionen
- Ein Grid realisiert Anwendungen auf potenziell inhomogenen Rechnern
- Die Unterscheidung zwischen Grid und Cluster ist häufig fließend und deshalb nicht wirklich relevant
  - Im Folgenden wird einheitlich von Clustern gesprochen

- Zur Ablage und Auswertung von Daten sind selbst mächtige Server-Maschinen auf Dauer überfordert
- Die Datenhaltung wird deshalb auf mehrere Server verteilt
- Die Lastverteilung übernimmt ein simpler Load Balancer oder ein komplexer "Master"
  - Aus Client-Sicht heraus tritt der Cluster damit immer noch als eine Einheit auf
- Im Optimum ist der Cluster damit eine quasi unendliche Datensenke mit beliebiger skalierbarer Rechnerkapazität
  - Damit eine Vorstufe zur "Cloud"

- Grenzen sind klar definiert
  - Kosten
  - Aktuelle Hardware-Grenzen
  - Anforderungen an Ausfallsicherheit



- Grenzen
  - Ausfallsicherheit fordert Replikation über Netzwerk
  - System-Administration und Überwachung



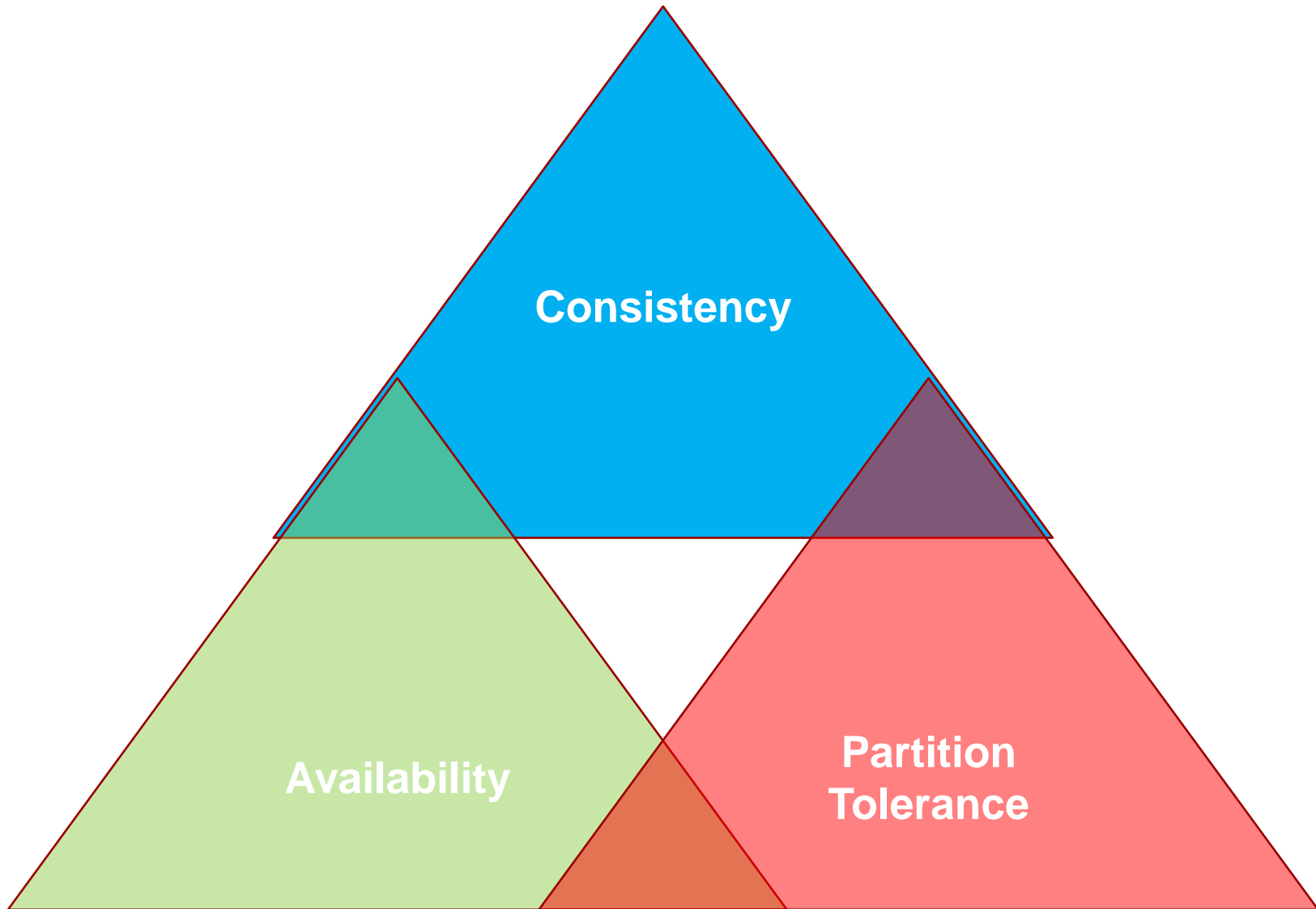
1.3

## **DAS CAP-THEOREM**

- Consistency
  - Alle Knoten haben jederzeit den selben Informationsstand
- Availability
  - Jeder Client bekommt garantiert Antwort
    - Solange zumindest ein Knoten im Cluster läuft
- Partition Tolerance
  - Das System toleriert
    - Den Ausfall von Knoten
    - Den Ausfall des Netzwerks zwischen Knoten



# Keine gemeinsame Schnittmenge!

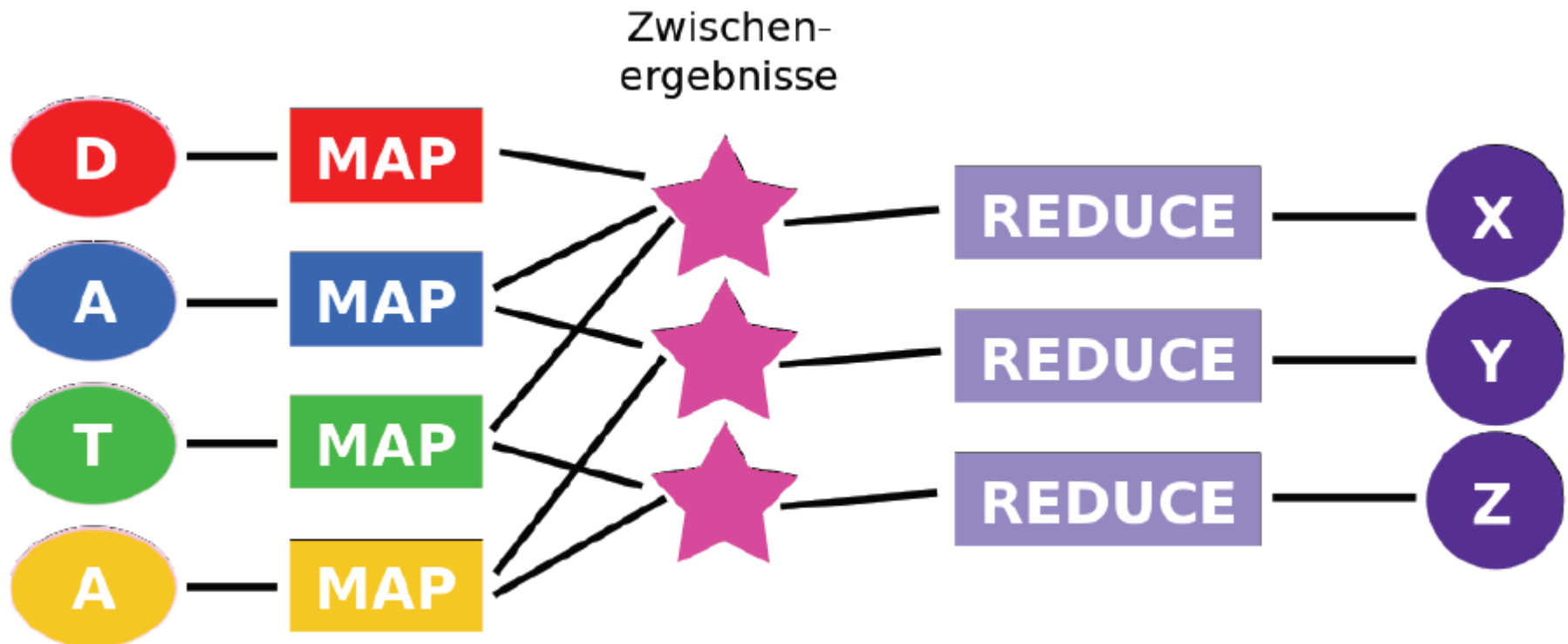


1.4

## **ANALYSE VON BIG DATA**

- Verlangt „neue“ Algorithmen
  - Aufteilung in Jobs
  - Diese werden parallelisiert
  - Operieren auf einem verteilten Datenbestand
- Alle genannten Unternehmen haben in den letzten 10 Jahren Algorithmen entwickelt bzw. eingesetzt
  - Und „netterweise“ der Open Source Community zur Verfügung gestellt
- Fast Data
  - Schneller Zugriff auf Daten-Selektionen
  - Schnelle Umsetzung neuer Abfrage-Anforderungen

- Bahnbrechende Entwicklung
  - Idee wird Google zugeschrieben
- Grundsätzliche Arbeitsweise
  - Operiert auf Mengen (Maps)
  - In einem ersten Schritt werden die Daten der Map in einer neuen Map aufbereitet
  - Diese wird in einem oder mehreren Reduce-Schritten zur Ergebnis-Struktur zusammengefasst



Quelle: <http://de.wikipedia.org>

1.5

# TRANSAKTIONALITÄT

- Atomic, Consistent, Isolated, Durable
- Features
  - Starke Konsistenz der Daten
  - Transaktionssteuerung
    - Transaction-Isolation
    - Bei Bedarf Zwei-Phasen-Commit
  - Relativ komplexe Entwicklung
- Relationale Datenbanken fokussieren auf ACID-Transaktionen

- Basically Available, Soft State, Eventual Consistency
  - Consistency
    - Alle Knoten haben **jederzeit** den selben Informationsstand
  - Eventually Consistent
    - Änderungen des Datenbestandes werden zeitlich versetzt an die anderen Knoten propagiert
    - "Eventual" nicht mit "eventuell" übersetzen!
- Features
  - Letzliche Konsistenz
  - Hohe Verfügbarkeit
  - Schnelligkeit
    - Bei Bedarf "Fire-and-forget"
  - Leichtere Entwicklung



1.6

## **WAS IST NOSQL?**

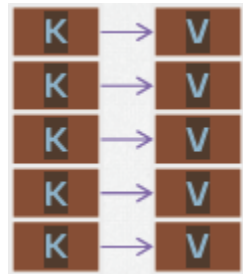
- No SQL!
  - Ursprüngliche Bedeutung
    - Etwas dogmatische Ablehnung relationaler Datenbank-Prinzipien
- Not only SQL
  - Schon deutlich abgeschwächt
  - Reduktion auf geeignete Problemstellungen
- No/ Not only relational
  - Der eigentlich richtige Begriff
  - Fokussierung auf die Daten-Modellierung, nicht auf die Abfragesprache

1.7

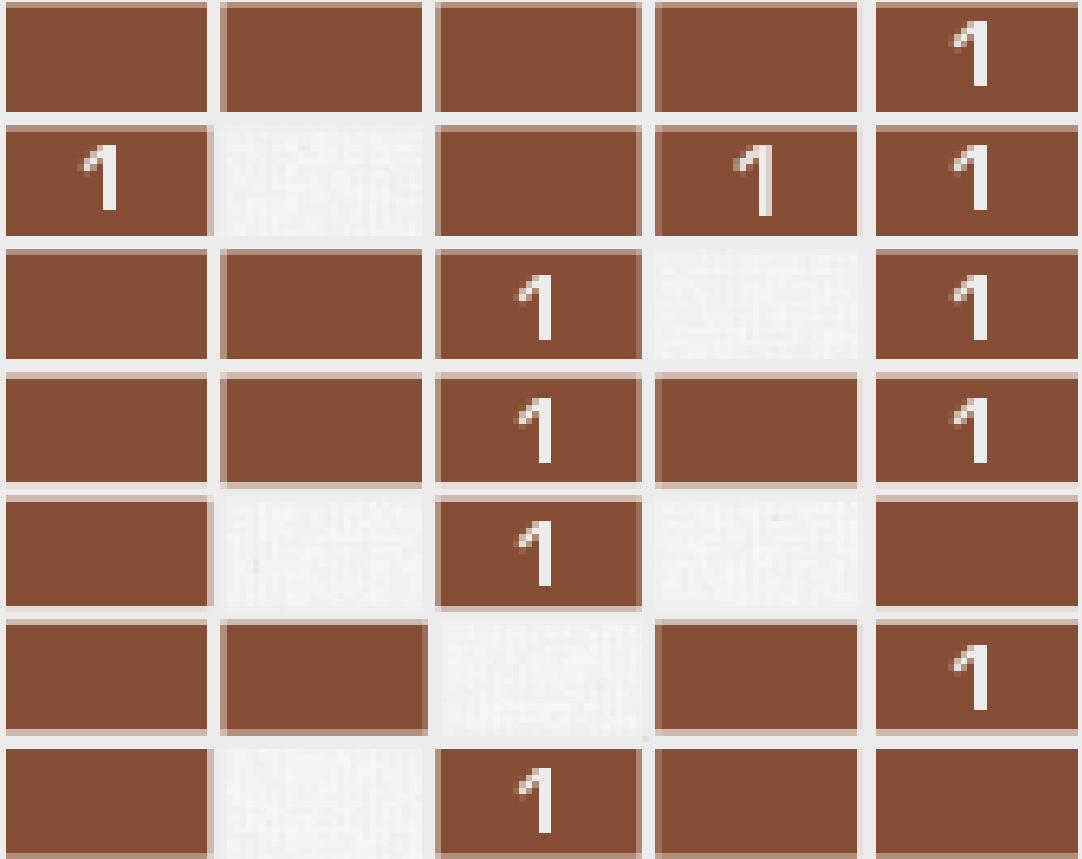
## **KLASSIFIZIERUNG**

- Key/Value
- Column-orientiert
- Dokumenten-orientiert
- Graphen-orientiert

- Ablage von Inhalten (=Value) unter einem eindeutigen Schlüssel (key)
  - Flache Struktur
  - Values werden als Byte-Array betrachtet
- Teilweise Verzicht auf Persistierung der Daten
  - Cache-Systeme
- Auslegung auf eine Vielzahl konkurrierender Zugriffe



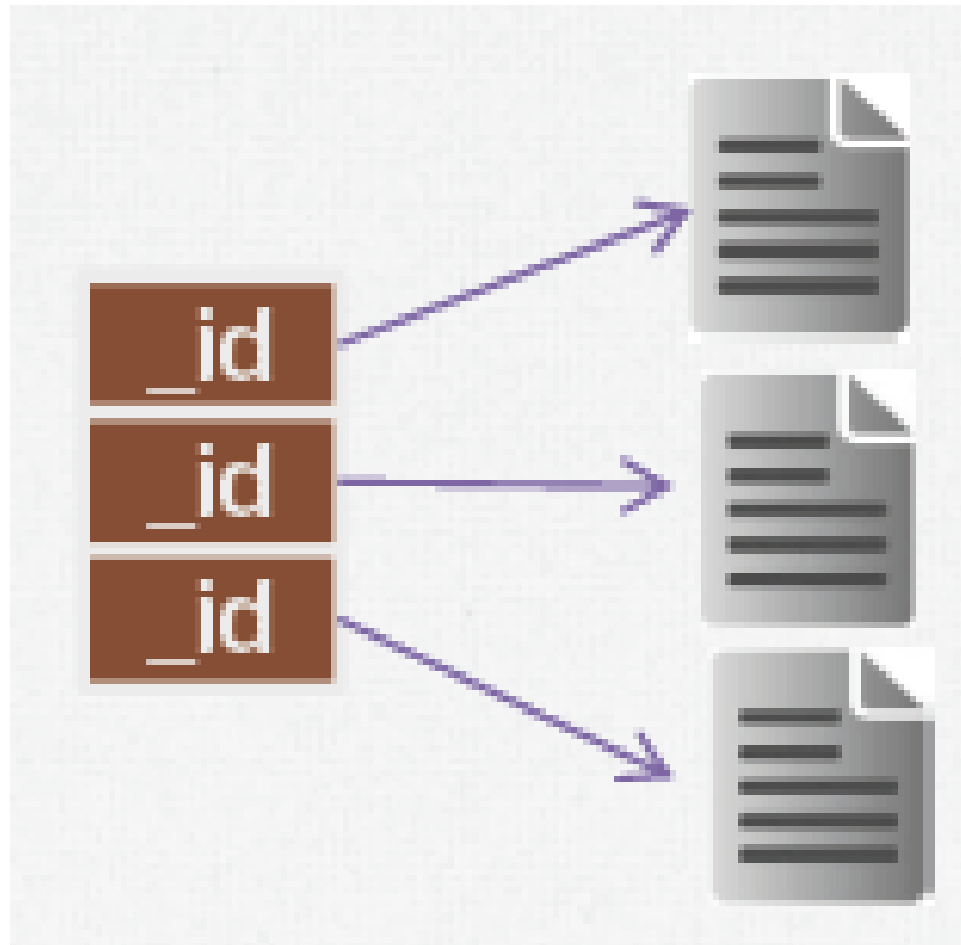
- Daten werden in Columns abgelegt
  - Eine Zeile besteht aber nicht zwangsläufig aus allen Columns
- Diese Columns können noch in Gruppen unterteilt werden
  - Vorteilhaft bei einer verteilten Datenhaltung
- Ausrichtung auf wirklich große Datenmengen



				1
1			1	1
		1		1
		1		1
		1		
				1
		1		



- Ablage der Daten in strukturierten Dokumenten
  - Nicht unbedingt XML!
  - Wesentlich weiter verbreitet ist JSON
    - Bzw. das binäre BSON



- Ablage der Daten in
  - Knoten
  - Beziehungen
  - Attribute
- Knoten und Beziehungen haben Attribute
- Knoten sind mit Beziehungen verknüpft und umgekehrt
- „Whiteboard-Friendly“
  - „Was Sie auf einem Whiteboard zeichnen können, können Sie in einer Graph-DB ablegen.“



1.8

## DATENMODELLIERUNG

- Daten werden abstrahiert modelliert
  - Klassenorientierte Programmierung
    - Java, C++/C#, ...
  - Tabellen-Schemata
  - XML-Schema
- Eine Umwandlung von Datentypen erfordert ein Umkopieren
  - „Migration“
- Damit sind Entwicklungszyklen und die Zeiten für die Einführung geänderter Features relativ lang

- „Duck Typing“: Nur vorhandene Eigenschaften und implementiertes Verhalten zählen
  - „Was gelb ist, quakt und watschelt ist eine Ente“
- Dies ist vorwiegend die Domäne von dynamischen Skript-Sprachen
  - Ruby, Python, JavaScript
- Ein Umkopieren von Daten ist nur noch in Ausnahmefällen notwendig
- Damit wird eine agile Software-Entwicklung unterstützt

- Was ist...
  - Besser?
  - Richtig?
  - Komfortabel?
- Leider keine pauschale Antwort möglich
  - „Wer lebend den Raum verlässt hat gewonnen“
- Aber:
  - Der Trend der letzten Jahre geht immer mehr Richtung Dynamik



## 2

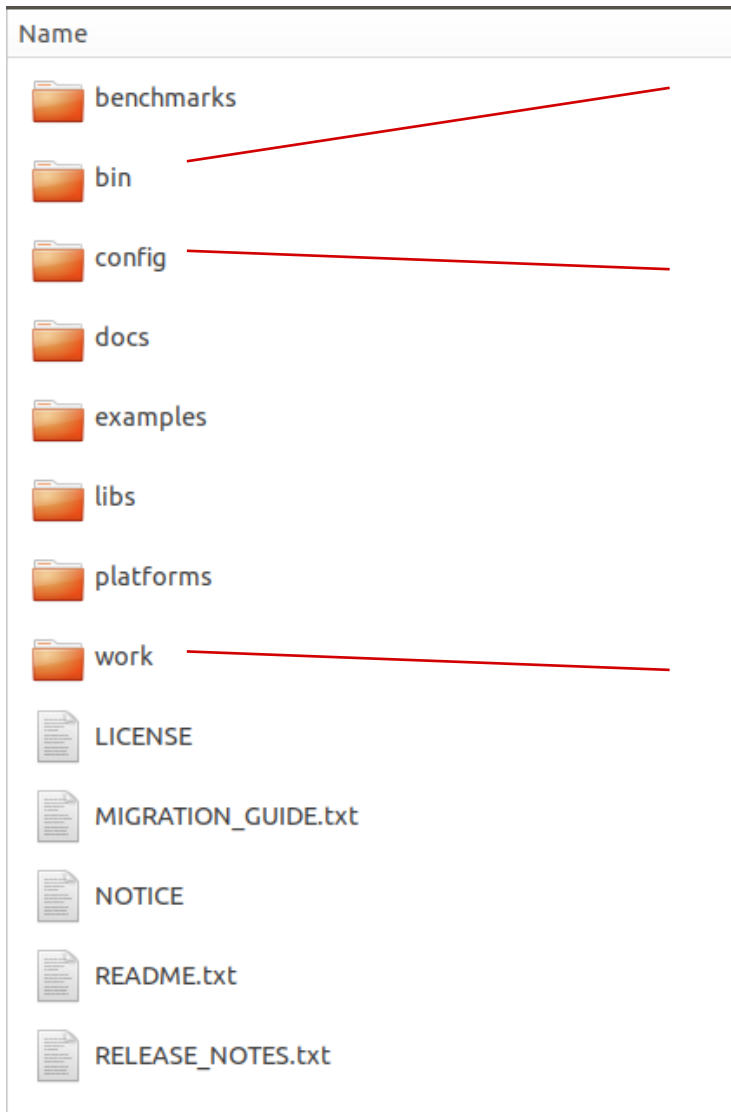
# APACHE IGNITE: EINE ÜBERSICHT

2.1

## **INSTALLATION**

- Läuft damit in einem Java Prozess
  - Die Java Virtual Machine
  - Vor der Installation von Apache Ignite ist folglich Java zu installieren
- Unterstützte stabile Java Version ist 8
  - Aktuelle Java-Version ist 13, allerdings hat Java 8 immer noch Long Time Support und ist in vielen Unternehmen die gesetzte Plattform

- Archiv von den Download-Seiten der Apache Community
  - <https://ignite.apache.org/download.cgi>
- Docker Image vom Docker Hub
  - <https://hub.docker.com/r/apacheignite/ignite>



Skripte zum Ausführen des Servers  
sowie weiterer Utilities

Konfigurationsdateien

Temporäre Dateien, insbesondere  
Logs

- GridGain ist die kommerzielle Ignite-Version
  - Implementierung zusätzlicher Features insbesondere im Bereich Security und Management
  - Support
- <https://www.gridgain.com/products/gridgain-vs-ignite>

2.2

**START**

- **Check der Java-Version**
  - `java -version`
    - `openjdk version "1.8.0_222"`
- **Starten eines einzelnen Ignite-Knoten mit einer Default-Konfiguration**
  - `ignite.sh`

```

[15:12:03]
[15:12:03]
[15:12:03]
[15:12:03]
[15:12:03]
[15:12:03] ver. 2.7.6#20190911-sha1:21f7ca41
[15:12:03] 2019 Copyright(C) Apache Software Foundation
[15:12:03]
[15:12:03] Ignite documentation: http://ignite.apache.org
[15:12:03]
[15:12:03] Quiet mode.
[15:12:03] ^-- Logging to file '/home/rainer/tools/apache_ignite/apache-ignite-2.7.6-bin/work/log/ignite-31a54e15.0.log'
[15:12:03] ^-- Logging by 'JavaLogger [quiet=true, config=null]'
[15:12:03] ^-- To see **FULL** console log here add -DIGNITE_QUIET=false or "-v" to ignite.{sh|bat}
[15:12:03]
[15:12:03] OS: Linux 4.15.0-70-generic amd64
[15:12:03] VM information: Java(TM) SE Runtime Environment 10.0.2+13 "Oracle Corporation" Java HotSpot(TM) 64-Bit Server VM 10.0
.2+13
[15:12:03] Please set system property '-Djava.net.preferIPv4Stack=true' to avoid possible problems in mixed environments.
[15:12:03] Initial heap size is 124MB (should be no less than 512MB, use -Xms512m -Xmx512m).

```



- Sehr ausgefeilte Laufzeitumgebung mit vielen Optimierungen
  - Just in Time-Compiler
  - HotSpot
- Plattformunabhängig
  - Linux
  - Windows
  - zOS
  - ...
- Große Menge von etablierten (Open Source-) Bibliotheken
  - Apache Ignite ist ein modular aufgebautes Spring-Projekt
- Detaillierte Bereitstellung von Metriken zur Beurteilung des Java-Prozesses
  - CPU
  - Memory
  - Threading

- Die automatische Speicherbereinigung der JVM bereinigt und defragmentiert den Heap-Speicher des Prozesses nach Bedarf
- Allerdings führt diese Garbage Collection insbesondere bei großen Speichergrößen zu unangenehmen Nebeneffekten
  - Kurzfristige Spitzen im CPU-Bedarf
  - Stop-the-World-Effekt
    - das heißt, die JVM ist in den Zeiten der Collection nicht erreichbar
    - Dauer des Effekts kann in ungünstigen/worst-case Server-Konfigurationen durchaus im Sekundenbereich liegen

- Apache Ignite ist Memory-zentriert und benötigt damit beträchtlich Speicher
  - Damit scheint Ignite gerade ein Produkt darzustellen, für das Java nicht gemacht ist
- Aber:
  - Die Garbage Collection ist ja nicht per se langsam
    - Moderne Java-Versionen optimieren diese stets weiter
  - Ein Ignite-Datenbanksystem ist stets als Cluster aufzufassen
    - Ein einzelner Knoten braucht damit gar nicht so viel Heap-Speicher
  - Java-Programme können auch Off-Heap-Memory benutzen
    - Das macht Ignite bereits in der Standard-Konfiguration
    - `Topology snapshot [ver=1, locNode=f8ee687c, servers=1, clients=0, state=ACTIVE, CPUs=4, offheap=1.5GB, heap=1.7GB`

## Off-Heap Memory in Java

📅 2. DEZEMBER 2019 👤 DR. RAINER SAWITZKI 📁 ALLGEMEIN

Die Java Virtual Machine organisiert und verwaltet die Objekte einer Anwendung selbst im so genannten Heap-Speicher. Diese wurde im Beitrag über [Referenzen und Objekte](#) beschrieben. Ein automatischer Hintergrundprozess, die [Garbage Collection](#), entfernt automatisch nicht mehr referenzierbare Objekte und bereinigt somit den Speicher.

“Native Memory” bzw Off-Heap Memory in Java ist jedoch ebenfalls möglich!

### Was ist “Off-Heap Memory”?

Das Off-Heap Memory ist, wie der Name auch sagt, außerhalb der des Heaps angesiedelt und wird deshalb nicht von der Garbage Collection bereinigt. Weiterhin ist es für einen Programmierer nicht möglich, Objekte direkt im Off-Heap Memory zu instanzieren. Damit ist die Benutzung deutlich komplexer und auch mit mehr Fehlerpotenzial behaftet als das Referenzen-Modell.

2.3

## **WEITERE WERKZEUGE**

- control
  - Überwachung und Kontrolle eines Clusters
  - <https://apacheignite-tools.readme.io/docs/control-script>
- igniterouter
  - Werkzeug für Thin Clients
  - In der aktuellen Version nicht benötigt
- ignite-tf.sh
  - Machine Learning mit Tensor Flow
- ignitevisorcmd
  - Konsolen-basiertes Management und Überwachung von Ignite
  - <https://apacheignite-tools.readme.io/docs/command-line-interface>
- sqlline
  - Ausführen von SQL-Befehlen in Ignite
  - <https://apacheignite-sql.readme.io/docs/sqlline>

- Separate Web Applikation zur Verwaltung des Ignite-Clusters
- Zusätzliche Installation eines eigenen Servers
  - <https://apacheignite-tools.readme.io/docs/ignite-web-console>

Connection Window Help

pid: 27053 org.apache.ignite.startup.cmdline.CommandLineStartup config/default-config.xml

Overview Memory Threads Classes VM Summary MBeans

▶ JMXImplementation  
 ▶ com.sun.management  
 ▶ java.lang  
 ▶ java.nio  
 ▶ java.util.logging  
 ▶ org.apache
 

- ▼ 764c12b6
  - ▶ "Thread Pools"
  - ▶ Clients
  - ▶ DataRegionMetrics
  - ▶ DataStorage
  - ▼ Kernal
    - ▶ ClusterLocalNodeMetricsMxBeanImpl
    - ▶ ClusterMetricsMxBeanImpl
    - ▶ FailureHandlingMxBeanImpl
    - ▼ IgniteKernal
      - ▶ **Attributes**
      - ▶ Operations
      - ▶ Ignition
      - ▶ SPIs
      - ▶ TransactionMetrics
      - ▶ Transactions

Attribute values

Name	Value
CheckpointSpiFormatted	[NoopCheckpointSpi []]
CollisionSpiFormatted	NoopCollisionSpi []
CommunicationSpiFormatted	TcpCommunicationSpi [connectGate...
Copyright	2019 Copyright(C) Apache Software F...
CurrentCoordinatorFormatted	[0:0:0:0:0:0:1%lo, 127.0.0.1, 172.16.1...
DeploymentSpiFormatted	LocalDeploymentSpi []
DiscoverySpiFormatted	TcpDiscoverySpi [addrRslvr=null, sock...
EventStorageSpiFormatted	org.apache.ignite.spi.eventstorage.N...
ExecutorServiceFormatted	8
FailoverSpiFormatted	[AlwaysFailoverSpi [maxFailoverAtte...
FullVersion	2.7.6-20190911
GridLoggerFormatted	GridLoggerProxy [igniteInstanceNam...
IgniteHome	/home/rainer/tools/apache_ignite/a...
InstanceName	
JdkInformation	OpenJDK Runtime Environment 1.8.0...
LifecycleBeansFormatted	[]
LoadBalancingSpiFormatted	[RoundRobinLoadBalancingSpi [balan...
LocalNodeId	f8ee687c-e576-4c2c-ac64-082541647...
LongJVMPauseLastEvents	<b>{1575443569234=54758209, 15754...</b>
LongJVMPausesCount	<b>4</b>
LongJVMPausesTotalDuration	<b>62067487</b>
MBeanServerFormatted	com.sun.jmx.mbeanserver.JmxMBea...
NodeInBaseline	false
OsInformation	Linux 4.15.0-70-generic amd64
OsUser	rainer
PeerClassLoadingEnabled	false
RebalanceEnabled	true
StartTimestamp	<b>1575383368774</b>

Refresh

pid: 27053 org.apache.ignite.startup.cmdline.CommandLin...



3

## **EINSATZBEREICHE VON APACHE IGNITE**

3.1

## **KEY-VALUE DATA GRID**

- Apache Ignite ist ein Key-Value Store
  - Darauf ist die Architektur des Datenbanksystems primär ausgerichtet
- Ein simpler Key-Value-Store kennt nur eine einzige Abfrage:
  - `select value from store where key = :key`

- JCache ist eine Java-Spezifikation von Oracle
  - JSR 107
- Features
  - Basic Cache Operationen
    - CRUD
  - Entry Processor
    - Der Cache kann Einträge selbst logisch aufbauen
    - Insbesondere sinnvoll, um bei Aktualisierungen nur die Änderungen übertragen zu müssen
  - Concurrent Map
    - Ein sehr elegantes Möglichkeit, innerhalb eines Programms den verteilten Cache zu programmieren
  - Events und Metriken
    - Jede Interaktion mit dem Cache wird durch Events signalisiert
    - Metriken geben einen detaillierten Einblick in das laufende System

- Scan-Queries
  - Im Wesentlichen die Übertragung eines Predicates, das von den Cluster-Knoten ausgeführt wird
- Text Queries
  - Ignite kann Cache-Einträge indizieren
    - Die Index-Einträge werden intern natürlich ebenfalls im Store abgelegt
  - Als Index-Engine wird Apache Lucene benutzt
- SQL-Queries
  - Auf Basis des Store realisiert Ignite eine SQL-konforme Datenbank
  - Distributed SQL

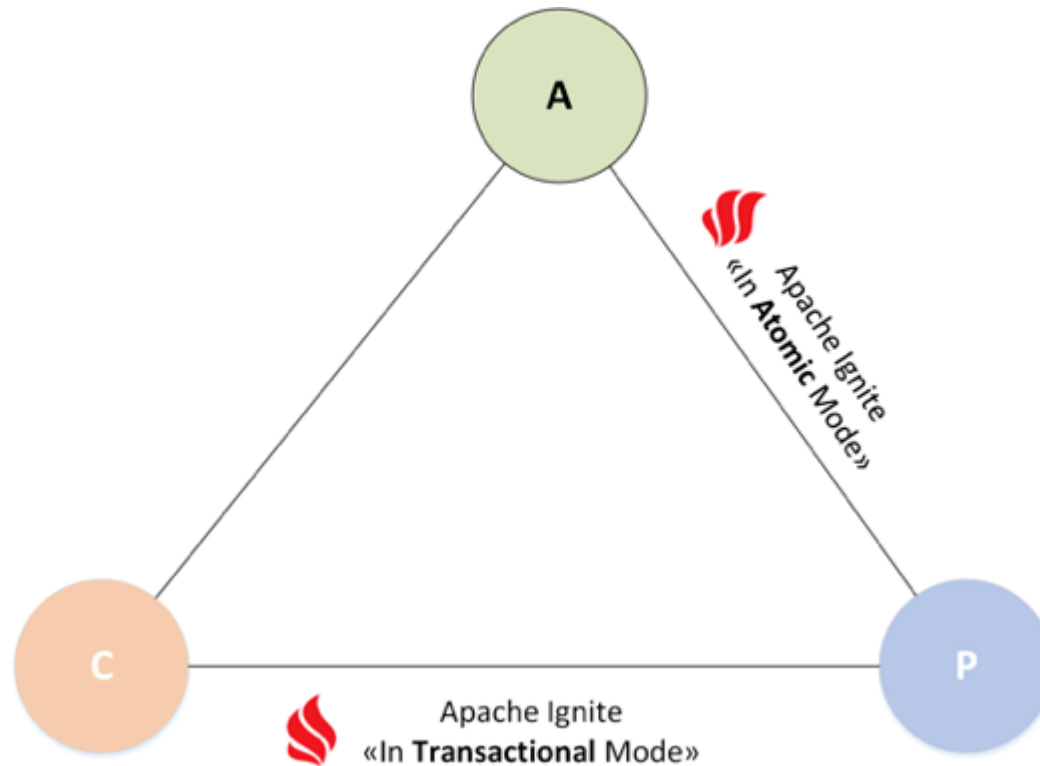
- Zusammengehörige Cache-Einträge werden möglichst jeweils auf gleichen Knoten gehalten
  - Sehr sinnvolle Optimierung für Abfragen
- Die Collocation wird über "Affinity Keys" definiert

3.2

## **TRANSAKTIONELLE SYSTEME**

- Aus Sicht des CAP-Theorems kann Ignite als AP oder CP konfiguriert werden
  - Ignite kann Daten über den verteilten Cache konsistent halten oder
  - die Verfügbarkeit garantieren
- Der Unterschied wird durch die Konfiguration des Atomicity Modes gesteuert

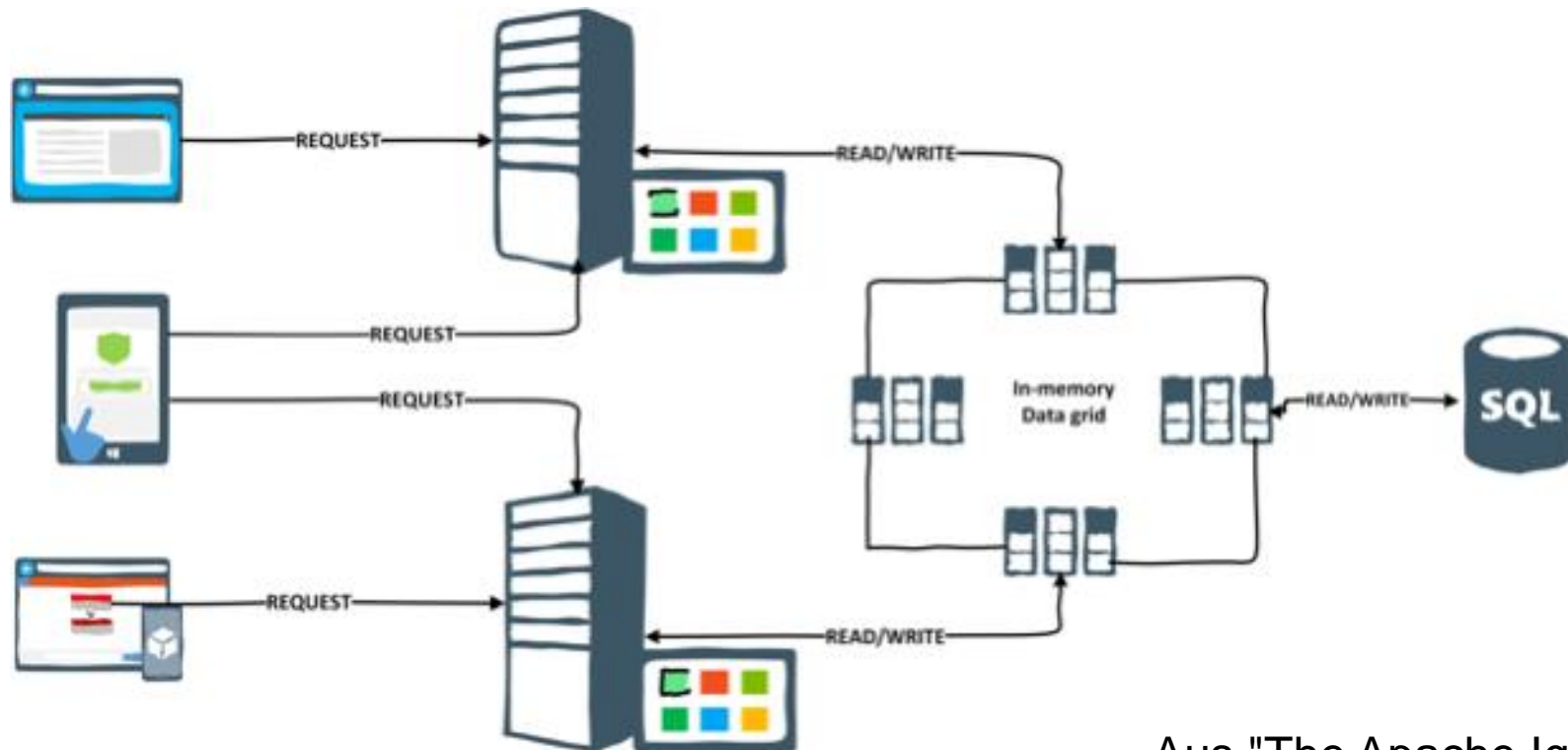




Aus "The Apache Ignite Book"

- Ignite unterstützt explizit Locks, falls Atomic Transactions benutzt werden
- Gesperrt werden können
  - Einzelne Einträge
  - Hierarchien
  - Komplette Caches
- Innerhalb einer laufenden Transaktion kann ein Lock nicht angefordert werden
  - Mit "Pessimistic Concurrency" lockt eine Transaktion jedoch automatisch die betroffenen Einträge

- Ignite dient hier als effizienter Puffer vor einem klassischen Datenbanksystem



Aus "The Apache Ignite Book"

- Ignite kann als Bestandteil einer komplexen Architektur benutzt werden, weitere Anforderungen zu realisieren
  - HTAP: Hybrid Transaction/Analytical Processing
  - Fast Data Processing
  - Lambda-Architektur
  - Machine Learning

4

## **DAS MEMORY GRID**

4.1

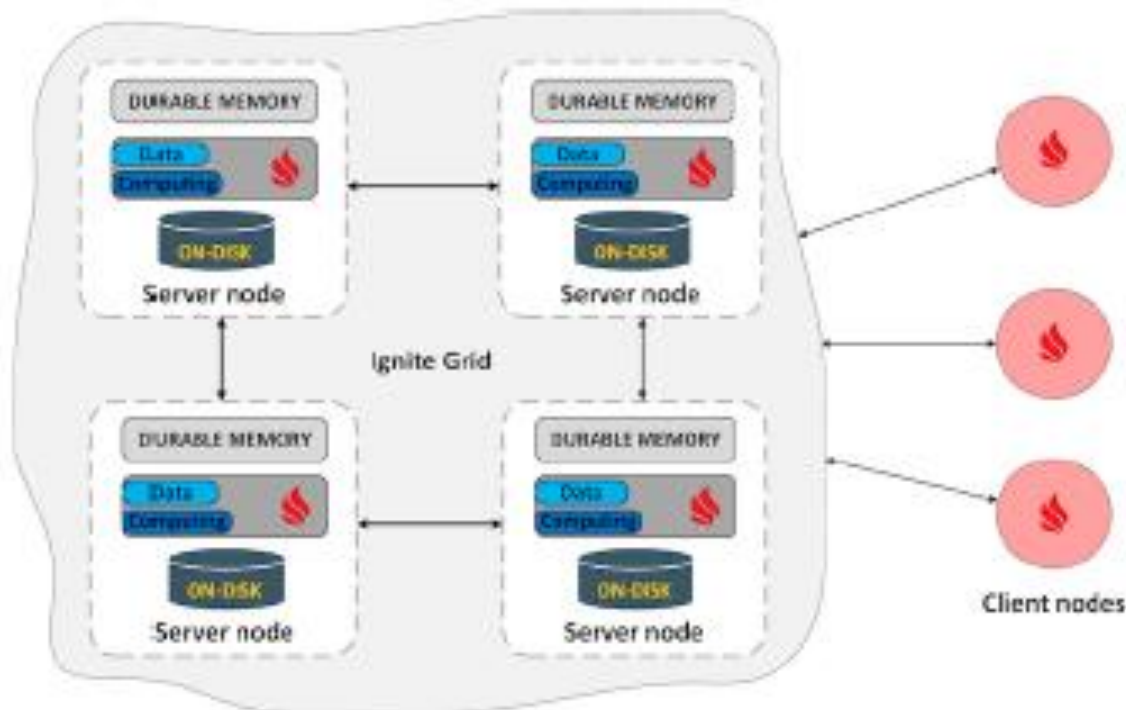
## FEATURES

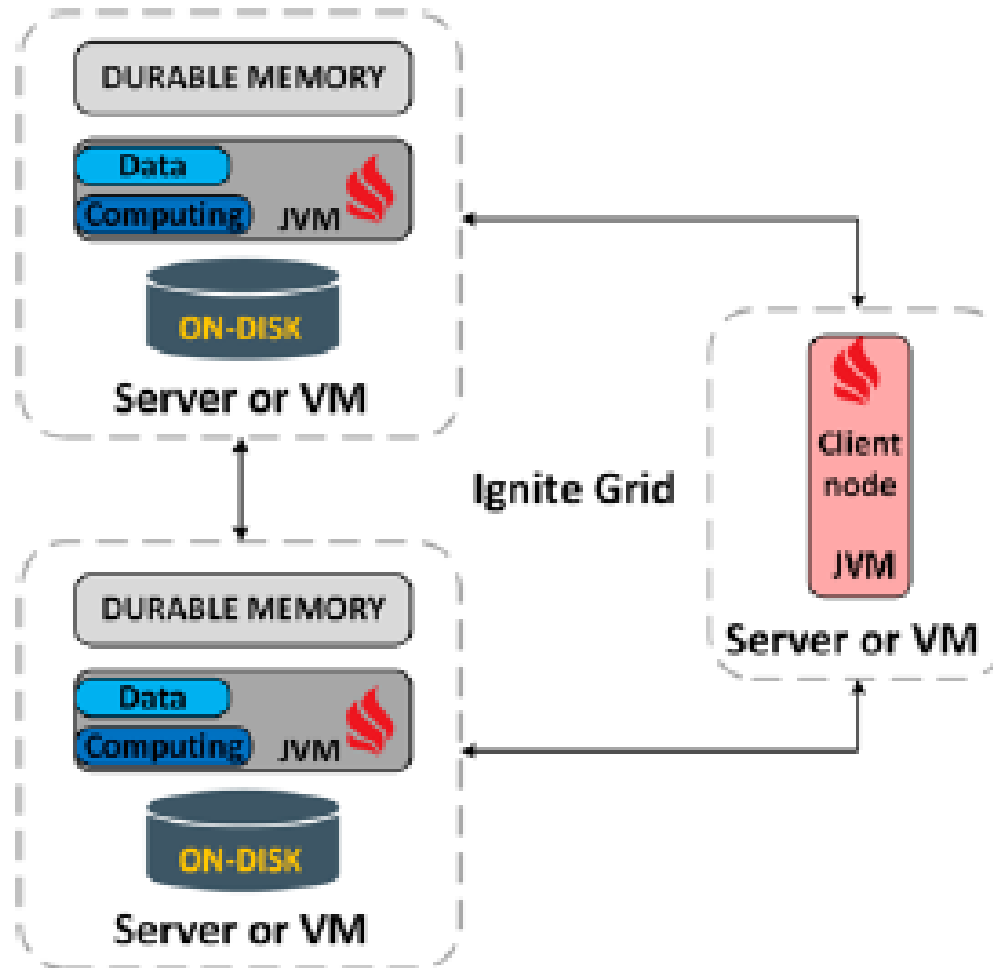
API	Java, .Net, C++, Rest, SQL, Python, Memcache
Drivers	JDBC, ODBC, Thin
Data structure	Queue, Set, Atomic types, Semaphore
Caching	In-cache (write behind) L2 COHING (Hibernate, MyBatis) Web session clustering
Computing	Compute task, in-memory Map/Reduce, distributed closure, Fork-join Service task
Partitioning	Partitioned Replicated
Engines	SQL (H2) Full text search (Lucene) Servlet container (jetty server)
Storage	On-heap Off-heap Disk
Streaming	Ignite data streamer, Kafka, Storm, Camel, Flume, MQTT
Bigdata	In-memory Map/Reduce, IGFS, Hadoop FS cache, Spark (RDD, Data frame)
Management	Visor Web console JMX
Networking	IPv4, IPv6
Deployment	JAR (JBK 8.9 Vendors, Oracle JDK, OpenJDK, IBM JDK) OS (Linux, Windows, MacOS, Oracle Solaris) Cluster deployment (AWS, Vmware, Docker, Mesos, Kubernetes, On premise, Google cloud)

- Der gesamte Cluster besteht aus gleichberechtigten Knoten
- Knoten können ohne den Betrieb zu unterbrechen dynamisch hinzugefügt oder gestoppt werden



- Server-Knoten enthalten die Daten und führen Berechnungen durch
- Ein Client-Knoten ist Bestandteil der Cluster-Topologie
  - Wird aber nur als lokaler Client genutzt, der mit den Server-Knoten kommuniziert





- Organisiert Daten in einzelnen Caches
  - Vergleichbar mit einer simplen Tabelle
    - Ignite ist ein Key-Value-Store...
- Zwei Modi pro Cache:
  - PARTITIONED
    - mit konfigurierbarer Anzahl von Backups
  - REPLICATED
    - Daten werden auf allen Knoten gehalten

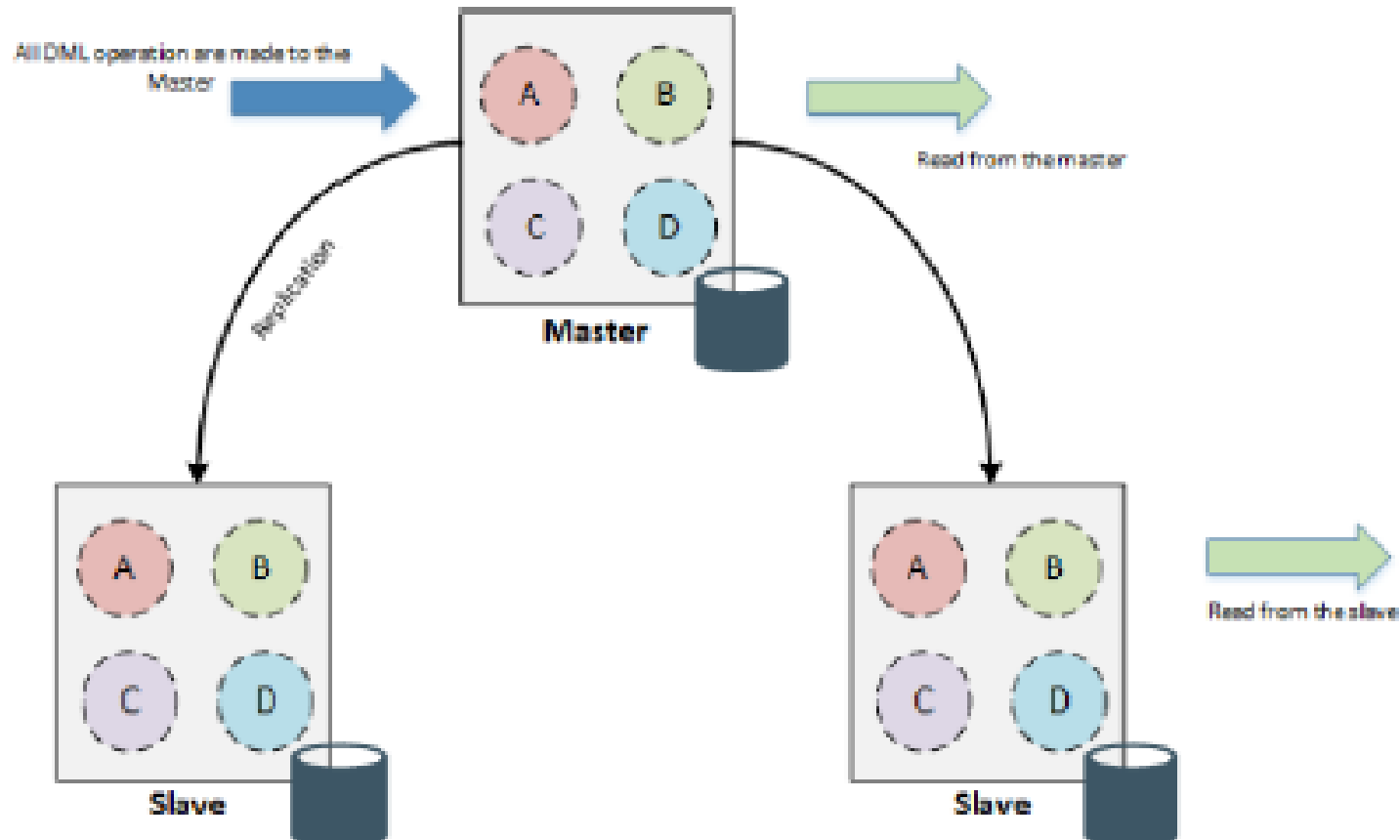
4.2

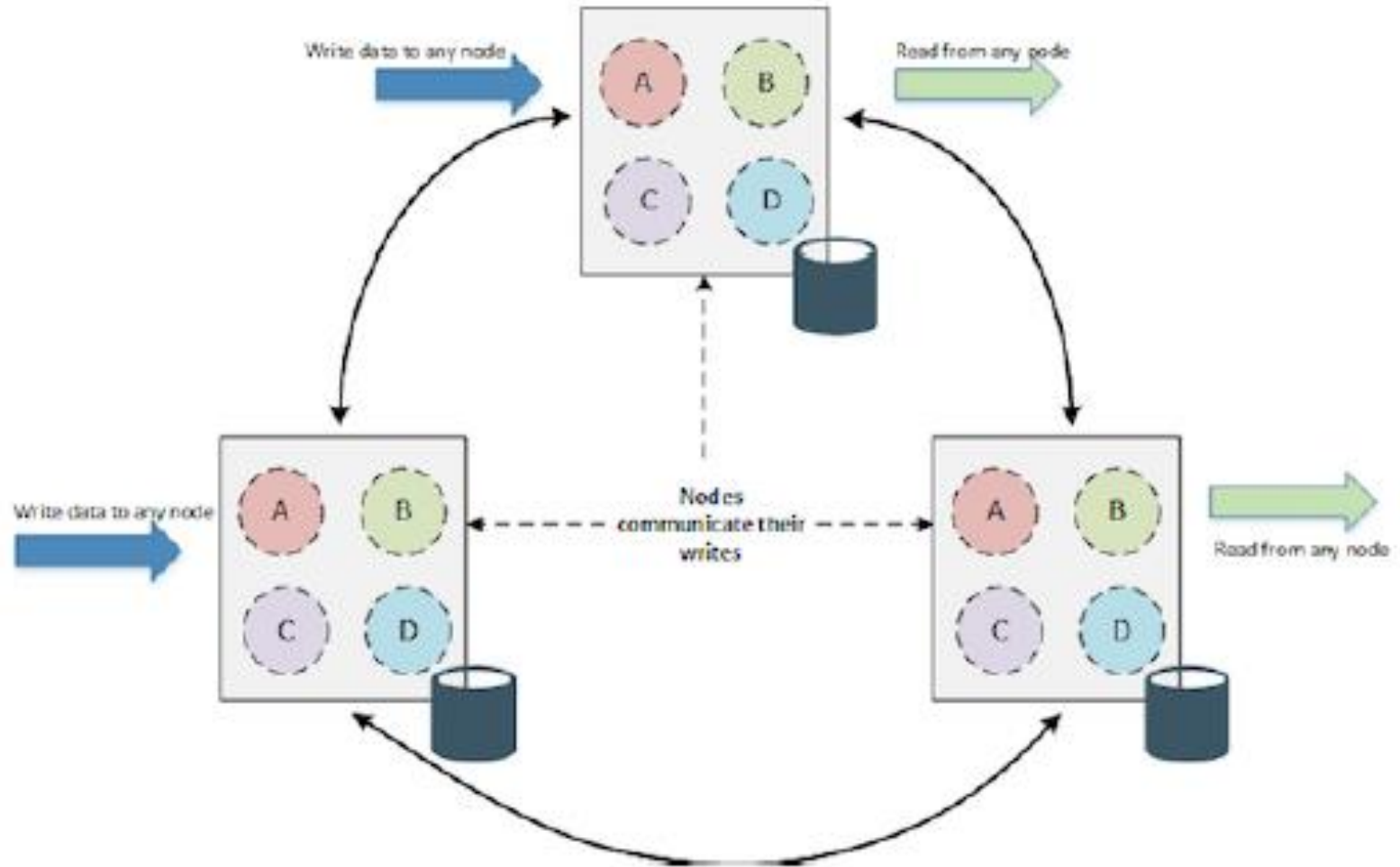
## **IN MEMORY DATENHALTUNG**

- Daten werden auf Grund nach einem Schlüssel gruppiert auf den Knoten des Clusters verteilt
- In anderen Datenbank-Systemen häufig auch Sharding genannt
  - MongoDB
  - Postgres
  - ...
- Notwendig hierfür ist die Distributed Hash Table
  - Diese ordnet den einzelnen Hash-Werten Knoten zu
  - Die Berechnung der Hash-Werte ist mit verschiedenen Algorithmen lösbar
    - Consistent Hashing
    - Rendezvous Hashing
  - Details siehe [https://en.wikipedia.org/wiki/Rendezvous\\_hashing](https://en.wikipedia.org/wiki/Rendezvous_hashing)

- Zur Datensicherheit werden Daten zwischen verschiedenen Knoten repliziert
- Zwei Ansätze
  - Master-Slave
  - Peer-to-Peer
- Hinweis
  - Ignite unterstützt nur Peer-to-Peer

# Master-Slave Replication







4.3

## **PERSISTENZ**

- Objekte im Heap-Speicher der JVM
  - Die schnellste Variante
- Datenstrukturen im Off-heap-Memory
  - Keine Störung durch die Garbage Collection
- In-Memory + 3rd party database
  - Klassische Cache-Architektur
- Ignite Persistence
  - Schreiben eines Write Ahead Logs (WAL) im Dateisystem

- Cache a-side
  - Der Client orchestriert den Cache
  - Daten, die der Cache nicht liefern kann werden vom Client aus der Datenbank gelesen und im Cache abgelegt
- Read through and Write through
  - Der Client benutzt ausschließlich den Cache
  - Dieser lädt die Daten selbstständig aus der Datenbank
- Write behind
  - Der Cache gruppiert Updates und sendet diese asynchron zur Datenbank

- Eine Strategie von Ignite, Daten effizient auf einem Speichermedium zu speichern
  - relativ neues Feature (ab Version 2.1)
  - früher konnten die Daten nur in einem relationalen Datenbanksystem abgelegt werden
- Implementiert mit einem Write Ahead Log
  - Schnell
- ACID-fähig

4.4

## **CONSISTENCY**

- Durch die Verteilung der Daten ist Konsistenz in einem Cluster per se nur mit Aufwand zu erreichen
- Wie bereits erwähnt kann Ignite als CP-System betrieben werden
  - Der Atomicity-Mode `TRANSACTIONAL`
- Was passiert aber in einem CP-System, wenn Partitionsfehler auftreten?
  - Knoten, die nicht mehr der Majorität des Clusters zugehörig sind, müssen deaktiviert werden, sind damit nicht mehr Available
  - Probleme
    - Wie wird die Majorität erkannt?
    - Split Brain: Mehrere Sub-Cluster arbeiten zumindest zeitweise parallel, ohne miteinander kommunizieren zu können

- Ausgangssituation:
  - In einer reinen In-Memory-Datenhaltung werden bei einem Neustart eines Knoten alle Daten über Netzwerk an diesen übermittelt
  - Wird Native Memory Persistence (oder eine Datenbank) benutzt , so restauriert sich ein Knoten selbstständig beim Start
- Zumindest ein Subset von Knoten hält die Daten auch persistent
- Dieses Subset ist die Baseline

## Apache Ignite Best Practices: Cluster Topology Management and Data Replication

Ivan Rakov  
Apache Ignite Committer  
Senior Software Engineer in GridGain Systems





5

# ANWENDUNGEN

5.1

# ÜBERSICHT

- Eine Spring-Anwendung
  - <https://spring.io>
- Die Konfiguration sind Spring-Beans-Definitionen im XML-Format

- Compute Grid
  - Ausführen von Code-Sequenzen, die vom Client an das Grid gesendet werden
  - Map-Reduce
- Service Grid
  - Bereitstellen von Services auf dem Ignite-Cluster
    - Cluster-Singleton
    - Node-Singleton
- SQL-Grid
  - Verteilte Datenbank mit SQL-Unterstützung
  - Interne-Datenhaltung bleibt aber Cache-basiert
    - Damit können nicht alle SQL-Befehle unterstützt werden

- Ignite-Anwendungen sind Java-Programme
  - also Bytecode
- Programmiersprachen
  - Java
  - Scala
  - auch andere Bytecode-Sprachen sind natürlich möglich
- API-Dokumentation unter  
<https://ignite.apache.org/releases/latest/javadoc/index.html>

## All Classes

### Packages

org.apache.ignite  
org.apache.ignite.binary  
org.apache.ignite.cache  
org.apache.ignite.cache.affinit  
org.apache.ignite.cache.evicti  
org.apache.ignite.cache.evicti  
org.apache.ignite.cache.evicti  
org.apache.ignite.cache.evicti  
org.apache.ignite.cache.hiber  
org.apache.ignite.cache.jta  
org.apache.ignite.cache.jta.jn  
org.apache.ignite.cache.jta.re

### All Classes

AboutDialog  
AbstractCommand  
AbstractContinuousQuery  
AbstractEvictionPolicy  
AbstractEvictionPolicyFactory  
AbstractFailureHandler  
AbstractLSQR  
AbstractMarshaller  
AbstractMatrix  
AbstractNodeNameAwareMars  
AbstractVector  
Accuracy  
Activators  
AdaptiveControlProbe

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

## Ignite 2.7.6

### Common Grid APIs

Package	Description
<b>org.apache.ignite</b>	Contains entry-point <b>Ignite &amp; HPC APIs</b> .
<b>org.apache.ignite.cluster</b>	Contains cluster-related classes.
<b>org.apache.ignite.configuration</b>	Contains configuration-related classes.
<b>org.apache.ignite.lang</b>	Contains general language constructs and functional APIs for distributed computations.
<b>org.apache.ignite.lifecycle</b>	Contains lifecycle-related classes.
<b>org.apache.ignite.resources</b>	Contains resource injection annotations.
<b>org.apache.ignite.scheduler</b>	Contains <b>Job Scheduling</b> functionality.
<b>org.apache.ignite.spi</b>	Contains common classes and interfaces for SPI implementations.
<b>org.apache.ignite.thread</b>	Contains threads-related utility classes.

### Messaging APIs

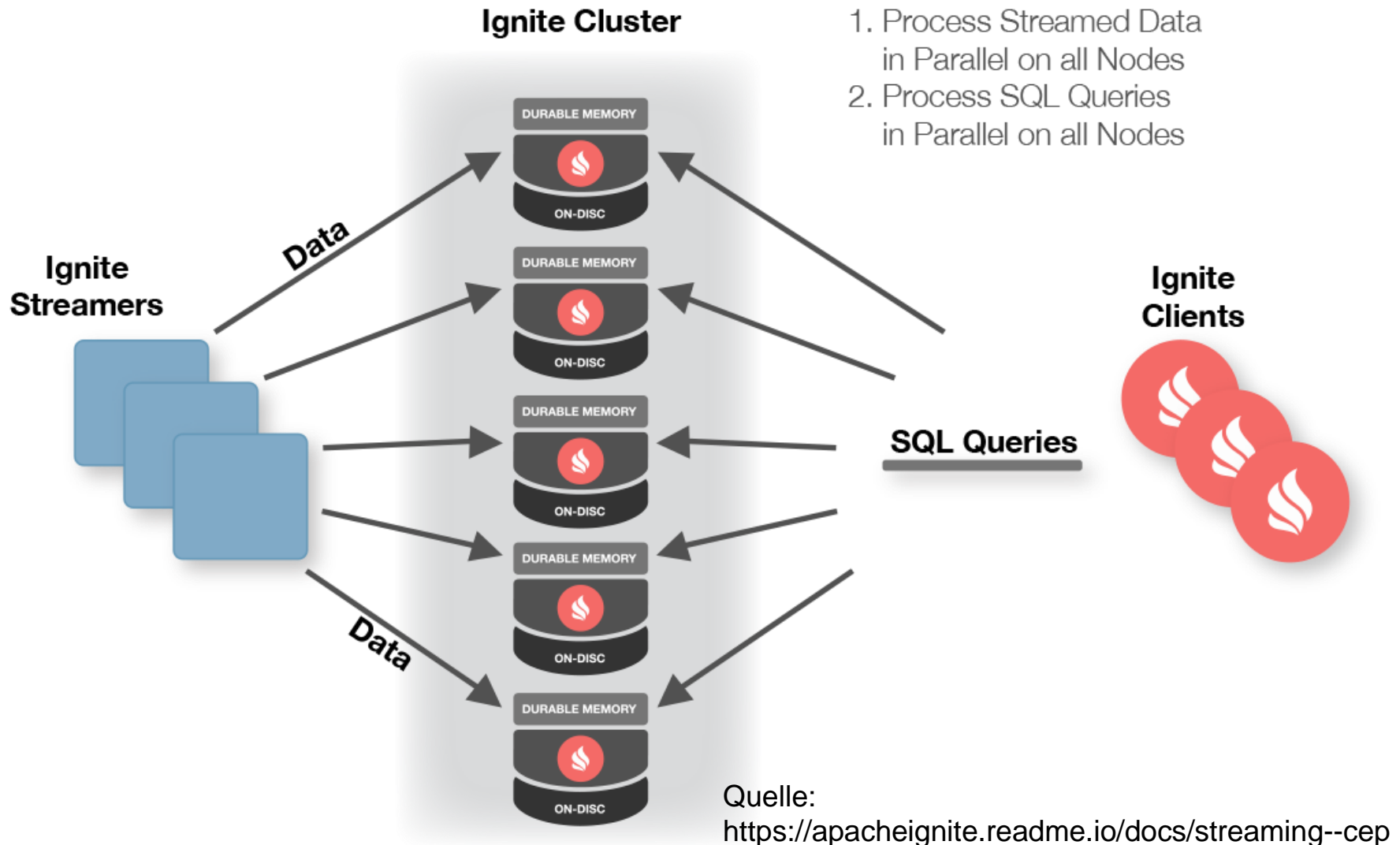
Package	Description
<b>org.apache.ignite.messaging</b>	Contains <b>Tonic-based Messaging</b> functionality.

- Treiber-Bibliotheken stehen für eine Vielzahl von Plattformen zur Verfügung
  - Java
  - C++
  - Ruby
  - PHP
- Auch Thin Clients können genutzt werden
  - REST-API
  - Allerdings mit eingeschränkter Funktionalität
    - insbesondere keine Transaktionalität

5.2

## **STREAMING**





- Data Loading
  - Laden der Daten aus einem Backend-System beim Hochfahren des Knotens
- Data Streaming
  - Schnittstellen zur Implementierung von Reader, Writer, Transformer
  - Fertige Implementierungen stehen zur Verfügung
    - Insbesondere die Apache Kafka-Integration

5.3

## BEISPIELPROGRAMME

- Bestandteil der Apache Ignite-Distribution
  - Verzeichnis `examples`
- Ebenfalls auf GitHub
  - <https://github.com/apache/ignite>
- Die meisten Beispiele sind Java-basierte Maven-Projekte
  - `pom.xml`
  - `src`-Verzeichnis
  - Paket `org.apache.ignite.examples`

- binary
- client
- cluster
- computegrid
- datagrid
- datastructures
- encryption
- events
- igfs
- messaging
- misc
- ml
- model
- persistentstore
- servicegrid
- springdata
- sql
- streaming
- util

6

# PRODUKTIONSUMGEBUNG

6.1

## **BETRIEBLICHE ASPEKTE**

- Docker-Images

- Docker Hub

- <https://hub.docker.com/r/apacheignite/ignite>

- Angepasste Images können selbstverständlich selbst gebaut werden

- Ausgangsbasis z.B. das Dockerfile des Standard-Images auf Docker Hub

```
# Start from a Java image.
FROM openjdk:8
# Ignite version
ENV IGNITE_VERSION 2.3.0
# Ignite home
ENV IGNITE_HOME /opt/ignite/apache-ignite-fabric-${IGNITE_VERSION}-bin
# Do not rely on anything provided by base image(s), but be explicit, if they
are installed already it is noop then
RUN apt-get update && apt-get install -y --no-install-recommends \
    unzip \
    curl \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /opt/ignite
RUN curl
https://dist.apache.org/repos/dist/release/ignite/${IGNITE_VERSION}/apache-
ignite-fabric-${IGNITE_VERSION}-bin.zip -o ignite.zip \
    && unzip ignite.zip \
    && rm ignite.zip
# Copy sh files and set permission
COPY ./run.sh $IGNITE_HOME/
RUN chmod +x $IGNITE_HOME/run.sh
CMD $IGNITE_HOME/run.sh
EXPOSE 11211 47100 47500 49112
```




- Kubernetes

- Ausführliche Metriken über JMX
  - REST-API beispielsweise mit Jolokia
    - <https://java.integrata-cegos.de/jolokia-simples-management-von-java-anwendungen/>
  - Anbindung an die Überwachungssoftware bzw. das Application Performance Management

## 6.2

# TUNING

[Java\\*](#) [C#/.NET](#) [C++](#) [SQL](#) [Integrations](#) [Ignite for Spark](#) [Tools](#)[Log In](#) [Javadoc](#) [Examples](#) [Ask a Question](#)[v2.7.6](#) > [Home](#) > [Preparing for Production](#) 

## SECURITY

[SSL/TLS](#)[Advanced Security](#)[Securing Data Deserialization](#)[Transparent Data Encryption](#)

## PRODUCTION READINESS

[Preparing for Production](#)[Capacity Planning](#)[Performance Tips](#)[Virtual Environments](#)[Durable Memory Tuning](#)[Garbage Collection Tuning](#)

# Preparing for Production

 [SUGGEST EDITS](#)

This section covers the main considerations to be taken into account when preparing to move a system to production.

- [Capacity Planning](#)
- [Performance Tips](#)
- [Virtual Environments](#)
- [Durable Memory Tuning](#)
- [Garbage Collection Tuning](#)

6.3

## **FAILOVER UND RECOVERY**



## Best Practices For Disaster Recovery and High Availability

Stan Lukyanov

Customer Solutions, GridGain Systems