NSF/IUCRC CAC PROJECT

INTEGRATED VISUALIZING, MONITORING, AND MANAGING HPC SYSTEMS

Jie Li

Doctoral Student, TTU

11/20/2020

Advisors:

Mr. Jon Hass, SW Architect, Dell Inc.

Dr. Alan Sill, Managing Director, HPCC, TTU

Dr. Yong Chen, Associate Professor, CS Dept, TTU

Dr. Tommy Dang, Assistant Professor, CS Dept, TTU

TABLES

- Convert 3 months of data from InfluxDB to TimescaleDB

```
name: measurements
name
----
CPUUsage
FanSensor
Health
JobsInfo
Load
MemUsage
NodeJobs
Power
SwapUsage
TempSensor
>
```

InfluxDB measurements

```
           List of relations
 Schema |    Name     | Type  |  Owner
--------+------------+-------+----------
 public | CPUUsage   | table | postgres
 public | FanSensor  | table | postgres
 public | Health     | table | postgres
 public | JobsInfo   | table | postgres
 public | Load       | table | postgres
 public | MemUsage   | table | postgres
 public | NodeJobs   | table | postgres
 public | Power      | table | postgres
 public | SwapUsage  | table | postgres
 public | TempSensor | table | postgres
(10 rows)
```

TimescaleDB tables

TABLES

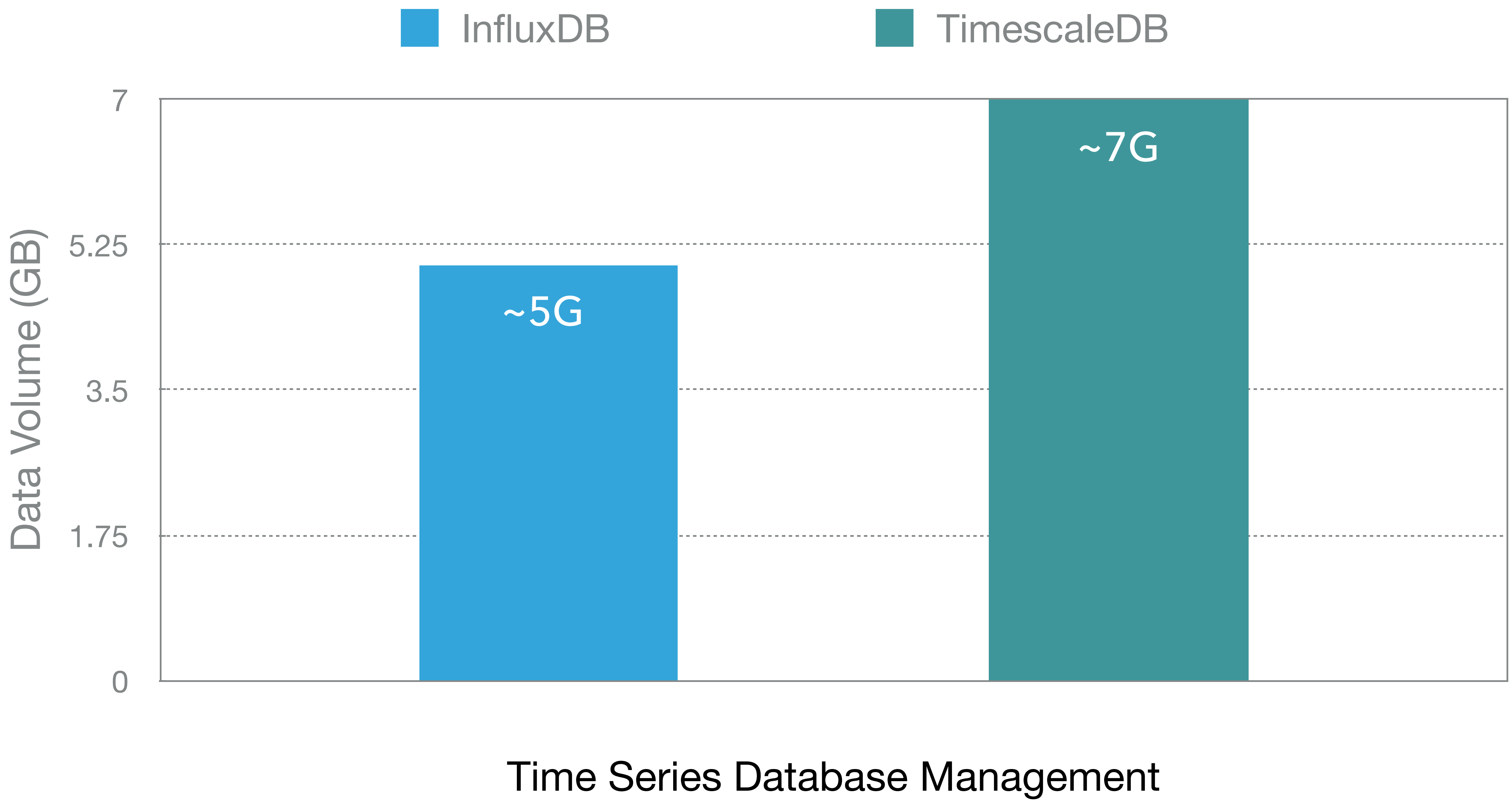
InfluxDB data points
in Power measurement

```
> select * from Power order by time desc limit 10
name: Power
time                Label     NodeId      Value
----                -----     ------      -----
1606318029590545408 NodePower 10.101.9.60 252
1606318029590545408 NodePower 10.101.9.59 284
1606318029590545408 NodePower 10.101.9.58 304
1606318029590545408 NodePower 10.101.9.57 291
1606318029590545408 NodePower 10.101.9.56 265
1606318029590545408 NodePower 10.101.9.55 233
1606318029590545408 NodePower 10.101.9.54 315
1606318029590545408 NodePower 10.101.9.53 298
1606318029590545408 NodePower 10.101.9.52 324
1606318029590545408 NodePower 10.101.9.51 314
>
```

TimescaleDB data points
in Power table

```
hpcc_metrics=# select * from "Power" order by time desc limit 10;
           time            |   Label   |    NodeId    | Value
---------------------------+-----------+--------------+-------
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.25 |   309
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.26 |   204
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.27 |   301
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.28 |   309
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.29 |   217
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.30 |   285
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.31 |   326
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.32 |   301
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.33 |   309
 2020-11-01 00:59:08.76888 | NodePower | 10.101.10.34 |   134
(10 rows)
```

DATA VOLUME


InfluxDB
TimescaleDB
Data Volume (GB)
7
5.25
3.5
1.75
0
~5G
~7G
Time Series Database Management

TIMESCALEDB-TABLES

- Reduce 10 tables to 3 tables
  - BMCMetrics table for storing the telemetry data
  - RMSMetrics table for storing the Slurm data
  - JobsInfo for storing the job metadata

List of relations

Schema	Name	Type	Owner
public	CPUUsage	table	postgres
public	FanSensor	table	postgres
public	Health	table	postgres
public	JobsInfo	table	postgres
public	Load	table	postgres
public	MemUsage	table	postgres
public	NodeJobs	table	postgres
public	Power	table	postgres
public	SwapUsage	table	postgres
public	TempSensor	table	postgres

(10 rows)

List of relations

Schema	Name	Type	Owner
public	BMCMetrics	table	postgres
public	RMSMetrics	table	postgres
public	JobsInfo	table	postgres

TIMESCALEDB-SCHEMA

time	NodeId	NodePower	FAN_1	Inlet Temp	CPU1 Temp	CPU2 Temp
2020-11-01 00:59:08.76888	10.101.10.25	309	9170	15	77	55
2020-11-01 00:59:08.76888	10.101.10.26	204	9170	14	54	41
2020-11-01 00:59:08.76888	10.101.10.27	301	9310	14	79	59
2020-11-01 00:59:08.76888	10.101.10.28	309	9170	14	77	56
2020-11-01 00:59:08.76888	10.101.10.29	217	9310	9	57	42
2020-11-01 00:59:08.76888	10.101.10.30	285	9310	9	71	56
2020-11-01 00:59:08.76888	10.101.10.31	326	9170	10	77	58
2020-11-01 00:59:08.76888	10.101.10.32	301	9170	10	80	53
2020-11-01 00:59:08.76888	10.101.10.33	309	9450	10	78	50
2020-11-01 00:59:08.76888	10.101.10.34	134	9800	10	38	34

(10 rows)

TimescaleDB – BMC Metrics table

TIMESCALEDB-SCHEMA

time	NodeId	JobList	CPUUsage	MemUsage
2020-11-01 00:59:08.76888	10.101.10.25	['1925771']	3.55	40.53
2020-11-01 00:59:08.76888	10.101.10.26	['1926327']	9.96	12.21
2020-11-01 00:59:08.76888	10.101.10.27	['1925169']	16.38	5.16
2020-11-01 00:59:08.76888	10.101.10.28	['1925596']	15.11	29.73
2020-11-01 00:59:08.76888	10.101.10.29	['1921878']	17.43	2.69
2020-11-01 00:59:08.76888	10.101.10.30	['1921878']	18.42	8.56
2020-11-01 00:59:08.76888	10.101.10.31	['1921772']	4.15	5.61
2020-11-01 00:59:08.76888	10.101.10.32	['1925771']	4.12	4.35
2020-11-01 00:59:08.76888	10.101.10.33	['1925596']	13.56	5.11
2020-11-01 00:59:08.76888	10.101.10.34	['1925771']	12.18	16.37

(10 rows)

TimescaleDB – RMS Metrics table

TIMESCALEDB-SCHEMA

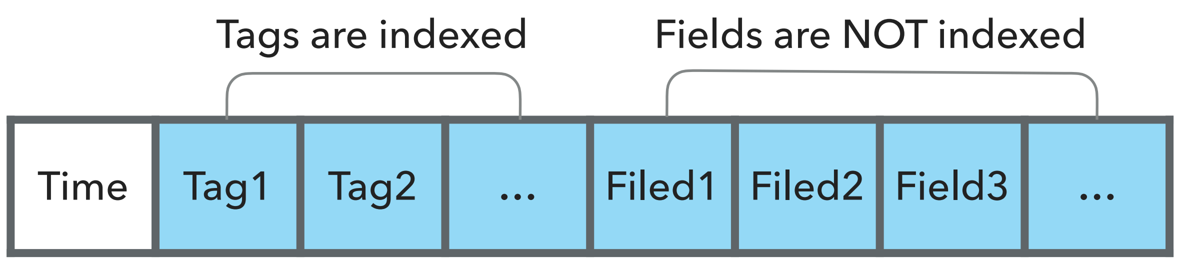
JobId	JobName	User	StartTime	SubmitTime	TotalNodes	CPUCores
1897748	ZSM_atom	ipandey	1594277911000000000	1594277911000000000	1	36
2100822	fluent	loboyd	1606329572000000000	1606329536000000000	1	36
2100600	QLOGIN	sojkwon	1606328982000000000	1606328979000000000	1	1
2100389	lammps	bdankesr	1606328790000000000	1606328127000000000	1	36
2100388	lrt_reg_88	cpokorny	1606329837000000000	1606328016000000000	1	18
2100387	lrt_reg_77	cpokorny	1606329828000000000	1606328010000000000	1	18
2100386	lrt_reg_66	cpokorny	1606329629000000000	1606328005000000000	1	18
2100385	lrt_reg_55	cpokorny	1606329620000000000	1606328000000000000	1	18
2100384	lrt_reg_44	cpokorny	1606329377000000000	1606327995000000000	1	18
2100383	lrt_reg_33	cpokorny	1606329368000000000	1606327990000000000	1	18

TimescaleDB – JobsInfo table


QUESTIONS?/COMMENTS?

DATA MODEL

InfluxDB	TimescaleDB
Non-relational DB (built from scratch in Go)	Relational DB (built on PostgreSQL)
floats, ints, strings, and booleans	floats, ints, strings, booleans, arrays, JSON, etc.
Only tags values are indexed	Indexable on all fields



Time	Tag - NodeIP	Field - JobList
1583792296	“101.10.1.1”	“[123456, 123457]”

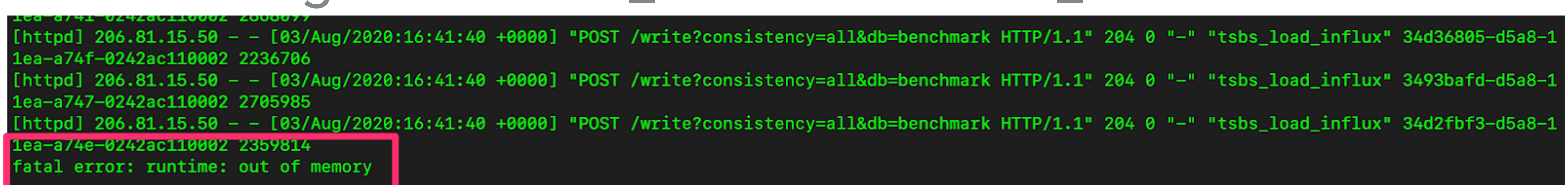
Time	Field - NodeIP	Field - JobList
1583792296	“101.10.1.1”	[123456, 123457]

NodeIP	Cluster	Rack	CPUs	...
“101.10.1.1”	“Quanah”	1	36	...

JobId	JobName	Start	NodeList	...
123456	“test”	1583792200	36	...

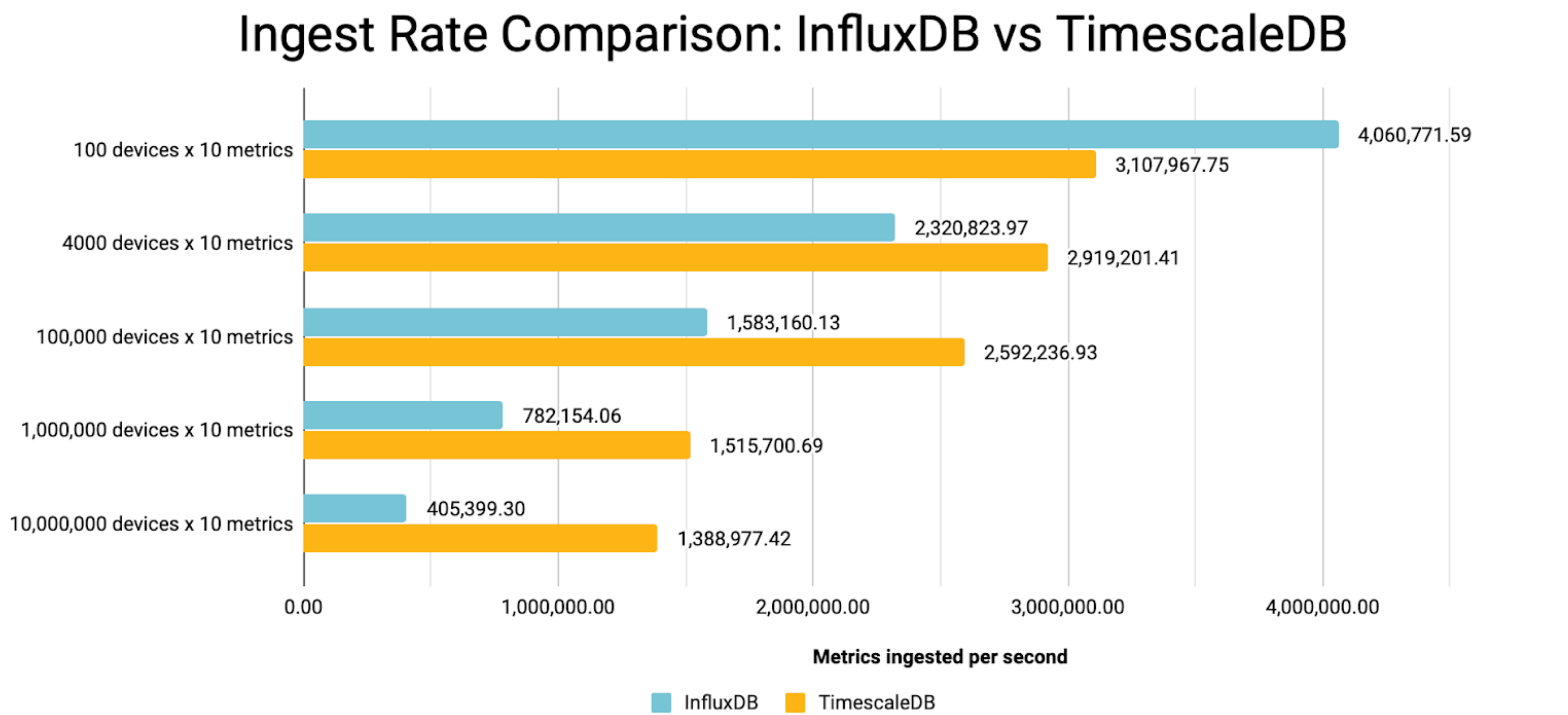
STABILITY

- Inserting batches into InfluxDB
  - Inserting batches of 10k into InfluxDB at high cardinalities will have write errors caused by timeouts, exceeding the maximum cache memory size, fatal out of memory errors.
  - Increasing maximum cache size and decreasing the batch size could solve these errors.
- Reading queries on InfluxDB
  - InfluxDB at high cardinalities could consume all available memory to run the query and crashed with an Out of Memory error.
- Writing large batches and reading queries on TimescaleDB do not have such issues. PostgreSQL limits system memory usage with settings like shared_buffers and work_mem.



Ref: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/.

PERFORMANCE



- InfluxDB outperforms TimescaleDB for workloads with low cardinality
- InfluxDB insert performance drops off dramatically as cardinality increases.
- TimescaleDB has ~3.5x the insert performance as InfluxDB

Ref: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/.

PERFORMANCE

Query Performance (measured in milliseconds)

	100 devices x 1 metric			100 devices x 10 metrics			4,000 devices x 10 metrics		
Simple rollups [1]	Influx	Timescale	Influx / Timescale	Influx	Timescale	Influx / Timescale	Influx	Timescale	Influx / Timescale
single-groupby-1-1-1	11.33	12.11	94%	5.49	7.76	71%	6.15	6.02	102%
single-groupby-1-1-12	32.87	13.36	246%	26.48	14.62	181%	32.61	22.68	144%
single-groupby-1-8-1	43.56	7.29	598%	13.04	10.17	128%	16.09	17.06	94%
single-groupby-5-1-1	—	—	—	12.4	6.67	186%	14.76	8.62	171%
single-groupby-5-1-12	—	—	—	82.8	17.87	463%	106.8	23.08	463%
single-groupby-5-8-1	—	—	—	49.32	12.51	394%	64.6	17.53	369%
Aggregates [2]									
cpu-max-all-1	—	—	—	13.84	13.69	101%	16.14	17.68	91%
cpu-max-all-8	—	—	—	95.36	56.61	168%	104.25	66.79	156%
Double rollups [3]									
double-groupby-1	500.55	272.46	184%	152.64	331.54	46%	6,050.85	11,060.68	55%
double-groupby-5	—	—	—	703.36	508.7	138%	31,801.62	22,479.91	141%
double-groupby-all	—	—	—	1393.91	869.81	160%	65,212.69	34,603.17	188%
Thresholds [4]									
high-cpu-1	2,652.17	304.9	870%	2,952.15	836.49	353%	180,235.94	35,049.85	514%
high-cpu-all	20.68	8.25	251%	29.5	11.42	258%	30.56	17.44	175%
Complex queries									
lastpoint [5]	367.45	7.55	4,867%	192.69	9.49	2,030%	10,514.64	147.36	7,135%
groupby-orderby-limit [6]	3,344.5	752.68	444%	2411.74	700.02	345%	114,419.32	27,990.85	409%

- Generally, Timescale outperforms InfluxDB.
- When simply rolling up a single metric, InfluxDB can sometimes outperform TimescaleDB
- TimescaleDB vastly outperforms InfluxDB for complex queries.

InfluxDB outperforms

TimescaleDB outperforms

Ref: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/.

CONCLUSION

- We do NOT need to use a non-TSDB to store static data (job data) if using TimescaleDB to store the HPC monitoring data.
- TimescaleDB is much more stable when writing and reading high-cardinality datasets.
- TimescaleDB performs better on writing and reading high-cardinality datasets.

> We may use TimescaleDB as the main storage solution for monitoring the RedRaider cluster.