Cassandra User Group Cologne

19:00 - Reception

19:20 - Cassandra libraries for Java developers
DuyHai Doan, Cassandra Evangelist at DataStax

20:20 - Apache Cassandra 3.0
Robert Stupp, Committer to Apache Cassandra, CIO contentteam AG

21:00 - Finish & Networking

contentteam DATASTAX PLANET CASSANDRA

contentteam

APACHE CASSANDRA 3.0

CASSANDRA USER GROUP COLOGNE

24.03.2015

Robert Stupp

- CIO contentteam
- Committer to Apache Cassandra
- Coding experience since 1985
- Internet and related technologies since 1992

- rstupp@contentteam.com
- @snazy

contentteam & DATASTAX

contentteam is a DataStax Solutions Partner

contentteam is active in Apache Cassandra community

AGENDA

1. Cassandra history (short)
2. Cassandra 3.0
3. Cassandra community
4. Apache Cassandra vs. DataStax Enterprise
5. One more thing :)

APACHE CASSANDRA HISTORY

CASSANDRA HISTORY

- Initially developed at Facebook to build a ”continuously available” database
  - Replication
  - Globally distributed
  - Masterless architecture
- Influenced by BigTable and Dynamo
- Today:
  Huge amount of working installation -
  few nodes up to 1000+ globally distributed nodes

CASSANDRA HISTORY

- Open sourced in 2008
- Version 0.3 - July 2009
- Version 0.6 - June 2010
- Version 0.7 - 2011
- Version 1.0 - October 2011 - introduction of DSE
- Version 1.2 - December 2012
- Version 2.0 - August 2013
- Version 2.1 - September 2014

CASSANDRA LIVE RELEASES

- Version 2.0 line
  critical bugfixes applied to 2.0 release
- Version 2.1 line
  bugfixes applied to 2.0 release
  some new, non-intrusive features
- Version 3.0
  lots of new features
  lots of improvements

APACHE CASSANDRA 3.0

DISCLAIMER

- Apache Cassandra 3.0 is **still in development**
- Features might be changed / revoked until 3.0 release

RELEASE DATE

- Apache Cassandra 3.0 will be released **when it is finished**
- Don’t ask for a release date - we don’t know it yet ;)

CASSANDRA 3.0 FEATURES

CASSANDRA 3.0 NEW FEATURES SUMMARY

- JSON support
- User-Defined-Functions + User-Defined-Aggregates
- Role based access control
- New row cache
- Lots of (small) performance improvements summing up to a huge improvement

- Altogether approx >100 tickets for 3.0
- Plus changes merged from 2.0 via 2.1 to 3.0 and 2.1 to 3.0

CASSANDRA 3.0 REMOVED FEATURES

- `cassandra-cli` is removed (as announced)

JSON Support

CASSANDRA 3.0

JSON SUPPORT

- Allows you to do **`INSERT`** and **`SELECT`** data using JSON data format
- Format your data to insert using JSON - can save a step to transform data

- Ease **web application development** - less transformation to/from Cassandra to the browser
- Also nice when using NodeJS

JSON SUPPORT - EXAMPLE

```
CREATE TYPE address (
  street text,
  city text,
  zip_code int,
  phones set<text>
);

CREATE TABLE users (
  id uuid PRIMARY KEY,
  name text,
  addresses frozen<map<text, address>>
);

INSERT INTO users JSON
  '{"id": "4b856557-7153",
   "name": "snazy",
   "addresses": {"work": {"street": "Im Mediapark 6",
                        "city": "Köln",
                        "zip_code": 50670,
                        "phones": ["+492214546200"]}}}';
```

this is JSON

JSON SUPPORT

```
cqlsh> SELECT JSON * FROM users;

[json]
------------------------------------------------------------
{"id": "4b856557-7153",
   "name": "snazy",
   "addresses": {"work": {"street": "Im Mediapark 6",
                        "city": "Köln",
                        "zip_code": 50670,
                        "phones": ["+492214546200"]}}}

(1 rows)
```

this is JSON

JSON SUPPORT

- JSON support **does not** introduce schema-free tables!!
- http://rustyrazorblade.com/2014/07/the-myth-of-schema-less/
- https://blog.compose.io/schema-less-is-usually-a-lie/

User-Defined-Functions (UDFs)

User-Defined-Aggregates (UDAs)

CASSANDRA 3.0

UDF

- UDF means **User** Defined Function
- **You** write the code that’s executed on Cassandra nodes
- Functions are **distributed** transparently to the whole cluster
- *You may not have to wait for a new release for new functionality :)*

UDF CHARACTERISTICS

- „Pure“
    - just input parameters
    - no state, side effects, dependencies to other code, etc
- Usually deterministic

Consider a Java function like…

```
import nothing;

public final class MyClass
{
  public static int myFunction ( int argument )
  {
    return argument * 42;
  }
}
```

This would be your UDF

UDF EXAMPLE

```
CREATE FUNCTION sinPlusFoo
(
      valueA    double,
      valueB    double
)
RETURNS double
LANGUAGE java
AS ’return Math.sin(valueA) + valueB;’;
```

arguments

return type

UDF language

Java code

Java works out of the box!

UDF/UDA

```
CREATE FUNCTION sin (
        value     double )
RETURNS double
LANGUAGE javascript
AS ’Math.sin(value);’;
```

JavaScript works, too

JavaScript code

JavaScript works out of the box!

JSR 223

- “Scripting for the Java Platform“
- UDFs can be written in **Java** and **JavaScript**
- Optionally: **Groovy**, **JRuby**, **Jython**, **Scala**
- Not: Clojure (JSR 223 implementation’s wrong)

BEHIND THE SCENES

- Builds Java (or script) source
- Compiles that code (Java class, or compiled script)
- Loads the compiled code
- Migrates the function to all other nodes
- Done - UDF is executable on any node

TYPES FOR UDFS

- Support for **all** Cassandra **types** for arguments and return value
- All means
  - Primitives (**`boolean`**, **`int`**, **`double`**, **`uuid`**, etc)
  - Collections (**`list`**, **`set`**, **`map`**)
  - Tuple types, User Defined Types

UDF - For what?

UDF INVOCATION

```
SELECT sumThat ( colA, colB )
    FROM myTable
    WHERE key = ...


SELECT sin ( foo )
    FROM myCircle
    WHERE pk = ...
```

Now your application can sum two values in one row - or create the sin of a value!

GREAT NEW FEATURES!

Okay - not really...

UDFS ARE GOOD FOR…

- UDFs on their own are just „nice to have“
- Nothing you couldn’t do better in your application

User Defined Aggregates !

Use **UDFs** to code your **own aggregation** functions

(Aggregates are things like SUM, AVG, MIN, MAX, etc)

Aggregates :

consume values from **multiple rows** & **produce** a **single result**

UDA EXAMPLE

```
CREATE AGGREGATE minimum ( int )
    STYPE int
    SFUNC minimumState;
```

arguments

state type

name of the state UDF

HOW AGGREGATES WORKS

```
SELECT minimum ( val ) FROM foo
```

1. **Initial state** is set to **`null`**
2. for **each row** the **state function** is called with current state and column value - returns **new state**
3. After all rows the **aggregate returns** the **last state**

MORE SOPHISTICATED

```
CREATE AGGREGATE average ( int )
    SFUNC averageState
    STYPE tuple<long,int>
    FINALFUNC averageFinal
    INITCOND (0, 0);
```

initial state value

name of the final UDF

HOW THAT WORKS

```
SELECT average ( val ) FROM foo …
```

1. **Initial state** is set to **`(0,0)`**

2. for **each row** the **state function** is called with current state + column value - returns **new state**

3. After all rows the **final function** is called with **last state**

4. **final function** calculates the **aggregate**

UDF/UDA

Now everybody can execute evil code on your cluster :)

PERMISSIONS ON UDFS

- There will be permissions to restrict (allow)
  - UDF creation (DDL)
  - UDF execution (DML)

SOME FINAL WORDS…

Keep in mind:

- JSR-223 has overhead - Java UDFs are much faster
- Do not allow everyone to create UDFs (in production)
- Keep your UDFs “**pure**“
- **Test** your UDFs and user defined aggregates **thoroughly**

MORE FINAL WORDS…

- **UDFs** and user defined aggregates are **executed** on the **coordinator node**
- **Prefer** to use **Java-UDFs** for performance reasons

DREAMS

UDFs could be useful for…

- Functional indexes
- Partial indexes
- Filtering
- Distributed GROUP BY
- etc etc

NOT IN C*
3.0 !

Role based access control

CASSANDRA 3.0

ROLE BASED ACCESS CONTROL

- Grant/revoke permissions to/from roles
- Grant roles to users

ROLE BASED ACCESS CONTROL

```
GRANT <permission> ON <resource>
TO [[USER] <username> | ROLE <rolename>]

REVOKE <permission> ON <resource>
FROM [[USER] <username> | ROLE <rolename>]

LIST <permissionOrAll> [ON <resource>]
[OF [[USER] <username> | ROLE <rolename>]
[NORECURSIVE]
```

MORE NEW AUTHENTICATION/AUTHORIZATION FEATURES

- Authentication/authorization has been reworked for Cassandra 3.0
- Much more options and possibilities
- See CASSANDRA-8394 for more information

New row cache

CASSANDRA 3.0

MOTIVATION NEW ROW CACHE

- Old row cache recommendation:
  "don't use it"
- Old row cache had data in off-heap,
  but management data in Java heap
- Resulted in a lot of additional GC pressure

NEW ROW CACHE

- All data (whole concurrent hash map) is off-heap
- Uses APL2 licensed https://github.com/snazy/ohc
- Works with really big row cache
- Works on really big machines

- But don’t expect a huge performance improvement
- Serialization of data to/from off-heap is still a bottleneck
- Will work on that bottleneck in future versions

More improvements

CASSANDRA 3.0

INTERNAL MODERNIZATION IN CASSANDRA 3.0

- Refactor and modernize storage engine (CASSANDRA-8099)
- Modernize schema tables (CASSANDRA-6717)

- CQL row read optimization
- Reduce GC pressure
- Make internal nomenclature intuitive

MORE PERFORMANCE IMPROVEMENTS IN CASSANDRA 3.0

Memory related

- Support direct buffer decompression for reads (CASSANDRA-8464)
- Avoid memory allocation when searching index summary (CASSANDRA-8793)
- Use preloaded jemalloc w/ Unsafe (CASSANDRA-8714)

Throughput/CPU related

- Improve concurrency of repair (CASSANDRA-6455, 8208)
- Select optimal CRC32 implementation at runtime (CASSANDRA-8614)

- plus many more

SOMETHING ELSE

Windows

- Cassandra 3.0 is tested on Windows
- 3.0 works definitely better on Windows that 2.1
- But still some issues to solve

My <u>personal</u> recommendation

- Use Cassandra on Linux

CASSANDRA >= 3.1 FEATURES

CASSANDRA FUTURE

- Global indexes
- More UDF related stuff
    - Function based indexes
    - Use UDFs in filtering clauses
    - Distributed aggregates
- RAMP transactions
- More internal improvements and optimizations
- Expect Thrift to disappear

- Remember:
  Everything is subject to change if not released ;)

CASSANDRA COMMUNITY

CASSANDRA COMMUNITY

- People writing code (of course ;))
- People doing talks and presentations
- People active on social media (Twitter, LinkedIn, SlideShare, YouTube)
- People active on mailing list
- People working in the background

- AND YOU !

HINTS

- Use the material provided by
  - DataStax
  - Planet Cassandra
  - ”the community”
- on/via
  - YouTube
  - SlideShare
  - DataStax academy
  - Webinars
  - Meetups

APACHE CASSANDRA "VS." DATASTAX ENTERPRISE

APACHE CASSANDRA VS. DATASTAX ENTERPRISE

- **Apache Cassandra** is open-source
- Support via user mailing list and tickets
- No commercial support

- **DataStax Enterprise** uses Apache Cassandra
- Adds graph database
- Adds enhanced security
- Adds analytics (Spark + Hadoop)
- Adds search (SolR)
- Adds commercial support

ONE MORE THING...

:)

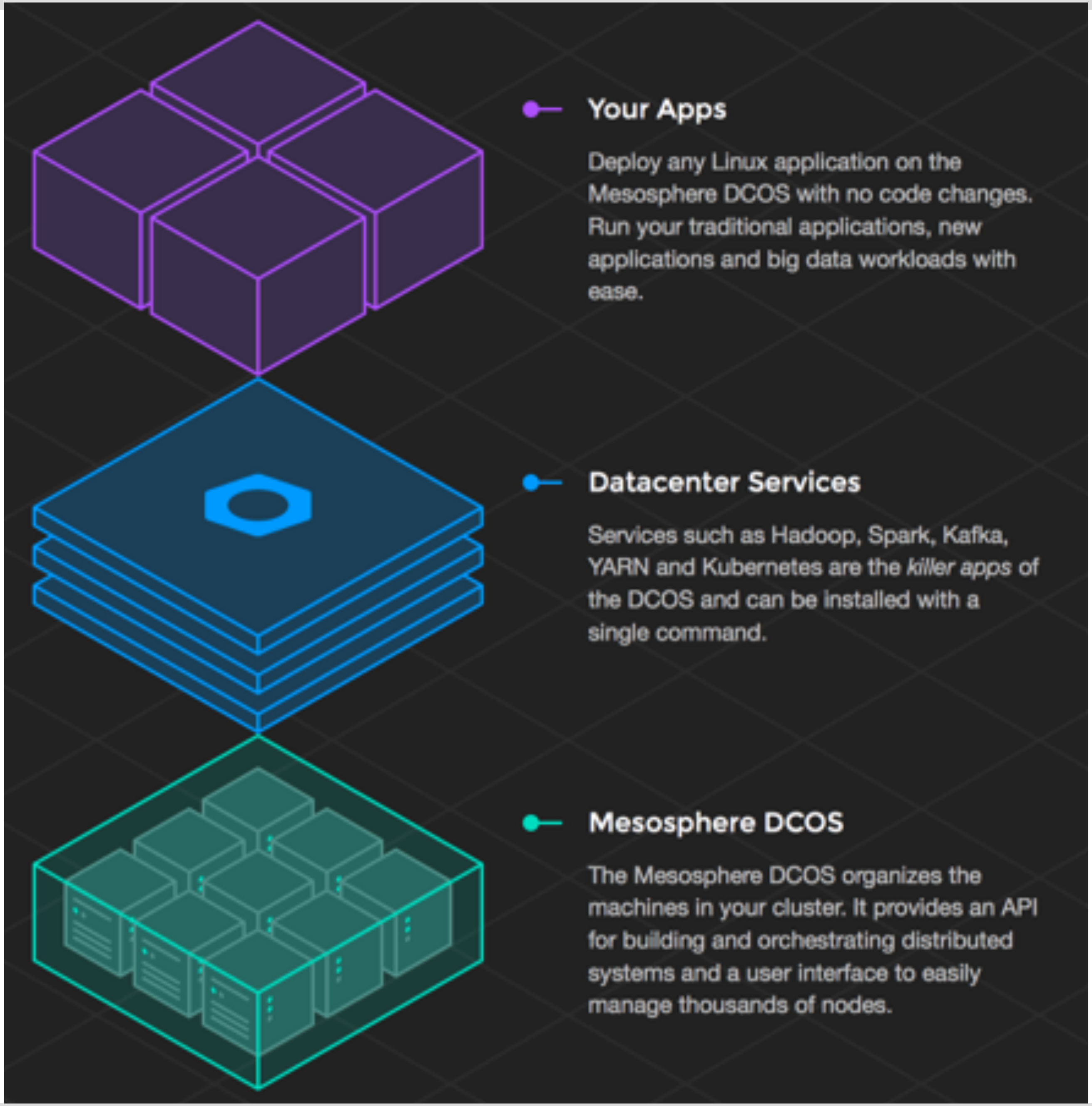
CASSANDRA-ON-MESOS

APACHE MESOS

Program against your **datacenter** like it’s a **single pool of resources**

Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively.

MESOSPHERE DCOS



Your Apps

Deploy any Linux application on the Mesosphere DCOS with no code changes. Run your traditional applications, new applications and big data workloads with ease.

Datacenter Services

Services such as Hadoop, Spark, Kafka, YARN and Kubernetes are the *killer apps* of the DCOS and can be installed with a single command.

Mesosphere DCOS

The Mesosphere DCOS organizes the machines in your cluster. It provides an API for building and orchestrating distributed systems and a user interface to easily manage thousands of nodes.

APACHE MESOS

- Run killer applications and services like Spark, Kafka and Cassandra

- In your own data center
- On Amazon EC2
- On Google GCE

CASSANDRA-ON-MESOS

- Allows to run Apache Cassandra on Apache Mesos
- Developed by mesosphere and contentteam
- Allows to spawn your Cassandra Cluster with a single command on Mesos
- Expect a first release-candidate this week !

Q & A

Robert Stupp
rstupp@contentteam.com
@snazy