ClickHouse in EOI

余志昌

yuzhichang@gmail.com

zhichang.yu@eoitek.com

Make Data Think

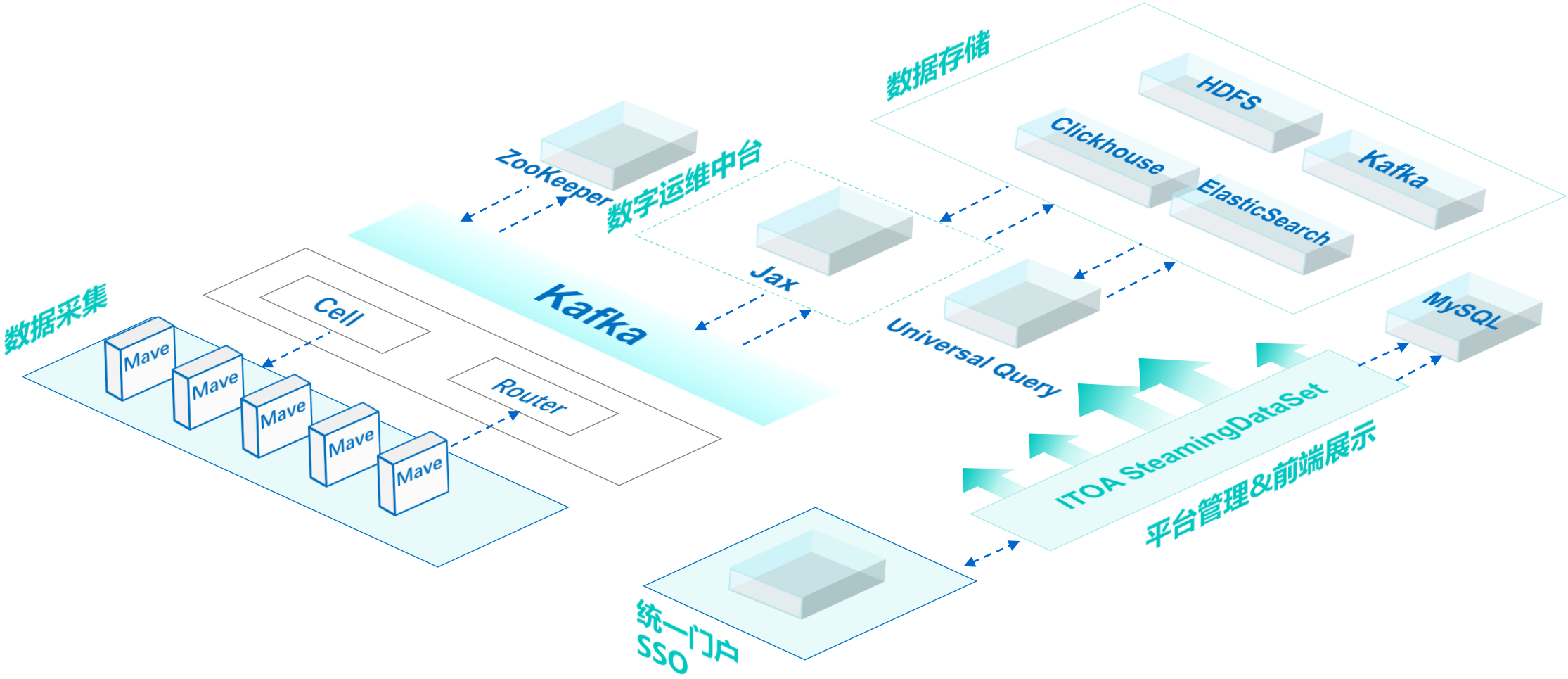
以AI激活运维数据智慧，助力客户数字化转型

Agenda

- EOI software architecture
- Data ingestion with clickhouse_sinker
- Cluster management with ckman
- ClickHouse on HDFS

逻辑架构

擎创科技


数据存储
HDFS
Clickhouse
Kafka
ElasticSearch
ZooKeeper
数字运维中台
Jax
Universal Query
MySQL
Kafka
Cell
Router
数据采集
Mave
Mave
Mave
Mave
Mave
ITOA SteamingDataSet
平台管理&前端展示
统一门户
SSO

功能层次

擎创科技

应用层

JAX Web

JAX RESTful API

作业编排

- Kafka Souce/Sink Job
- Transform Job
- ES Source/Sink Job
- Aggregation Job
- JDBC Source/Sink Job
- Regex match Job
- SQL Job
- Custom extended Job

作业管理/监控

- 指标监控
- 作业调试
- 定时调度
- 指标告警
- 状态检查
- 数据探索
- 作业启停
- 作业诊断
- 数据血缘

计算/处理框架

Apache Flink

- Data Stream
- Data Table
- Operator
- Stream SQL

Spark

- Data Frame
- Data Table
- Operator
- RDD

presto

clickhouse_sinker

资源管理层

HADOOP/YARN

Kubernetes

Standalone

数据库/持久化

HIVE

ClickHouse

VERTICA

OSS backed by EOI

- clickhouse_sinker(https://github.com/housepower/clickhouse_sinker)
  - yuzhichang(余志昌), sundy-li(李本旺)

- ckman(https://github.com/housepower/ckman)
  - yuzhichang(余志昌), YenchangChan(陈衍长)

Why not Kafka Engine built in ClickHouse?

擎创科技

- Kafka Engine is complicated, buggy and hard to debug.
- Kafka Engine runs inside the db process, lowers the database stability.
- Kafka Engine doesn't support custom sharding policy.
- Neither Kafka Engine nor clickhouse_sinker support exactly-once.

clickhouse_sinker features

- Uses native ClickHouse client-server TCP protocol. (written in Golang)
- Support multiple message format: json, csv.
- Support multiple Kafka security mechanisms: SSL, SASL/PLAIN, SASL/SCRAM, SASL/GSSAPI.
- Every message is routed to a determined clickhouse shard.
- At-least-once delivery guarantee.
- Handling ClickHouse replica single-point-failure, Kafka consumer group rebalace, Kafka partition changes etc.
- Config or detect fields mapping between message and table.
- Detect new fields in message and add columns to table accordingly.
- Support Prometheus style metrics.
- Support load balance among clickhouse_sinker instances.

clickhouse_sinker supported data types

ClickHouse data type	default value	compatible Json data type	valid range
Int8, Int16, ...	0	Bool, Number	Int8 [-128,127], ...
Float32, Float64	0.0	Number	Float32 [-MaxFloat32,MaxFloat32], ...
String, ...	""	Bool, Number, String, Object, Array	N/A
Date, DateTime, ...	EPOCH	Number, String	[EPOCH,MaxUint32_seconds_since_epoch)
Nullable(T)	NULL	(The same as T)	(The same as T)
Array(T)	[]	(The same as T)	(The same as T)

clickhouse_sinker benckmark

擎创科技

config	thoughput(rows/s)	writer total cost	clickhouse cost per node
1 kafka partition, 1 sinker	142 K	11.0 cpu, 8 GB	0.3 cpu
2 kafka partition, 1 sinker	159 K	14.0 cpu, 14 GB	0.7 cpu
4 kafka partition, 1 sinker	25~127 K	2~22 cpu, 16 GB	1 cpu
2 kafka partition, 2 sinker	275 K	22 cpu, 8 GB	1.3 cpu
4 kafka partition, 2 sinker	301 K	25 cpu, 18 GB	1.5 cpu

flink pipeline benckmark

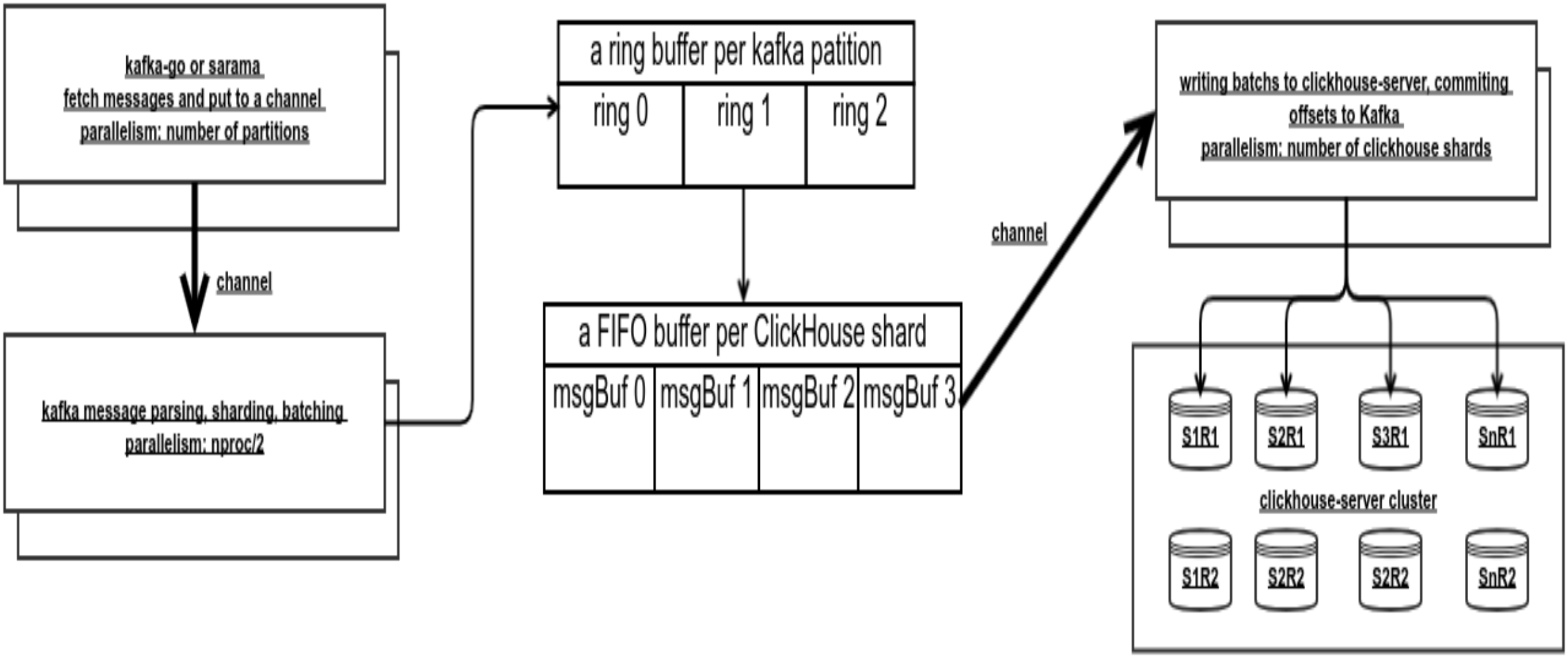
Kafka Source -> JSON decode -> DateTime formart conversion -> Interger type conversion -> JDBCSinkJob

config	thoughput(rows/s)	writer total cost	clickhouse cost per node
1 kafka partition, pipeline Parallelism: 20	44.7 K	13.8 cpu, 20 GB	1.1 cpu

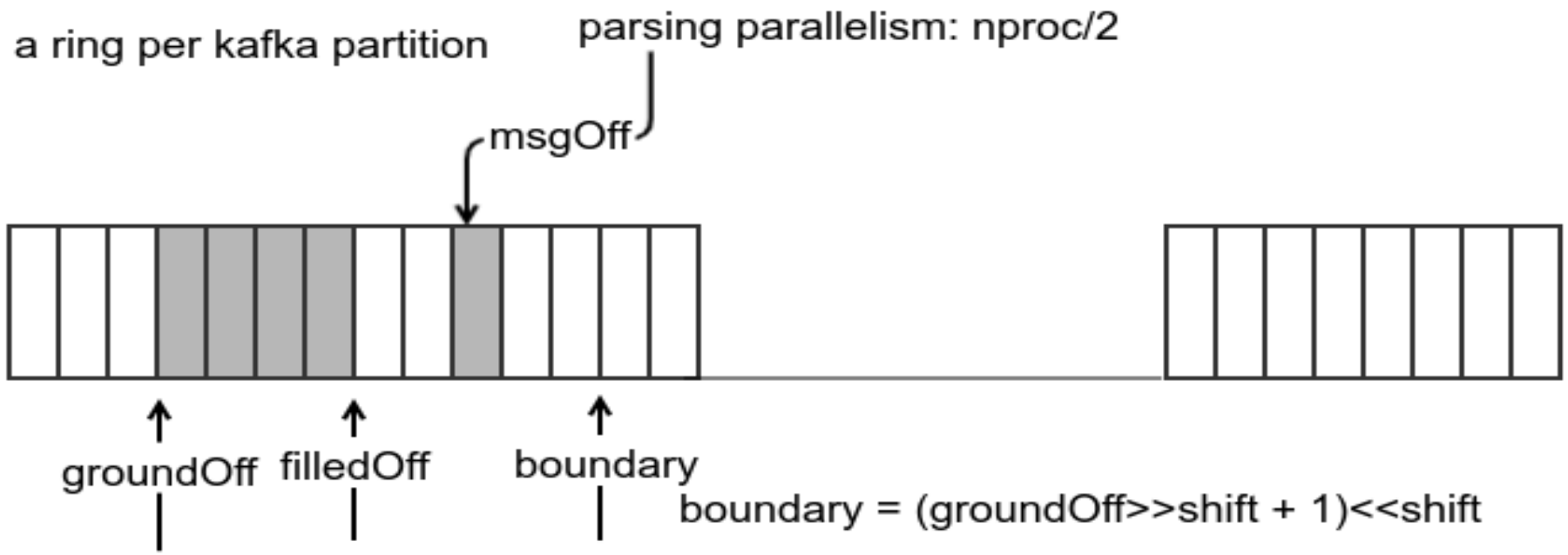
Conclusion

- clickhouse_sinker is 3x fast as the Flink pipeline, and cost much less connection and cpu overhead on clickhouse-server.

clickhouse_sinker architecture



clickhouse_sinker parsing



each put moves filledOff as far as possible
genBatchOrShard if filledOff reach boundary or flush timer fire
ensure msg order inside a batch in order to let clickhouse-server sorting happy

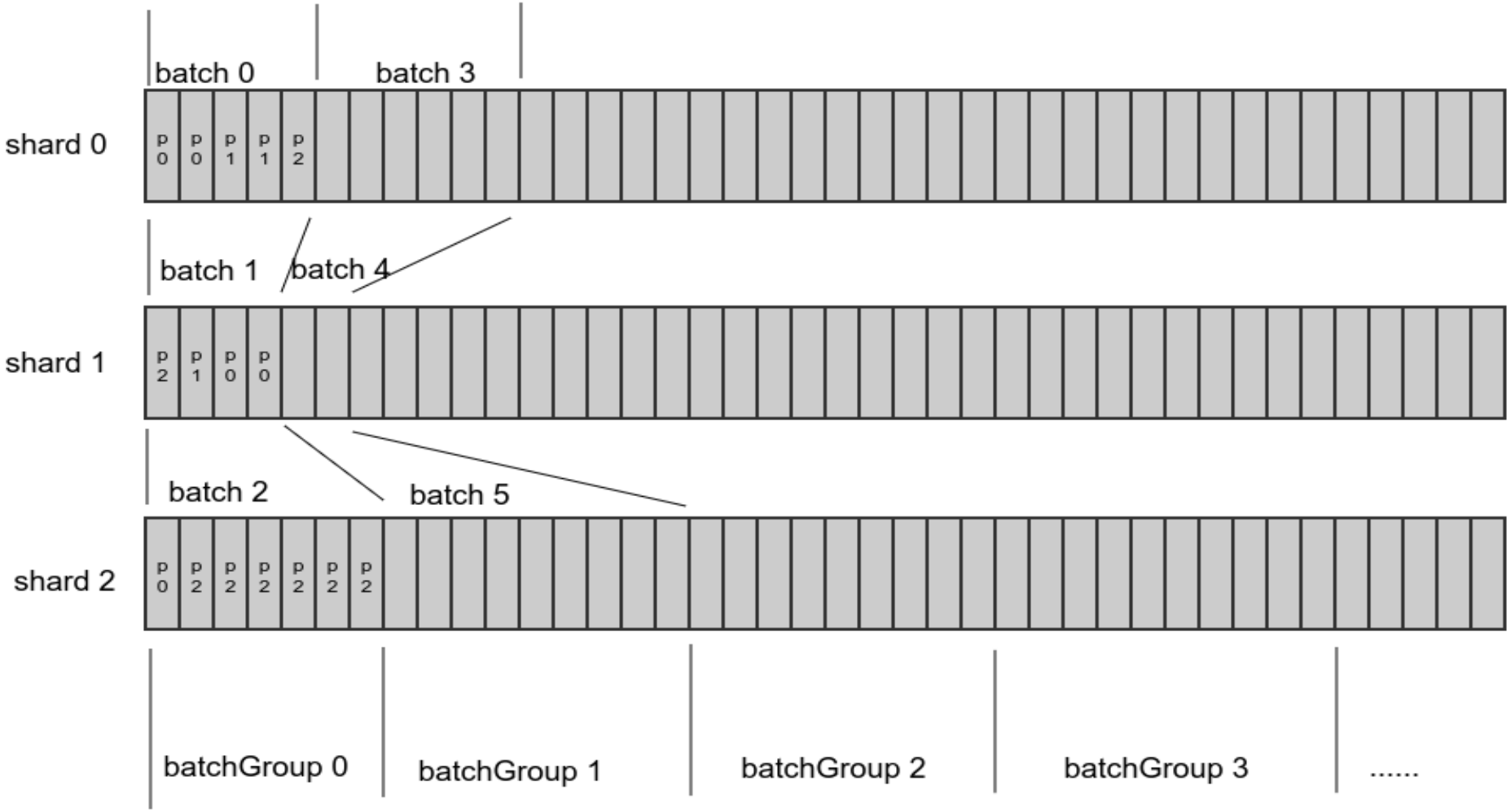
clickhouse_sinker sharding

Every message is routed to a determined ClickHouse shard.

- default: (kafka_offset/roundup(buffer_size))%clickhouse_shards
- stripe: (uint64(shardingKey)/stripe_size)%clickhouse_shards
- hash: xxhash64(string(shardingKey))%clickhouse_shards

- roundup() round upward an unsigned integer to the the nearest 2^n.
- shardingKey value is a column name.

clickhouse_sinker batch group



BatchGroup consists of multiple batches.
The `before` relationship could be impossible if messages of a partition are distributed to multiple batches.
So those batches need to be committed after ALL of them have been written to clickhouse.

Prometheus metrics


```
{
    "__name__":"node_cpu_core_throttles_total",
    "timestamp":"2021-10-27T14:54:32.288+08:00",
    "value":0,
    "core":"5",
    "instance":"192.168.102.116:9100",
    "job":"testscrape",
    "package":"1"
}
```


- A datapoint consisit of:

  - metric name and a list of labels

  - timestamp and value

⑩Different metrics have different labels!

Prometheus metrics, sulotion 1

```
CREATE TABLE prom_extended (
    timestamp DateTime,
    value Float64,
    __name__ String,
    job String,
    instance String
) ENGINE = ReplacingMergeTree
PARTITION BY toYYYYMMDD(timestamp)
ORDER BY (timestamp, __series_id);


SELECT toStartOfInterval(timestamp, INTERVAL 5 minute) AS ts,
    avg(value) FROM prom_extended
    WHERE timestamp ⩾ addDays(now(), -1) AND
    __name__='XXX' and instance='XXX' AND ip ='XXX'
GROUP BY ts;
```

- wide-table, 1-1 mapping of label and column
- Add columns as needed via clickhouse_sinker

- Benchmark:
  - Ingestion thoughput decrease to ~1/5(~60 labels in total)
  - Aggregation of last 24h costs 3.77s.

Prometheus metrics, sulotion 2

```
CREATE TABLE prom_metric (
    timestamp DateTime64(3),
    value Float64,
    __series_id UInt64
)  ENGINE = ReplacingMergeTree
PARTITION BY toYYYYMMDD(timestamp)
ORDER BY (__series_id, timestamp);

CREATE TABLE prom_metric_series (
    __series_id UInt64,
    labels String,
    __name__ String,
    job String,
    instance String
) ENGINE = ReplacingMergeTree
ORDER BY (__series_id);
```

- Tow tables, one for datapoints, one for series.
- Add columns to series table as needed via clickhouse_sinker
- Calculate series ID via clickhouse_sinker

- Benchmark:
  - Ingestion thoughput no decreasion(~60 labels in total)
  - Aggregation of last 24h costs 0.10s.

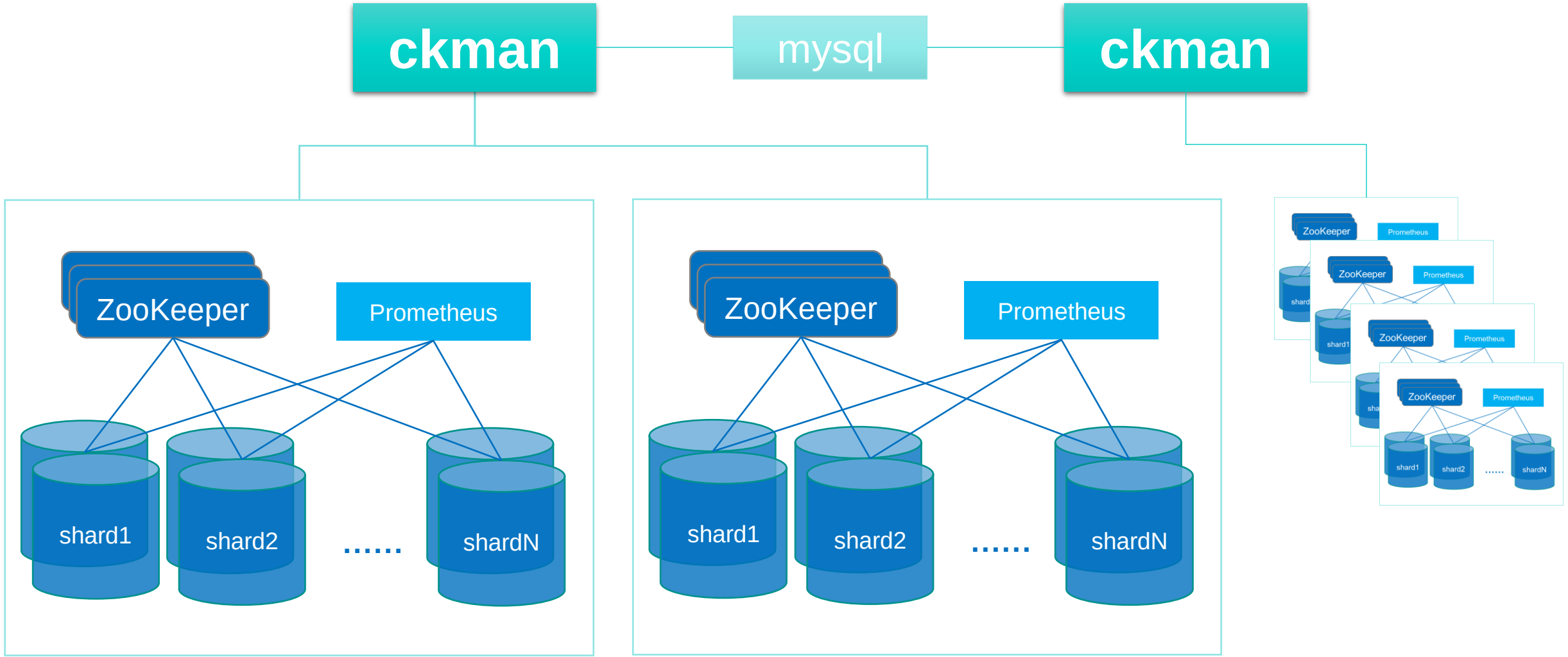
ClickHouse vs OpenTSDB

- OpenTSDB characters:
  - Several Hive tables on HDFS.
  - Allocate an 24bit UID for each metric name, lable key, label value
  - Store a hour of datapoints of a series to one Hive row.
  - No pre-aggregation.
- Benchmark
  - Aggregation of last 24h costs 0.20s.
  - Much slower for larger dataset.
- ClickHouse beats OpenTSDB!

ckman features

- Web console for ops:
  - Deploy, upgrade, destory, start, stop cluster
  - Scale in/out cluster
  - Rebalance, archive, purge data
- Monitor status of ClickHouse node, table, ZooKeeper
- API for cluster and table management
- Simple query console

ckman architecture



ClickHouse on HDFS

- 21.10 is good enough to try ClickHouse on HDFS:
  - PR#25918 HDFS zero-copy replication
  - PR#28268 HDFS NameNode HA
- 21.11 introduced:
  - PR#29205 try async read for remote fs disks
- DiskHDFS Benchmark (HDFS cluster: 3 physical hosts)
  - MOVE PARTITION - from local to HDFS, 550~650MB/s
  - MOVE PARTITION - from HDFS to local, 320MB/s
  - INSERT INTO local - 450K rows/s
  - INSERT INTO hdfs - 200K rows/s

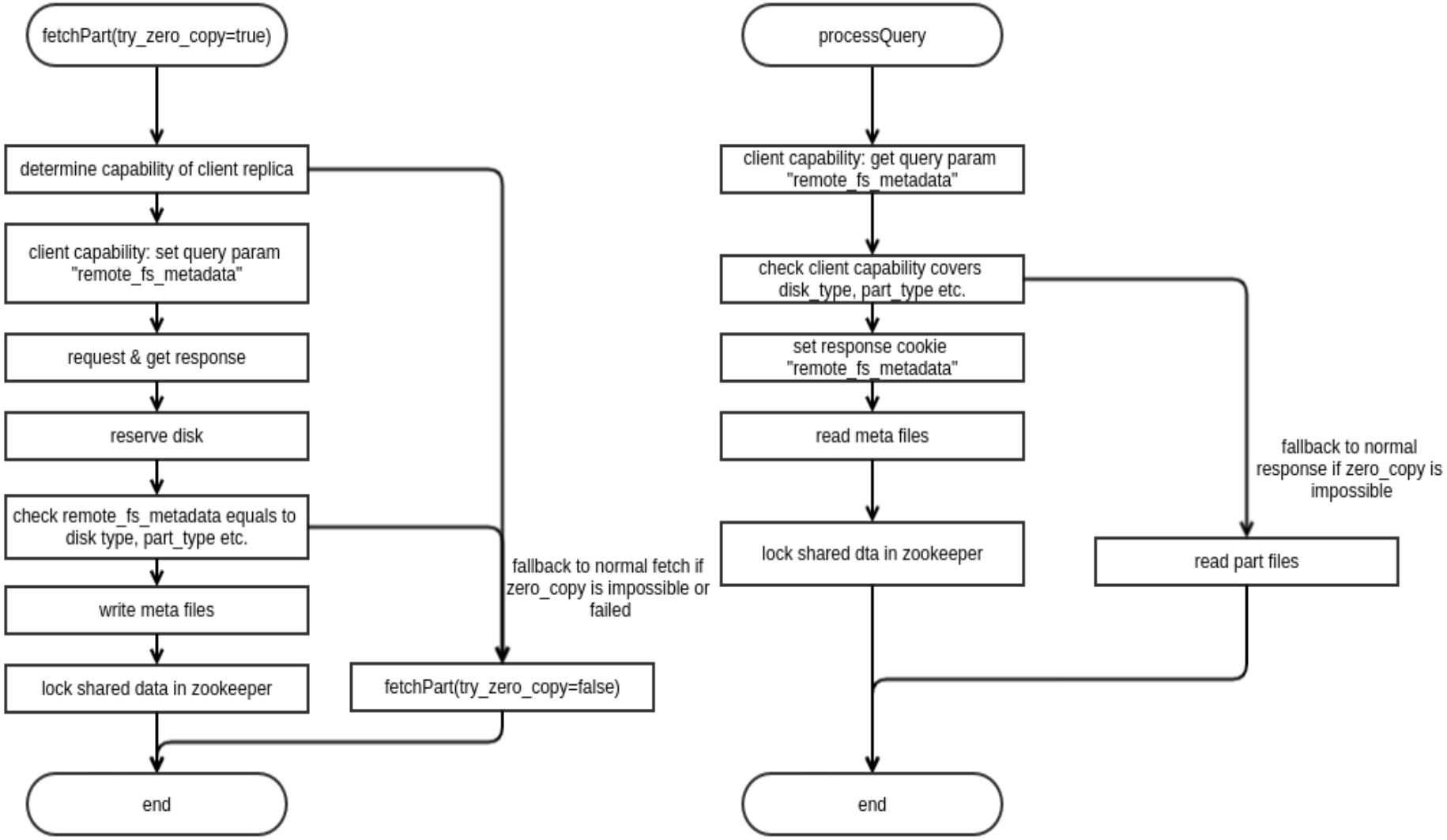
HDFS zero-copy replication

- All clickhouse nodes share the same HDFS disk:

```
<storage_configuration>
    <disks>
        <hdfs1>
            <type>hdfs</type>
            <endpoint>hdfs://hdfs1:9000/clickhouse1/</endpoint>
        </hdfs1>
    </disks>
    <policies>
        <hybrid>
            <volumes>
                <main>
                    <disk>default</disk>
                </main>
                <external>
                    <disk>hdfs1</disk>
                </external>
            </volumes>
        </hybrid>
    </policies>
</storage_configuration>

<merge_tree>
    <allow_remote_fs_zero_copy_replication>1</allow_remote_fs_zero_copy_replication>
</merge_tree>
```

HDFS&S3 zero-copy replication



ClickHouse on HDFS TODO

- PR#22012 introduced table function s3Cluster.
  - Impl hdfsCluster to read Parquet files parallelly to beat HBase?
- PR#25615 DiskS3 seek to reduce data read
  - Port to DiskHDFS?

擎创科技

MAKE DATA THINK

以AI激活运维数据智慧　助力客户数字化转型

www.eoitek.com
info@eoitek.com
4008 215 724