



TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies



CASSANDRA

Lukáš Matějů
20.5.2024 | DPB



ČÁST I.: OPAKOVÁNÍ



OPAKOVÁNÍ

- Cassandra prakticky

- vkládání

- INSERT INTO, COPY
 - i z JSON formátu
 - i do kolekcí
 - set, list, map

```
INSERT INTO activity (home_id, datetime, event,  
code_used) VALUES ('H01474777', '2014-05-21 07:32:16',  
'alarm set', '5599');
```

```
cqlsh> SELECT * FROM cycling.cyclist_category;
```

```
cqlsh> SELECT category, points, lastname FROM cycling.cyclist_category;
```

- čtení

- SELECT

- WHERE klauzule pro podmínky
 - z pravidla musí obsahovat partition key
 - případně clustering columns

```
UPDATE home_security.home SET phone = '310-883-7197'  
WHERE home_id = 'H01474777';
```

```
UPDATE home_security.home SET phone = '310-883-7197',  
contact_name = 'Mr. Drysdale' WHERE home_id = 'H01474777';
```

- aktualizace

- UPDATE, ALTER

- UPDATE je jen další zápis

```
cqlsh> DELETE FROM cycling.calendar WHERE race_id = 200;
```

```
TRUNCATE cycling.user_activity;
```

- odstranění

- DELETE, TRUNCATE, DROP

```
DROP TABLE cycling.cyclist_name;
```

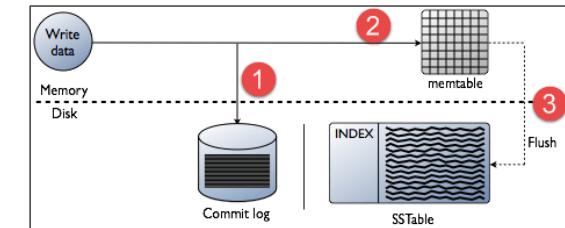
<https://www.udemy.com/course/apache-cassandra>

https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useQueryColumnsSort.html



OPAKOVÁNÍ

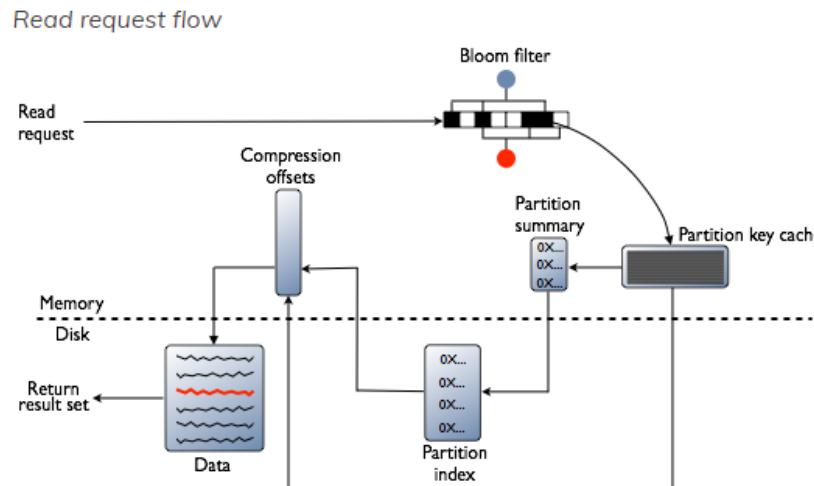
- ukládání dat na disk
 - při zápisu dat do tabulky jsou data zapsána do commit logu a do paměti
 - commit log
 - uložený na disku
 - slouží pro obnovu dat v případě selhání uzlu (durability)
 - po zápisu do SSTables archivován, smazán nebo recyklován
 - paměť (memtable)
 - mezipaměť zpětného zápisu
 - po naplnění je zapsána na disk jako SSTables
 - existuje pro každou tabulku na každém uzlu
 - SSTable
 - setříděná string tabulka uložená na disku
 - obsah je možné zobrazit pomocí .\tools\bin\sstabledump
 - vynutit zápis z memcache do SSTable je možné přes .\bin\nodetool flush <jméno>



<https://www.guru99.com/cassandra-architecture.html>

OPAKOVÁNÍ

- čtení dat
 - Cassandra musí kombinovat výsledky z aktuální memtable a z SSTables
 - zpracování v několika fázích



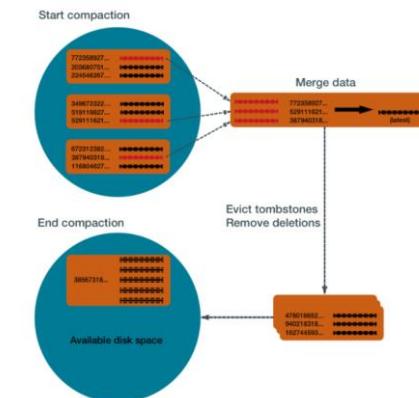
- Check the memtable
- Check row cache, if enabled
- Checks Bloom filter
- Checks partition key cache, if enabled
- Goes directly to the compression offset map if a partition key is found in the partition key cache, or checks the partition summary if not
- If the partition summary is checked, then the partition index is accessed
- Locates the data on disk using the compression offset map
- Fetches the data from the SSTable on disk

<https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlAboutReads.html>



OPAKOVÁNÍ

- odstranění dat
 - při zadání příkazu pro odstranění je vytvořen tzv. tombstone
 - označuje data pro smazání
 - proč ale nejsou data smazána ihned?
 - při okamžitém smazání by mohlo dojít ke zpětné replikaci z jiného uzlu
 - uzly (i ty momentálně neběžící) tak dostávají čas, aby se o označení dozvěděly
 - doba mezi označením a možným smazáním dána ve vlastnostech tabulky
 - gc_grace_seconds (v základu 10 dní)
 - pro samotné smazání musí nastat compaction
 - compaction nastává při sloučení SSTables
 - zlepšení výkonu čtení z důvodu menšího počtu SSTables
 - navrácení diskové kapacity smazáním dat
 - prováděno automaticky



<https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/dml/dmlHowDataMaintain.html>

OPAKOVÁNÍ

- sekundární indexy
 - možnost definovat nad vybranými sloupcí
 - kromě partition key a clustering columns, které už indexované jsou
 - umožňují filtrování nad danými sloupcí
 - použít WHERE klauzule
 - pro každý sekundární index je vytvořena skrytá tabulka na každém uzlu
 - pro přístup
- nezrychlují prohledávání, ale umožňují jej i nad ostatními sloupcí
 - pro zrychlení je možnost vytvořit tabulku přímo odpovídající dotazu
 - CREATE INDEX <jméno indexu> ON <jméno tabulky> (<jméno sloupce>)
 - nad libovolným sloupcem
 - možnost definovat i nad kolekcemi
 - po vytvoření použití automatické



OPAKOVÁNÍ

- vybraná téma
 - statické sloupce
 - sloupce, jejichž hodnoty jsou konstantní pro všechny řádky v partition
 - počítadla
 - agregační funkce
 - funkce definované uživatelem
 - lightweight transakce
 - IF NOT EXISTS, IF
 - materialized view
 - speciální tabulka s daty, která jsou vkládána a aktualizována podle zdrojové tabulky
 - tabulky se liší primárním klíčem a vlastnostmi
 - umožňuje provádět jiné optimalizované dotazy
 - a zároveň udržuje data aktuální podle zdrojové tabulky

```
CREATE TABLE courses (
    id varchar,
    name varchar STATIC,
    module_id int,
    module_name varchar,
    PRIMARY KEY (id, module_id)
);
```

Insert

```
INSERT INTO users (id, first_name, last_name)
VALUES ('john-doe', 'John', 'Doe')
IF NOT EXISTS;
```

<https://app.pluralsight.com/library/courses/cassandra-developers>





ČÁST II.: CASSANDRA V CLUSTERU





HARDWAROVÉ POŽADAVKY

- není vyžadován žádný speciální hardware
- je ale potřeba zajistit alespoň minimální doporučené požadavky
 - pro paměť (RAM), CPU, úložiště
 - výkonná distribuovaná databáze
 - paměť
 - doporučeno pro každý uzel alespoň 8 GB
 - v produkci běžně 32 GB na uzel
 - více paměti zlepšuje výkon operací čtení
 - nacachování více dat vede k nižší potřebě přístupovat na disk
 - procesor
 - Cassandra je schopná efektivně využívat více jader
 - minimální počet doporučených jader je 4
 - v produkci běžně 8 a více



HARDWAROVÉ POŽADAVKY

- je ale potřeba zajistit alespoň minimální doporučené požadavky
 - pro paměť (RAM), CPU, úložiště
 - úložiště
 - velký vliv na výkon
 - potřeba zvážit typ a kapacitu disku a jestli je úložiště sdílené
 - vhodné nepoužívat sdílené disky
 - obsluha ostatních databází zpomalí obsluhu specifické databáze
 - SSD jsou preferované
 - rychlejší než klasické disky, i ty je ale možné bez problémů použít
 - z hlediska výkonu je většinou lepší mít více uzlů s méně daty než pár uzlů s hromadou dat
 - doporučené množství dat na uzel se pohybuje mezi 500 GB až 1 TB
 - liší se podle typu disku
 - pro zlepšení výkonu zápisu je vhodný samostatný disk pro commit log



HARDWAROVÉ POŽADAVKY

- je ale potřeba zajistit alespoň minimální doporučené požadavky
 - [DataStax Enterprise Reference Architecture](#)

Low End Specification (Cassandra only)

Server	Intel Xeon or AMD modern processor (64 bit) 4 cores or more
RAM	**16 GB to 32 GB of ECC DRAM per node
Storage	SATA III hard disks or SAS drives
Network	Gigabit Ethernet
Data Size	30 GB to 500 GB per node on SATA III Up to 1 TB per node on SAS

High End Specification (Cassandra only)

Server	Intel Xeon or AMD modern processors (64 bit) 8 cores or more
RAM	128 GB or 256 GB ECC DRAM per node
Storage	SATA III hard disks or SAS drives or SSDs
Network	10-Gigabit Ethernet
Data Size	300 GB to 1 TB or more per node Up to 5 TB per node using SSD exclusively, for key-based access pattern

<http://www.odbms.org/wp-content/uploads/2014/06/WP-DataStax-Enterprise-Reference-Architecture-2014.pdf>





PŘÍDÁNÍ UZLŮ DO CLUSTERU

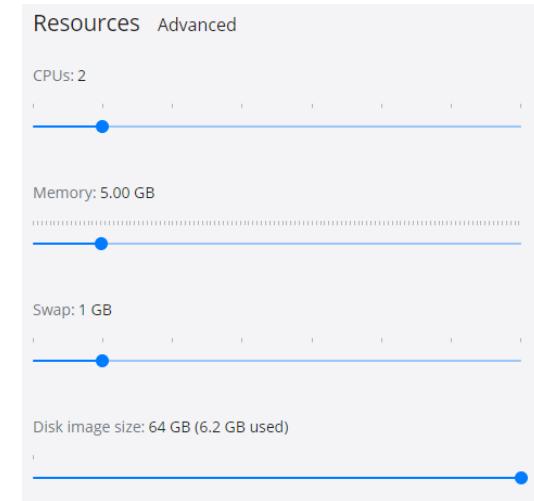
- každý uzel má stejnou funkcionalitu
 - uzly jsou nezávislé, ale propojené s ostatními
 - mohou přijímat dotazy na čtení i zápis
 - při výpadku uzlu dotazy obslouženy z jiných uzlů
- přidání uzlu (bootstrapping) do clusteru je snadné
 - uzly musí být v síti, která jim umožňuje komunikaci
 - rychlejší komunikace znamená vyšší výkon
 - porty využívané Cassandrou
 - 7000 – vnitřní komunikace
 - 9042 – nativní port
 - 7199 – JMX monitoring
 - potřeba zajistit
 - stejné jméno clusteru
 - IP adresu a síťový přístup k alespoň jednomu z uzlů clusteru (seed node)



PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - využívá originální Cassandra image
 - cluster
 - 3 uzly, 2 v prvním datacentru, 1 ve druhém
 - jména uzlů: cas1, cas2, cas3
 - jména datacenter: datacenter1, datacenter2
 - použití GossipingPropertyFileSnitch
 - umožňuje replikaci mezi datacentry
 - info předáváno Dockeru pomocí parametru -e
 - potřeba alespoň 5 GB RAM (1,5 GB na uzel + obsluha)
 - stažení docker image

```
docker pull cassandra
```



```
D:\Prezentace>docker pull cassandra
Using default tag: latest
latest: Pulling from library/cassandra
Digest: sha256:bacc4f8e419a18c23ea56fb7501367bd91ae10d12f53ea249251f86d7706924a
Status: Image is up to date for cassandra:latest
docker.io/library/cassandra:latest
```



PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - spuštění prvního uzlu (vytvoření clusteru)

```
docker run --name cas1 -p 9042:9042 -e CASSANDRA_CLUSTER_NAME=MyCluster  
-e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e CASSANDRA_DC=datacenter1 -d cassandra
```

```
D:\Prezentace>docker exec -ti cas1 nodetool status  
Datacenter: datacenter1  
=====  
Status=Up/Down  
|/ State=Normal/Leaving/Joining/Moving  
-- Address     Load      Tokens    Owns (effective)  Host ID          Rack  
UN 172.17.0.2  70.71 KiB  256       100.0%           82811efa-4821-49c5-ad66-3c7fe2154df9  rack1
```

- -p publikuje port kontejneru hostiteli
 - umožní použití zvenčí
- pro přidání dalších uzlů potřeba zjistit IP adresu prvního uzlu

```
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" cas1
```

```
D:\Prezentace>docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" cas1  
172.17.0.2
```





PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - přidání druhého uzlu

```
docker run --name cas2 -e CASSANDRA_SEEDS=172.17.0.2 -e CASSANDRA_CLUSTER_NAME=MyCluster -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e CASSANDRA_DC=datacenter1 -d cassandra
```

- navíc parametr CASSANDRA_SEEDS odkazující na IP adresu prvního uzlu
- přidání do stejného datacentra jako první uzel

```
D:\Prezentace>docker exec -ti cas1 nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID            Rack
UN  172.17.0.3   75.83 KiB  256        100.0%           6a05ed73-667b-4842-ae9d-e446cbbaebba  rack1
UN  172.17.0.2   75.72 KiB  256        100.0%           82811efa-4821-49c5-ad66-3c7fe2154df9  rack1
```



PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - seed node (uzel)
 - běžný uzel
 - umožňuje skrze IP adresu připojení dalších uzlů do clusteru
 - IP adresa alespoň jednoho seed uzlu je potřeba pro přidání dalších uzlů
 - definovány v seeds vlastnosti v Cassandra.yaml jako seznam oddělený čárkami

```
seeds: "192.168.159.101,192.168.159.102"
```

- často více než jeden záznam pro případ výpadku jednoho z uzlů
- není potřeba mít uvedené všechny uzly

```
docker run --name cas2 -e CASSANDRA_SEEDS=172.17.0.2 -e CASSANDRA_CLUSTER_NAME=MyCluster -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e CASSANDRA_DC=datacenter1 -d cassandra
```



PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - přidání třetího uzlu do jiného datacentra

```
docker run --name cas3 -e CASSANDRA_SEEDS=172.17.0.2 -e CASSANDRA_CLUSTER_NAME=MyCluster -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e CASSANDRA_DC=datacenter2 -d cassandra
```

```
D:\Prezentace>docker exec -ti cas1 nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID            Rack
UN 172.17.0.3   75.83 KiB  256       63.0%           6a05ed73-667b-4842-ae9d-e446cbbabba  rack1
UN 172.17.0.2   95.02 KiB  256       72.0%           82811efa-4821-49c5-ad66-3c7fe2154df9  rack1
Datacenter: datacenter2
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID            Rack
UN 172.17.0.4   75.86 KiB  256       64.9%           c784b328-6fff-42fc-a56a-001baec8e420  rack1
```

- po přidání se uzel začne starat o svoje rozsahy tokenů
- nejprve začne přijímat požadavky na zápis
- následně i dotazy na čtení





PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - vytvoření keyspace

```
docker exec -ti casl cqlsh

CREATE KEYSPACE mykeyspace WITH replication = {
    'class' : 'NetworkTopologyStrategy',
    'datacenter1' : 1,
    'datacenter2' : 1
};
```

- NetworkTopologyStrategy umožňuje umístění ve více datových centrech
 - 1 kopie dat v každém datacentru
- vytvoření tabulky

```
CREATE TABLE mykeyspace.mytable (
    id int PRIMARY KEY,
    name text
);
```





PŘÍDÁNÍ UZLŮ DO CLUSTERU

- praktická ukázka pomocí Dockeru
 - distribuce keyspace na clusteru

```
D:\Prezentace>docker exec -ti casl nodetool status mykeyspace
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID            Rack
UN  172.17.0.3   253.25 KiB  256      46.1%          6a05ed73-667b-4842-ae9d-e446cbbaebba  rack1
UN  172.17.0.2   297.64 KiB  256      53.9%          82811efa-4821-49c5-ad66-3c7fe2154df9  rack1
Datacenter: datacenter2
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load      Tokens  Owns (effective)  Host ID            Rack
UN  172.17.0.4   247.19 KiB  256     100.0%         c784b328-6ff8-42fc-a56a-001baec8e420  rack1
```

- 1 kopie dat v každém datacentru
 - uzly v datacentru 1 se o replikovaná data dělí
 - datacentrum 2 obsahuje celá replikovaná data



PŘÍDÁNÍ UZLŮ DO CLUSTERU

- úklid
 - po připojení nového uzlu a převzetí části dat z ostatních uzlů nejsou data na původních uzlech odstraněna
 - důvodem je zachování dat v případě, že by došlo k brzkému selhání nového uzlu
 - pro odstranění dat je možné použít nodetool cleanup

```
nodetool [connection_options] cleanup
[-j num_jobs] [--] [keyspace_name table_name [table_name ...]]
```

https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/tools/nodetool/toolsCleanup.html

```
bin/nodetool -h 192.168.159.101 cleanup
```

<https://www.udemy.com/course/apache-cassandra>





CASSANDRA-STRESS

- nástroj pro stress-testing
 - součást Cassandra
 - ./tools/bin/cassandra-stress
 - čtení i zápis dat

```
cassandra-stress write n=1000000 cl=one -mode native cql3 -node 172.17.0.2
sh cassandra-stress write n=1000000 cl=one -mode native cql3 -node 172.17.0.2 -schema "replication(strategy=NetworkTopologyStrategy, datacenter1=1, datacenter2=1)"
```

- příkaz vytvoří keyspace s replikačním faktorem 1 s několika tabulkami (podle parametrů)
- následně zapíše milion řádků na uzel s IP adresou 172.17.0.2
- uzel data rozdistribuuje mezi zbylé uzly clusteru

```
Connected to cluster: MyCluster, max pending requests per connection 128, max connections per host 8
Datacenter: datacenter1; Host: /172.17.0.3; Rack: rack1
Datacenter: datacenter1; Host: /172.17.0.2; Rack: rack1
Datacenter: datacenter2; Host: /172.17.0.4; Rack: rack1
Created keyspaces. Sleeping 1s for propagation.
Sleeping 2s...
Warming up WRITE with 50000 iterations...
Failed to connect over JMX; not collecting these stats
Running WRITE with 200 threads for 1000000 iteration
Failed to connect over JMX; not collecting these stats
type      total ops,    op/s,   pk/s,   row/s,   mean,   med,   .95,   .99,   .999,   max,   time,   stderr, errors, gc: #,   max ms,   sum ms,   sdv ms,   mb
total,        835,     835,     835,    45.9,   32.1,  121.4,  198.6,  216.4,  227.7,  1.0,  0.0000,    0,    0,    0,    0,    0
total,      4075,    3240,    3240,    54.2,   49.9,  115.6,  145.5,  208.0,  238.7,  2.0,  0.42062,    0,    0,    0,    0,    0
total,      7819,    3744,    3744,    59.4,   45.8,  169.2,  284.4,  311.7,  316.9,  3.0,  0.29686,    0,    0,    0,    0,    0
```



CASSANDRA-STRESS

- nástroj pro stress-testing
 - rozložení zobrazitelné přes nodetool status

```
C:\Users\Rimmer>docker exec -ti cas1 cqlsh
Connected to MyCluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.10 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> DESCRIBE KEYSPACES

system_schema  system      keyspace1      system_traces
system_auth     mykeyspace  system_distributed
```

```
Results:
Op rate           : 5,120 op/s [WRITE: 5,120 op/s]
Partition rate   : 5,120 pk/s [WRITE: 5,120 pk/s]
Row rate          : 5,120 row/s [WRITE: 5,120 row/s]
Latency mean     : 38.8 ms [WRITE: 38.8 ms]
Latency median   : 32.4 ms [WRITE: 32.4 ms]
Latency 95th percentile : 85.3 ms [WRITE: 85.3 ms]
Latency 99th percentile : 170.1 ms [WRITE: 170.1 ms]
Latency 99.9th percentile : 311.4 ms [WRITE: 311.4 ms]
Latency max       : 416.0 ms [WRITE: 416.0 ms]
Total partitions  : 1,000,000 [WRITE: 1,000,000]
Total errors      : 0 [WRITE: 0]
Total GC count    : 0
Total GC memory   : 0.000 KiB
Total GC time     : 0.0 seconds
Avg GC time       : NaN ms
StdDev GC time    : 0.0 ms
Total operation time : 00:03:15
```

```
D:\>docker exec -ti cas1 nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens      Owns (effective)  Host ID      Rack
UN 172.17.0.3  115.89 MiB  256        50.2%          fb9be963-cee3-4b1a-ba02-2c2f0d5bd254  rack1
UN 172.17.0.2  116.55 MiB  256        49.8%          b4f2b578-1eb6-4ca9-bbca-430f52f0a481  rack1
Datacenter: datacenter2
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens      Owns (effective)  Host ID      Rack
UN 172.17.0.4  233.51 MiB  256        100.0%         0167ac14-af9a-4678-a9ef-d52ca0441537  rack1
```





CASSANDRA-STRESS

```
cqlsh> DESCRIBE KEYSPACE keyspace1
CREATE KEYSPACE keyspace1 WITH replication = {'class': 'NetworkTopologyStrategy', 'datacenter1': '1', 'datacenter2': '1'} AND durable_writes = true;
CREATE TABLE keyspace1.counter1 (
    key blob,
    column1 text,
    "C0" counter static,
    "C1" counter static,
    "C2" counter static,
    "C3" counter static,
    "C4" counter static,
    value counter,
    PRIMARY KEY (key, column1)
) WITH COMPACT STORAGE
    AND CLUSTERING ORDER BY (column1 ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'enabled': 'false'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
CREATE TABLE keyspace1.standard1 (
    key blob PRIMARY KEY,
    "C0" blob,
    "C1" blob,
    "C2" blob,
    "C3" blob,
    "C4" blob
) WITH COMPACT STORAGE
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'enabled': 'false'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

- spousta možností customizace



CASSANDRA-STRESS

- nástroj pro stress-testing
 - read testuje čtení, mixed čtení i zápis

```
cassandra-stress mixed n=1000000 cl=one -mode native cql3 -node 172.17.0.2
```

		id, type	total ops,	op/s,	pk/s,	row/s,	mean,	med,	.95,	.99,	.999,	max,	time,	stderr,
Results:	:	4,529 op/s [READ: 2,263 op/s, WRITE: 2,265 op/s]												
Op rate	:	4,529 op/s [READ: 2,263 op/s, WRITE: 2,265 op/s]												
Partition rate	:	4,529 pk/s [READ: 2,263 pk/s, WRITE: 2,265 pk/s]												
Row rate	:	4,529 row/s [READ: 2,263 row/s, WRITE: 2,265 row/s]												
Latency mean	:	25.8 ms [READ: 25.8 ms, WRITE: 25.8 ms]												
Latency median	:	22.2 ms [READ: 22.3 ms, WRITE: 22.2 ms]												
Latency 95th percentile	:	54.7 ms [READ: 54.6 ms, WRITE: 54.9 ms]												
Latency 99th percentile	:	91.0 ms [READ: 91.8 ms, WRITE: 90.0 ms]												
Latency 99.9th percentile	:	156.4 ms [READ: 156.4 ms, WRITE: 156.4 ms]												
Latency max	:	222.3 ms [READ: 222.3 ms, WRITE: 220.6 ms]												
Total partitions	:	100,000 [READ: 49,981, WRITE: 50,019]												
Total errors	:	0 [READ: 0]												
Total GC count	:	0	4 threadCount, READ,	50258,	1198,	1198,	1.7,	1.2,	4.4,	8.9,	21.5,	112.3,	41.9,	0.03829,
Total GC memory	:	0.000 KIB	4 threadCount, WRITE,	49742,	1186,	1186,	1.5,	1.1,	4.0,	7.8,	20.8,	113.0,	41.9,	0.03829,
Total GC time	:	0.0 seconds	4 threadCount, total,	100000,	2384,	2384,	1.6,	1.1,	4.2,	8.4,	21.1,	113.0,	41.9,	0.03829,
Avg GC time	:	NaN ms	8 threadCount, READ,	49519,	1757,	1757,	2.2,	1.7,	5.4,	9.6,	25.6,	70.0,	28.2,	0.04192,
StdDev GC time	:	0.0 ms	8 threadCount, WRITE,	50481,	1791,	1791,	2.1,	1.6,	5.2,	9.4,	22.7,	68.2,	28.2,	0.04192,
Total operation time	:	00:00:22	8 threadCount, total,	100000,	3548,	3548,	2.2,	1.6,	5.3,	9.5,	23.9,	70.0,	28.2,	0.04192,
			16 threadCount, READ,	49630,	1753,	1753,	4.5,	3.4,	11.6,	20.9,	49.9,	94.4,	28.3,	0.02565,
			16 threadCount, WRITE,	50370,	1779,	1779,	4.4,	3.3,	11.5,	21.2,	50.8,	96.6,	28.3,	0.02565,
			16 threadCount, total,	100000,	3532,	3532,	4.5,	3.3,	11.5,	21.1,	50.4,	96.6,	28.3,	0.02565,
			24 threadCount, READ,	50391,	2015,	2015,	5.9,	4.7,	14.5,	24.7,	57.2,	144.4,	25.0,	0.05086,
			24 threadCount, WRITE,	49609,	1984,	1984,	5.8,	4.6,	14.3,	24.5,	51.9,	149.9,	25.0,	0.05086,
			24 threadCount, total,	100000,	3999,	3999,	5.9,	4.7,	14.4,	24.6,	55.3,	149.9,	25.0,	0.05086,
			36 threadCount, READ,	51000,	2324,	2324,	7.9,	6.4,	18.3,	32.2,	73.8,	117.0,	21.9,	0.01578,
			36 threadCount, WRITE,	49000,	2233,	2233,	7.8,	6.4,	18.2,	31.8,	74.2,	124.7,	21.9,	0.01578,
			36 threadCount, total,	100000,	4556,	4556,	7.8,	6.4,	18.3,	31.9,	74.2,	124.7,	21.9,	0.01578,
			54 threadCount, READ,	50376,	2235,	2235,	11.6,	9.8,	26.1,	43.3,	82.6,	119.8,	22.5,	0.04700,
			54 threadCount, WRITE,	49624,	2202,	2202,	11.5,	9.8,	25.5,	42.1,	82.1,	123.3,	22.5,	0.04700,
			54 threadCount, total,	100000,	4438,	4438,	11.6,	9.8,	25.9,	42.9,	82.6,	123.3,	22.5,	0.04700,
			81 threadCount, READ,	49963,	2212,	2212,	17.8,	15.3,	39.1,	66.5,	105.8,	143.3,	22.6,	0.03032,
			81 threadCount, WRITE,	50037,	2215,	2215,	17.7,	15.2,	39.0,	65.0,	103.5,	149.3,	22.6,	0.03032,
			81 threadCount, total,	100000,	4426,	4426,	17.7,	15.3,	39.1,	65.9,	104.9,	149.3,	22.6,	0.03032,
			121 threadCount, READ,	49981,	2263,	2263,	25.8,	22.3,	54.6,	91.8,	156.4,	222.3,	22.1,	0.82039,
			121 threadCount, WRITE,	50019,	2265,	2265,	25.8,	22.2,	54.9,	90.0,	156.4,	220.6,	22.1,	0.82039,
			121 threadCount, total,	100000,	4529,	4529,	25.8,	22.2,	54.7,	91.0,	156.4,	222.3,	22.1,	0.82039,



MONITOROVÁNÍ CLUSTERU

- různé nástroje pro monitorování
 - nodetool
 - nástroj přibalený ke Cassandra
 - z příkazové řádky
 - JConsole
 - nástroj obsažený v JDK (Java Development Kit)
 - OpsCenter
 - GUI aplikace od Datastax
 - nástroje komunikují s Cassandra přes JMX
 - Java Management Extensions
 - Java technologie umožňující monitorování a správu Java aplikací a služeb
 - přes JMX Cassandra zpřístupňuje různé metriky a příkazy využitelné k monitoringu a správě clusteru



MONITOROVÁNÍ CLUSTERU

- nodetool
 - nástroj přibalený ke Cassandře
 - ./bin/nodetool
 - spousta možností pro monitorování

```
C:\Users\Rimmer>docker exec -ti cas1 nodetool
usage: nodetool [(-pfw <passwordFilePath> | --password-file <passwordFilePath>)]
                [(-u <username> | --username <username>)]
                [(-pw <password> | --password <password>)] [(-h <host> | --host <host>)]
                [(-p <port> | --port <port>)] <command> [<args>]

The most commonly used nodetool commands are:
  assassinate          Forcefully remove a dead node without re-replicating any data. Use as a last resort if you cannot removenode
  bootstrap            Monitor/manage node's bootstrap process
  cleanup              Triggers the immediate cleanup of keys no longer belonging to a node. By default, clean all keyspaces
  clearsnapshot        Remove the snapshot with the given name from the given keyspaces. If no snapshotName is specified we will remove all snapshots
  compact              Force a (major) compaction on one or more tables or user-defined compaction on given SSTables
  compactionhistory   Print history of compaction
  compactionstats     Print statistics on compactions
  decommission         Decommission the *node I am connecting to*
  describecluster     Print the name, snitch, partitioner and schema version of a cluster
  describing           Shows the token ranges info of a given keyspace
  disableautocompaction Disable autocompaction for the given keyspace and table
  disablebackup        Disable incremental backup
  disablebinary        Disable native transport (binary protocol)
  disablegossip       Disable gossip (effectively marking the node down)
  disablehandoff      Disable storing hinted handoffs
  disablehintsfordc  Disable hints for a data center
  disablethrift       Disable thrift server
  drain               Drain the node (stop accepting writes and flush all tables)
  enableautocompaction Enable autocompaction for the given keyspace and table
  enablebackup         Enable incremental backup
  enablebinary         Reenable native transport (binary protocol)
  enablegossip        Reenable gossip
  enablehandoff       Reenable future hints storing on the current node
  enablehintsfordc   Enable hints for a data center that was previously disabled
  enablethrift        Reenable thrift server
  failuredetector    Shows the failure detector information for the cluster
```



MONITOROVÁNÍ CLUSTERU

- nodetool
 - nodetool status
 - monitorování stavu clusteru

```
C:\Users\Rimmer>docker exec -ti cas1 nodetool status
Datacenter: datacenter1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load    Tokens     Owns      Host ID            Rack
UN  172.17.0.3   56.28 MiB  256        ?  6a05ed73-667b-4842-ae9d-e446cbbabba  rack1
UN  172.17.0.2   84.34 MiB  256        ?  82811efa-4821-49c5-ad66-3c7fe2154df9  rack1
Datacenter: datacenter2
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load    Tokens     Owns      Host ID            Rack
UN  172.17.0.4   85.12 MiB  256        ?  c784b328-6ff8-42fc-a56a-001baec8e420  rack1
```

- nodetool ring
 - zobrazí rozsahy tokenů pro jednotlivé uzly
- nodetool tablestats
 - zobrazí statistiky pro jednotlivé keyspace a tabulky
- nodetool tablehistograms



MONITOROVÁNÍ CLUSTERU

- nodetool
 - nodetool info
 - informace o konkrétním uzlu

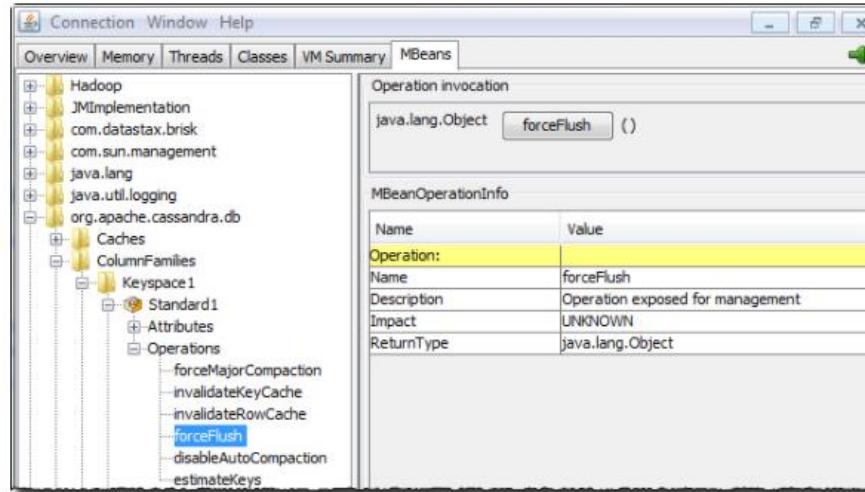
```
C:\Users\Rimmer>docker exec -ti casl nodetool info
ID : 82811efa-4821-49c5-ad66-3c7fe2154df9
Gossip active : true
Thrift active : false
Native Transport active: true
Load : 84.2 MiB
Generation No : 1619009893
Uptime (seconds) : 157043
Heap Memory (MB) : 746.83 / 1216.00
Off Heap Memory (MB) : 0.48
Data Center : datacenter1
Rack : rack1
Exceptions : 0
Key Cache : entries 18058, size 1.38 MiB, capacity 60 MiB, 15644 hits, 34125 requests, 0.458 recent hit rate, 14400 save period in seconds
Row Cache : entries 0, size 0 bytes, capacity 0 bytes, 0 hits, 0 requests, NaN recent hit rate, 0 save period in seconds
Counter Cache : entries 0, size 0 bytes, capacity 30 MiB, 0 hits, 0 requests, NaN recent hit rate, 7200 save period in seconds
Chunk Cache : entries 16, size 1 MiB, capacity 272 MiB, 325 misses, 998 requests, 0.674 recent hit rate, NaN microseconds miss latency
Percent Repaired : 100.0%
Token : (invoke with -T--tokens to see all 256 tokens)
```

- nodetool netstats
- nodetool tsststats
- nodetool help
 - pro zobrazení dalších možností



MONITOROVÁNÍ CLUSTERU

- JConsole
 - nástroj obsažený v JDK
 - jconsole v příkazové řádce
 - umožňuje nahlízení do Java procesů
 - po připojení poskytuje interface ke zpřístupněným metrikám

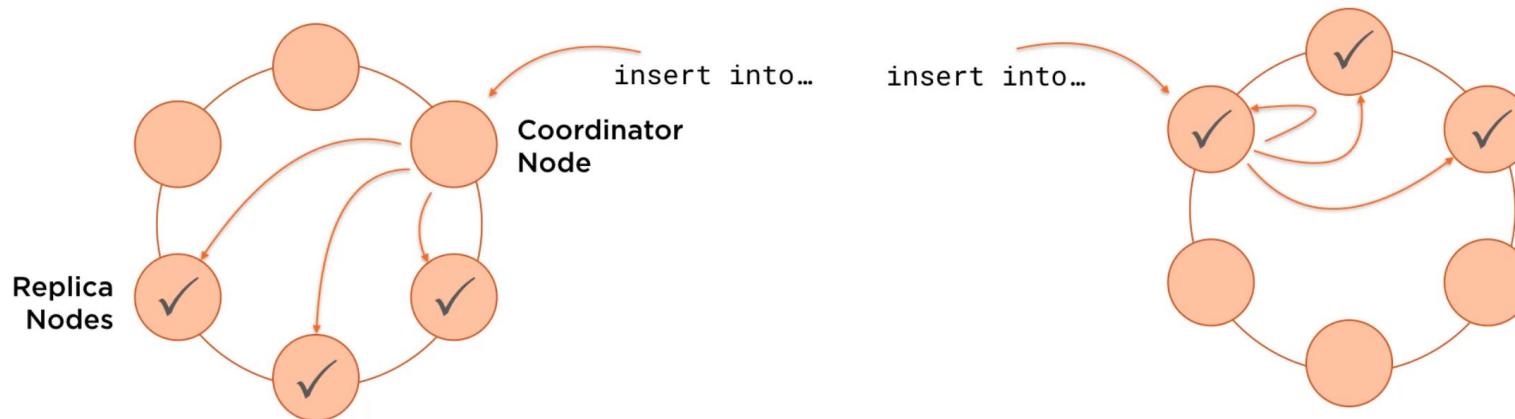


<https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/operations/opsMonitoring.html>



CLUSTER – ČTENÍ A ZÁPIS

- v rámci dotazů čtení a zápisu přejímají některé uzly speciální roli
 - uzel, ke kterému se uživatel připojí, se nazývá koordinační uzel
 - coordinator node
 - během připojení obstarává komunikaci mezi klientem a clusterem
 - předává dotazy a sbírá odpovědi
 - pro další připojení může být koordinačním uzlem jiný uzel



<https://app.pluralsight.com/library/courses/cassandra-developers>

OPRAVA UZLŮ

- node repair
- ./bin/nodetool repair
- řeší problém s neaktuálností dat uzlu
 - slouží k aktualizaci na současná data
 - synchronizuje data mezi uzly porovnáním rozsahu tokenů příslušných dat
 - streamuje nesynchronizované sekce (porovnání s merkle trees)
 - nutné v případě replikačního faktoru vyššího než 1
 - proč mohou být data neaktuální?
 - výpadek uzlu
 - zvýšení replikačního faktoru keyspace
 - změna rozsahu tokenů uzlu
 - hrozí, že neaktuální data přepíší aktuální data



OPRAVA UZLŮ^º

- dva typy oprav
 - plné
 - pracují se všemi daty v rozsahu tokenů, které jsou opravované
 - inkrementační
 - základní typ
 - opravují jen data, která byla zapsána od poslední inkrementační opravy
 - jakmile jsou data opravou označena jako opravená, už nebudou opraveny znovu
 - postačující pro chybějící zápisy
 - nechrání ale před vadným diskem, chybou obsluhy nebo bugy v Cassandře
 - plná oprava
- náročná operace
 - vhodné dobrě naplánovat
 - v době nízké zátěže databáze
 - nespouštět opravu všech uzlů najednou



OPRAVA UZLŮ

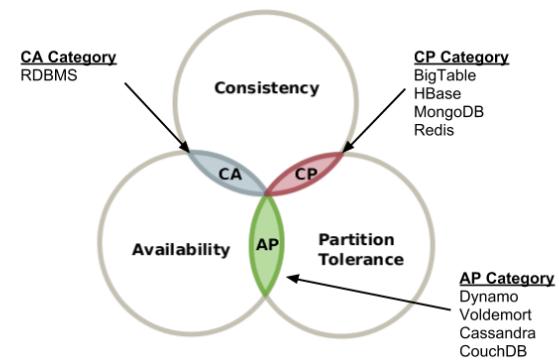
- oprava uzlu by měla být provedena na každém uzlu alespoň jednou za dobu definovanou v gc_grace_seconds
 - zabrání zpětné propagaci dat označených na smazání z neaktuálního uzlu
 - gc_grace_seconds je v základním nastavení 10 dní
- doporučení
 - inkrementační po 1-3 dnech
 - plná po 1-3 týdnech

```
nodetool repair  
  
nodetool repair -full  
  
nodetool repair [options] <keyspace_name>
```



LADITELNÁ KONZISTENCE

- CAP teorém
 - říká, že pro distribuovaný datový sklad nelze zajistit více jak dvě z následujících tří vlastnosti
 - konzistence – consistency (C)
 - všechny uzly mají stejná data
 - každé čtení vrací poslední výsledek, nebo chybu
 - dostupnost – availability (A)
 - na každý dotaz je vrácena (nechybová) odpověď
 - odolnost k přerušení – partition tolerance (P)
 - systém funguje dál i při zdržení či ztrátě zprávy
 - nezbytná pro big data



<https://changeaas.wordpress.com/tag/nosql/>





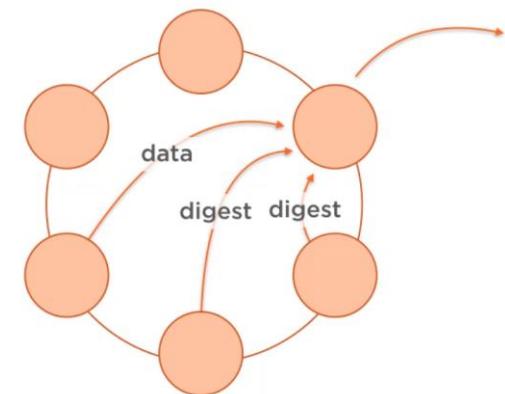
LADITELNÁ KONZISTENCE

- konzistence je nastaviteľná v Cassandre
 - Cassandra primárne volí AP z CAP
 - dostupnosť a odolnosť k prerušeniu
- ovplyvňuje presnosť dát v prípade viac než jedné repliky dát
- definovaná na úrovni čítania a zápisu
 - v základe nastavena na ONE
 - čítanie – len jeden z uzlov obsahujúcich repliky musí odpovedať
 - zápis – len jedna z replik musí potvrdiť prijatie dát
- porušuje C



LADITELNÁ KONZISTENCE

- další možnosti
 - TWO, THREE
 - stejné jako u ONE, ale pro 2, respektive 3, repliky
 - QUORUM, LOCAL_QUORUM
 - většina replik musí odpovědět
 - local jen pro jediné datacentrum
 - podpora i pro silnou konzistenci
 - ALL
 - všechny repliky musí odpovědět
 - EACH_QUORUM, LOCAL_ONE, ...
- aktuální nastavená konzistence
 - v cqlsh pomocí CONSISTENCY;
- změna pomocí CONSISTENCY <úroveň>;



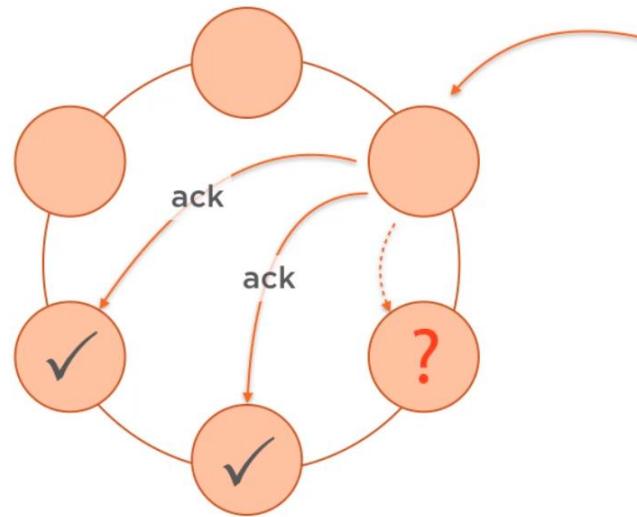
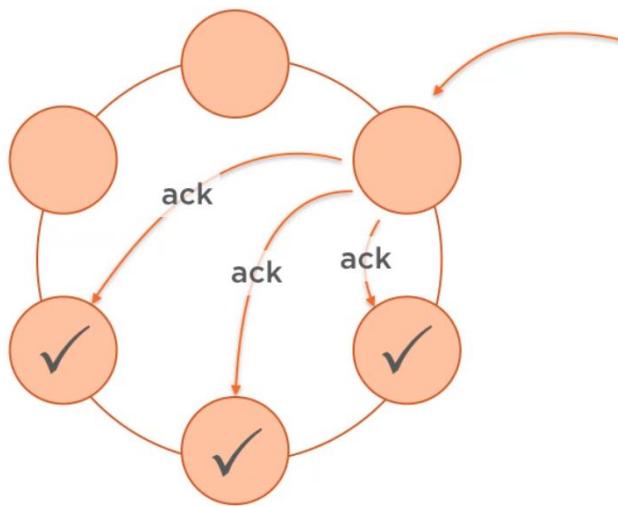
<https://app.pluralsight.com/library/courses/cassandra-developers>

```
cqlsh> CONSISTENCY;  
Current consistency level is ONE.
```

```
cqlsh> CONSISTENCY TWO;  
Consistency level set to TWO.
```



LADITELNÁ KONZISTENCE



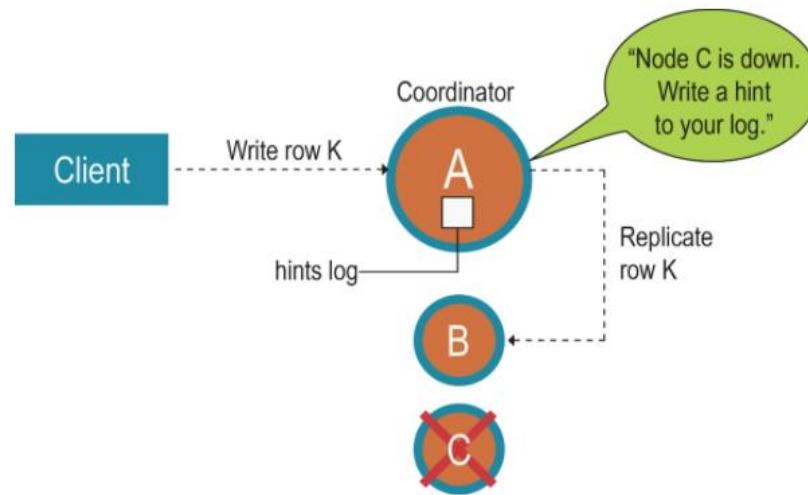
<https://app.pluralsight.com/library/courses/cassandra-developers>

HINTED HANDOFF

- technika umožňující zápisy dat i v případě výpadku jednoho z uzlů, na který mají data být zapsána
 - v základním nastavení zapnuta
 - koordinační uzel dočasně ukládá data místo nefunkčního uzlu
 - v základu 3 hodiny
 - po znovuspuštění uzlu koordinační uzel předá data
 - uzel zapíše
 - nepočítá se do možností konzistence
 - ONE, QUOROM, ALL, ...
 - výjimkou je možnost ANY, kterou splňuje
 - i v případě, že k zápisu nikdy nedojde
 - použití ANY jen v případech, kdy zápis není důležitý
 - nastavení v `cassandra.yaml`



HINTED HANDOFF



<https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/operations/opsRepairNodesHintedHandoff.html>

OPRAVA PŘI ČTENÍ

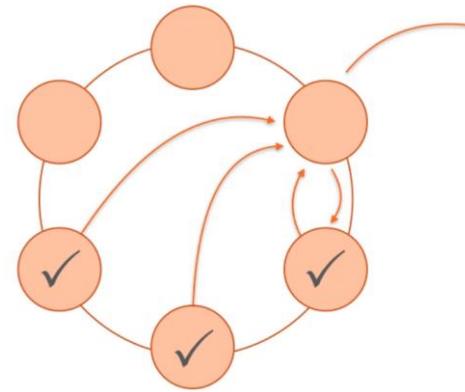
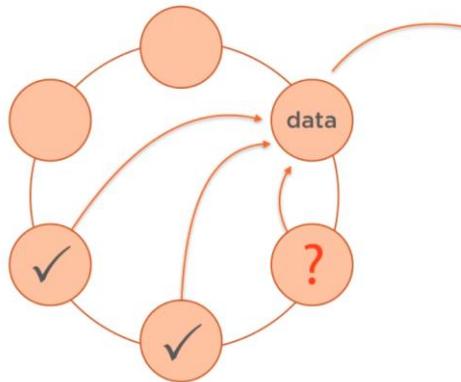
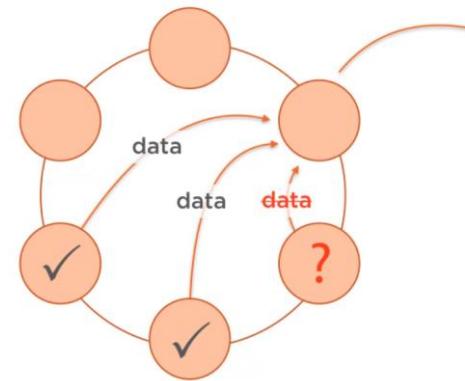
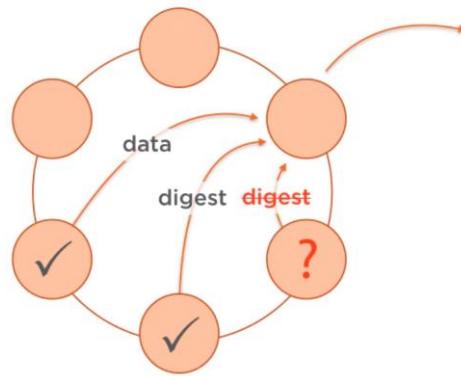
- **read repair**
 - možnost opravy jako součást dotazu čtení
 - porovnává požadovaná data z replik a následně updatuje zastaralé repliky
 - definováno na úrovni tabulek
 - v základu zapnuto s hodnotou 10 %
 - každé 10. čtení spustí opravu při čtení
 - prováděno v popředí
 - blokuje odpověď, dokud není oprava dokončena
 - klientovi jsou vrácena aktuální data
 - neslouží jako náhrada za opravy uzlů

```
cqlsh> DESCRIBE TABLE vehicle_tracker.location;  
  
CREATE TABLE location (  
    vehicle_id text,  
    date text,  
    time timestamp,  
    latitude double,  
    longitude double,  
    PRIMARY KEY ((vehicle_id, date), time)  
) WITH CLUSTERING ORDER BY (time DESC) AND  
bloom_filter_fp_chance=0.010000 AND  
caching='KEYS_ONLY' AND  
comment='' AND  
dclocal_read_repair_chance=0.000000 AND  
gc_grace_seconds=864000 AND  
index_interval=128 AND  
read_repair_chance=0.100000 AND
```

<https://www.udemy.com/course/apache-cassandra>



OPRAVA PŘI ČTENÍ



LADITELNÁ KONZISTENCE

- možnost tedy dosáhnout eventuální i silné konzistence
 - záleží na aplikaci
 - potřeba rozmyslet

(Write Consistency + Read Consistency) > Replication Factor

ONE	QUORUM	3	✗
ONE	ALL	3	✓
ONE	ONE	3	✗
QUORUM	QUORUM	3	✓

<https://app.pluralsight.com/library/courses/cassandra-developers>



ODPOJENÍ UZLU

- uzly mohou být odpojeny z různých důvodů
 - kapacita není potřeba, údržba HW, selhání HW, ...
- Cassandra si s odpojením dokáže poradit
 - pro plánované odstavení uzlu
nodetool decommission
 - pro neočekávaný výpadek uzlu
nodetool removenode
 - v obou případech Cassandra přiřadí rozsahy tokenů odpojeného uzlu zbylým uzlům a replikuje patřičná data



ODSTAVENÍ UZLU

- využíváno pro plánované odpojení uzlu z clusteru
- nodetool decommission
 - Cassandra přiřadí rozsahy tokenů odpojovaného uzlu ostatním uzelům
 - následně jsou data z odpojovaného uzlu streamována na ostatní uzly
 - neodstraňuje data z uzlu
- návrat uzlu
 - ideálním řešením je data vymazat a nechat uzel připojit jako nový
 - pro vymazání je potřeba smazat data, commit log a saved_caches
 - zamezí zdlouhavým opravám uzlu kvůli neaktuálnosti dat

ODSTRANĚNÍ UZLU

- využíváno pro odstranění mrtvého uzlu
- nodetool removenode
 - Cassandra přiřadí rozsahy tokenů mrtvého uzlu ostatním uzelům
 - následně jsou rozsahy zaplněny odpovídajícími daty z replik
 - v případě absence repliky dochází ke ztrátě dat

```
Every 2.0s: bin/nodetool ... Tue Jun 3 11:56:34 2014
RemovalStatus: Removing token (-9129580477490204828).
Waiting for replication confirmation from [/192.168.15
9.102,/192.168.159.101].
```

<https://www.udemy.com/course/apache-cassandra>



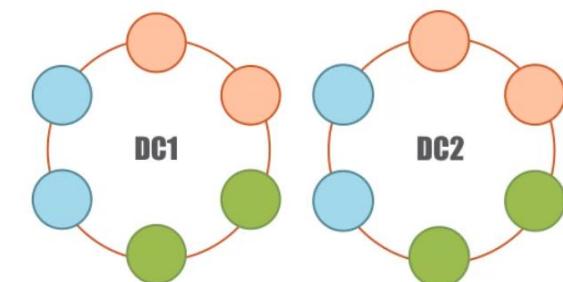
VÍCE DATACENTER

- pro umístění clusteru ve více datacentrech musí být splněno
 - snitch nemůže být SimpleSnitch
 - poskytuje uzlům informaci o topologii sítě
 - základní snitch, ale použitelná jen pro jedno datové centrum
 - snitch musí být stejná na všech uzlech
 - v praxi nejčastěji GossipingPropertyFileSnitch
 - umožňuje i více datacenter
 - informace o datacentru a racku pro daný uzel uloženy v `cassandra.rackdc.properties`
 - pomocí gossip informace předávána zbylým uzlům
- pro každý uzel musí být definované datacentrum a rack
 - většinou při definici snitch
 - pro GossipingPropertyFileSnitch je to právě v `cassandra.rackdc.properties`
 - specifikuje topologii daného uzlu

dc = DC2

rack = RAC1

<https://www.udemy.com/course/apache-cassandra>



<https://app.pluralsight.com/library/courses/cassandra-developers>



VÍCE DATACENTER

- pro umístění clusteru ve více datacentrech musí být splněno
 - replikační strategie pro všechny keyspace musí být změněna
 - NetworkTopologyStrategy
 - umožňuje specifikování počtu replik v každém datacentru pro keyspace

```
docker run --name cas3 -e CASSANDRA_SEEDS=172.17.0.2 -e CASSANDRA_CLUSTER_NAME=MyCluster -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch -e CASSANDRA_DC=datacenter2 -d cassandra
```

```
docker exec -ti cas1 cqlsh

CREATE KEYSPACE mykeyspace WITH replication = {
    'class' : 'NetworkTopologyStrategy',
    'datacenter1' : 1,
    'datacenter2' : 1
};
```





ČÁST III.: CASSANDRA V CLOUDU



CLOUD

- možnost využít libovolného poskytovatele cludu
 - Amazon Web Service, Google Cloud Platform, Microsoft Azure
 - DataStax Astra
 - customizovatelná DBaaS postavená na Cassandře
 - možnost vyzkoušet zdarma (v hodnotě \$25 měsíčně)

Enter the Basic Details

Database Name *	Keypspace Name *
MyDatabase	MyKeyspace
Give it a memorable name - this can't be changed later.	
Name your keyspace to reflect your data model.	

Select a Provider and Region

Google Cloud	AWS	Amazon Web Services	Microsoft Azure
Area		Region	
North America	0 of 2 regions selected	St. Ghislain, Belgium	europe-west1
Europe, Middle East, and Africa	1 of 1 region selected	>	
Asia Pacific	0 of 1 region selected		

Current Plan
Pay as you go

Read Requests	\$0.39/1M
Write Requests	\$1.15/1M
Storage	\$0.27/GB
Data Transfer	\$0.16/GB

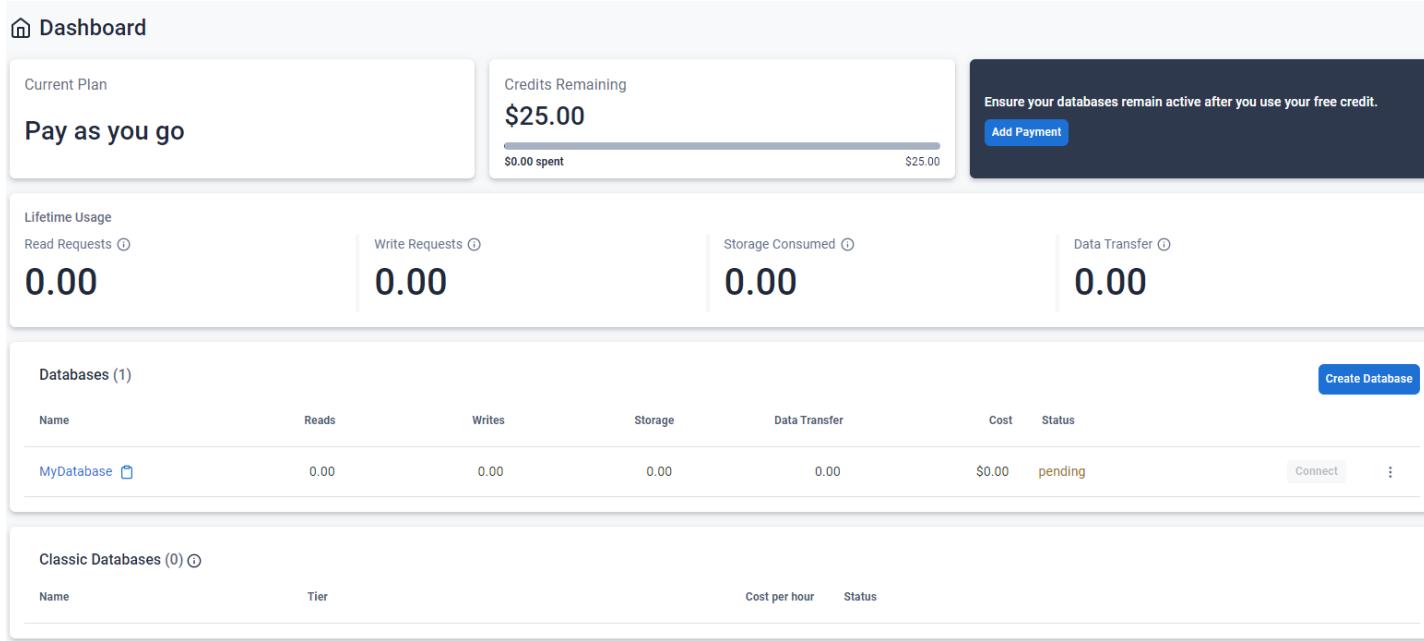
ⓘ Heads up: providers and regions can have different costs.

Create Database



CLOUD

- možnost využít libovolného poskytovatele cludu
 - [DataStax Astra](#)
 - obecný přehled



The screenshot shows the DataStax Astra Cloud Dashboard. At the top left, it says "Current Plan: Pay as you go". To the right, it displays "Credits Remaining: \$25.00" with a progress bar from \$0.00 spent to \$25.00. A dark blue callout box says "Ensure your databases remain active after you use your free credit." with a "Add Payment" button. Below this, there are four boxes for "Lifetime Usage": "Read Requests: 0.00", "Write Requests: 0.00", "Storage Consumed: 0.00", and "Data Transfer: 0.00". Under "Databases (1)", there is a table with columns: Name, Reads, Writes, Storage, Data Transfer, Cost, and Status. One row shows "MyDatabase" with values 0.00, 0.00, 0.00, 0.00, \$0.00, and pending status. There is a "Create Database" button. At the bottom, there is a section for "Classic Databases (0)" with a table having columns: Name, Tier, Cost per hour, and Status.



CLOUD

- možnost využít libovolného poskytovatele cludu
 - [DataStax Astra](#)
 - jednotlivé databáze, CQL konzole

Lifetime Usage for MyDatabase

Status Active

Read Requests ⓘ	Write Requests ⓘ	Storage Consumed ⓘ	Data Transfer ⓘ	Database Cost
0.00	0.00	0.00	0.00	\$0.00

Regions

Provider	Region	Capacity Units	Cluster ID
AWS	eu-central-1	1	8fd660b5-6809-492f-94a

Overview Health Connect **CQL Console** Settings

Connected as lukas.matejul@gmail.com.
Connected to cndb at cassandra.ingress:9042.
[cqlsh 6.8.0 | DSE DB 4.0.0.6811 | CQL spec 3.4.5 | Native protocol v4]
Use HELP for help.
token@cqlsh>

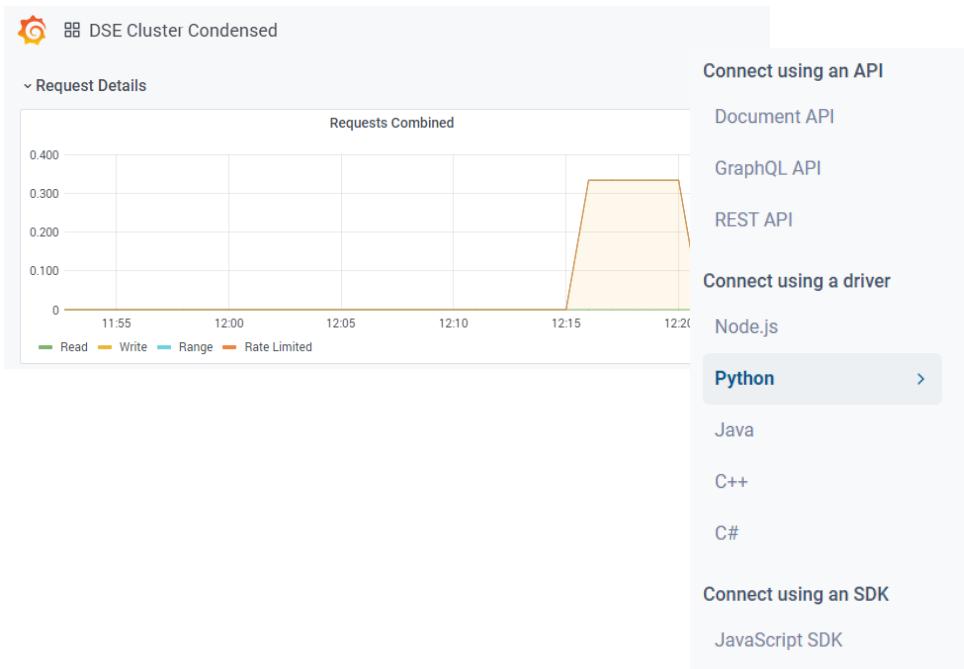
Keyspaces

Keyspace Name
MyKeyspace
1 total



CLOUD

- možnost využít libovolného poskytovatele cludu
 - [DataStax Astra](#)
 - statistiky, možnosti připojení



The screenshot shows the DataStax Astra interface. On the left, there's a chart titled 'Requests Combined' showing request rates over time (from 11:55 to 12:20). The legend indicates four types of requests: Read (green), Write (yellow), Range (light blue), and Rate Limited (orange). A significant spike in the orange 'Rate Limited' series occurs between 12:15 and 12:20. To the right of the chart, there are sections for connecting using APIs (Document API, GraphQL API, REST API) and drivers (Node.js, Python, Java, C++, C#). A callout box highlights the 'Python' option. Further down, there's a section for connecting using an SDK (JavaScript SDK).

Download your Secure Connect Bundle
If you have multiple regions, you will have the option to download a bundle for each region.

Using the Python driver to connect to your database

Use the Python driver to connect to your database and begin building your own application.

Prerequisites

1. Use the **Download Bundle** button at the top of this page to obtain connection credentials to your database.
2. Download and install a supported Python version. Python 2.7, 3.4, 3.5 and 3.6 are supported.
3. An Application Token (create a new one [here](#)) with the appropriate role set (RO User is needed for example below).

Steps

1. Install the DataStax Python driver:





ČÁST IV.: ZÁVĚREM...



ZÁPOČET

- náplň cvičení
 - samostatná práce
 - 10 povinných úloh
 - 10 bonusových úloh
 - za plné vypracování bonusové úlohy uděleno **0,5** bodu
- podmínky udelení zápočtu
 - vypracování a odevzdání všech 10 základních úloh
 - za každý týden opožděného odevzdání je **-0,5** bodu
 - splněná docházka (max 2 absence)
 - každá další absence znamená **-3** body



ZKOUŠKA

- zkouška
 - prezenční, písemná
 - 20 bodů, 10 otázek po 2 bodech
 - body ze cvičení jsou přenášeny ke zkoušce
 - zaměřená na základní koncepty probírané v rámci předmětu
- hodnocení
 - dvě varianty
 - jen za bonusové body
 - 5 bodů -> 1 4,5 bodů -> 2 4 body -> 3
 - v případě absolvování písemné zkoušky
 - maximum 25 bodů (20 + 5)
 - ≥ 21 bodů -> 1 ≥ 19 bodů -> 1- ≥ 17 bodů -> 2 ≥ 15 bodů -> 2- ≥ 13 bodů -> 3
 - < 13 bodů -> 4





Děkuji za pozornost.
Otázky?

