MeSS '22

International workshop on MDE for Smart IoT Systems

co-located with *STAF 2022*

Nantes, 5-8 July 2022

Key-Value vs Graph-based data lakes for realizing Digital Twin Systems

Daniel Pérez-Porras
Universidad de Málaga, Spain

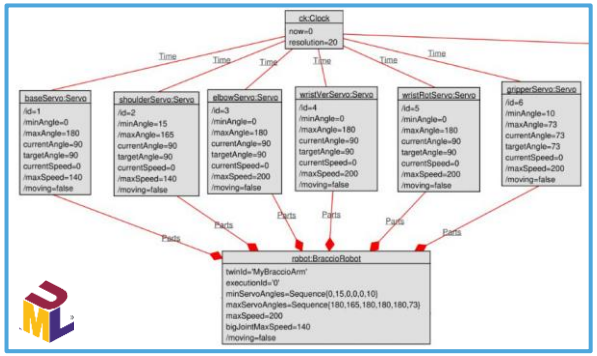
Paula Muñoz
Universidad de Málaga, Spain

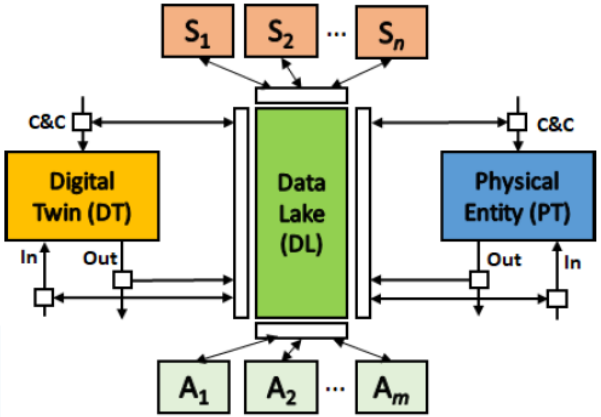
Javier Troya
Universidad de Málaga, Spain

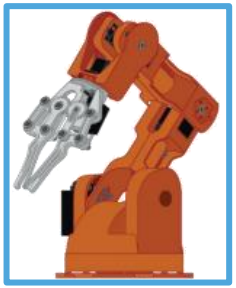
Antonio Vallecillo
Universidad de Málaga, Spain

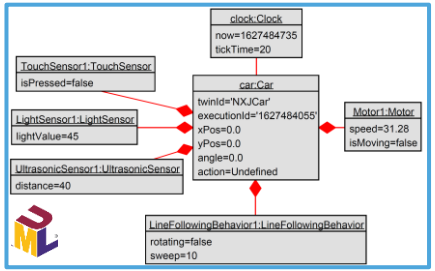
Digital Twin Systems

A Digital Twin is a comprehensive digital representation of an actual system, service or product (the Physical Twin), synchronized at a specified frequency and fidelity [1].







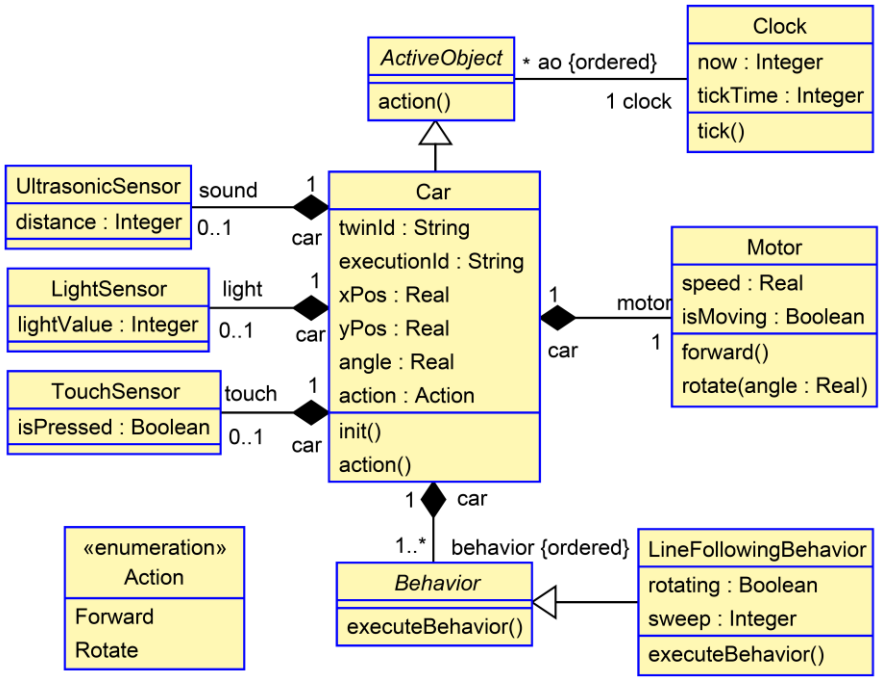


https://tinyurl.com/MeSSDTArch

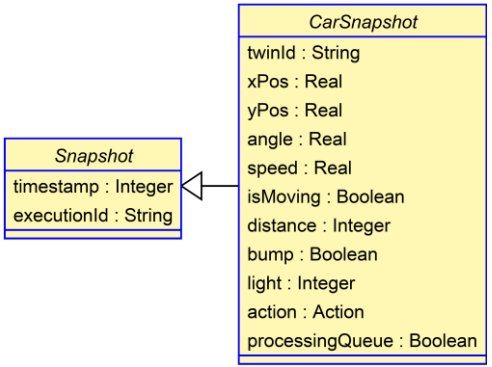
[1] Digital Twin Consortium, “Glossary of digital twins”, 2021. [Online]. Available: https://www.digitaltwinconsortium.org/glossary/index.html
[2] P. Muñoz, J. Troya, A. Vallecillo, Using UML and OCL Models to Realize High-Level Digital Twins, in: Proc. of ModDiT2021@MODELS’21, IEEE, 2021, pp. 212–220

Information format – Lego NXJ Car

Car's Digital Twin model



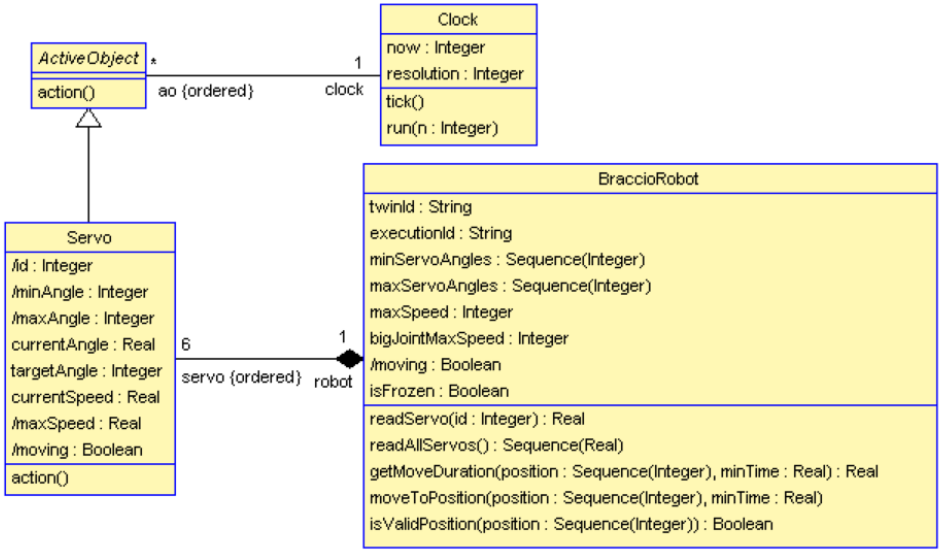
Car's Snapshots



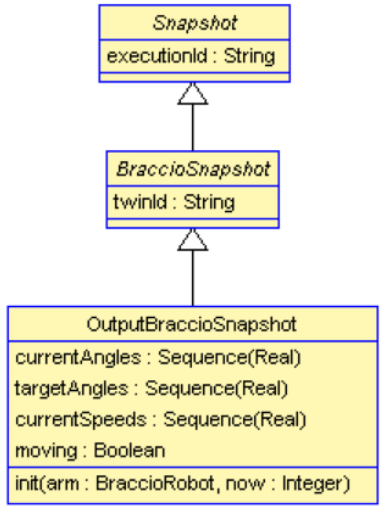

```
InputCarSnapshot5:InputCarSnapshot
timestamp=1627484835
executionId='1627484055'
twinId='NXJCar'
xPos=27.81
yPos=0.0
angle=0.0
speed=31.28
isMoving=true
distance=89
bump=false
light=45
action=#Forward
processingQueue=false
```

Information format – Arduino Braccio

Braccio's Digital Twin model



Braccio's Snapshots



outputSnapshot:OutputBraccioSnapshot

```
timestamp=1656492966
isProcessed=false
whenProcessed=Undefined
executionId='test'
twinId='MyBraccioArm'
currentAngles=Sequence{90,90,90,90,90,73}
targetAngles=Sequence{90,90,90,90,90,73}
currentSpeeds=Sequence{0,0,0,0,0,0}
moving=false
```

redis as Data Lake

Key-value database

- You need to know the keys to query the values.
- To make more complex queries you need additional registers or composed keys

snapshotType : twinId : executionId : timestamp

```
HGETALL PTOutputSnapshot:SSSCar:1627484055:16274848179

1) "twinId"              2) "SSSCar"
3) "executionId"         4) "1627484055"
5) "timestamp"           6) "16274848179"
7) "xPos"                8) "2.996127561"
9) "yPos"                10) "15.51560811"
11) "angle"              12) "4.71"
13) "speed"              14) "62.56"
15) "isMoving"           16) "TRUE"
17) "distance"           18) "35"
19) "bump"               20) "FALSE"
21) "light"              22) "45"
23) "action"             24) "#Forward"
25) "processingQueue"    26) "FALSE"
```

Snapshots in which the car was at a distance between 0 and 100

```
ZRANGEBYSCORE PTOutputSnapshot:SSSCar_DISTANCE_LIST 0 100

1) "PTOutputSnapshot:SSSCar:1627484055:16274847352"
2) "PTOutputSnapshot:SSSCar:1627484055:16274848139"
3) "PTOutputSnapshot:SSSCar:1627484055:16274848148"
4) "PTOutputSnapshot:SSSCar:1627484055:16274848158"
5) "PTOutputSnapshot:SSSCar:1627484055:16274848168"
```

All Snapshots for a given Physical Twin

```
ZRANGEBYLEX PTOutputSnapshot_PROCESSED
"[PTOutputSnapshot:SSSCar:"
"[PTOutputSnapshot:SSSCar:\xff"

1) "PTOutputSnapshot:SSSCar:1627484055:16274847352"
2) "PTOutputSnapshot:SSSCar:1627484055:16274847359"
3) "PTOutputSnapshot:SSSCar:1627484055:16274847368"
4) "PTOutputSnapshot:SSSCar:1627484055:16274847380"
5) "PTOutputSnapshot:SSSCar:1627484055:16274847391"
```

redis as Data Lake

Advantages

- **Lightweight**, easy to deploy
- **High-responsiveness.** It is optimized to deliver fast responses to a massive amount of petitions.

Disadvantages

- **Rigid.** You need to define the database's structure beforehand according to the queries you are planning to make.
- **Duplicated information** to perform queries.

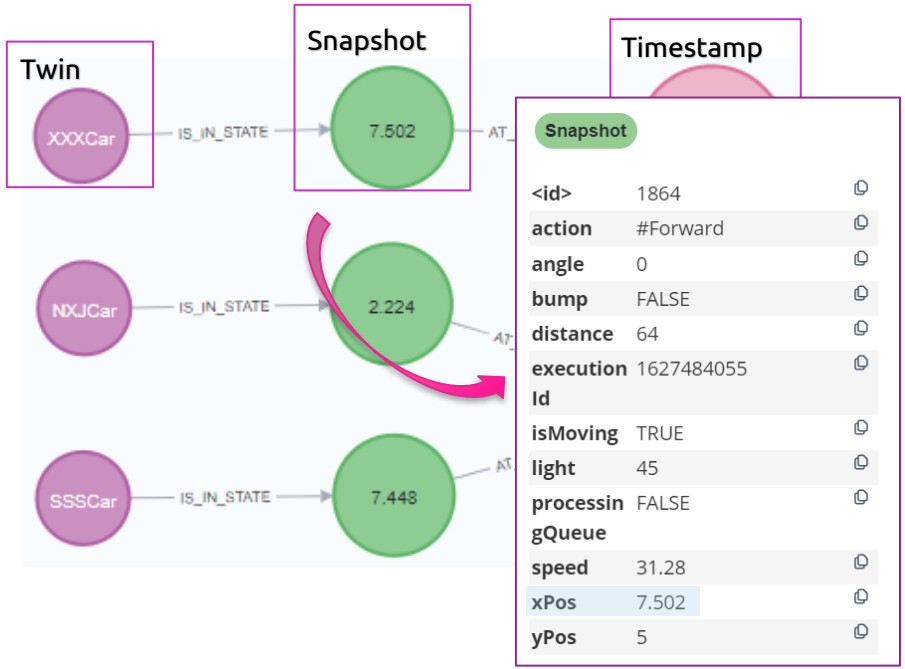
neo4j as Data Lake

Graph database

- ❖ You only need to know the database structure to query for values.
- ❖ It enables to make complex queries using the graph relationships.

Snapshots for any Digital Twin, which occur between two given

```
MATCH (before:Timestamp)
WHERE before.value ⩾ 16274847350
MATCH (after:Timestamp)
WHERE after.value ⩽ 16274847540
AND before.value < after.value
MATCH (before)-[:NEXT]→(after)
MATCH (dt:DigitalTwin)-[:IS_IN_STATE]→(snp)-[:AT_THE_TIME]→(before)
MATCH (dt2:DigitalTwin)-[:IS_IN_STATE]→(snp2)-[:AT_THE_TIME]→(after)
WHERE dt <> dt2
RETURN dt, snp, dt2, snp2, before, after
LIMIT 2
```



Comparative analysis neo4j vs redis

Q1: Does a given car follow the line during a given time window?

```
MATCH (dt: DigitalTwin {name:'NXJCar'})
MATCH (snp: Snapshot)
MATCH (ts: Timestamp)
MATCH (dt)-[:IS_IN_STATE]→(snp)-[:AT_THE_TIME]→(ts)
WHERE ts.value ⩾ 16274847350
AND ts.value ⩽ 16274847490
RETURN avg(snp.light)
```

```
╒═══════════════════╕
│"avg(snp.light)"   │
╞═══════════════════╡
│45.0               │
└───────────────────┘
```

Comparative analysis neo4j vs redis

Q1: Does a given car follow the line during a given time window?

Q2: Average value of a given servo for all robot arms.

```
MATCH (ts:Timestamp)
WHERE ts.value ⩽ 20 AND ts.value ⩾ 0
MATCH (dt:DigitalTwin)
MATCH (dt)-[:IS_IN_STATE]→(snp)-[:AT_THE_TIME]→(ts)
WITH dt, snp.currentAngles_3 as snapshots
RETURN dt, avg(snapshots)
```

"dt"	"avg(snapshots)"
{"name":"braccio"}	94.125
{"name":"braccio2"}	107.16666666666666

Comparative analysis neo4j vs redis

Q1: Does a given car follow the line during a given time window?

Q2: Average value of a given servo for all robot arms.

Q3: Number of cars in a squared area during a given time window.

```
MATCH (snp:Snapshot)
MATCH (ts:Timestamp)
WHERE ts.value ⩽ 16274847380 AND ts.value ⩾ 16274847350
AND snp.xPos ⩾ 3 AND snp.xPos ⩽ 6
AND snp.yPos ⩾ 0 AND snp.yPos ⩽ 3
MATCH (dt)-[:IS_IN_STATE]→(snp)-[:AT_THE_TIME]→(ts)
WITH DISTINCT dt
RETURN count(dt)
```

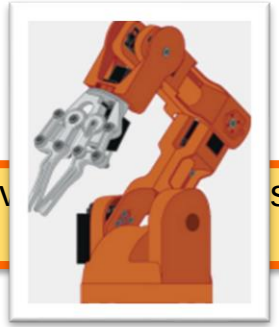
```
┌──────────┐
│"count(dt)"│
├──────────┤
│2         │
└──────────┘
```

Comparative analysis neo4j vs redis

Q1

"dt.name"	"upperBound.value"	"lowerBound.value"	"avg(s.currentAngles_3)"
"braccio"	17	7	94.66666666666667
"braccio2"	17	7	108.66666666666667
"braccio"	24	14	66.2

Average v servo for all



Q3: Number of cars in a squared area during a given time window.

```
MATCH timePeriod=(lowerBound:Timestamp)
    -[:NEXT*2..15]→(upperBound:Timestamp)
WHERE upperBound.value - lowerBound.value = 10
WITH *, nodes(timePeriod) as timestamps
MATCH (dt:DigitalTwin)-[:IS_IN_STATE]
      →(snp:Snapshot)-[:AT_THE_TIME]→(ts:Timestamp)
WHERE ts in timestamps
WITH upperBound, lowerBound, timestamps, dt,
     collect(snp) as snapshots
UNWIND snapshots as s
RETURN dt.name, upperBound.value, lowerBound.value,
    avg(s.currentAngles_3)
```

Q4: Average value of a given servo for each robot arm in the system during a certain period.

Comparative analysis neo4j vs redis

Q1: Does a given car foll
during a given time w

```
MATCH (snp : Snapshot)
WITH *, point({x: snp.xPos, y: snp.yPos, crs: 'cartesian'}) AS p1
MATCH (snp2 : Snapshot)
WITH *, point({x: snp2.xPos, y: snp2.yPos, crs: 'cartesian'}) AS p2
WITH snp, snp2, point.distance(p1,p2) AS dist
WHERE dist < 3
MATCH (dt : DigitalTwin)
MATCH (dt)-[:IS_IN_STATE]→(snp)
MATCH (dt2 : DigitalTwin)
MATCH (dt2)-[:IS_IN_STATE]→(snp2)
WHERE dt <> dt2
MATCH (snp)-[:AT_THE_TIME]→(ts)←[:AT_THE_TIME]-(snp2)
RETURN dt, dt2, snp, snp2, ts
```

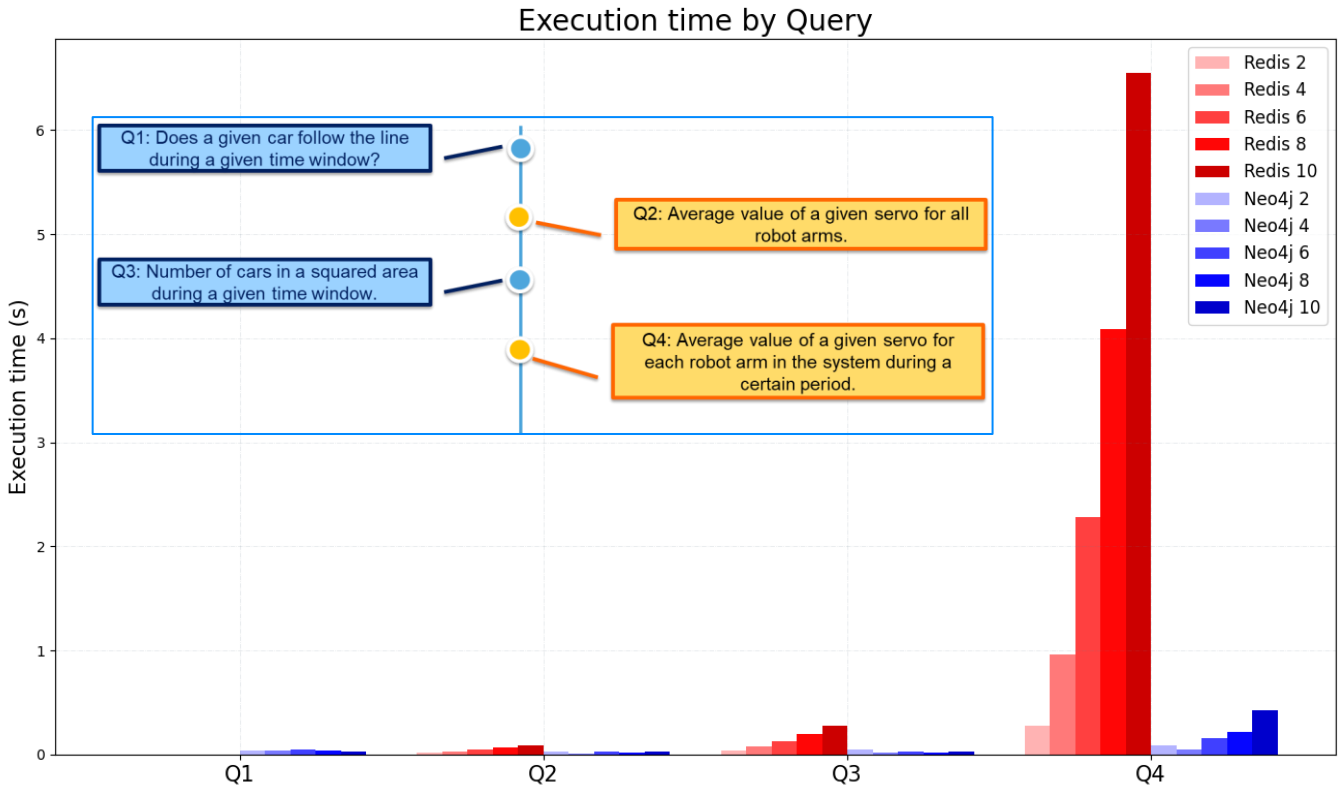
ue of a given servo for all
bot arms.

Q3: Number of cars in a s
during a given time w

Q4: A given servo for
each ystem during a
od.

Q5: Number of collisions between cars during a given time window.

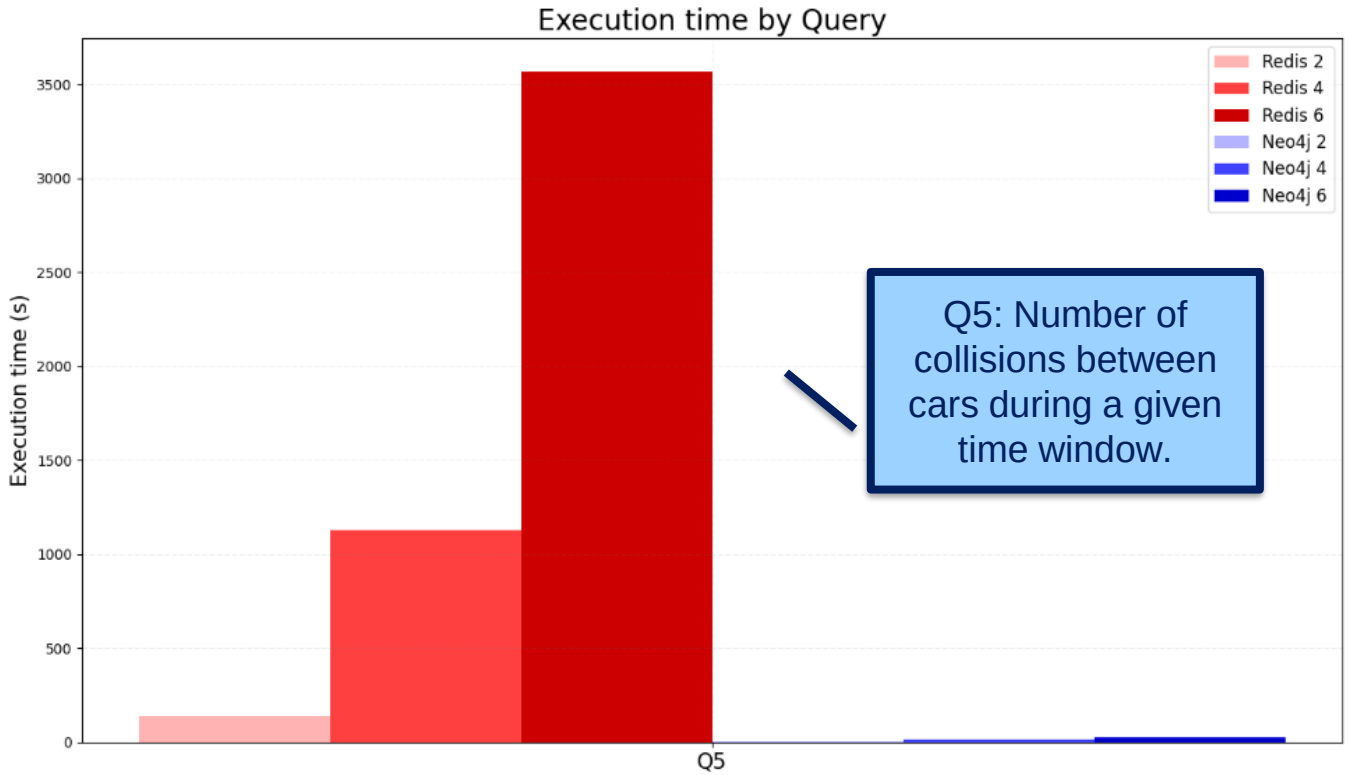
Comparative analysis neo4j vs redis



# Robots	# Snaps /robot	# Snaps
2	50	100
4	100	400
8	200	1600
16	400	6400
32	800	25600

# Cars	# Snaps/car	# Snaps
6	213	1278
12	426	5112
24	852	20448
48	1704	81792
96	3408	327168

Comparative analysis  vs 



# Robots	# Snaps/robot	# Snaps
2	50	100
4	100	400
8	200	1600

# Cars	# Snaps/car	# Snaps
6	213	1278
12	426	5112
24	852	20448

Conclusions and future work

- **Redis** is really convenient when you are not expecting complex queries or aggregated data from the database, since you will have to rely on manual postprocessing of the information.
- **Neo4j** allows complex queries with good performance and great expressivity by using a simple structure to model time information.

- In our **future work**,
  - We plan on comparing this approach with one using a time-series database.
  - Try real-time analysis using a working system to measure performance in ingestion and querying in a real environment.

https://tinyurl.com/MESSRedisVSNeo

MeSS '22
International workshop on MDE for Smart IoT Systems

co-located with *STAF 2022*
Nantes, 5-8 July 2022

Thank you for your attention

Key-Value vs Graph-based data lakes for realizing Digital Twin Systems

Daniel Pérez-Porras
Universidad de Málaga, Spain

Paula Muñoz
Universidad de Málaga, Spain

Javier Troya
Universidad de Málaga, Spain

Antonio Vallecillo
Universidad de Málaga, Spain

SOFTWARE
UNIVERSIDAD DE MÁLAGA

Comparative analysis neo4j vs redis

Q1: Does a given car follow the line during a given time window?

Q2: Average value of a given servo for all robot arms.

Q3: Number of cars in a squared area during a given time window.

Q4: Average value of a given servo for each robot arm in the system during a certain period.

Q5: Number of collisions between cars during a given time window.