

HandsOnSession Dokumentation

CockroachDB

Ziele der HandsOnSession

Die HandsOnSession dient der praktischen Umsetzung der dazugehörigen Präsentation im Modul 3 PSE. Die Präsentation ist im Anschluss an diese Dokumentation nochmals hinterlegt. Im Wesentlichen sollen folgende Techniken gezeigt werden.

- Starten eines Clusters mit mindestens 3 Knoten.
- Veranschaulichung der CRUD-Operatoren in CockroachDB
- Nutzung der ADMIN UI und das Erweitern von Clustern
- Automatisiertes, dynamische Verhalten bei Einstellungsänderungen
- Automatisiertes, dynamische Verhalten bei Cluster-Fehlern

Starten eines Clusters

Schritt 1: Erklärung

- Zunächst bauen wir ein Cluster mit 3 Knoten auf. Dazu benötigen wir mindestens 4 Terminals (Bsp. PowerShell)
- In 3 von 4 Terminals geben wir die Befehle zum Konfigurieren der einzelnen Knoten inklusive Clusterinformationen ein
 - `cockroach start --insecure --listen-addr=ABC--http-addr=localhost:XYZ --join=localhost:ABC,localhost:DEF,localhost:GHI --store=<Name>`
- Für jeden Knoten aktivieren wir ein eigenes Terminal

Knoten	ABC	DEF	GHI	XYZ	<Name>
1	26257	26258	26259	8080	Node1
2	26258	26257	26259	8081	Node2
3	26259	26257	26258	8082	Node3

- Im 4. Terminal initialisieren wir das Cluster. Die Knoten starten, bauen ein Cluster auf und erstellen einen lokalen Speicher. Befehl: „`cockroach init --insecure`“

Starten eines Clusters

Schritt 1: Bilder

- Hier gezeigt ist die Konfiguration des Knoten 1 und das Starten des Knotens. Dies wird quittiert mit „Node will now attempt to join a running cluster, or wait for `cockroach init`.“

```
PS C:\Users\Marvin> cd d:\master_pse\m3_datenbanken\cr_uni
PS D:\master_pse\m3_datenbanken\cr_uni> cockroach start --insecure --listen-addr=localhost:26257
--http-addr=localhost:8080 --join=localhost:26257,localhost:26258,localhost:26259 --store=node1

*
* WARNING: RUNNING IN INSECURE MODE!
*
* - Your cluster is open for any client that can access localhost.
* - Any user, even root, can log in without providing a password.
* - Any user, connecting as root, can read or write any data in your cluster.
* - There is no network encryption nor authentication, and thus no confidentiality.
*
* Check out how to secure your cluster: https://www.cockroachlabs.com/docs/v20.1/secure-a-cluster.html
*
*
* INFO: initial startup completed.
* Node will now attempt to join a running cluster, or wait for `cockroach init`.
* Client connections will be accepted after this completes successfully.
* Check the log file(s) for progress.
*
```

Starten eines Clusters

Schritt 1: Bilder

- Hier gezeigt ist die Initialisierung des Clusters über das freie Terminal, das Aufstarten der Knoten und die automatisch angelegten Speicher der einzelnen Knoten.

```
PS D:\master_pse\m3_datenbanken\cr_uni> cockroach init --insecure
Cluster successfully initialized
PS D:\master_pse\m3_datenbanken\cr_uni>

CockroachDB node starting at 2020-11-15 14:59:42.3224244 +0000 UTC (took 294.8s)
build:          CCL v20.1.8 @ 2020/10/21 15:56:58 (go1.13.9)
webui:          http://localhost:8080
sql:            postgresql://root@localhost:26257?sslmode=disable
RPC client flags: D:\Master_PSE\M3_Datenbanken\cockroach-v20.1.8.windows-6.2-amd64\cockroach.exe
<client cmd> --host=localhost:26257 --insecure
logs:           D:\master_pse\m3_datenbanken\cr_uni\node1\logs
temp dir:        D:\master_pse\m3_datenbanken\cr_uni\node1\cockroach-temp030055331
external I/O path: D:\master_pse\m3_datenbanken\cr_uni\node1\extern
store[0]:         path=D:\master_pse\m3_datenbanken\cr_uni\node1
storage engine:   rocksdb
status:           initialized new cluster
clusterID:        cd60f870-75a8-45f3-a034-1c2fdada00a9
nodeID:          1
```

Dieser PC > Volume (D:) > Master_PSE > M3_Datenbanken > CR_uni

Name	Änderungsdatum	Typ	Größe
node1	15.11.2020 15:54	Dateiordner	
node2	15.11.2020 16:00	Dateiordner	
node3	15.11.2020 16:00	Dateiordner	

CRUD Operationen

- Die gängigen SQL-Befehle werden vorausgesetzt. Daher werden hier nur die einfachsten Befehle aus Gründen der Vollständigkeit in Form der Terminalein und –ausgaben gezeigt.
- CockroachDB bietet dabei die Möglichkeit, integrierte Datensätze zu nutzen. Diese werden über „workloads“ geladen. Diese dienen im weiteren der Veranschaulichung.
- Beispiele:
 - Movr: „cockroach workload init movr;
 - Startrek: cockroach workload init startrek;

Laden der integrierten Datensätze

movr

```
PS D:\master_pse\m3_datenbanken\cr_uni> cockroach workload init movr;
I201115 15:12:28.712592 1 workload/workloads/sql/dataload.go:140 imported users (0s, 50 rows)
I201115 15:12:28.747164 1 workload/workloads/sql/dataload.go:140 imported vehicles (0s, 15 rows)
I201115 15:12:29.105865 1 workload/workloads/sql/dataload.go:140 imported rides (0s, 500 rows)
I201115 15:12:29.467065 1 workload/workloads/sql/dataload.go:140 imported vehicle_location_histories (0s, 1000 rows)
I201115 15:12:29.833303 1 workload/workloads/sql/dataload.go:140 imported promo_codes (0s, 1000 rows)
I201115 15:12:29.840837 1 workload/workloads/sql/workloads/sql.go:113 starting 8 splits
I201115 15:12:30.511322 1 workload/workloads/sql/workloads/sql.go:113 starting 8 splits
I201115 15:12:31.125838 1 workload/workloads/sql/workloads/sql.go:113 starting 8 splits
PS D:\master_pse\m3_datenbanken\cr_uni> cockroach workload init startrek;
I201115 15:12:45.061253 1 workload/workloads/sql/dataload.go:140 imported episodes (0s, 79 rows)
I201115 15:12:45.405865 1 workload/workloads/sql/dataload.go:140 imported quotes (0s, 200 rows)
```

startrek

- „Movr“ ist ein fiktives Mobilitätsunternehmen.
- Relationen: users, vehicles, rides, vehicle_location_histories, promo_codes
- „Startrek“ ist eine Sammlung von Episoden der Serie „Startrek“
- Relationen: episodes, quotes

CRUD Operationen

- Nun greifen wir auf die erstellten Datenbanken und deren Datensätze zu.
- Schritt 1: Anzeigen aller Datenbanken
- Schritt 2: Auswählen einer Datenbank und alle Relationen anzeigen lassen
- Schritt 3: Anzeigen von Datensätzen innerhalb von gewählter Relation
- Schritt 4: Anzeigen von Datensätzen anderer Datenbanken


```

root@:26257/defaultdb> show databases;
database_name
-----
defaultdb
movr
postgres
startrek
system
(5 rows)

```

1

```

root@:26257/defaultdb> use movr;
SET

Time: 1.5359ms

root@:26257/movr> show tables;
table_name
-----
promo_codes
rides
user_promo_codes
users
vehicle_location_histories
vehicles
(6 rows)

```

2

```

root@:26257/movr> select * from users limit(10);

```

id	city	name	address	credit_card
ae147ae1-47ae-4800-8000-000000000022	amsterdam	Tyler Dalton	88194 Angela Gardens Suite 94	4443538758
b3333333-3333-4000-8000-000000000023	amsterdam	Dillon Martin	29590 Butler Plain Apt. 25	3750897994
b851eb85-1eb8-4000-8000-000000000024	amsterdam	Deborah Carson	32768 Eric Divide Suite 88	8107478823
bd70a3d7-0a3d-4000-8000-000000000025	amsterdam	David Stanton	80015 Mark Views Suite 96	3471210499
c28f5c28-f5c2-4000-8000-000000000026	amsterdam	Maria Weber	14729 Karen Radial	5844236997
1eb851eb-851e-4800-8000-000000000006	boston	Brian Campbell	92025 Yang Village	9016427332
23d70a3d-70a3-4800-8000-000000000007	boston	Carl Mcguire	60124 Palmer Mews Apt. 49	4566257702
28f5c28f-5c28-4600-8000-000000000008	boston	Jennifer Sanders	19121 Padilla Brooks Apt. 12	1350968125
2e147ae1-47ae-4400-8000-000000000009	boston	Cindy Medina	31118 Allen Gateway Apt. 60	6464362441
33333333-3333-4400-8000-00000000000a	boston	Daniel Hernandez MD	51438 Janet Valleys	0904722368

```

(10 rows)

```

3

```

root@:26257/movr> select * from startrek.episodes limit(10);

```

id	season	num	title	stardate
1	1	1	The Man Trap	1531.1
2	1	2	Charlie X	1533.6
3	1	3	Where No Man Has Gone Before	1312.4
4	1	4	The Naked Time	1704.2
5	1	5	The Enemy Within	1672.1
6	1	6	Mudd's Women	1329.8
7	1	7	What Are Little Girls Made Of?	2712.4
8	1	8	Miri	2713.5
9	1	9	Dagger of the Mind	2715.1
10	1	10	The Corbomite Maneuver	1512.2

```

(10 rows)

```

4

CRUD Operationen

Nun manipulieren wir die Daten

- Schritt 1: Neue Relation erstellen
- Schritt 2: Daten einfügen in neue Relation
- Schritt 3: Daten verändern in neue Relation
- Schritt 4: Datensätze in der neuen Relation löschen
- Schritt 5: Neue Relation löschen

```

root@:26257/testdb> CREATE TABLE Persons (
    ->     ID int NOT NULL PRIMARY KEY,
    ->     LastName varchar(255) NOT NULL,
    ->     FirstName varchar(255),
    ->     Age int
    -> );

```

CREATE TABLE

Time: 41.2802ms

```

root@:26257/testdb> show tables;

```

```

  table_name
-----
  persons
(1 row)

```

1

```

root@:26257/testdb> insert into persons values (1,'bermel','marvin', 24)
,(2, 'mustermann', 'max', 99), (20, 'Mueller', 'Josef', 30);
INSERT 3

```

Time: 10.4135ms

```

root@:26257/testdb> select * from persons;

```

```

  id |  lastname |  firstname |  age
-----+-----+-----+-----
   1 | bermel    | marvin     |   24
   2 | mustermann | max        |   99
  20 | Mueller   | Josef      |   30
(3 rows)

```

Time: 3.1517ms

2

```

root@:26257/testdb> update persons set firstname = 'fabian' where age < 31;

```

UPDATE 2

Time: 23.2539ms

```

root@:26257/testdb> select * from persons;

```

```

  id |  lastname |  firstname |  age
-----+-----+-----+-----
   1 | bermel    | fabian     |   24
   2 | mustermann | max        |   99
  20 | Mueller   | fabian     |   30
(3 rows)

```

3

```

root@:26257/testdb> delete from persons where firstname = 'fabian';

```

DELETE 2

Time: 22.3854ms

```

root@:26257/testdb> select * from persons;

```

```

  id |  lastname |  firstname |  age
-----+-----+-----+-----
   2 | mustermann | max        |   99
(1 row)

```

Time: 1.8684ms

4

```

root@:26257/testdb> drop table persons;

```

DROP TABLE

Time: 264.1989ms

5

```

root@:26257/testdb> drop table testdb;

```

ERROR: relation "testdb" does not exist
SQLSTATE: 42P01

```

root@:26257/testdb> drop database testdb;

```

ERROR: rejected: DROP DATABASE on current database (sql_safe_updates = true)
SQLSTATE: 01000

```

root@:26257/testdb> use defaultdb;

```

SET

Time: 1.4837ms

```

root@:26257/defaultdb> drop database testdb;

```

DROP DATABASE

Time: 143.6045ms

```

root@:26257/defaultdb> show databases;

```

```

  database_name
-----
  defaultdb
  movr
  postgres
  startrek
  system
(5 rows)

```

Time: 3.7742ms

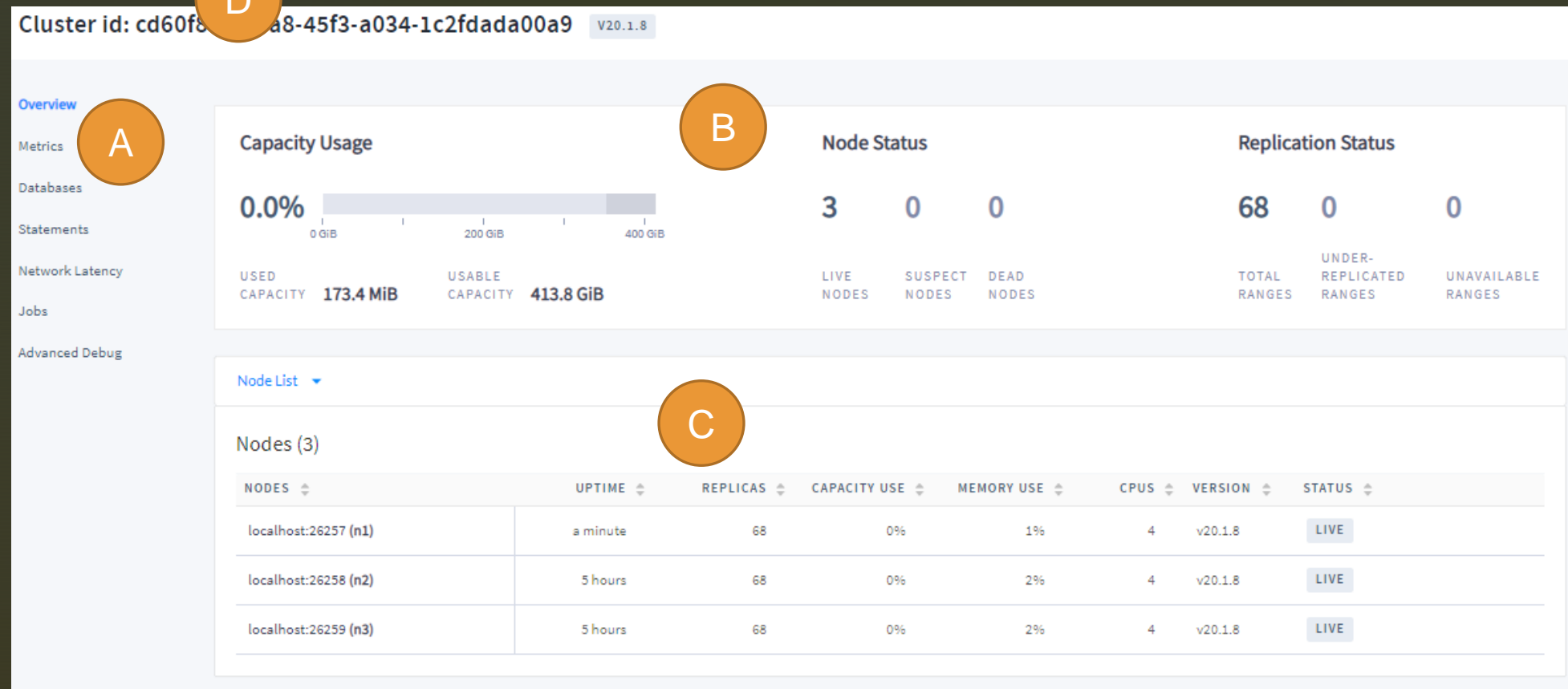
6

Dynamisches Verhalten Admin UI und Cluster-Erweiterung

Nun schauen wir uns die Admin UI, erläutern die Grundlegenden Funktionen und schauen uns das dynamische Verhalten des Clusters bei einer Erweiterung an.

- Schritt 1: Wir starten ein Cluster und laden Daten
- Schritt 2: Wir erweitern das Cluster
- Schritt 3: Wir beobachten das Cluster-Verhalten

Overview

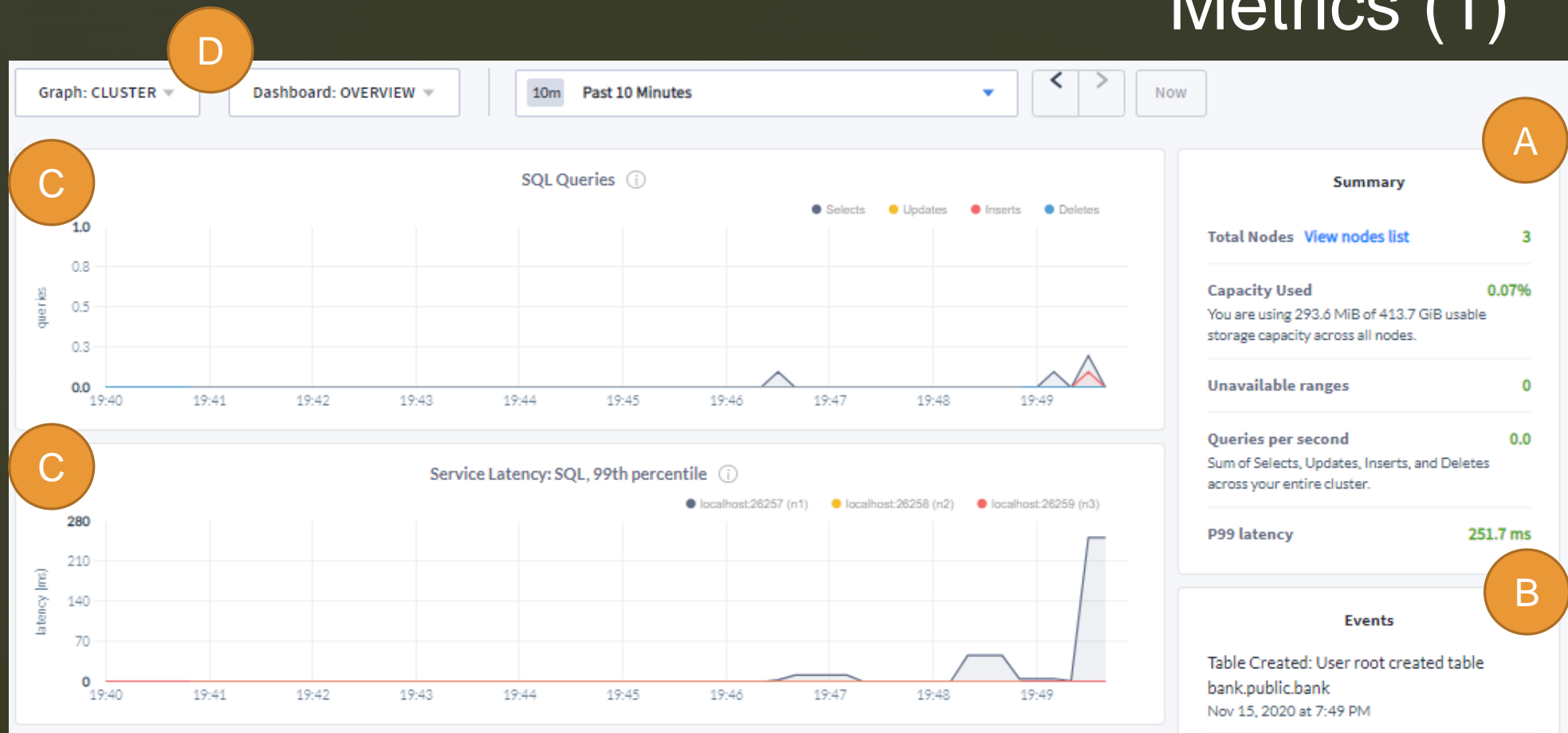


A: Auswahl der Oberflächen

B: Zusammenfassung der Datenqualität

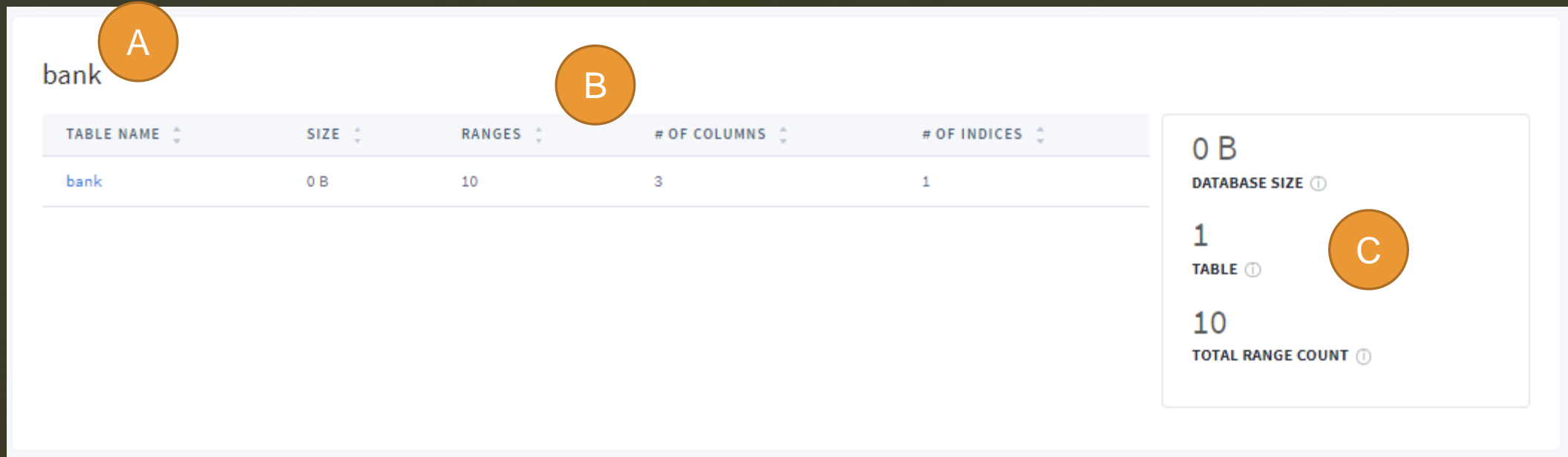
C: Informationen zum laufenden Cluster hinsichtlich Inbetriebnahme

Metrics (1)



- A: Zusammenfassende Informationen über Cluster-Zustand
- B: Chronologische Anzeige der letzten Operationen/Events
- C: Graphische Anzeige von Anfragen und Latenzen
- D: Einstellungen der graphischen Anzeigen

Databases (1)



bank

TABLE NAME	SIZE	RANGES	# OF COLUMNS	# OF INDICES
bank	0 B	10	3	1

0 B
DATABASE SIZE

1
TABLE

10
TOTAL RANGE COUNT

The screenshot shows a database management interface. At the top left, the word 'bank' is displayed. Below it is a table with five columns: 'TABLE NAME', 'SIZE', 'RANGES', '# OF COLUMNS', and '# OF INDICES'. The table contains one row with the values 'bank', '0 B', '10', '3', and '1'. To the right of the table is a sidebar containing three summary statistics: '0 B DATABASE SIZE', '1 TABLE', and '10 TOTAL RANGE COUNT'. Three orange circles are overlaid on the image: circle 'A' is positioned over the word 'bank' at the top left; circle 'B' is positioned over the 'RANGES' column header; and circle 'C' is positioned over the '1 TABLE' summary statistic in the sidebar.

A: Name der Datenbank

B: Inhalte der Datenbank

C: Informationen zur Speichernutzung

Databases (2)

bank.bank

A

OverviewGrants

```
CREATE TABLE bank (  
  id INT8 NOT NULL,  
  balance INT8 NULL,  
  payload STRING NULL,  
  CONSTRAINT "primary" PRIMARY KEY (id ASC),  
  FAMILY fam_0_id_balance_payload (id, balance, payload)  
)  
  
-----  
  
CONFIGURE ZONE USING  
  range_min_bytes = 134217728,  
  range_max_bytes = 536870912,  
  gc.ttlseconds = 90000,  
  num_replicas = 3,  
  constraints = [''],  
  lease_preferences = [['']
```

0 B
Size

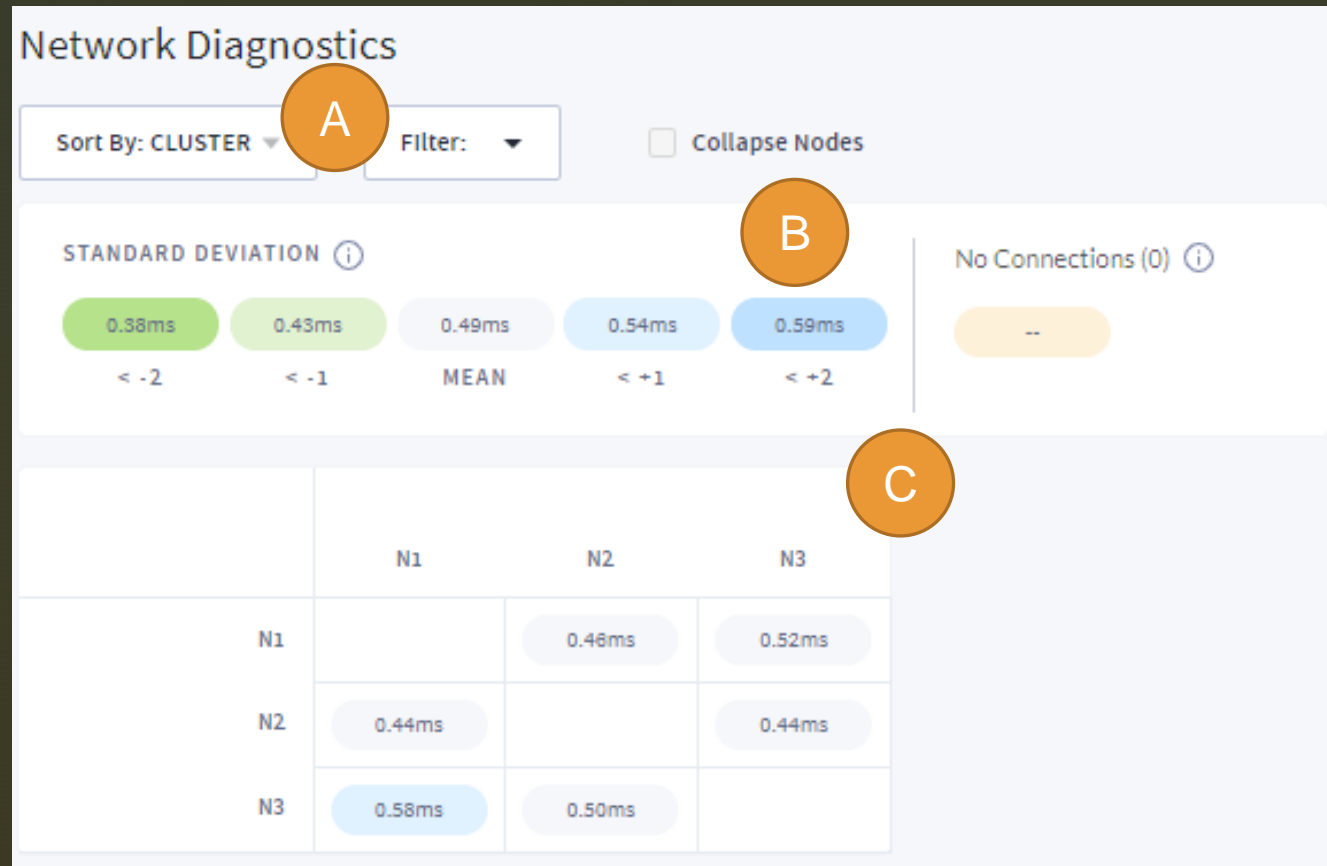
3
Replicas

10
Ranges

C

- A: Name der Tabelle
- B: Aufbau der Tabelle
- C: Informationen zur Speichernutzung

Databases (2)



A: Einstellungen

B: Mittlere Latenz der Knoten zum Gesamtcluster

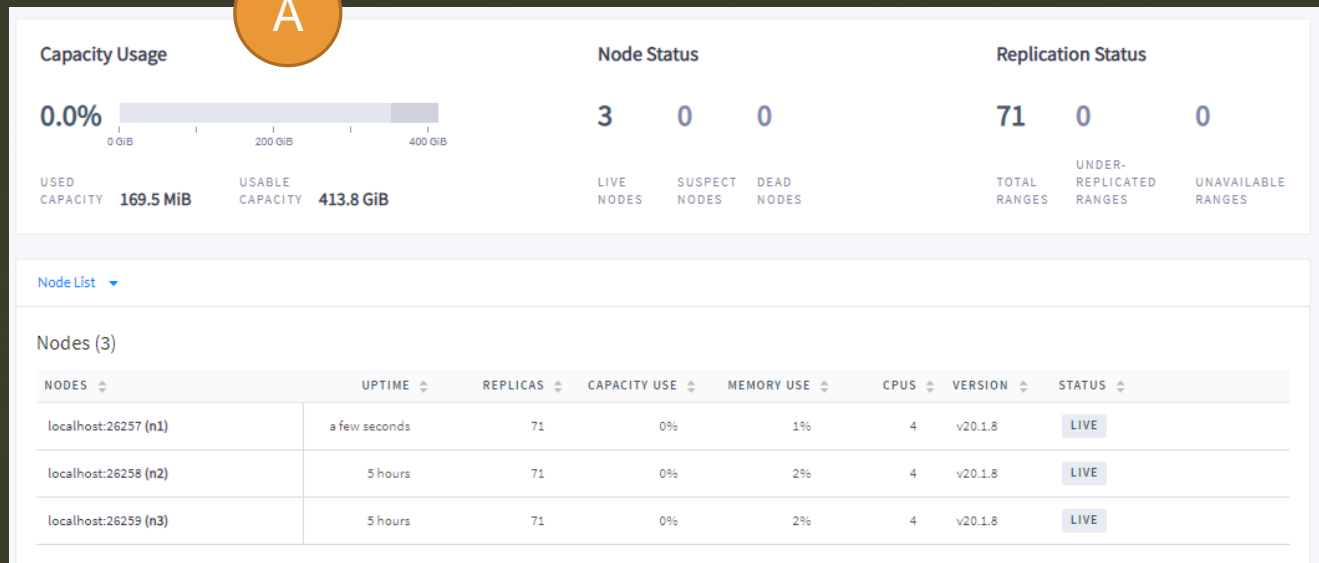
C: Mittlere Latenz der Knoten zu anderen Knoten

Dynamisches Verhalten

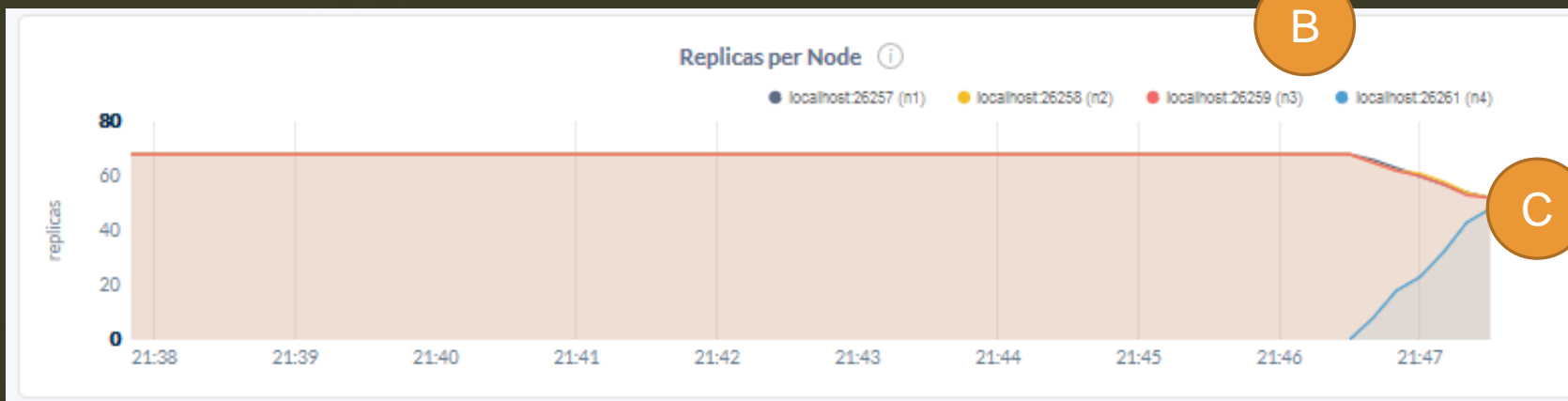
Vorgehen Clustererweiterung/-veränderung

- Zur Demonstration des dynamischen Verhaltens wurde ein Cluster mit beliebigen Datenbanken und Datensätzen angelegt
- Es sind die Standardparameter für Replikationen (Faktor 3) und Ranges (CRDB V20.1.8 →64MB) eingestellt
- Die Knoten-Totzeit wurde auf das Minimum, 75s, gestellt.
(Befehl SQL: SET CLUSTER SETTING server.time_until_store_dead = '1m15s';
- Zunächst erweitern wir das Cluster von 3 auf 4 Knoten.
- Erweiterung auf 5 Knoten
- Im weiteren Verlauf wird der Replikationsfaktor auf 5 erhöht
(Befehl SQL: ALTER DATABASE xyz CONFIGURE ZONE USING num_replicas = 5;
- Das Verhalten der Datenbank wird beobachtet

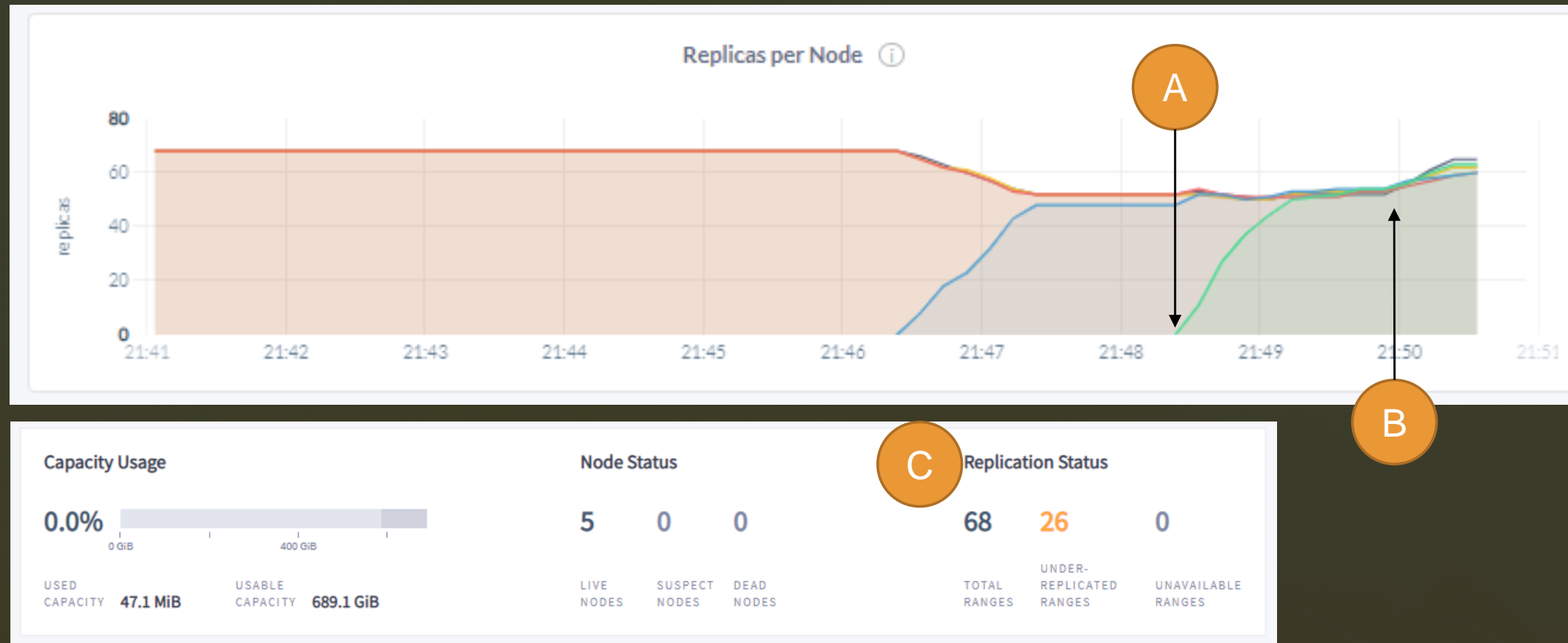
Dynamisches Verhalten (1)



A: Altes Cluster (3 Knoten)
B: Neues Cluster (4 Knoten)
C: Ausgleich Replikationen
von ca. 71 auf 68 pro Knoten



Dynamisches Verhalten (2)



A: Erweiterung um 5.Knoten

B: Änderung des Replikationsfaktors von 3 → 5

C: Momentaufnahme nach Änderung des Replikationsfaktor. 30% der „Ranges“ sind unzureichend repliziert

Dynamisches Verhalten (3)

Latenzbeeinflussung

A: Durch die Umstellung des Replikationsfaktor entsteht erhebliche interne Kommunikation der Datenbank
→ erhöhte Latenz



Dynamisches Verhalten (4)

Ausgabe über SQLShell

A

```
D:\master_pse\m3_datenbanken\cr_uni> cockroach node status --all --insecure
id | address | sql_address | build | started_at | updated_at | locality | is_available | is_live | replicas_leaders | replicas_lease
holders | ranges | ranges_unavailable | ranges_underreplicated | live_bytes | key_bytes | value_bytes | intent_bytes | system_bytes | gossip_replicas | is_decommissioning | is_draining
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
1 | localhost:26257 | localhost:26257 | v20.1.8 | 2020-11-16 16:55:15.725551+00:00 | 2020-11-16 17:00:48.835306+00:00 | 42482 | true | true | false | 16 | false
16 | 51 | 0 | 0 | 6196138 | 463647 | 5825786 | 0 | 0 | 51 | false | false
2 | localhost:26258 | localhost:26258 | v20.1.8 | 2020-11-16 16:55:17.790955+00:00 | 2020-11-16 17:00:46.334029+00:00 | 41491 | true | true | false | 11 | false
11 | 51 | 0 | 0 | 1045074 | 327514 | 810995 | 0 | 0 | 51 | false | false
3 | localhost:26259 | localhost:26259 | v20.1.8 | 2020-11-16 16:55:18.264569+00:00 | 2020-11-16 17:00:46.811957+00:00 | 44481 | true | true | false | 12 | false
12 | 49 | 0 | 0 | 5949547 | 350680 | 5692302 | 0 | 0 | 49 | false | false
4 | localhost:26260 | localhost:26260 | v20.1.8 | 2020-11-16 16:55:27.801137+00:00 | 2020-11-16 17:00:47.37871+00:00 | 39994 | true | true | false | 12 | false
12 | 49 | 0 | 0 | 898691 | 225606 | 766520 | 0 | 0 | 49 | false | false
5 | localhost:26261 | localhost:26261 | v20.1.8 | 2020-11-16 16:56:49.285175+00:00 | 2020-11-16 17:00:47.840681+00:00 | 44510 | true | true | false | 11 | false
11 | 48 | 0 | 0 | 5836211 | 248689 | 5680992 | 0 | 0 | 48 | false | false
(5 rows)
```

B

```
D:\master_pse\m3_datenbanken\cr_uni> cockroach node status --all --insecure
id | address | sql_address | build | started_at | updated_at | locality | is_available | is_live | replicas_leaders | replicas_lease
holders | ranges | ranges_unavailable | ranges_underreplicated | live_bytes | key_bytes | value_bytes | intent_bytes | system_bytes | gossip_replicas | is_decommissioning | is_draining
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
1 | localhost:26257 | localhost:26257 | v20.1.8 | 2020-11-16 16:55:15.725551+00:00 | 2020-11-16 17:03:48.836334+00:00 | 55857 | true | true | false | 14 | false
14 | 56 | 0 | 0 | 9196754 | 471008 | 8833360 | 0 | 0 | 56 | false | false
2 | localhost:26258 | localhost:26258 | v20.1.8 | 2020-11-16 16:55:17.790955+00:00 | 2020-11-16 17:03:46.334082+00:00 | 56605 | true | true | false | 11 | false
11 | 56 | 0 | 0 | 1079872 | 338393 | 849233 | 0 | 0 | 56 | false | false
3 | localhost:26259 | localhost:26259 | v20.1.8 | 2020-11-16 16:55:18.264569+00:00 | 2020-11-16 17:03:46.813432+00:00 | 55836 | true | true | false | 13 | false
13 | 52 | 0 | 0 | 8967484 | 363015 | 8712223 | 0 | 0 | 52 | false | false
4 | localhost:26260 | localhost:26260 | v20.1.8 | 2020-11-16 16:55:27.801137+00:00 | 2020-11-16 17:03:47.36248+00:00 | 50992 | true | true | false | 13 | false
13 | 52 | 0 | 0 | 941147 | 238498 | 810403 | 0 | 0 | 52 | false | false
5 | localhost:26261 | localhost:26261 | v20.1.8 | 2020-11-16 16:56:49.285175+00:00 | 2020-11-16 17:03:47.82833+00:00 | 53004 | true | true | false | 11 | false
11 | 50 | 0 | 0 | 8716281 | 269104 | 8554581 | 0 | 0 | 50 | false | false
(5 rows)
```

Über die SQLShell kann man sich ebenfalls das dynamische Verhalten des Clusters anzeigen lassen.

„A“ zeigt die Informationen der Knoten mit dem Replikationsfaktor 3

„B“ zeigt die Informationen der Knoten mit dem Replikationsfaktor 5

Befehl: <cockroach node status --all --insecure>

Dynamisches Verhalten Vorgehen Cluster-Fehler

Nun schauen wir uns über die Admin UI das Verhalten des Clusters bei einem Ausfall eines Knotens an

- Schritt 1: Wir starten ein Cluster mit 4 Knoten und laden Daten
- Schritt 2: Wir beenden einen Knoten

Befehl: STRG+C im Terminal des gewählten Knotens

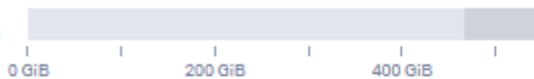
- Schritt 3: Wir beobachten das Cluster-Verhalten

Dynamisches Verhalten Cluster-Fehler (1)

Schritt 1: Aufgebautes Cluster

Capacity Usage

0.0%



USED
CAPACITY **125.4 MiB**

USABLE
CAPACITY **548.1 GiB**

Node Status

4

0

0

LIVE
NODES

SUSPECT
NODES

DEAD
NODES

Replication Status

72

0

0

TOTAL
RANGES

UNDER-
REPLICATED
RANGES

UNAVAILABLE
RANGES

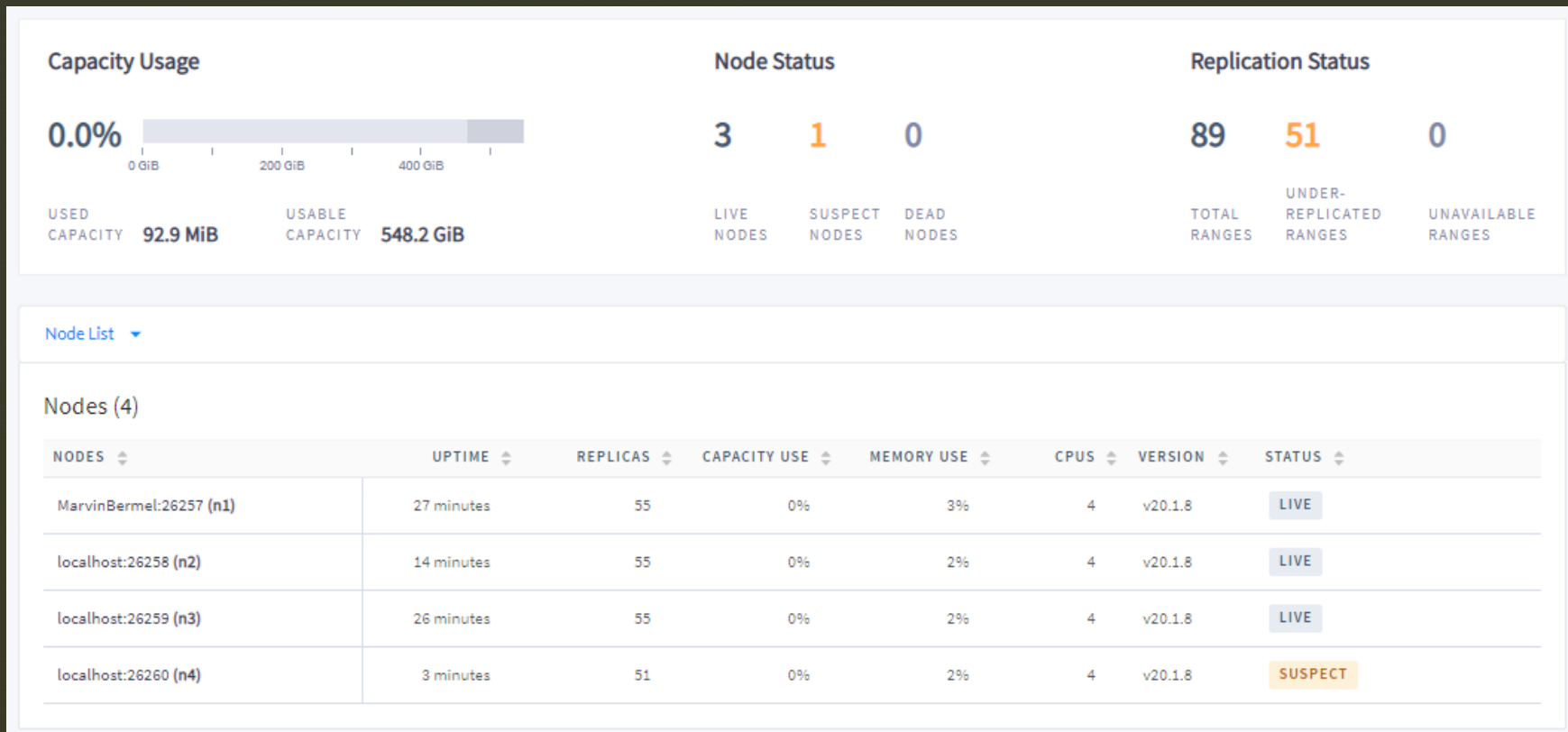
Node List ▾

Nodes (4)

NODES ▾	UPTIME ▾	REPLICAS ▾	CAPACITY USE ▾	MEMORY USE ▾	CPUS ▾	VERSION ▾	STATUS ▾
MarvinBermel:26257 (n1)	14 minutes	55	0%	3%	4	v20.1.8	LIVE
localhost:26258 (n2)	2 minutes	55	0%	2%	4	v20.1.8	LIVE
localhost:26259 (n3)	14 minutes	55	0%	2%	4	v20.1.8	LIVE
localhost:26260 (n4)	a minute	51	0%	1%	4	v20.1.8	LIVE

Dynamisches Verhalten Cluster-Fehler (2)

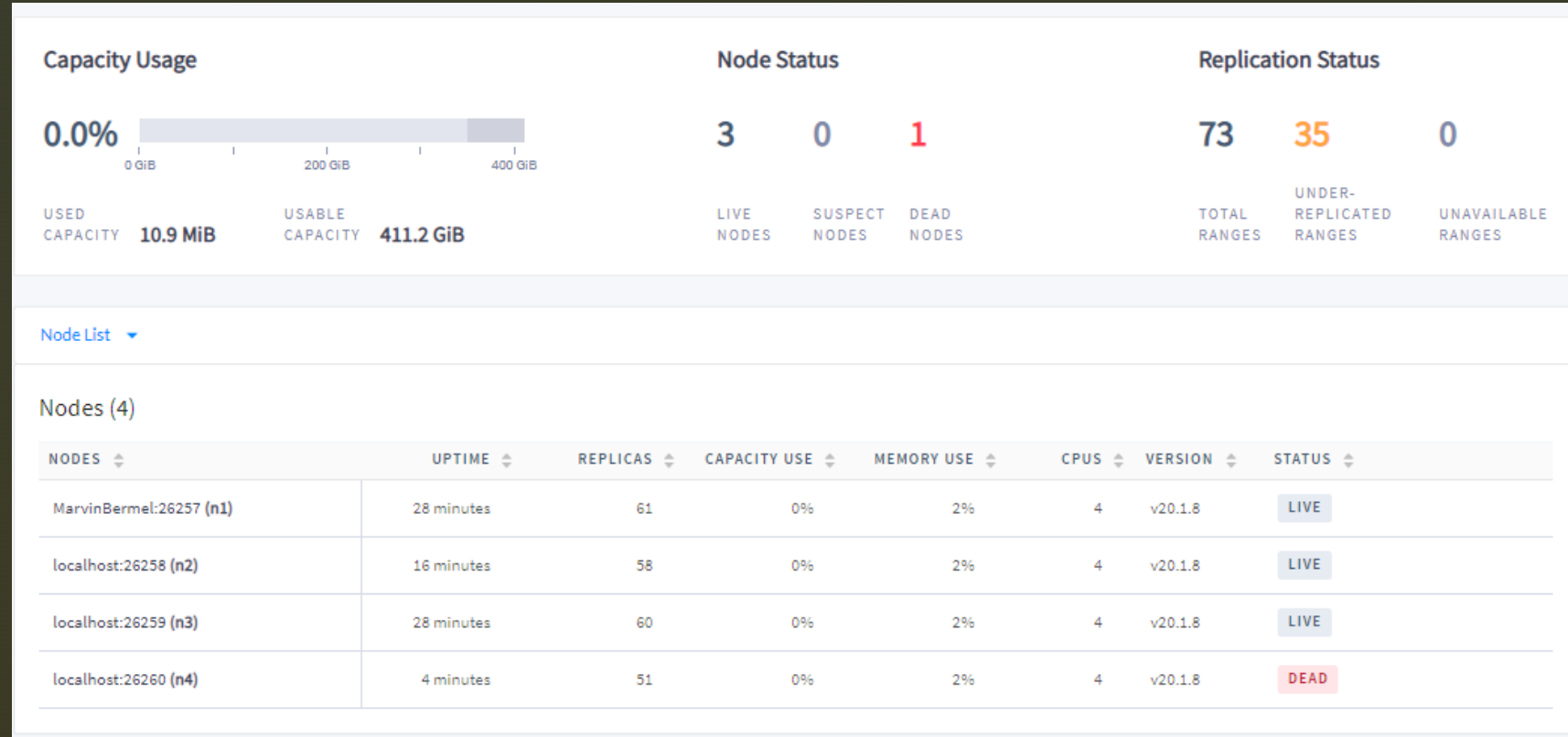
Schritt 2: Fehlerhafter Knoten



Der fehlerhafte Knoten wurde erkannt („1 SUSPECTE NODES“), die unzureichenden Replikationen werden nicht nachgebildet

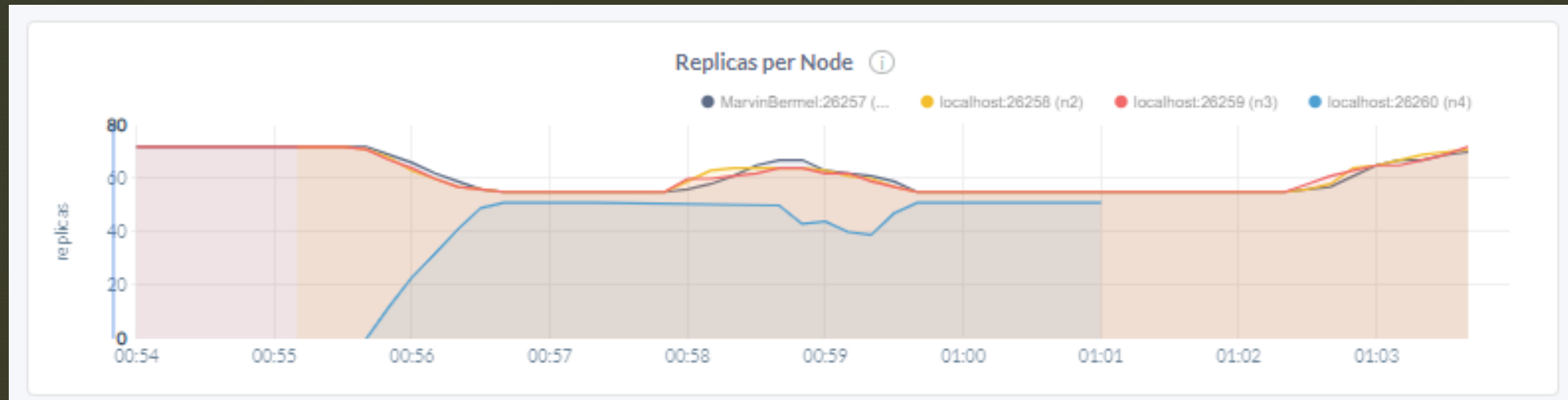
Dynamisches Verhalten Cluster-Fehler (3)

Schritt 3: Toter Knoten



Der fehlerhafte Knoten wurde als tot erkannt („1 DEAD NODES“), die unzureichenden Replikationen werden nachgebildet

Dynamisches Verhalten Cluster-Fehler (3) Schritt 3: Toter Knoten



Graphische Darstellung des Cluster-Verhaltens. Nach Ablauf der Totzeit beginnt das Cluster automatisch die unzureichend replizierten „Ranges“ selbstständig zu replizieren