InfluxDB

陈 超 CrazyJvm
chenchao@qiniu.com
SDCC 2016

Agenda

- basic concept
- how to use
- storage engine
- cluster design
- what we did
- conclusion

Agenda

- ***basic concept***
- how to use
- storage engine
- cluster design
- what we did
- conclusion

TSDB

- A time series database (TSDB) is a software system that is optimized for handling time series data, arrays of numbers indexed by time (a datetime or a datetime range). In some fields these time series are called profiles, curves, or traces.

https://en.wikipedia.org/wiki/Time_series_database

DB-Engines Ranking of Time Series DBMS

19 systems in ranking, March 2016

Rank Mar 2016	Rank Feb 2016	Rank Mar 2015	DBMS	Database Model	Score Mar 2016	Score Feb 2016	Score Mar 2015
1.	1.	1.	InfluxDB	Time Series DBMS	3.95	+0.32	+2.79
2.	2.		RRDtool	Time Series DBMS	2.60	-0.08	
3.	3.		Graphite	Time Series DBMS	1.57	-0.01	
4.	4.		OpenTSDB	Time Series DBMS	1.39	-0.02	
5.	5.	↓ 2.	Kdb+	Multi-model	1.24	-0.07	+0.42
6.	6.		KairosDB	Time Series DBMS	0.16	-0.01	
7.	↑ 9.		Druid	Time Series DBMS	0.15	+0.06	
8.	8.		Axibase	Time Series DBMS	0.14	+0.03	
9.	↓ 7.		Prometheus	Time Series DBMS	0.13	+0.01	
10.	10.		Riak TS	Time Series DBMS	0.03	-0.03	
11.	11.		TempoIQ	Time Series DBMS	0.01	-0.02	
12.	12.		Blueflood	Time Series DBMS	0.00	±0.00	
12.	12.		Cityzen Data	Time Series DBMS	0.00	±0.00	
12.	12.		Hawkular Metrics	Time Series DBMS	0.00	±0.00	
12.	12.		Infiniflux	Time Series DBMS	0.00	±0.00	
12.	12.		Newts	Time Series DBMS	0.00	±0.00	
12.	12.		SiteWhere	Time Series DBMS	0.00	±0.00	
12.	12.		TimeSeries.Guru	Time Series DBMS	0.00	±0.00	
12.			Yanza	Time Series DBMS	0.00		

What TSDB should do?

- query time-related data efficiently
- easy to downsampling
- handle expiring data automatically

Why InfluxDB?

- No external dependencies
- Easy to get started
- Elegant restful api
- Powerful query language
- Horizontally scale
- Written in pure Go :)
- ……

Concept

- measurement

Concept

- tag
- tagset
- tags are indexed

Concept

- series = measurement + tagset

Concept

- field

Concept

- timestamp

Concept

- Continuous queries

Concept

- Retention policy

Concept

- line protocol
- measurement[,tag1,tag2,…] field1[,field2,…] ts
- cpu,host=qn00001 value=0.1 1434055562000000000

Agenda

- basic concept
- ***how to use***
- storage engine
- cluster design
- what we did
- conclusion

write point

- curl -i -XPOST 'http://localhost:8086/write?db=mydb' --data-binary 'cpu_load_short,host=server01,region=us-west value=0.64 1434055562000000000'

create & show retention

```
CREATE RETENTION POLICY two_hours ON food_data DURATION 2h REPLICATION 1
```

```
SHOW RETENTION POLICIES ON food_data
```

write batch points

```
curl -i -XPOST 'http://localhost:8086/write?db=mydb' --data-binary
'cpu_load_short,host=server02 value=0.67
cpu_load_short,host=server02,region=us-west value=0.55 1422568543702900257
cpu_load_short,direction=in,host=server01,region=us-west value=2.0
1422568543702900257'
```

```
curl -i -XPOST 'http://localhost:8086/write?db=mydb' --data-binary @cpu_data.txt
```

query data

```
curl -G 'http://localhost:8086/query?pretty=true' --data-urlencode "db=mydb" --
data-urlencode "q=SELECT value FROM cpu_load_short WHERE region='us-west'"
```

```
curl -G 'http://localhost:8086/query?pretty=true' --data-urlencode "db=mydb" --
data-urlencode "q=SELECT value FROM cpu_load_short WHERE region='us-
west';SELECT count(value) FROM cpu_load_short WHERE region='us-west'"
```

create continuous queries

```
CREATE CONTINUOUS QUERY cq_30m ON food_data
BEGIN
SELECT mean(website) AS mean_website,mean(phone) AS mean_phone INTO
food_data."default".downsampled_orders FROM orders GROUP BY time(30m)
END
```

query language

SELECT * FROM h2o_feet WHERE time > now() - 7d

SELECT * FROM h2o_feet WHERE location = 'coyote_creek' AND water_level > 8

SELECT MEAN(water_level) FROM h2o_feet GROUP BY location

SELECT MEAN(water_level) FROM h2o_feet GROUP BY time(10m), location

SELECT water_level INTO h2o_feet_copy FROM h2o_feet WHERE location = 'coyote_creek'

SELECT water_level INTO h2o_feet_copy FROM h2o_feet WHERE location = ‘coyote_creek' limit 10

……

about update

- Do not update old data frequently!
- If so, use redis or MongoDB etc

Agenda

- basic concept
- how to use
- ***storage engine***
- cluster design
- what we did
- conclusion

- LSM Tree ===> mmap B+ Tree ===> TSM Tree

LevelDB BoltDB tsm

B+ Tree

- appends in the keyspace are efficient
- appends happening in individual time series
- end up looking more like random inserts than append only inserts.

LSM Tree

- used by Cassandra, HBase, LevelDB etc ...
- Log(WAL) & Mem Tables & SSTables

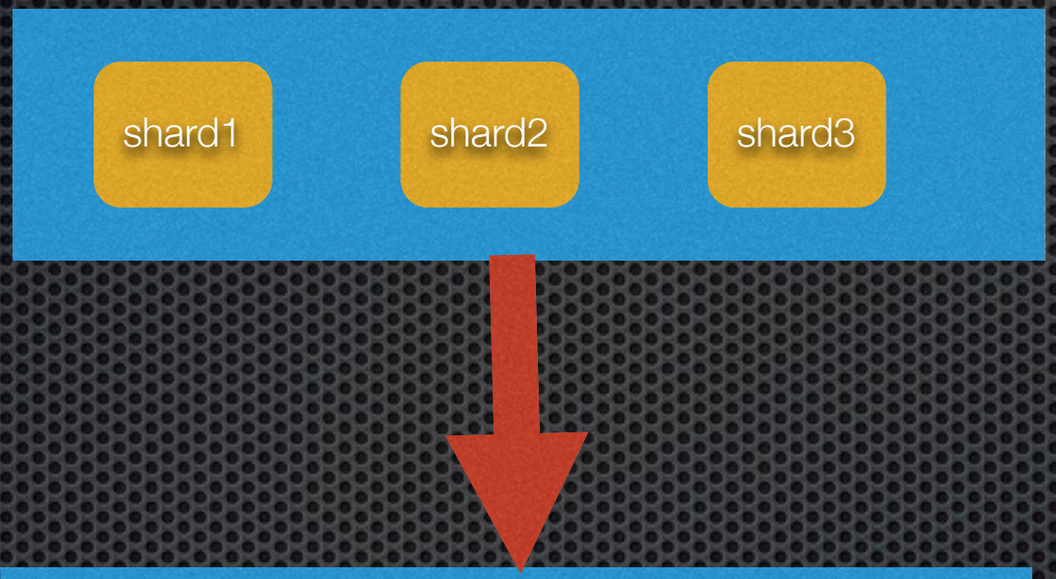
LevelDB (based on LSM)

- high write throughput √
- built in compression √
- do not support hot backups × （RocksDB & HyperLevelDB）
- fail to handle retention × (Delete is expensive in LSM)
- too many open files × (Due to shard design & LevelDB itself)

Shard & Shard Group

start1

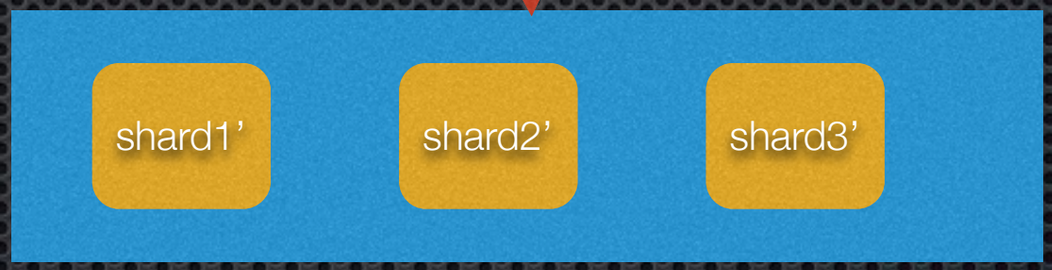
end1



shard group1

start2

end2



shard group2

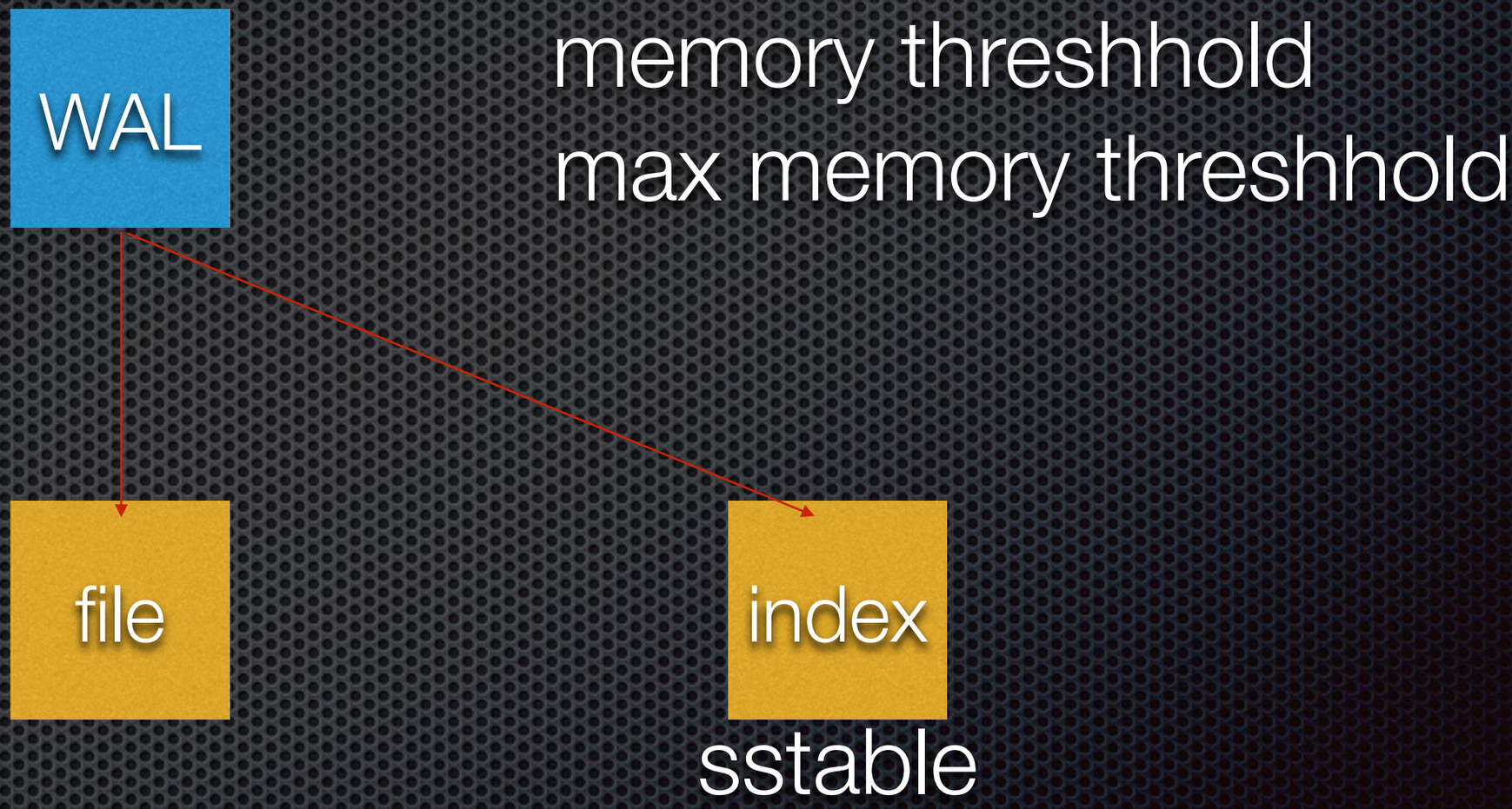
mmap B+ Tree & BoltDB

- BoltDB is a pure Golang database heavily inspired by *LMDB*
- *BoltDB used a single file as the database √*
- *High write & read throughput √*
- *IOPS spike × (introduce WAL)*

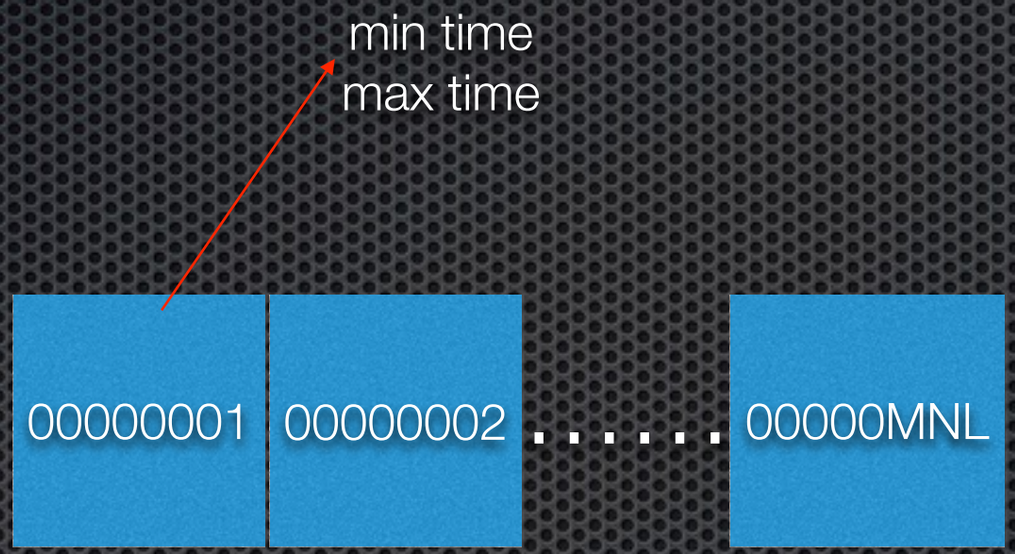
Current solution — TSM

- Just refine LSM!

Current solution — TSM



Current solution — TSM

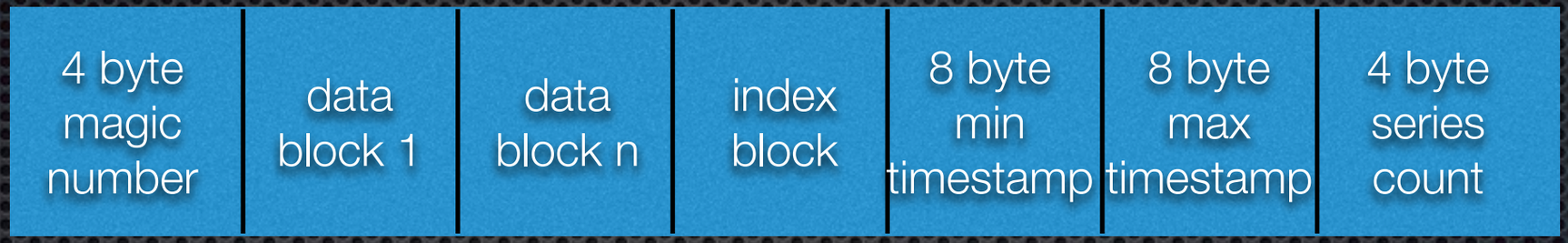


- data files
- metadata files (load data file when start)

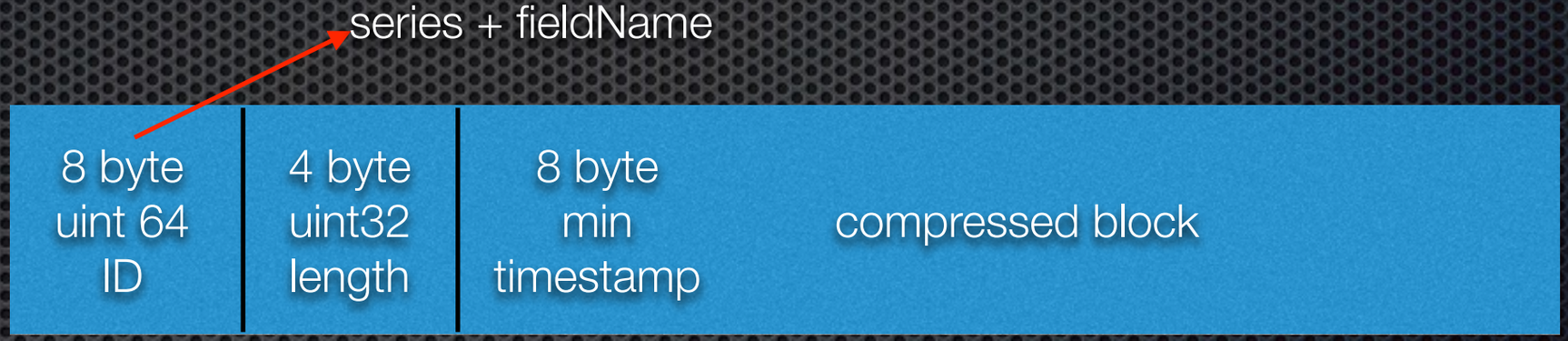
data files: non-overlapping & continuous time ranges

Dive into data files

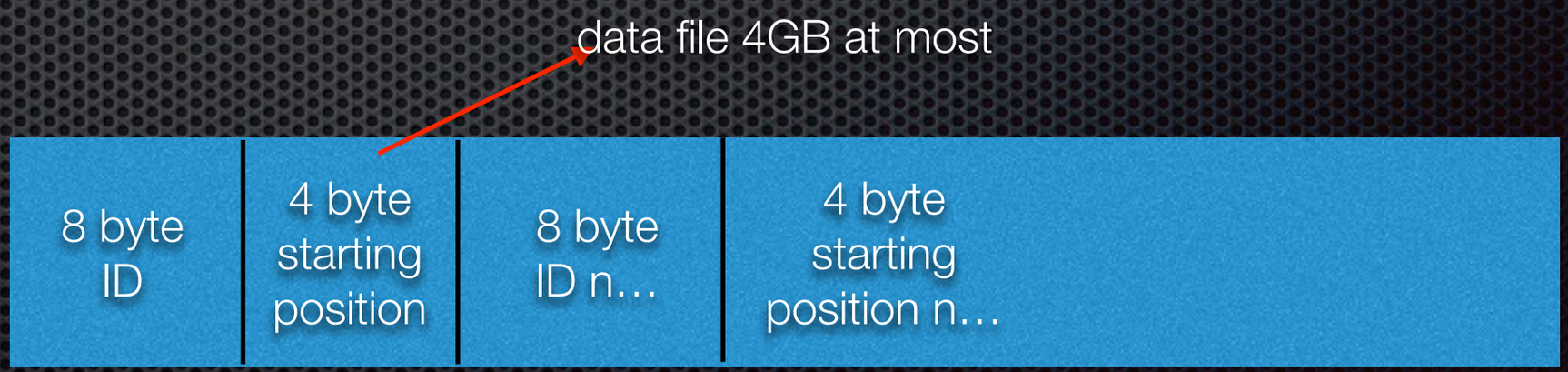
data files:



data block:

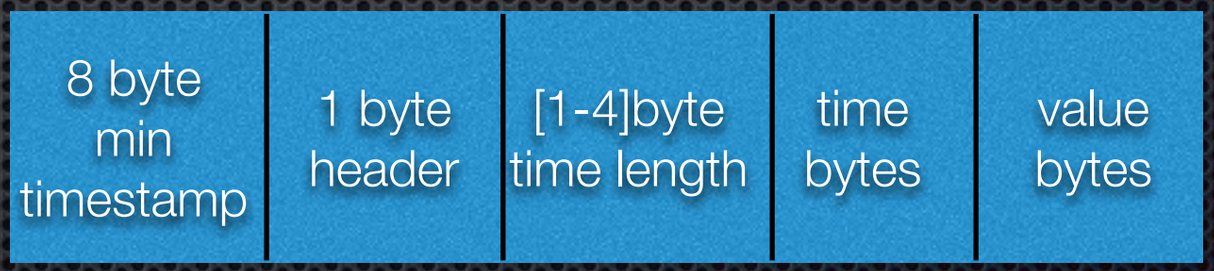


index block:



Current solution — TSM

- compaction
- update
- delete
- compression



Agenda

- basic concept
- how to use
- storage engine
- ***cluster design***
- what we did
- conclusion

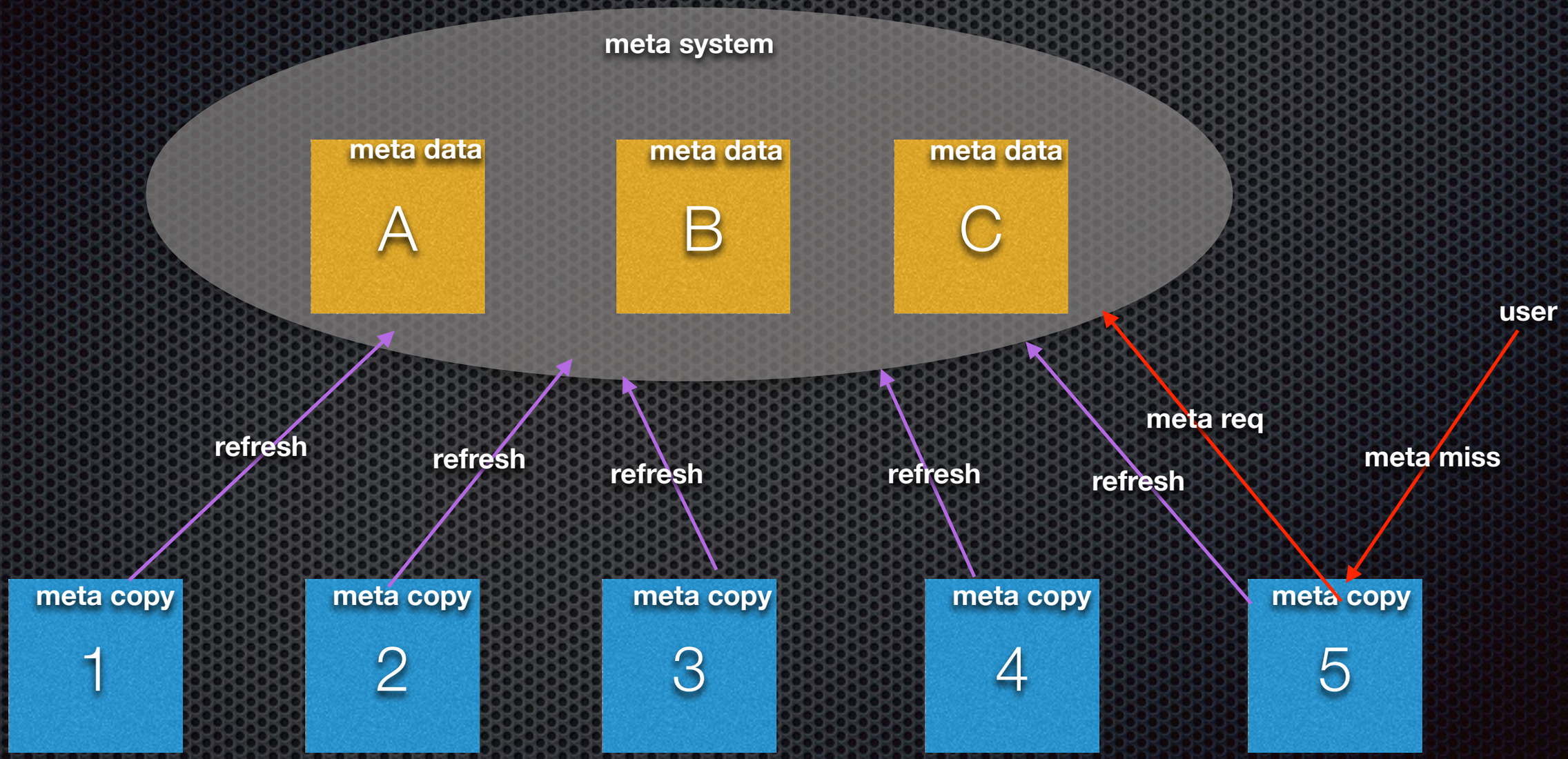
CAP?

- P is required
- CP & AP
- cluster metadata ——> CP
- writes ——> AP

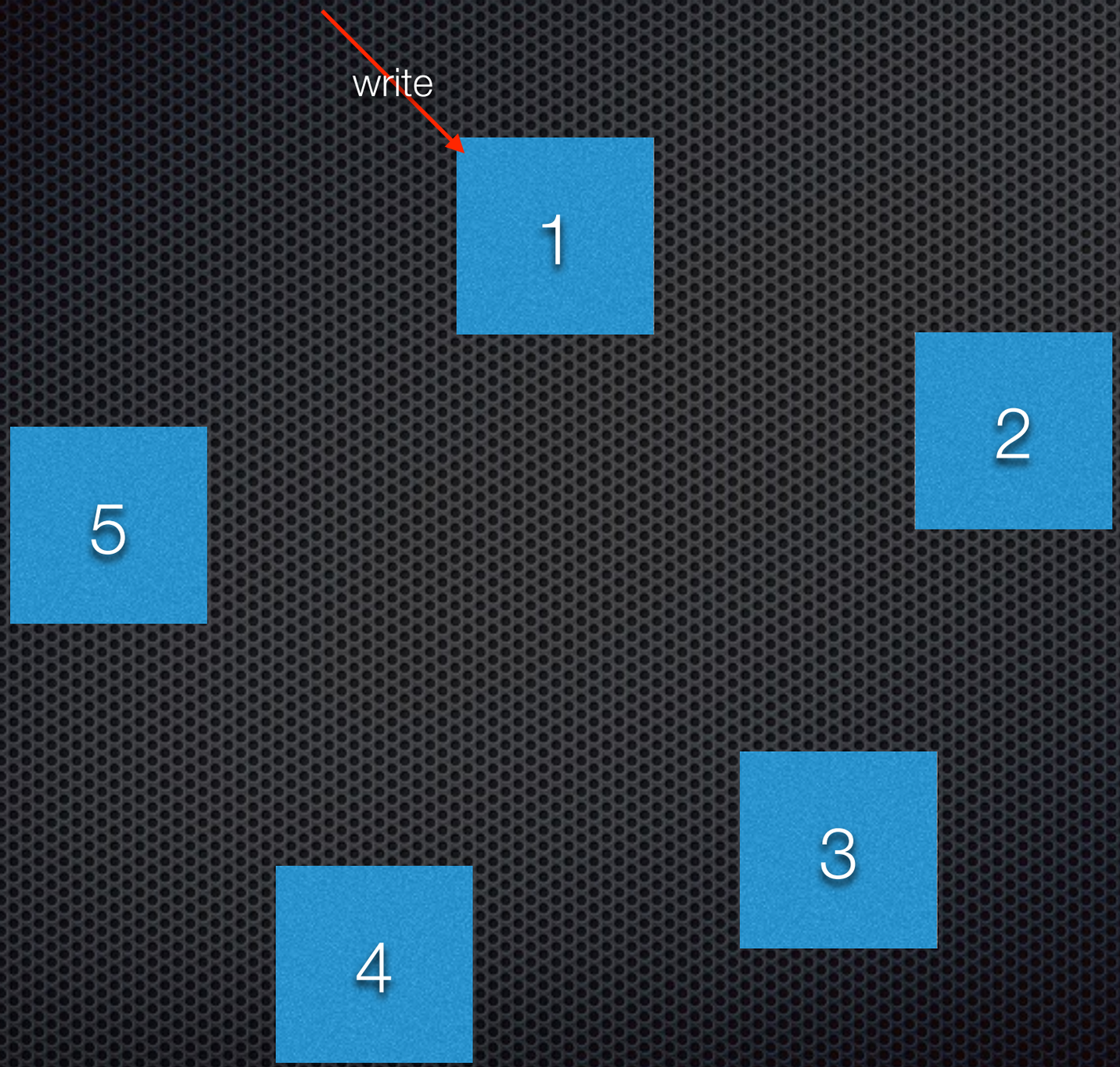
metadata system—AP

- raft cluster
- Servers in the cluster – unique id, hostname, if it's running the cluster metadata service
- Databases, Retention Policies, and continuous queries
- Users and permissions
- Shard Groups – start and end time, shards
- Shards – unique shard id

metadata system—AP



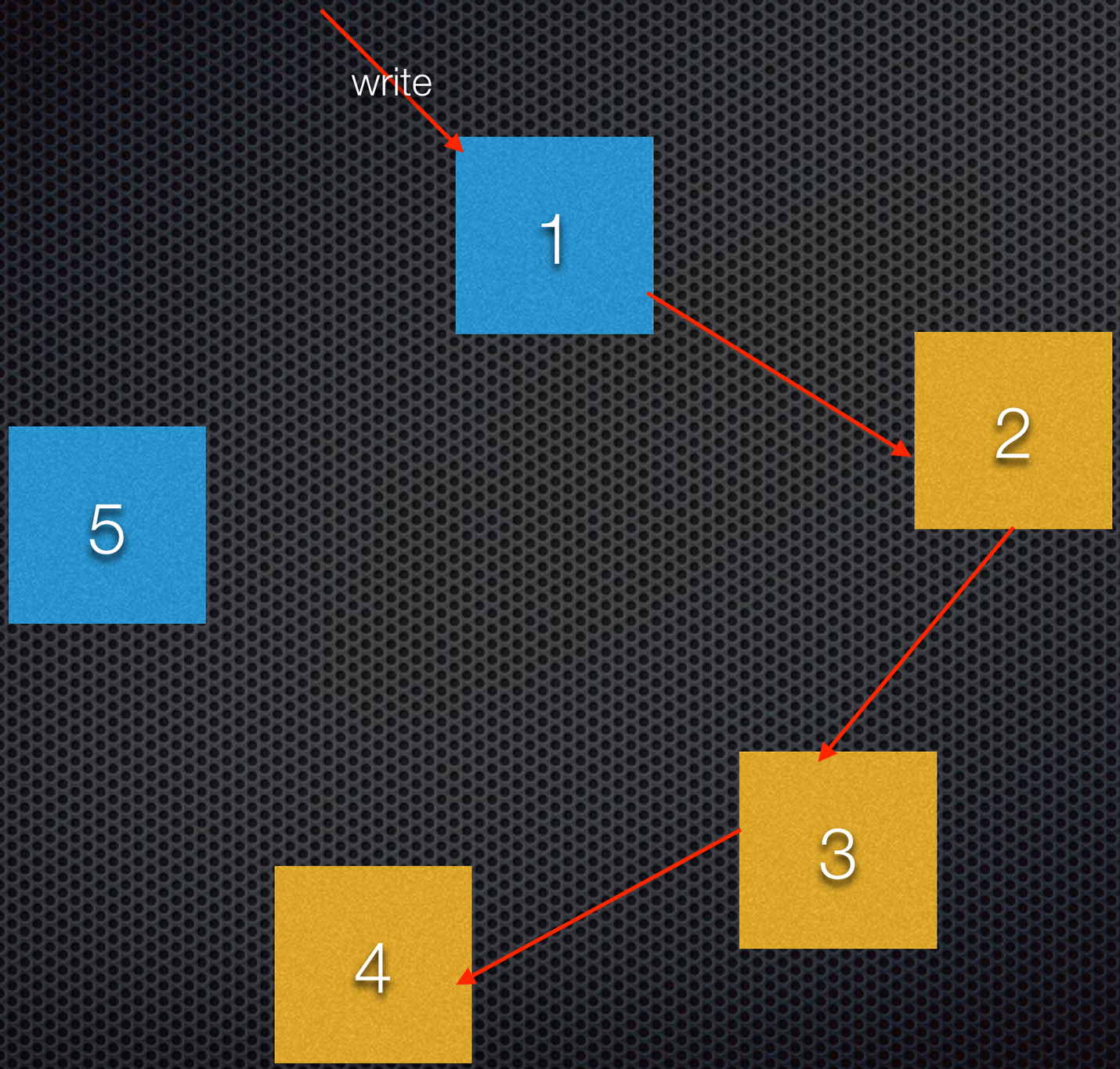
write — CP



consistency level

- Any
- One
- Quorum
- All

write — CP



Fault tolerance

- hinted handoff (ttl + full queue)
- anti-entropy repair (merkle tree)
- conflict resolution (greater value wins)

Agenda

- basic concept
- how to use
- storage engine
- cluster design
- ***what we did***
- conclusion

Our contributions to InfluxDB

- https://github.com/influxdata/influxdb/pull/4818
- https://github.com/influxdata/influxdb/pull/4815
- https://github.com/influxdata/influxdb/pull/4833
- https://github.com/influxdata/influxdb/pull/4940
- https://github.com/influxdata/influxdb/pull/4817
- https://github.com/influxdata/influxdb/pull/4984
- https://github.com/influxdata/influxdb/pull/5013
- .....

just show part of cluster related PRs

Our cluster solution

- design new solutions
- agent(heartbeat, influxd info, disk info etc....)
- scheduler, executor, adapter, ...
- support more useful functions
- introduce Kafka
- ....

Notice!

cluster will not be opensource anymore after 0.11!

Agenda

- basic concept
- how to use
- storage engine
- cluster design
- what we did
- ***conclusion***

Conclusion

- very easy to install & use …

Conclusion

- graceful restful API

Conclusion

- become more stable (so many bugs before, OMG!)

Conclusion

- If you just wanna standalone mode? Perfect!

Conclusion

- the best or nothing!

Conclusion

- InfluxDB + Spark

Thank you !