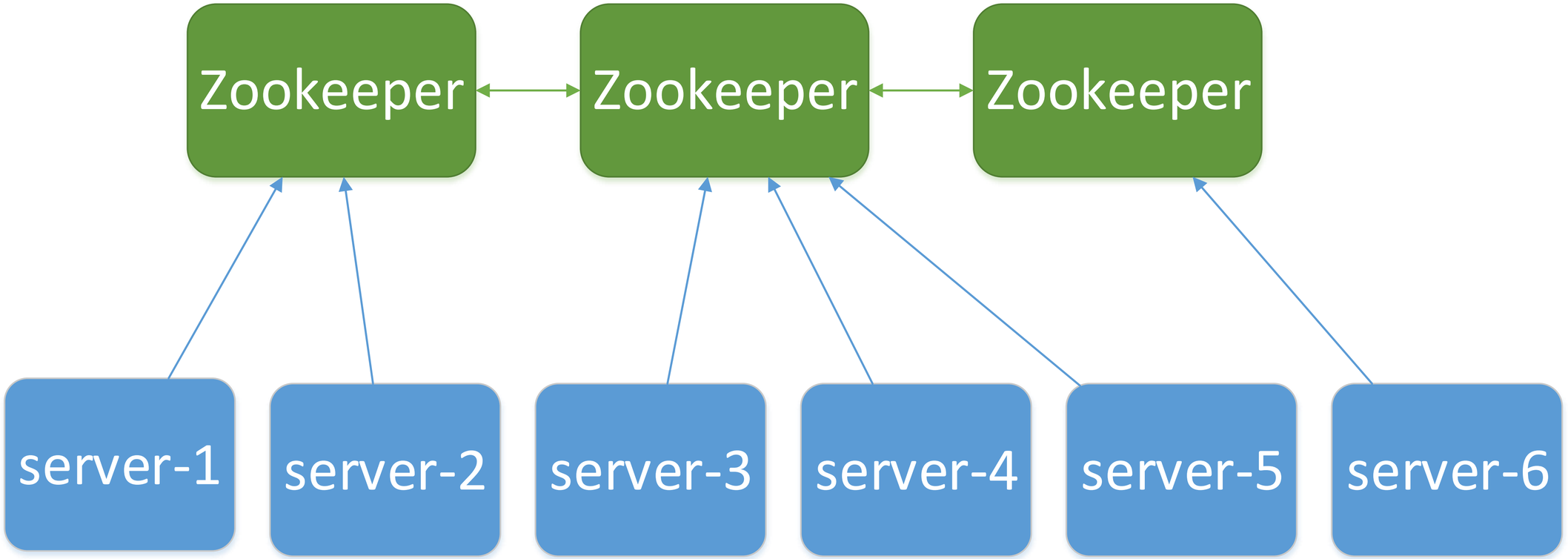
Как не наступить на распределенные грабли или просто хорошие практики по работе с Kafka, Zookeeper и HBase

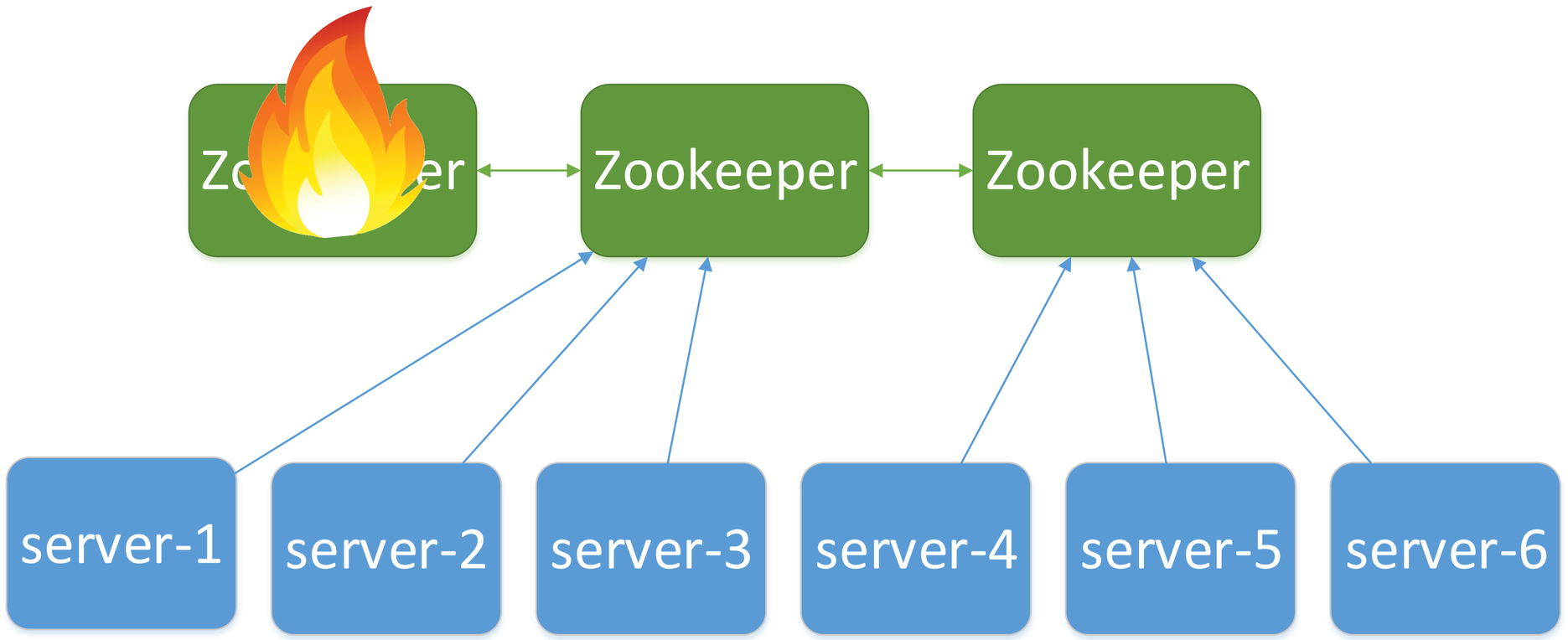
Работа с очередями или persistent хранилищами становится не такой очевидной, когда дело касается распределенных систем. Вопросы consistency и data availability требуют особого внимания, когда речь идет об обработке критичных данных. Мы расскажем о подводных камнях, которые нам попались, а также поделимся интересными практиками решения типовых задач.


ZOO

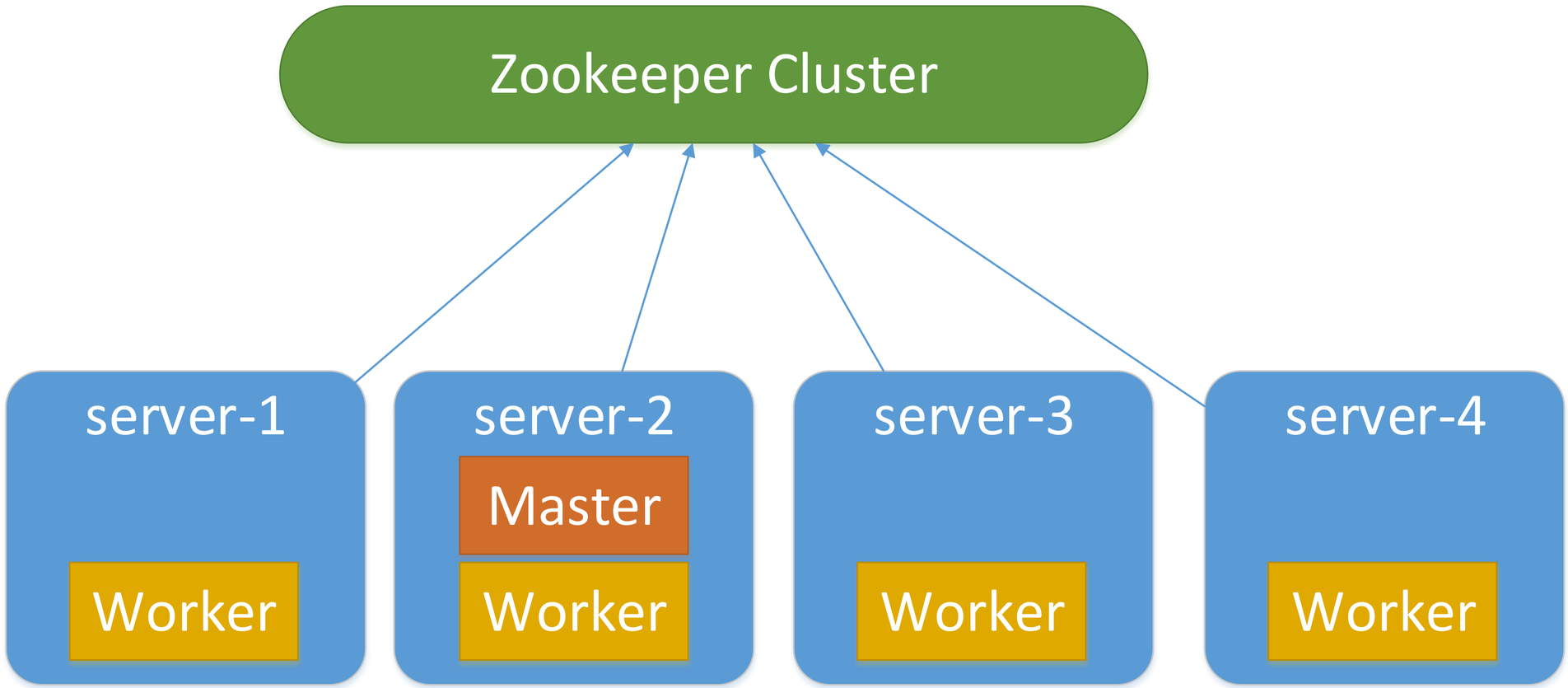

Zookeeper
Zookeeper
Zookeeper
server-1
server-2
server-3
server-4
server-5
server-6

Consul


ZOO


Zookeeper
Zookeeper
server-1
server-2
server-3
server-4
server-5
server-6

distributed-job-manager




```
job-manager
    └ locks
        └ work-pooled
            └ async.report.building.job
            └ elasticsearch.upload.job
            └ fraud.user.event.export.job
            └ notification.processing.job
            └ statistics.log.kafka.reader.job
            ...
    └ assignment-version
    └ workers
        └ 20
            ...
        └ 3
            └ available
                └ work-pooled
                    └ async.report.building.job
                    └ cdr.upload.service
                    └ deferred.sms.notification.job
                    └ elasticsearch.upload.job
                    └ email.sender.job
                    ...
            └ assigned
                └ work-pooled
                    └ async.report.building.job
                    └ elasticsearch.upload.job
```


```
job-manager
    └ locks
        └ work-pooled
            └ async.report.building.job
            └ elasticsearch.upload.job
            └ fraud.user.event.export.job
            └ notification.processing.job
            └ statistics.log.kafka.reader.job
            ...
    └ assignment-version
    └ workers
        └ 20
            ...
        └ 3
            └ available
                └ work-pooled
                    └ async.report.building.job
                    └ cdr.upload.service
                    └ deferred.sms.notification.job
                    └ elasticsearch.upload.job
                    └ email.sender.job
                    ...
            └ assigned
                └ work-pooled
                    └ async.report.building.job
                    └ elasticsearch.upload.job
```


```
job-manager
    └ locks
        └ work-pooled
            └ async.report.building.job
            └ elasticsearch.upload.job
            └ fraud.user.event.export.job
            └ notification.processing.job
            └ statistics.log.kafka.reader.job
            ...
    └ assignment-version
    └ workers
        └ 20
            ...
        └ 3
            └ available
                └ work-pooled
                    └ async.report.building.job
                    └ cdr.upload.service
                    └ deferred.sms.notification.job
                    └ elasticsearch.upload.job
                    └ email.sender.job
                    ...
            └ assigned
                └ work-pooled
                    └ async.report.building.job
                    └ elasticsearch.upload.job
```


```
job-manager
    └ locks
        └ work-pooled
            └ async.report.building.job
            └ elasticsearch.upload.job
            └ fraud.user.event.export.job
            └ notification.processing.job
            └ statistics.log.kafka.reader.job
            ...
    └ assignment-version
    └ workers
        └ 20
            ...
        └ 3
            └ available
                └ work-pooled
                    └ async.report.building.job
                    └ cdr.upload.service
                    └ deferred.sms.notification.job
                    └ elasticsearch.upload.job
                    └ email.sender.job
                    ...
            └ assigned
                └ work-pooled
                    └ async.report.building.job
                    └ elasticsearch.upload.job
```

distributed-job-manager

Zookeeper Cluster

server-1

Worker

server-2

Master

Worker

server-3

Worker

server-4

Worker

jepsen

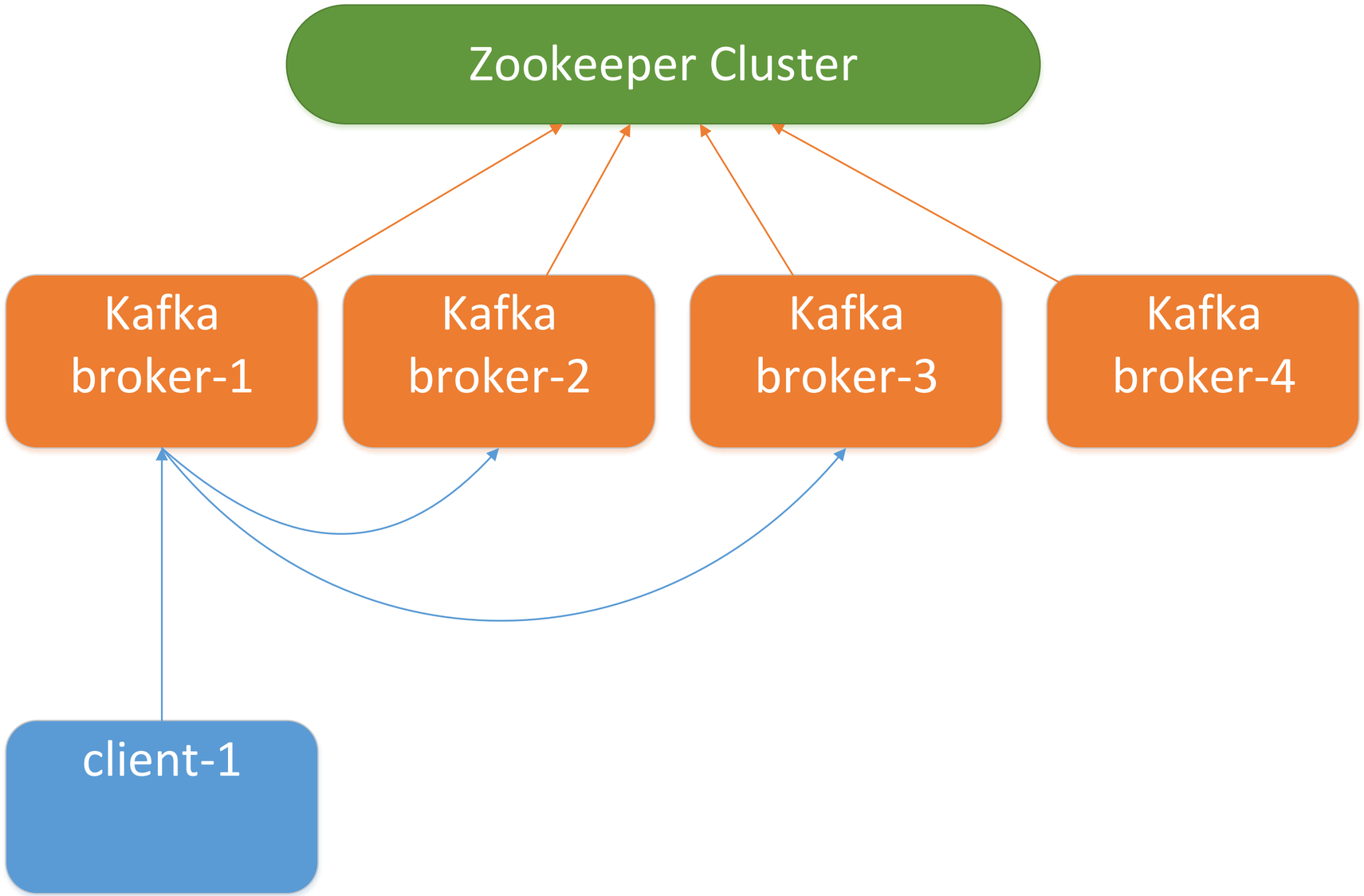

```
class PersistentExpiringDistributedLock{
    public boolean expirableAcquire(long acquirePeriod, long timeout) throws Exception {...}
    public boolean checkAndProlong(long prolongationPeriod) throws Exception {...}
}
```

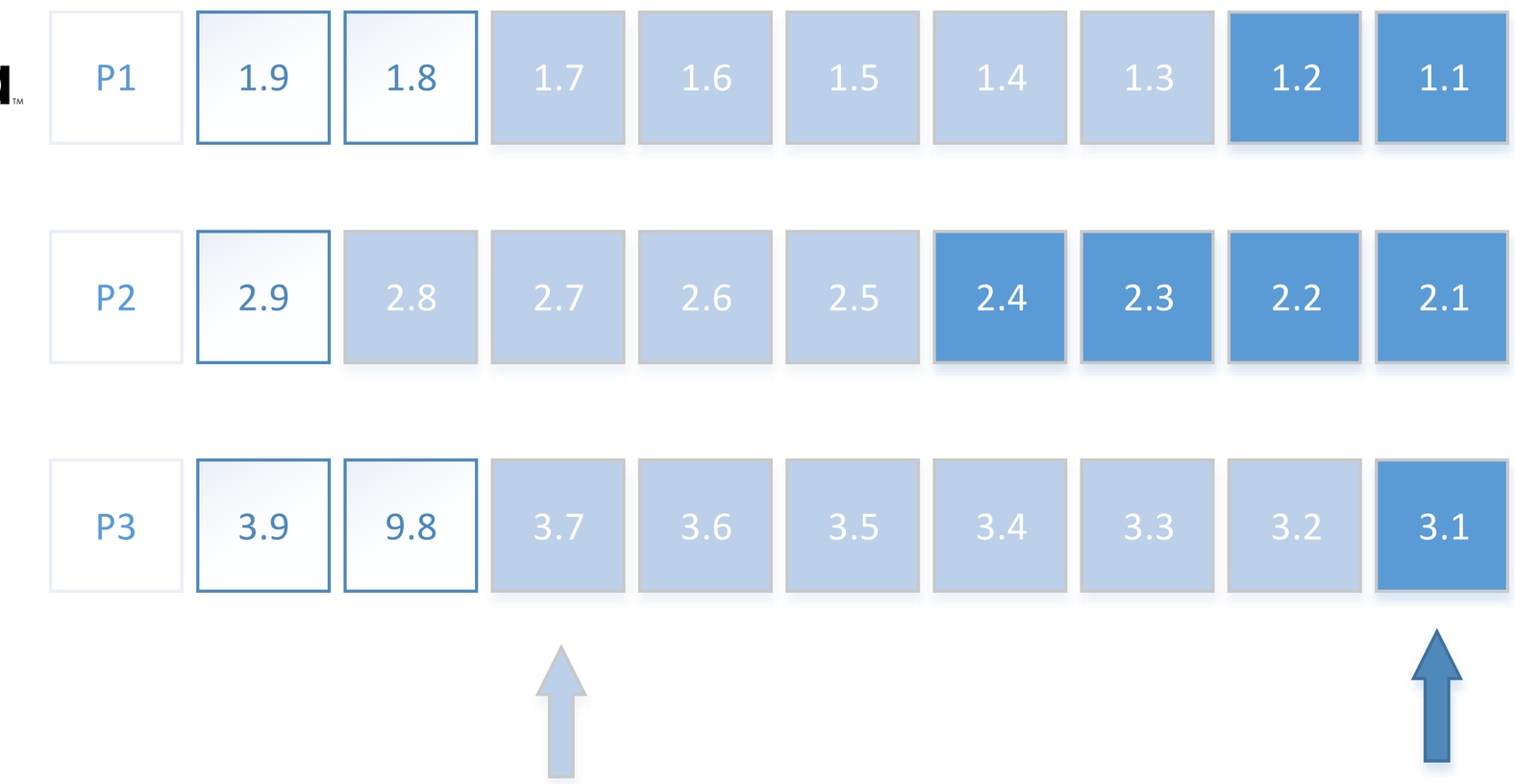
```
PersistentExpiringDistributedLock lock = getDistributedLock();

if (lock.expirableAcquire(MINUTES.toMillis(30), SECONDS.toMillis(5))) {
    while (haveWorkToDo()) {
        // do 10 minutes work
        doWork();
        // prolong persistent lock
        if (!lock.checkAndProlong(MINUTES.toMillis(30))) {
            //something gone wrong
            //gracefully stop task
            return;
        }
    }
}
```

~~Ephemeral~~


```
job-manager
    └ locks
        └ ...
    └ assignment-version
    └ workers
        └ 20
            ...
        └ 3
            └ available
                └ work-pooled
                    └ async.report.building.job
                    └ cdr.upload.service
                    └ deferred.sms.notification.job
                    └ elasticsearch.upload.job
                    └ email.sender.job
                    ...
            └ assigned
                └ work-pooled
                    └ async.report.building.job
                    └ elasticsearch.upload.job
```

```
public class ZooKeeper {
    ...
    public List<OpResult> multi(Iterable<Op> ops) throws Exception{...}
    ...
}
```


Zookeeper Cluster
Kafka broker-1
Kafka broker-2
Kafka broker-3
Kafka broker-4
client-1



2.8

2 – partition number

8 – item number


APACHE
kafka
APACHE
HBASE

P1	1.9 25	1.8 22	1.7 19	1.6 16	1.5 13	1.4 10	1.3 7	1.2 4	1.1 1
P2	2.9 26	2.8 23	2.7 20	2.6 17	2.5 14	2.4 11	2.3 8	2.2 5	2.1 2
P3	3.9 27	9.8 24	3.7 21	3.6 18	3.5 15	3.4 12	3.3 9	3.2 6	3.1 3

1.1 1	1.2 4	1.3 7	1.4 10	1.5 13	1.6 16	1.7 19	2.1 2	2.2 5	2.3 8


FIX
APACHE
kafka


P1
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1


1.7
1.6
1.5
1.4
1.3


P1
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1

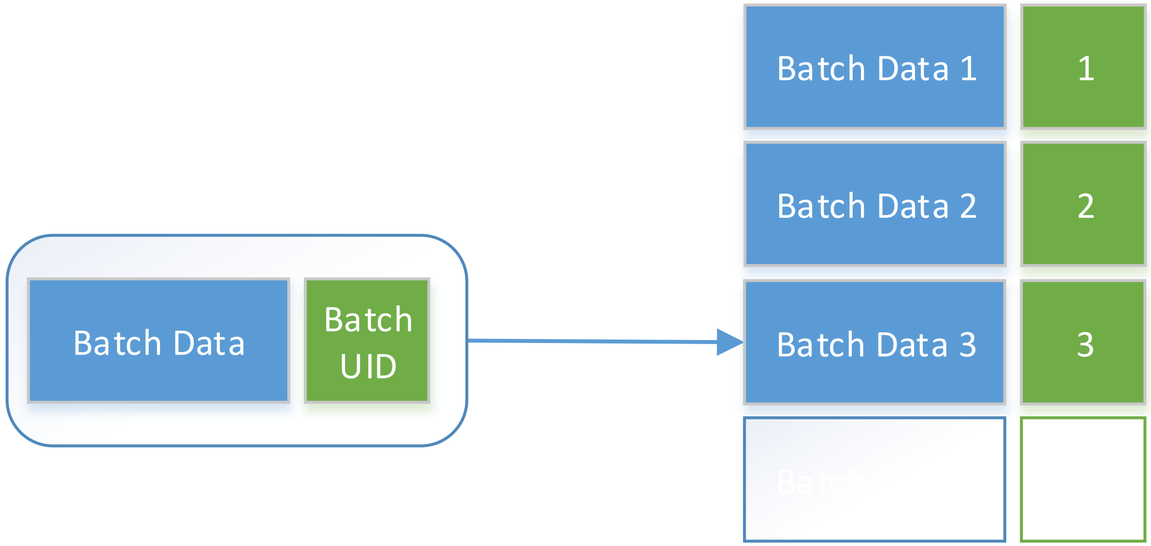

1.7
1.6
1.5
1.4
1.3

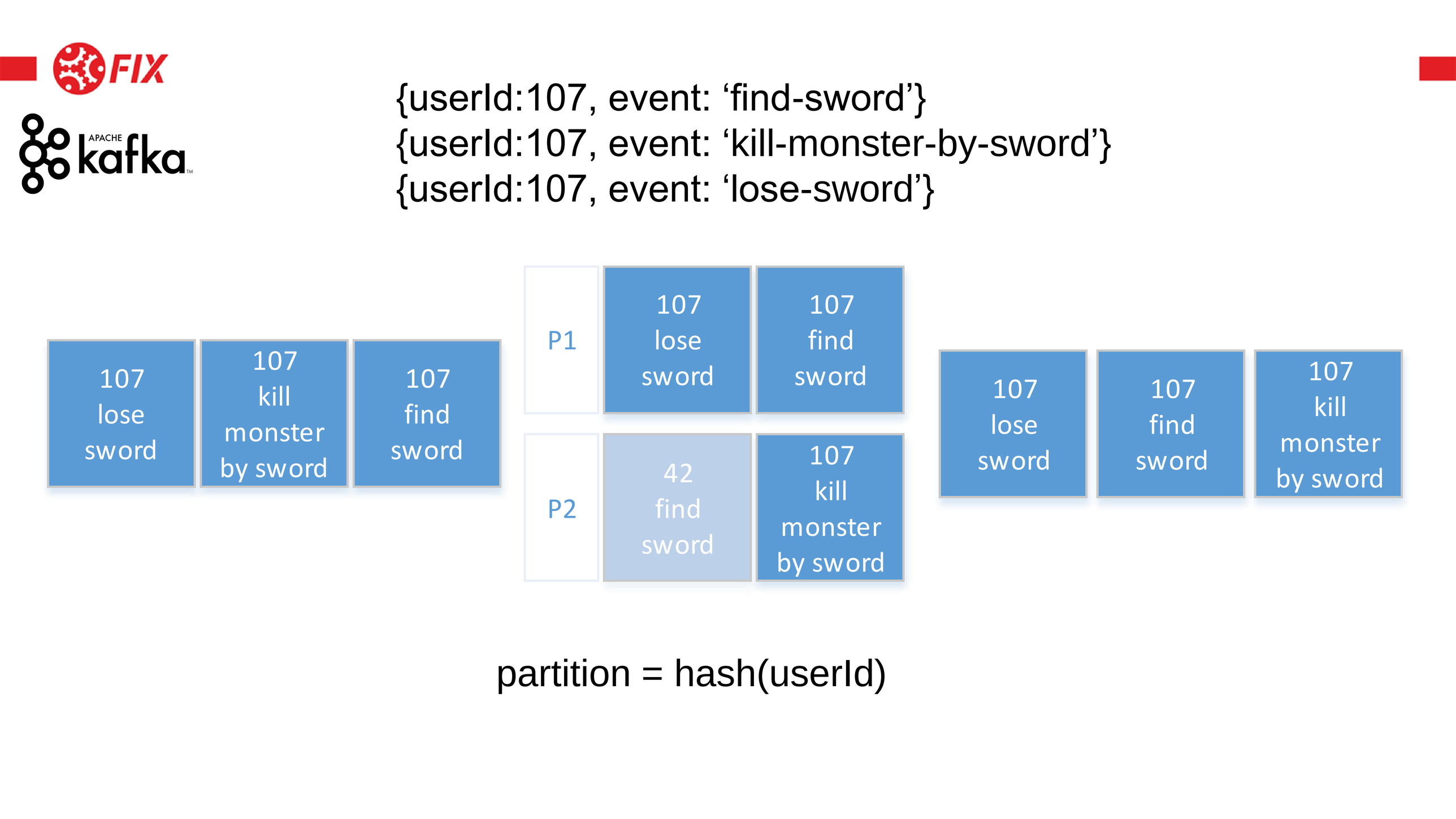

?
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1


A
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1


B
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1

Idempotency




{userId:107, event: ‘find-sword’}
{userId:107, event: ‘kill-monster-by-sword’}
{userId:107, event: ‘lose-sword’}
107 lose sword
107 kill monster by sword
107 find sword
P1
107 lose sword
107 find sword
P2
42 find sword
107 kill monster by sword
107 lose sword
107 find sword
107 kill monster by sword
partition = hash(userId)

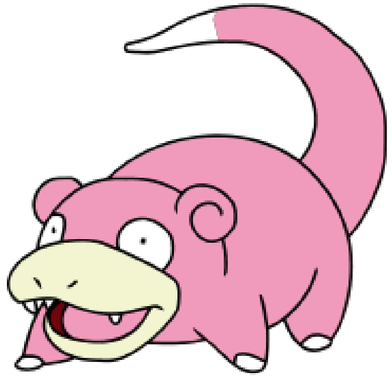
```
KafkaConsumer consumer = newConsumer();
while (!isNeedToShutdown()) {
    val records = consumer.poll(SECONDS.toMillis(10));
    processRecords(records);
    // if processRecords take too much time kafka session will die
    consumer.commitSync();
}
```

KStreams?

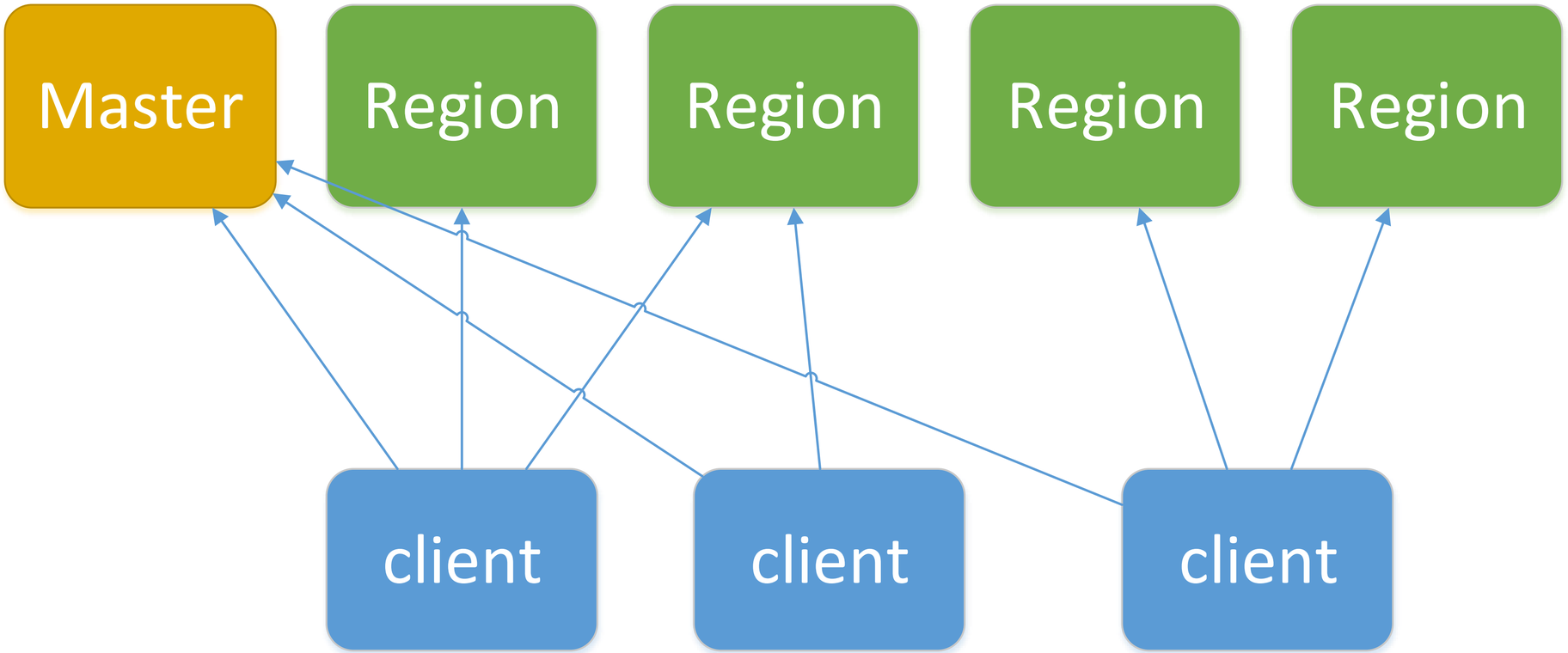
KSt ns?

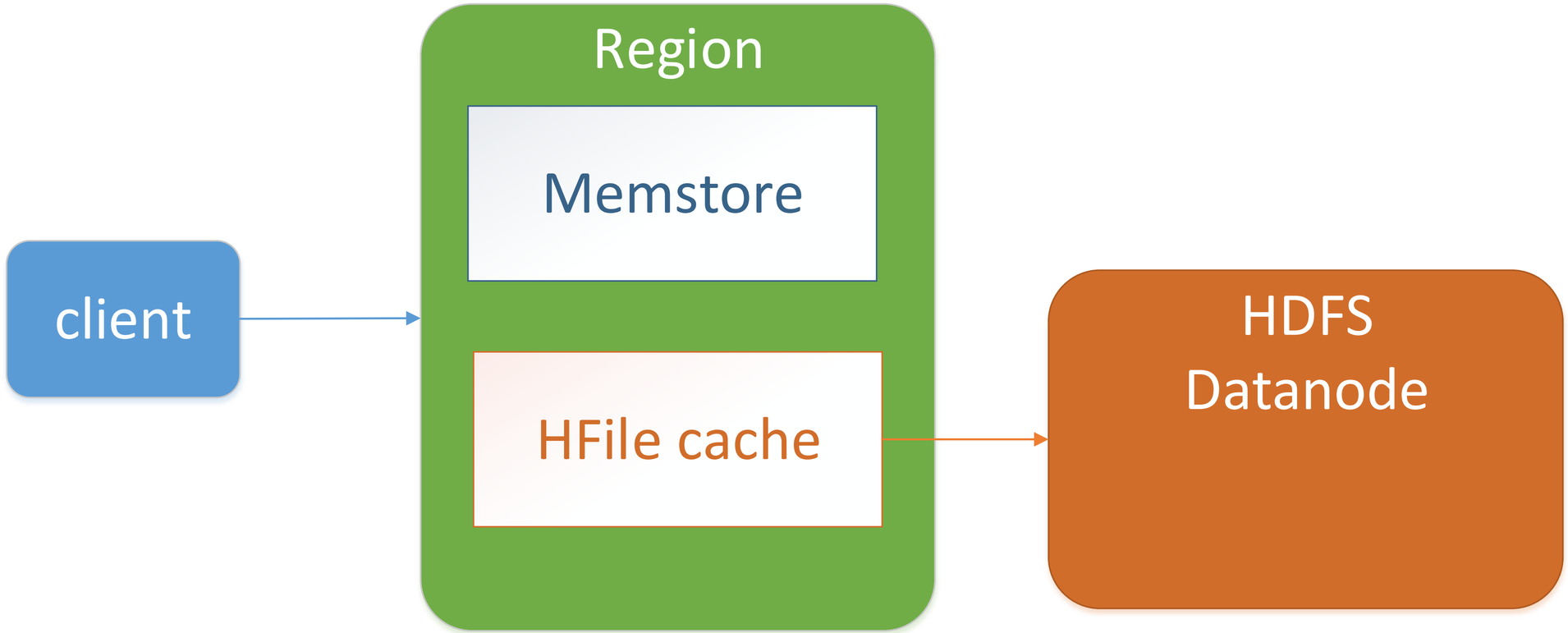
```
Map config = new HashMap();
config.put(ConsumerConfig.GROUP_ID_CONFIG, group);
config.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
config.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
...
KafkaConsumer consumer = new KafkaConsumer(config);
```

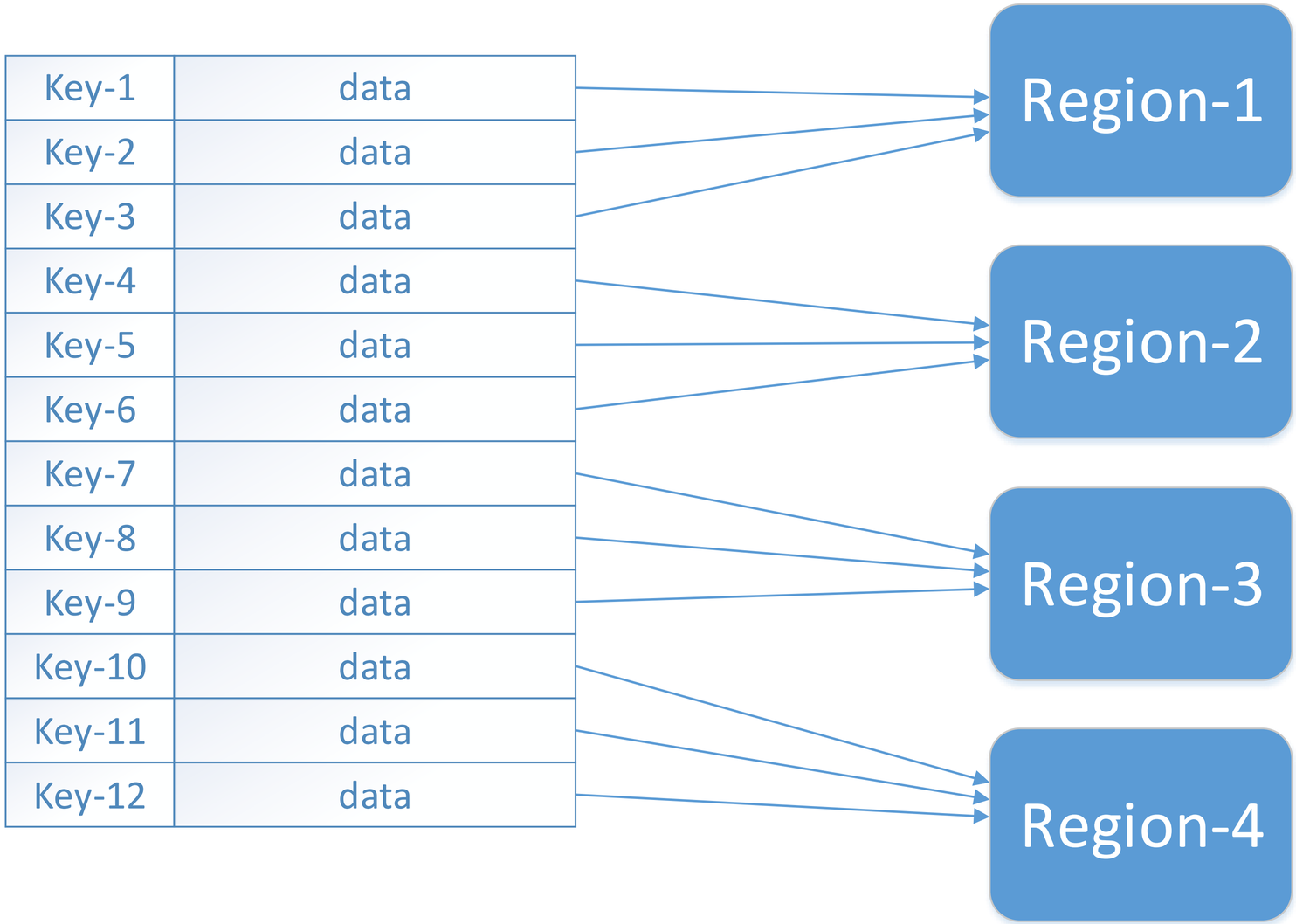
CAS!



CAS!


Master
Region
Region
Region
Region
client
client
client


Region
Memstore
client
HFile cache
HDFS
Datanode


Key-1 data
Key-2 data
Key-3 data
Key-4 data
Key-5 data
Key-6 data
Key-7 data
Key-8 data
Key-9 data
Key-10 data
Key-11 data
Key-12 data
Region-1
Region-2
Region-3
Region-4

partition	timestamp	UID
5bit	43bit	64bit
1	10:00	0x3CAFEBABE…
1	11:17	0xC4AFEBABE…
1	23:05	0xCA5FEBABE…
2	10:00	0xCAF6EBABE…
2	10:01	0xCAFE3BABE…
2	11:45	0xCAFEB7ABE…
2	12:15	0xCAFEBAB9E…
2	13:10	0xCAFEBABE5…
3	10:00	0x8CAFEBABE…
3	10:03	0xC4AFEBABE…
3	12:17	0xCA6FEBABE…
3	15:22	0xCAF5EBABE…

2	10:00	0xC4AFEBABE…
random	17:00	0xCAFE6BABE…

timestamp	counter	server_id
43bit	13bit	7bit
273 years	0-8191	0-127

Keep row key and column name as small as possible

FIX

APACHE HBASE

OPENTSDB

event_type	timestamp	UID
11bit	43bit	64bit

FIX

APACHE HBASE

9 21:10	8 20:19	7 20:18	6 20:12	5 19:53	4 12:45	3 11:17	2 10:05	1 10:00

{ login: «Пицот метроф красной тряпки», name: «Иван», score: 45 },
{ login: «Двести пийсят метроф красной тряпки», name: «Коля», score: 42 }

key = (crc32(login) << 32) | javaHash(login) // 64bit hash key
Index = UUID

RowKey
Hash
64bit

Column		Value
type	index	
0x1	0x1CAFEBABE…	Пицот метроф красной тряпки
0x1	0x2CAFEBABE…	Двести пийсят метроф красной тряпки
0x2	0x1CAFEBABE…	{ name:«Иван», score: 45 }
0x2	0x2CAFEBABE…	{ name:«Коля», score: 42 }

http://github.com/ru-fix

kasfandiyarov@gmail.com

СПАСИБО ЗА ВНИМАНИЕ