Developing Scalable Services and Managing their Transactions with CockroachDB

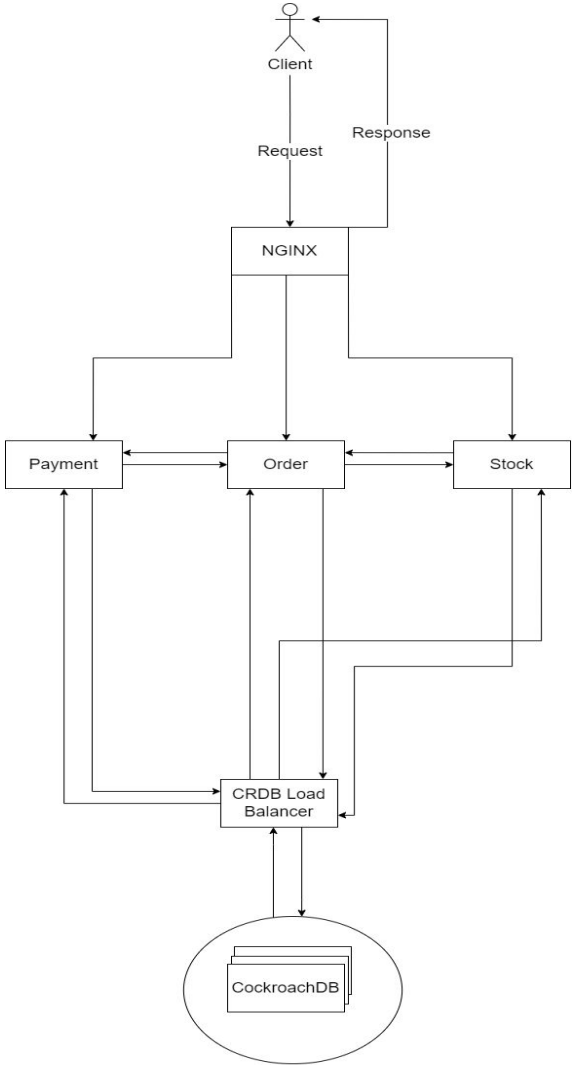
Adam Kadiev
Ayush Patandin
Jakub Nguyen
Mohamed Rashad
Kevin Nanhekhan

TU Delft

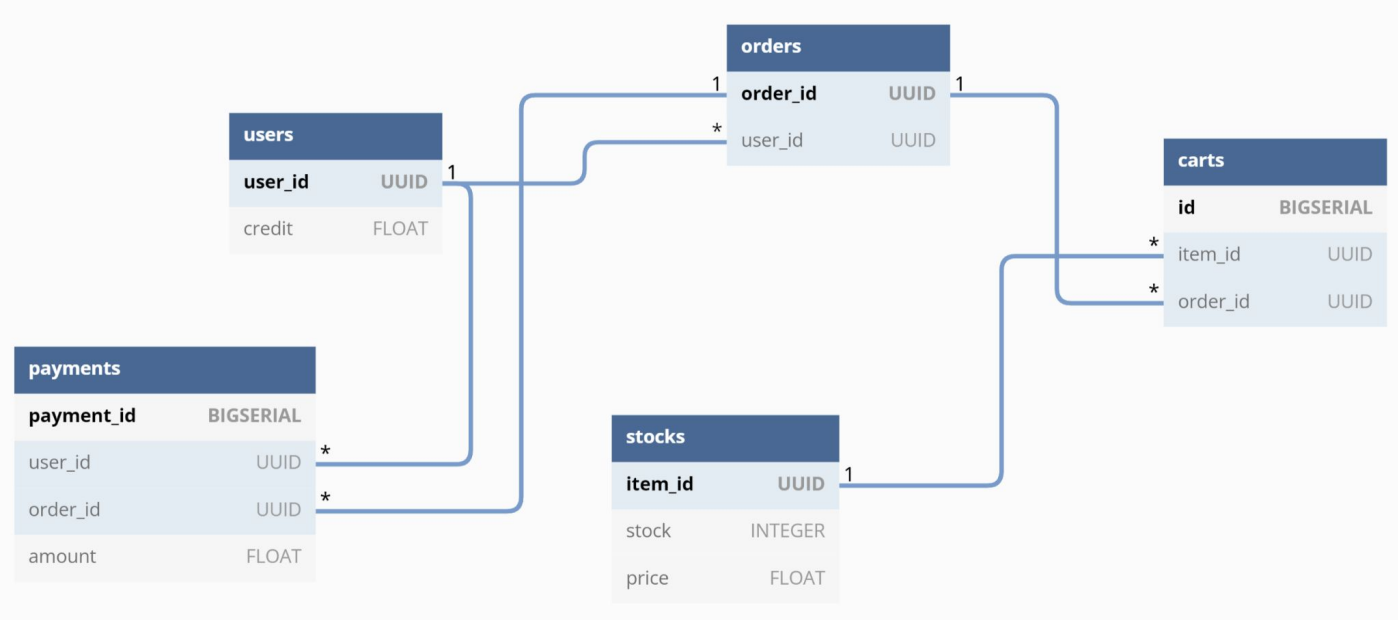
20-06-2022

System Design

- Services are stateless
  - Each service can scale independently
- Interservice communication through REST calls
  - Services use Flask API
- Database used is a relational database
  - Key-value store under the hood
  - Distributed database
  - Load balancer
  - Adheres to ACID principles



CockroachDB table overview

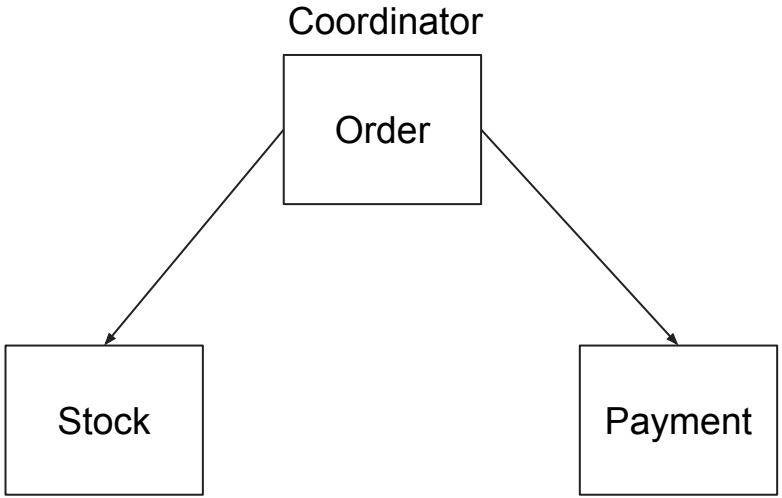


Method of transaction execution

- SQLAlchemy ORMs
  - SQLAlchemy sessions
  - Model Schema (Python Classes)
- Transactions using Two-Phase Commit
  - Services become stateful
  - Difficult to scale
- Transactions Managed by database
  - Services are stateless
  - Easy to scale

Method used for consistency

- Two-Phase Commit
  - Slow due to synchronous REST calls
- Transactions
  - Automatic retries
  - ACID guarantees



Scalability

- Horizontal Pod Scaling in Kubernetes
  - Given CPU usage, creates new instance of services
  - Minimum and maximum amount of pods specified

Fault-tolerance

- CockroachDB replication and partitioning

Results: 2PC

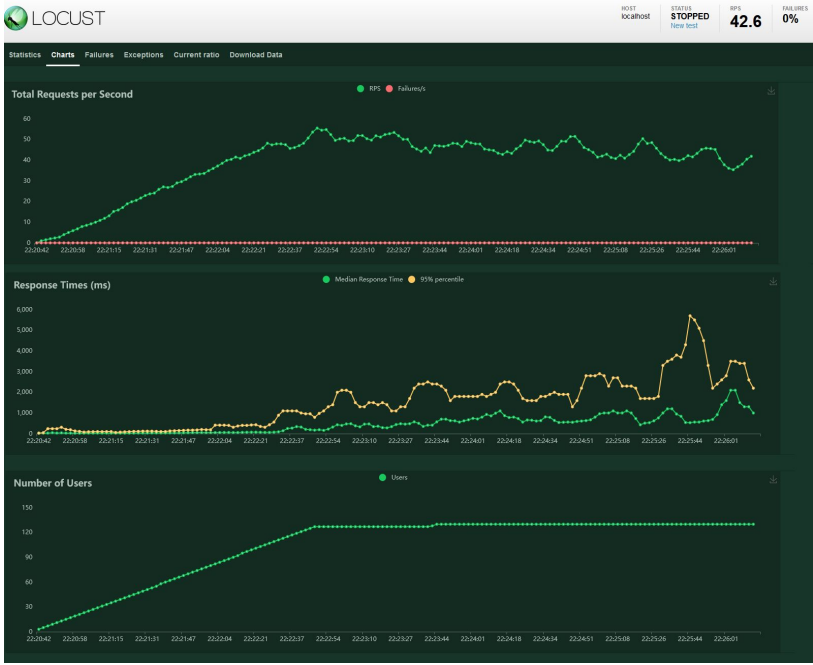
- Delayed requests due to DB block
- 0 failures for 5 spawned users
- Prepared transactions in checkout

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/orders/addItem/[order_id]/[item_id]	71	0	17	25	37	18	13	37	0	0.3	0
POST	/orders/checkout/[order_id]	35	0	100	180	250	120	78	249	7	0.3	0
POST	/orders/create/[user_id]	37	0	18	27	62	21	13	62	52	0	0
POST	/payment/add_funds/[user_id]/[amount]	37	0	20	32	48	22	13	48	14	0.1	0
POST	/payment/create_user	39	0	16	21	32	17	13	32	51	0.2	0
POST	/stock/add/[item_id]/[number]	80	0	17	27	47	20	14	47	0	0.7	0
POST	/stock/item/create/[price]	80	0	15	28	100	20	12	102	51	0.7	0
	Aggregated	**379**	**0**	**17**	**62**	**180**	**29**	**12**	**249**	**23**	**2.3**	**0**

Results: Scalable Version

- 4 order and payment deployments
- 1 stock deployment
- Faster than 2PC
- Few inconsistencies for ~1000 users

  → resolvable by performing single transaction in checkout




```
Consistency test - Creating tmp folder...
Consistency test - tmp folder created
Consistency test - Populating the databases...
populate - Creating items ...
populate - Items created
populate - Creating users ...
populate - Users created
Consistency test - Databases populated
Consistency test - Starting the load test...
stress - Creating orders...
stress - Orders created ...
stress - Running concurrent checkouts...
stress - Concurrent checkouts finished...
Consistency test - Load test completed
Consistency test - Starting the consistency evaluation...
verify - Stock service inconsistencies in the logs: -4
verify - Stock service inconsistencies in the database: 0
verify - Payment service inconsistencies in the logs: 4
verify - Payment service inconsistencies in the database: 4.0
Consistency test - Consistency evaluation completed
```

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/orders/additem/[order_id]/[item_id]	2312	0	360	1800	3900	678	17	6562	0	7.7	0
POST	/orders/checkout/[order_id]	1497	0	700	2100	4800	958	44	6728	10	6	0
POST	/orders/create/[user_id]	1530	0	340	1700	3600	657	16	6538	52	4.7	0
DELETE	/orders/removeItem/[order_id]/[item_id]	325	0	420	2000	3900	761	18	6492	0	1.5	0
POST	/payment/add_funds/[user_id]/[amount]	1325	0	73	190	440	94	18	650	14	3.4	0
POST	/payment/create_user	1551	0	60	160	350	82	16	542	51	3.9	0
POST	/stock/add/[item_id]/[number]	2254	0	760	1800	2400	826	18	2877	0	6.9	0
GET	/stock/find/[item_id]	133	0	760	1700	2600	821	9	2741	39	0.5	0
POST	/stock/item/create/[price]	2283	0	760	1800	2400	836	16	2907	51	7.5	0
POST	/stock/subtract/[item_id]/[number]	133	0	720	1800	2400	828	23	2443	0	0.5	0
	Aggregated	**13343**	**0**	**310**	**1700**	**3100**	**637**	**9**	**6728**	**24**	**42.6**	**0**

Q&A