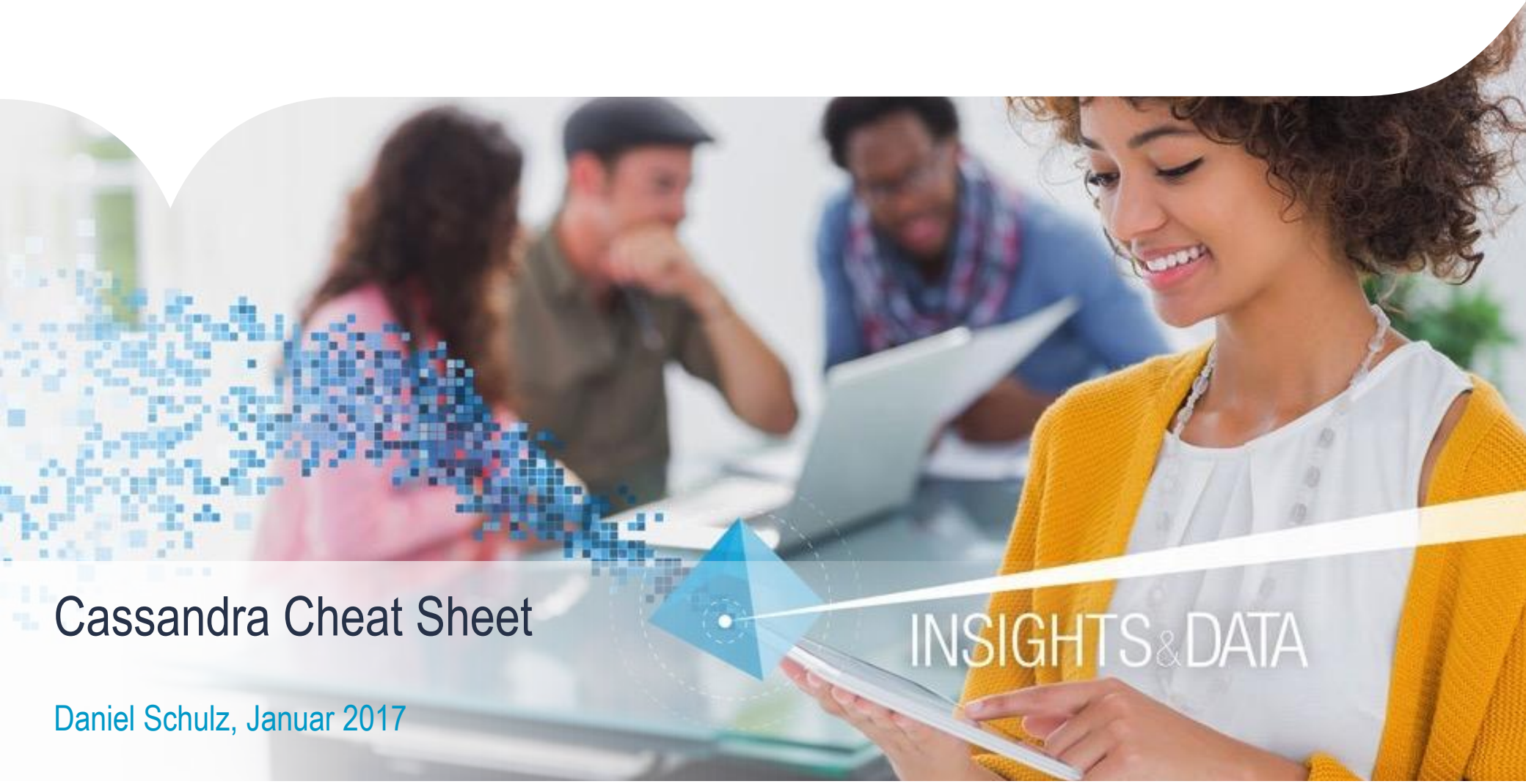
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



Cassandra Cheat Sheet

Daniel Schulz, Januar 2017

People matter, results count.

Agenda – Architecture

- Features of Apache Cassandra
- Features of DataStax' Cassandra resp. Enterprise
- Distributions Overview
- DataStax Enterprise Components

Cassandra Architecture Basics

- Cassandra is a horizontally scalable distributed Key/Value columnar database
- Cassandra knows awareness towards nodes, racks and Data Centers (DCs)
- It runs on commodity hardware as Java application w/ or wo/ root privileges
- Cassandra is often abbreviated C* because the double meaning C standing for its name and for „Tunable Consistency“ according to Brewer‘s Theorem (CAP) alike
- Unlike its main competitors HBase and Hadoop, Cassandra does not need Kerberos to guarantee data security, privacy and integrity

Features of Apache Cassandra

- Superior performance on HTAP (mixed) workloads
- Very good performance on OLAP and OLTP workloads comparable to competitors like Hadoop and HBase or better
- High-available data store – safe upon node failures and network partitions; has no Single Point of Failure
- Tunable, effective strong consistency – even when configured inconsistent
- Assured data integrity with optional Kerberos integration
- Local user management or usage of LDAP systems
- Multi-tenancy through Data Center and Rack Awareness
- Possible SMACK architecture –
  Spark, Mesos, Akka, Cassandra and Kafka for modern data analysis stacks

Features of Apache Cassandra

- Easily configurable Master-Master-replication
- Complete in-flight data encryption within one DataCenter (DC), between DCs and from client to the DC with TLS certificates, Trust Stores and basic authentication
- Client session between client and one or more Data Centers w/ auto-failover and rediscovery of newly joined nodes
- Compatible w/ DataStax‘ open-source drivers and connectors

Features of DataStax' Cassandra resp. Enterprise

- All features from Apache Cassandra
- Lean Big Data stack
- In-memory database – per table configurable
- Auditing safety
- TDE – Transparent Data Encryption for in-rest data encryption
- Complete in-flight data encryption within one DC, between DCs, from client to the DC and between all solution components of DSE with TLS certificates, Trust Stores and basic authentication
- Hub-and-Spoke architecture with local datastores
- No Single Point of Failure in Cassandra, Spark and Solr
- Integration w/ Spark, Solr, Hadoop and maintenance components
- Enterprise-ready support

Components of DataStax Enterprise

- Cassandra
- DSE Analytics – Spark with integration
- DSE Graph – graph database based on DSE Cassandra
- DSE Search – Solr with integration
- Hadoop v1 with integration and BYOH support
- OpsCenter as a maintenance tool
- DataStax-Agents to couple Cassandra with OpsCenter
- DevCenter to query data from Cassandra
- DataStax Studio to query data from DSE Graph

Distributions Overview

Two major Cassandra distributions

- Apache Cassandra (Apache 2 license)
  - open-source Cassandra w/ client tools
- DataStax Enterprise (commercial product w/ enterprise support)
  - open-source, improved, tested Cassandra w/ client tools as in Apache Cassandra,
  - plus OpsCenter for maintenance and backups,
  - plus DataStax-Agents to connect the former,
  - plus DSE Analytics (Spark-integration) w/ auto-failover through Session-Replication,
  - plus DSE Search (Solr-integration) for document indexing, retrieval and geo spactial features like fencing,
  - plus DSE Graph horizontally scalable graph database on Cassandra,
  - plus DevCenter comparable to SQLDeveloper for developers and testers,
  - plus DataStax Studio to submit DSE Graph queries,
  - plus enterprise support for DataStax‘ open-source product line

Architectonical Features of DataStax Enterprise

Enterprise-ready, distributor-supported, hardened, tested

- Cassandra – as a data store
- Sqoop – for bidirectional data transfer
  between Cassandra and RDBMS resp. the local file system
- Hadoop v1 w/ HDFSv1 & MRv1 – for OLAP workloads and BYOH-integration
- Spark (DSE Analytics)
- Solr (DSE Search)
- DSE Graph as a graph database on Cassandra

Open-Source Architecture Components

- Java Cassandra driver:
  github.com/datastax/java-driver
- Spark/Cassandra connector:
  github.com/datastax/spark-cassandra-connector
- Full list of DataStax‘ open-sourced product line for various programming languages and platforms:
  github.com/datastax

Agenda – Basics

- Terminology
- Schema Design

Cassandra Basics

- Cheat Sheet: hakkalabs.co/articles/c-cheat-sheet

Term	Description	MySQL Equivalent (if exists)
Column family	Also known as a table in CQL, a column family is a container for columns and rows.	Table
Commit log	A sequential log of writes and the first place writes are written and ensures that writes are durable in the case of a crash or node failure.	Transaction log
Compaction	The process of merging multiple SSTables (as they are immutable), into a new unified sstable continuing the still active data from the the two source sstables. There are different strategies available for performing compactions.	N/A
Consistency level	The number of replicas that must participate in a given read or write query. To better control the behavior and outcome of eventual consistency, a consistency level (CL) provides tunable consistency and is provided with each query. Example consistency levels are ANY, ONE, TWO THREE, QUORUM, LOCAL_QUORUM, EACH_QUORUM, SERIAL, and ALL. A query performed at CL=TWO would require a read result to be the most recent data from two of the closest replicas or require a write to be written to both the commit log and memory table for at least two replica nodes.	N/A
Coordinator	The node responsible for communicating with other replicas in the cluster for a given client read or write query.	N/A
CQL	The Cassandra Query Language is a structured language for interfacing with Cassandra.	SQL
Eventual consistency	The concept that all replica nodes will "eventually" return the newest and same data for a given query.	N/A

Cassandra Basics

- Cheat Sheet: hakkalabs.co/articles/c-cheat-sheet

Term	Description	MySQL Equivalent (if exists)
Flushing	The process of sorting a memtable by key and then writing the data sequentially to disk in the form of an SSTable.	Flushing
Gossip	The protocol that all nodes in the cluster use to know the state information for all other nodes.	N/A
Keyspace	A container for all application data.	Database
Memtable	A per column family, bounded, write-back, in-memory cache of rows that can be looked up by key. When the memtable is full, it is flushed and serialized to a SSTable.	Closest would be issuing SET AUTOCOMMIT=0 and not flushing the database after every transaction as by default MySQL behavior.
Native Binary Protocol	The native binary protocol (available in 1.2+) is one of two methods for clients to communicate with Cassandra. CQL queries are made over the native binary protocol.	MySQL Binary Protocol
Partitioner	A partitioner is a set of logic that determines how data is distributed across the nodes in the cluster. It is essentially a hash function that computes the token (or hash) for each row key.	N/A
Replication factor	How many replicas or nodes in the cluster should be responsible for each row key.	N/A
Ring	The concept that nodes are responsible for a given token range, evenly divided around a ring or evenly spaced hashes.	N/A
Row key	Also known as a primary key in CQL, a row key is a way to uniquely identify a row of columns.	Primary key
SSTable	An immutable (written once and not modifiable) per column family file that contains all or a portion of data for a column family	InnoDB (.idb) files or MyISAM (.myd) files

Cassandra Basics

- Cheat Sheet: hakkalabs.co/articles/c-cheat-sheet

Term	Description	MySQL Equivalent (if exists)
Thrift protocol	A RPC based protocol. Thrift was the original client protocol available in Cassandra. Thrift has it's own set of API methods, but it also supports the execution of CQL queries.	N/A
Token	A token is a single hash value from a larger token range (or hash range). Each Cassandra node is responsible for sets of tokens that define what replica nodes are responsible for what data.	N/A

Cassandra Tools for Development & Maintenance

Command Line tools for both Cassandra distributions:

- `nodetool` –
  checking the cluster state and node health using `` `nodetool status` ``
- `cqlsh` –
  CQL shell like `SQLplus` for Oracle

Command Line tools for DSE Cassandra:

- `dse` –
  central tool for all Cassandra, Spark, etc. interaction with DSE
- `dsetool` –
  checking cluster state like in nodetool with the now deprecated `` `dsetool ring` ``

Cassandra Basics

- Terminology and Cassandra sub components explained:
  - zcourts.com/2011/12/25/cassandra-terminology-cheat-sheet/#sthash.iZPFkYrr.PG4VAfdE.dpbs
  - zcourts.com/2011/12/13/cassandra-cheat-sheets/#sthash.v415sFsw.FkfvMgvF.dpbs

Cassandra Schema Design

- Pay special attention to your Primary Key and its Partition Key and Clustering Columns in particular
  - The Primary Key needs to be provided for select statements from left to right – more on this one in the section „Cassandra Query Limitations on Primary Keys & Indices“
  - Save your data in a form you like them later on to query
- 1st NF *(First Normal Form)*
  - Unlike in RDBMS, in big data solutions always denormalize and hence possibly duplicate your data

Agenda – Getting Started

- Development Systems
- Test- & Integration Systems
- Regression Tests & Production Systems
- Free-of-charge Academy Courses

Development: Prerequisites of both Cassandra distributions

- Having Java 8 installed
- Having set `$JAVA_HOME` variable to the respective root directory and/or added `$JAVA_HOME/bin` to the user `PATH` as well – so that
  - `` `echo $JAVA_HOME` `` yields a non-empty response and
  - `` `which java` `` is found from any directory there is

Development: Getting Started w/ Apache Cassandra

- Installing Apache Cassandra: cassandra.apache.org/doc/latest/getting_started/installing.html
- Configuring both Cassandra distributions: cassandra.apache.org/doc/latest/getting_started/configuring.html
- Data manipulation on both Cassandra distributions: cassandra.apache.org/doc/latest/getting_started/querying.html
- Start & stop Apache Cassandra: wiki.apache.org/cassandra/RunningCassandra

Development: Getting Started w/ DataStax Enterprise

- Installing
  - Installing DSE in various ways: docs.datastax.com/en/latest-dse/datastax_enterprise/install/installTOC.html
  - Installing DSE from the DSE installer w/ root privileges as a service: docs.datastax.com/en/latest-dse/datastax_enterprise/install/installGUIdse.html
  - Installing DSE from the DSE installer wo/ root privileges: docs.datastax.com/en/latest-dse/datastax_enterprise/install/installNoSudoDse.html
- Configuring both Cassandra distributions:
  - cassandra.apache.org/doc/latest/getting_started/configuring.html
  - docs.datastax.com/en/cassandra/3.x/cassandra/configuration/configTOC.html
- Data manipulation on both Cassandra distributions: cassandra.apache.org/doc/latest/getting_started/querying.html
- Start & stop DSE‘s Cassandra: docs.datastax.com/en/cassandra/3.0/cassandra/initialize/referenceStartStopTOC.html

Development: Getting Started w/ both Cassandra distributions

- It is recommended to permanently set environment variables:
  - `CASSANDRA_HOME` to the Cassandra root directory,
  - `CASSANDRA_CONF` to Cassandra's `CASSANDRA_HOME/conf` directory and
  - add them its bin directory to the user PATH as well:
    `export PATH=$CASSANDRA_HOME/bin:$PATH`

  for ease of use

Development: Starting up the Cluster

- Boot your cluster using:
  - `$CASSANDRA_HOME/bin/cassandra` (Apache C*) or
  - `$DSE_HOME/bin/dse cassandra` (DSE C*)
- Checking the cluster state and node health using
  - `` `nodetool status` `` (both)

Development: First Steps w/ an Installed Cassandra

- Get already present databases

```
DESCRIBE keyspaces;
```

- Create your own database

```
CREATE KEYSPACE capgemini_db WITH replication = {
'class':'SimpleStrategy', 'replication_factor' : 1 };
```

- Use this database:

```
USE capgemini_db;
```

- Create a custom table:

```
CREATE TABLE users_by_id(id uuid PRIMARY KEY,
username text, password text, displayname text,
email text);
```

- Optional in the first place: create an index:

```
CREATE INDEX users_by_id_index_username
ON users_by_id( username );
```

Development: First Steps w/ an Installed Cassandra

- Load data into you custom table

```
 insert into users_by_id(id, username, password,
displayName, email ) values( uuid(), ‘serge.kampf',
'secret', ‘Serge Kampf', ‘serge.kampf@capgemini.com');
```

- Get back your data

```
 select * from users_by_id;
```

Test: Adding Users, Authentication & Authorization

- Configuring authentication on both distributions:
  - docs.datastax.com/en/cassandra/3.0/cassandra/configuration/secureConfigNativeAuth.html
  - datastax.com/dev/blog/a-quick-tour-of-internal-authentication-and-authorization-security-in-datastax-enterprise-and-apache-cassandra
  - wiki.apache.org/cassandra/SimpleAuthenticator
- Configuring DSE Unified Authentication:
  - docs.datastax.com/en/latest-dse/datastax_enterprise/unifiedAuth/unifiedAuthConfig.html
  - docs.datastax.com/en/datastax_enterprise/4.5/datastax_enterprise/reference/refDseYaml.html
- Swift walk-through on creating users: duckduckgo.com/?q=cassandra+cheat+sheet&ia=cheatsheet

Test: Cleaning Up Spark Temporary Directories

- When using Spark: enable temporary data cleanup so your persistent disks do not run out of free space in a couple of hours:
  set `spark.worker.cleanup.enabled` to `true` in `SPARK_WORKER_OPTS`
  spark.apache.org/docs/latest/spark-standalone.html
- Consider changing as well:
  - `spark.worker.cleanup.interval`
  - `spark.worker.cleanup.appDataTtl`

Test: Building a Multi-Node Data Center

- Many nodes, one Data Center: docs.datastax.com/en/cassandra/3.0/cassandra/initialize/initSingleDS.html
- Many nodes, many Data Centers: docs.datastax.com/en/cassandra/3.0/cassandra/initialize/initMultipleDS.html
- Many virtual nodes on one physical machine wo/ Virtualization technology: datastax.com/dev/blog/running-multiple-datastax-enterprise-nodes-in-a-single-host

Production: Recommended Production Settings

- Recommended production settings for independent of the platform: docs.datastax.com/en/archived/cassandra/2.0/cassandra/install/installRecommendSettings.html
- Recommended production settings for Linux: docs.datastax.com/en/landing_page/doc/landing_page/recommendedSettingsLinux.html
- Recommended production settings for Windows: docs.datastax.com/en/landing_page/doc/landing_page/recommendedSettingsWin.html

Free-of-charge Academy Courses

- Self-paced online courses on Apache Cassandra, DataStax Enterprise, its respective features and certification information: academy.datastax.com/courses
- DataStax‘ YouTube account for press information: youtube.com/user/DataStaxMedia

Agenda – Best Practices

- Best Practices
- Hardware & OS Requirements

Cassandra Best Practices

- For production systems, it is recommended to follow the Append-only pattern in databases: do not remove or change existing columns, tables and Keyspaces; only add new ones upon changes
- One Keyspace only: it is best to put all data in one consolidated Keyspace as e.g. Spark relies on working with data from exactly one Keyspace – not multiple ones
- Materialized Views force data duplication in the background
- Each Data Center needs to store all data and data duplication resulting from Materialized Views – hence the Replication Factor may change from DC to DC
- Early on when having multiple nodes, consider using the Keyspace Replication Class „NetworkTopologyStrategy“ (NTS) over „SimpleStrategy“ (default) as the former can be scaled more easily; the latter is needed for all systems having just a single node

Cassandra Best Practices

- Production clusters are recommended to have five to six nodes at least; the Replication Factor (RF) is the absolute, technical minimum node count needed
- Increasing the RF yields better cluster usage and higher throughput rates but needs higher Consistency Levels (CL) to guarantee string consistency
- Lower CLs yield higher throughput as fewer replicas need to arrive before the Coordinator node replies back to the client
- Enabling Virtual Nodes (vnodes) makes the data more evenly distributed across the cluster
- Restrict each DC to one workload only – one DC for OLAP jobs and another for OLTP queries so both do not interfer each other
- Spark analytical jobs may use different queues as a workaround for some OLAP workloads in an e.g. OLTP workload cluster

Cassandra Best Practices

- NTP *(Network Time Protocol)* – synchronized clocks are profoundly needed
- Local, fast and many disks on each node; prefer many SSDs per node – no RAID storage, no SAN/NAS, etc.
- Consider using three SSDs per virtual node: one for data, commit log and the Operating System respectively
- No Swapping to disk – disable the Operating System's swap
- Set appropriate user limits resp. quotas
- Fast network
- Much transient RAM memory – much persistent memory
- Almost all modern Linux distributions in current versions are supported: docs.datastax.com/en/landing_page/doc/landing_page/supportedPlatforms.html
- Both major Java distributions are supported:
  Oracle JDK/JRE 1.8.0 u40 or later and
  OpenJDK 8

Cassandra Best Practices

- Recommended production settings for independent of the platform: docs.datastax.com/en/archived/cassandra/2.0/cassandra/install/installRecommendSettings.html
- Recommended production settings for Linux: docs.datastax.com/en/landing_page/doc/landing_page/recommendedSettingsLinux.html
- Recommended production settings for Windows: docs.datastax.com/en/landing_page/doc/landing_page/recommendedSettingsWin.html

Cassandra Best Practices

- Hardware requirements: docs.datastax.com/en/latest-dse/datastax_enterprise/install/installTARdse.html
- Supported Operating Systems: docs.datastax.com/en/landing_page/doc/landing_page/supportedPlatforms.html
- You may use Kernel virtualization
  - „Fat“ Hardware virtualization is not recommended but does not void your warranty; e.g. VirtualBox, VMware, HyperV, etc. are less favored – please avoid Hardware virtualization and put Cassandra directly on physical machines for data locality, less overhead and fast network access
  - Lean Kernel virtualization, like in Docker and OpenShift, is preferrable to Hardware virtualization and there exists guidelines from the vendor DataStax

Agenda – Technical Details

- CQL, CLI & Management tools
- Data Types & UDFs
- Compaction Strategies, Snitches, Seed Providers, Network Strategies
- Limitations
- Cassandra for the Impatient RDBMS Switcher
- Replication Factor vs Consistency Level
- Data Duplication
- Write & Read Paths
- Further Reading

Cassandra Tools for Development & Maintenance

Command Line tools for both Cassandra distributions:

- `nodetool` –
  checking the cluster state and node health using `` `nodetool status` ``
- `cqlsh` –
  CQL shell like `SQLplus` for Oracle

Command Line tools for DSE Cassandra:

- `dse` –
  central tool for all Cassandra, Spark, etc. interaction with DSE
- `dsetool` –
  checking cluster state like in nodetool with the now deprecated `` `dsetool ring` ``

Cassandras Default Data Types

Cassandra Data Type	Explanation
ascii	ASCII character string
bigint	8 byte number
blob	Byte array data – up to 1 MiB recommended; up to 2 GiB possible; no validation
boolean	Either true or false – default is false
counter	Integer number to incrementally scale up or down – only use CQL command „update“
decimal	Variable-precision number
double	8 byte floating point number w/ double precision
date	Date string wo/ time information; time information can be supplied using „time“ and both toghether are supplied in „datetime“

Cassandras Default Data Types

Cassandra Data Type	Explanation
frozen	Frozen encapsulates multiple values into a single one; as the consolidated type is treated as BLOB in the background, updates to individual sub components require overwriting the entire field
float	4 byte floating point number w/ single precision
inet	IP address in IPv4 or IPv6 format
int	4 byte integer number
list	A non-empty, ordered collection
map	A non-empty key/value map in JSON-style
set	A non-empty, unordered collection
smallint	2 byte integer number

Cassandras Default Data Types

Cassandra Data Type	Explanation
text	UTF8 encoded string of variable length wo/ specifying its respective length – pendant to VARCHAR or VARCHAR2; Cassandras “text” and “varchar” can be used interchangeably
time	time string wo/ date information; date information can be supplied using „date“ and both toghether are supplied in „datetime“
timestamp	Date w/ time in milli seconds relative to Linux‘ Epoch in 8 bytes
timeuuid	Type 1 UUID w/ time information – having an upper limit upon creating them in batches but being guaranteed unique within this JVM and revealing the MAC address of the computer producing them (Type 1 UUID: Date/Time/MAC)
tinyint	1 byte integer number
tuple	A group of 2 or 3 fields – since Cassandra 2.1

Cassandras Default Data Types

Cassandra Data Type	Explanation
UUID	Type 1 or Type 4 UUID – the former having an upper limit upon creating them in batches but being guaranteed unique within this JVM and revealing the MAC address of the computer producing them (Type 1 UUID: Date/Time/MAC); the latter allowing collisions within one JVM instance but having no upper limits and does not leak identity information (Type 4 UUID: random)
varchar	See above in „text“
varint	Arbitrary-precision integer number

Cassandras Default Data Types

- Sources:
  - codefoundries.com/developer/cassandra/cassandra-cheatsheet.html
  - docs.datastax.com/en/cql/3.3/cql/cql_reference/cql_data_types_c.html
- Feel free to create your own data types using CREATE TYPE: docs.datastax.com/en/cql/3.1/cql/cql_reference/cqlRefcreateType.html
- You can create UDFs (User Defined Functions) as well – but from Cassandra 3 on, they are deactivated by default and not recommended using: docs.datastax.com/en/cql/3.3/cql/cql_using/useCreateUDF.html

Compaction Strategies

- SizeTieredCompactionStrategy (STCS)
  - Best for insert-intense workloads
- LeveledCompactionStrategy (LCS)
  - Best for read- and update-intense workloads
- DateTieredCompactionStrategy (DTCS) – use with care
  - Best to use for data stored and retrieved by date/time information
- Custom Compaction Strategies possible to implement

- It is not recommended altering existing Compaction Strategies on existing tables; rather migrate data between old and newly created tables instead

- Further reading: instaclustr.com/blog/2016/01/27/apache-cassandra-compaction/

Snitches

- SimpleSnitch
  - For one DC deployments only
- RackInferringSnitch
  - Takes each nodes IP address and defines data locality based upon that – e.g. in IPv4 based on the third quad being the rack and the forth being the node identifier
- Various cloud provider snitches for Amazon AWS, Google Cloud and Cloudstack
- Custom Snitches possible to implement

- Further reading: docs.datastax.com/en/archived/cassandra/2.0/cassandra/architecture/architectureSnitchesAbout_c.html

Seed Providers

- SimpleSeedProvider
  - Takes seed IPs in one DC
- Kubernetes Seed Providers for e.g. OpenShift or SDN/Cloud networks
  - Finds services using Kubernetes‘ API
  - Source: github.com/kubernetes/kubernetes/tree/master/examples/storage/cassandra
- Custom Seed Providers possible to implement

Network Strategies

- SimpleStrategy
  - Only applicable to one DC – cannot scale out for different workloads wo/ data migration
- NetworkTopologyStrategy
  - Can be used with more than one node
  - Easily to scale out to multiple DCs – it is wise to use this in environments w/ multiple nodes

- It is not recommended altering existing Network Strategies on existing Keyspaces; rather migrate data between old and newly created Keyspaces respectively

Limitations of Cassandra & CQL

- Most prominent limitations on Cassandra & CQL are:
  - neither join functionality – *can be circumvented in Spark e.g. through DSE* –,
  - nor are there table references hence no integrity checks,
  - nor transactions – *can be circumvented using Batches for guaranteed atomicity* –,
  - nor “random” select statements on table columns which are not the Primary Key or do not have an index – *can be circumvented in Spark e.g. through DSE*
- Other important limitations:
  - Maximum table name length is 48 characters; table & Keyspace namens are recommended to be in `lowercase_with_underscore_deelimiters` or the case-sensitive respective name needs to be put parenthesis „`camelCaseName`“
  - Each BLOB is recommended to be at most 1 MiB big; 2 GiB is maximal possible

Limitations of Cassandra & CQL

- No distribution pre-packed with a Message Queue
- Whole client session can be encrypted or not – but unlike in Oracle, not only the authentication can be secured
- CQL is not fully SQL-kompatibel but similar to it – *SQL support can be gained through Spark from DSE*
- DataStax‘ JDBC driver for Cassandra is deprecated – programming language access to Cassandra only – *ODBC & JDBC support can be gained through Spark from DSE*
- All limitations w/ the rest having reasonable high bars in practice are listed here: docs.datastax.com/en/cql/3.3/cql/cql_reference/refLimits.html

Cassandra Query Limitations on Primary Keys & Indices

Put indices only on columns:

- With low cardinality of the value range – e.g. person sexes female/male, on person‘s departments, on base colors, etc.
- Avoid indices when data is frequently changed or removed
- Queries can retrieve specific values rather than ranges of values – Clustering columns can retrieve connected ranges only
- The attributes from the Primary Key must incorporate all precessor attributes – „from left to right“ only
- Make sure narrow down your lookup query even when accessing an index
- Source: docs.datastax.com/en/cql/3.1/cql/ddl/ddl_when_use_index_c.html

Cassandra for the Impatient RDBMS Switcher

Cassandra jargon to RDBMS jargon

- Keyspaces are Databases resp. Schemata
- Column Family (CF) are Tables
- Prepared Statements are predefined statements with placeholders to later on with appropriate data, which can hence preparation be executed very efficiently
- Cassandra knows indices but no histograms
- Cassandra partitions are a differing concept to Oracle patitions;
  Cassandra partitions are collections of data rows with the very same Partition Key resp. a common part in the Primary Key

Cassandra for the Impatient RDBMS Switcher

Keep in mind

- Primary Key consists of Partition Key and an or more optional Cluster Columns
  - Partition Key can be one or more attributes (compound) and define the node those data will be stored on; each one is one partition
  - Clustering Columns (CC) can be one or more attributes (compound) and define the ordering of those rows within each partition; like sort ascending or descending by what attribute
  - Update/insert queries need he entire Primary Key
  - Select queries need the entire Partition Key and the Clustering Columns from left to right
  - One index can be used upon select queries as well
- Materialized Views…
  - Can circumvent problems querying existing data by other attributes
  - But come at the cost of data duplication

Cassandra for the Impatient RDBMS Switcher

Keep in mind

- While Oracle can serve as a Message Queue, this is considered an Anti-Pattern for Cassandra; hence the SMACK stack w/ Cassandra as a data store uses Kafka as a means to distribute messages

Replication Factor vs Consistency Level

- Replica: each replica is the same – there is no „master data instance“, an original date and some copies of that or s.th. like that; there is a logical date, which is persisted [1,2,3,…) times in equal, eye-to-eye physical data instances
- The Replication Factor (RF) defines the amount each data instance is replicated in rest across each Data Center (DC) – so having one row thrice makes all disks in sum this item three times for one DC, six times for two DCs, etc.
- The Consistence Level (CL) defines the count of data instances to await while querying data in flight to define the latest state of it; while selecting data in a query always any replica will be asked for and the CL defines how many replicas to compare the latest timestamp to find out the most recent version

Replication Factor vs Consistency Level

- The higher the CL the higher the latency for that query and the lower its High Availability properties
- So both RF & CL values are recommended to keep at a reasonable level, like
  - RF = 3 (default) and
  - Use CL = 2 upon reading and writing for strong consistency or
  - Use CL = 1 upon reading and writing for loose consistency – is very likely to be nearly always strongly consistent in practice, according to the Netflix CL One Experiment: slideshare.net/planetcassandra/cassandra-day-sv-2014-a-netflix-experiment-eventual-consistency-hopeful-consistency-with-apache-cassandra

- Overview over CLs: docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_config_consistency_c.html

Data Duplication

Data Duplication Occurs in Cassandra for

- Each Data Center (full logical data state per DC)
- Each Materialized View, the originating table is basically replicated
- Each Snapshot taken when data then is changed
- Each Backup is taken, because OpsCenter saves backups basically first and permanently on local worker node disks –
  and then possibly exports to the final medium
- The RF, which is the underlying baseline duplication

Write & Read Paths

- Cassandras Write & Read Path overview: docs.datastax.com/en/cassandra/3.x/cassandra/dml/dmlIntro.html
- Cassandras Write Path: docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_write_path_c.html
- Cassandras Read Path: docs.datastax.com/en/cassandra/3.x/cassandra/dml/dmlAboutReads.html

Further Reading

- Cassandra RefCard: dzone.com/refcardz/apache-cassandra
- CQL command overview: tutorialspoint.com/cassandra/cassandra_cqlsh.htm
- CQL Reference Guide: datastax.com/wp-content/uploads/2012/01/DS_CQL_web.pdf
- CLI Reference Guide: datastax.com/wp-content/uploads/2012/01/DS_CLI_web.pdf
- Nodetool/Mgmt Reference Guide: datastax.com/wp-content/uploads/2012/01/DS_nodetool_web.pdf
- Cheat-Sheet w/ architecture, CLI tools & performance considerations: https://github.com/alpinejoe/cassandra-cheat-sheet
- Cheat Sheet: hakkalabs.co/articles/c-cheat-sheet

Further Reading

- Introduction w/ Hadoop-integration for DSE: itjumpstart.files.wordpress.com/2015/03/4487-rc153-010d-cassandra.pdf
- Introduction to Cassandra: dzone.com/articles/introduction-apache-cassandras
- Elaborate Cassandra tutorial: tutorialspoint.com/cassandra
- Compact tutorial on getting started w/ Cassandra: cheatsheet.logicalwebhost.com/cassandra-howto
- Compact tutorial on getting started w/ Cassandras CLI: https://sites.google.com/site/cassandratutorial/client-libraries/command-line-interface

Further Reading

- Running Cassandra on Docker resp. OpenShift environments: datastax.com/wp-content/uploads/resources/DataStax-WP-Best_Practices_Running_DSE_Within_Docker.pdf
- Developer Guidelines for Spark:
  - github.com/databricks/scala-style-guide
  - cloudera.com/documentation/enterprise/5-6-x/PDF/cloudera-spark.pdf
  - github.com/apache/spark/blob/master/scalastyle-config.xml

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

About Capgemini

With more than 180,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2015 global revenues of EUR 11.9 billion.

Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

www.capgemini.com