

INTRODUCTION TO THE MYSTERIES OF

CLICKHOUSE REPLICATION



by Robert Hodges

Quick Introduction to Altinity

- Premier provider of software and services for ClickHouse
 - Enterprise support for ClickHouse and ecosystem projects
 - ClickHouse Feature Development (Geohashing, codecs, tiered storage, log cleansing...)
 - Software (Kubernetes, cluster manager, tools & utilities)
 - POCs/Training
- Main US/Europe sponsor of ClickHouse community

Introduction to ClickHouse

Understands SQL

Runs on bare metal to cloud

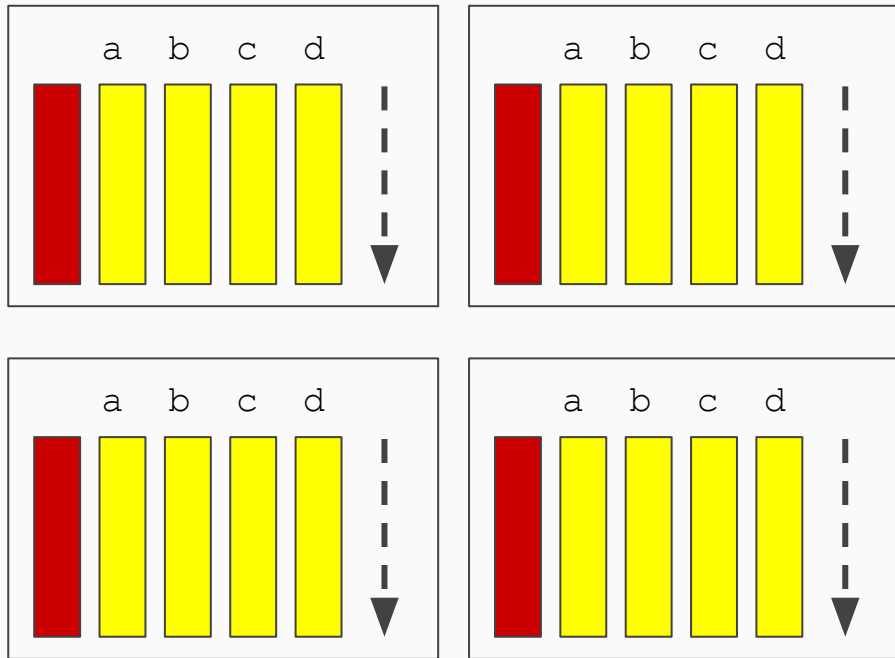
Stores data in columns

Parallel and vectorized execution

Scales to many petabytes

Is Open source (Apache 2.0)

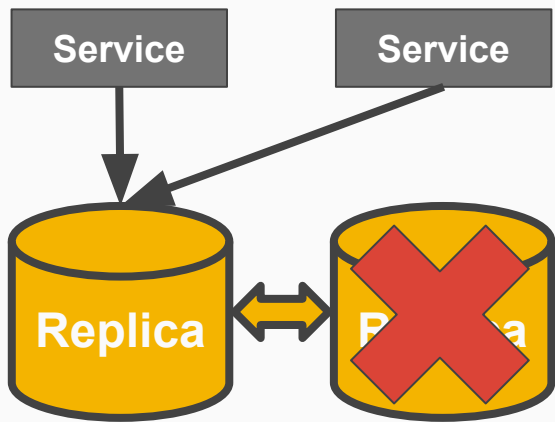
Is WAY fast!



Replication Concepts

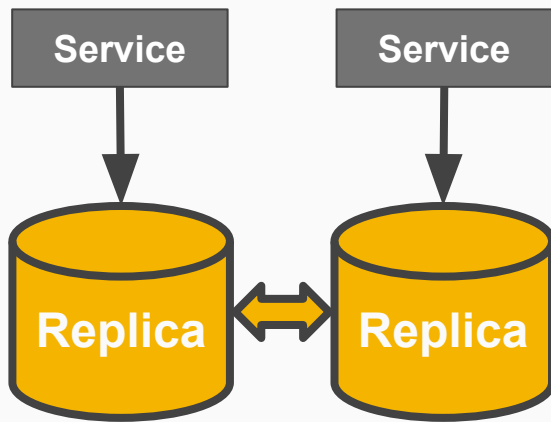
Replication is fundamental to distributed data

High Availability



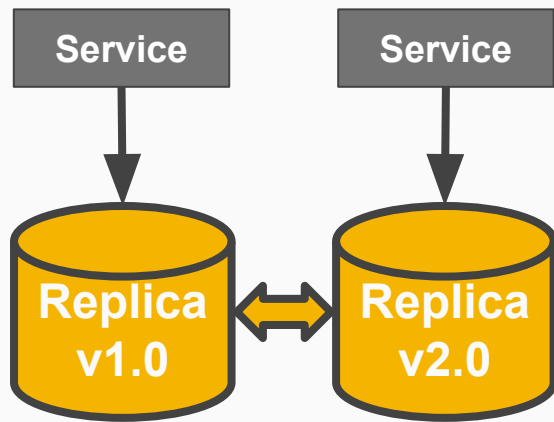
Keep going if a node fails

Load Scaling



Spread traffic across nodes

Migration/Upgrade



Upgrade node version or resources

Replication is different from sharding

Sharded

Shard 1	Shard 2
Shard 3	Shard 4

Data sharded 4
ways without
replication

Replicated

Replica 1	Replica 2
Replica 3	Replica 4

Data replicated 4
times without
sharding

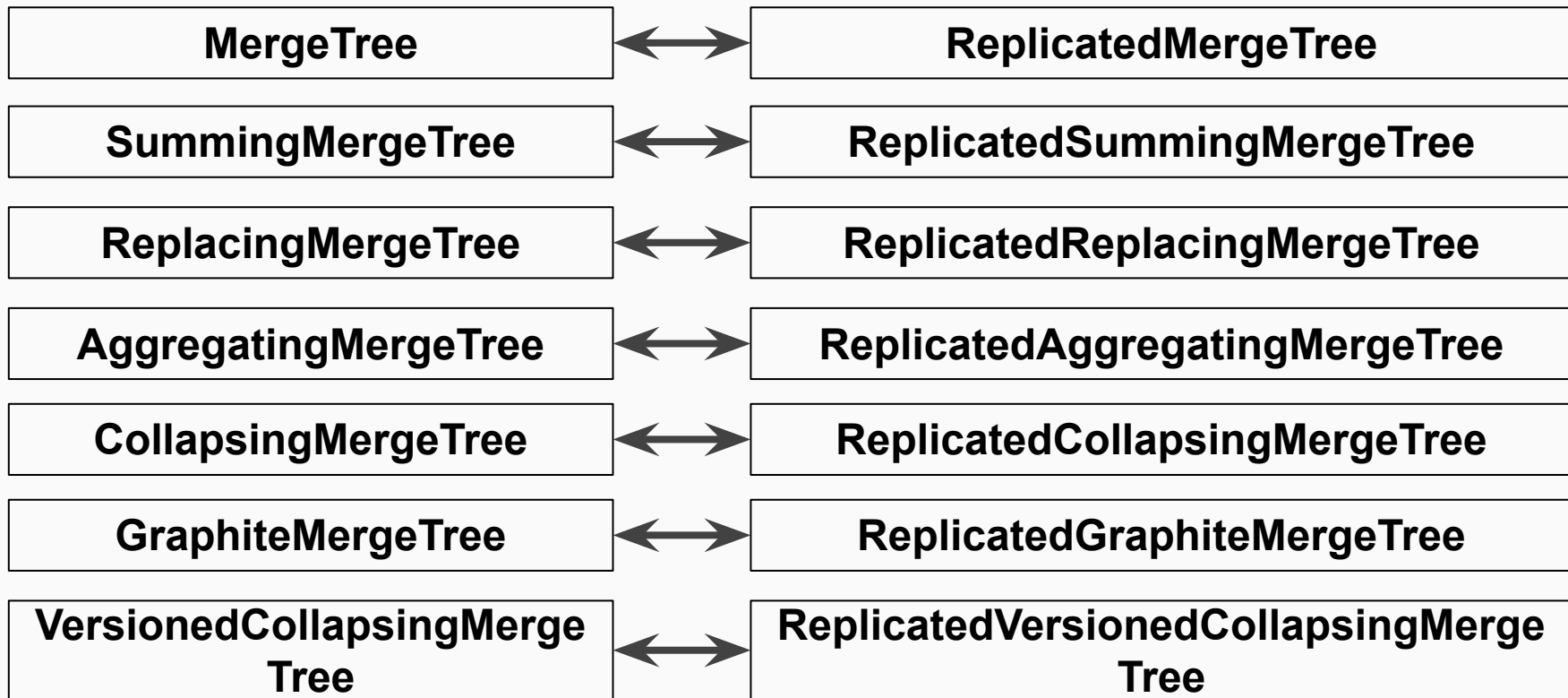
Sharded and Replicated

Shard 1 Replica 1	Shard 2 Replica 1
Shard 1 Replica 2	Shard 2 Replica 2

Data sharded 2
ways and
replicated 2 times

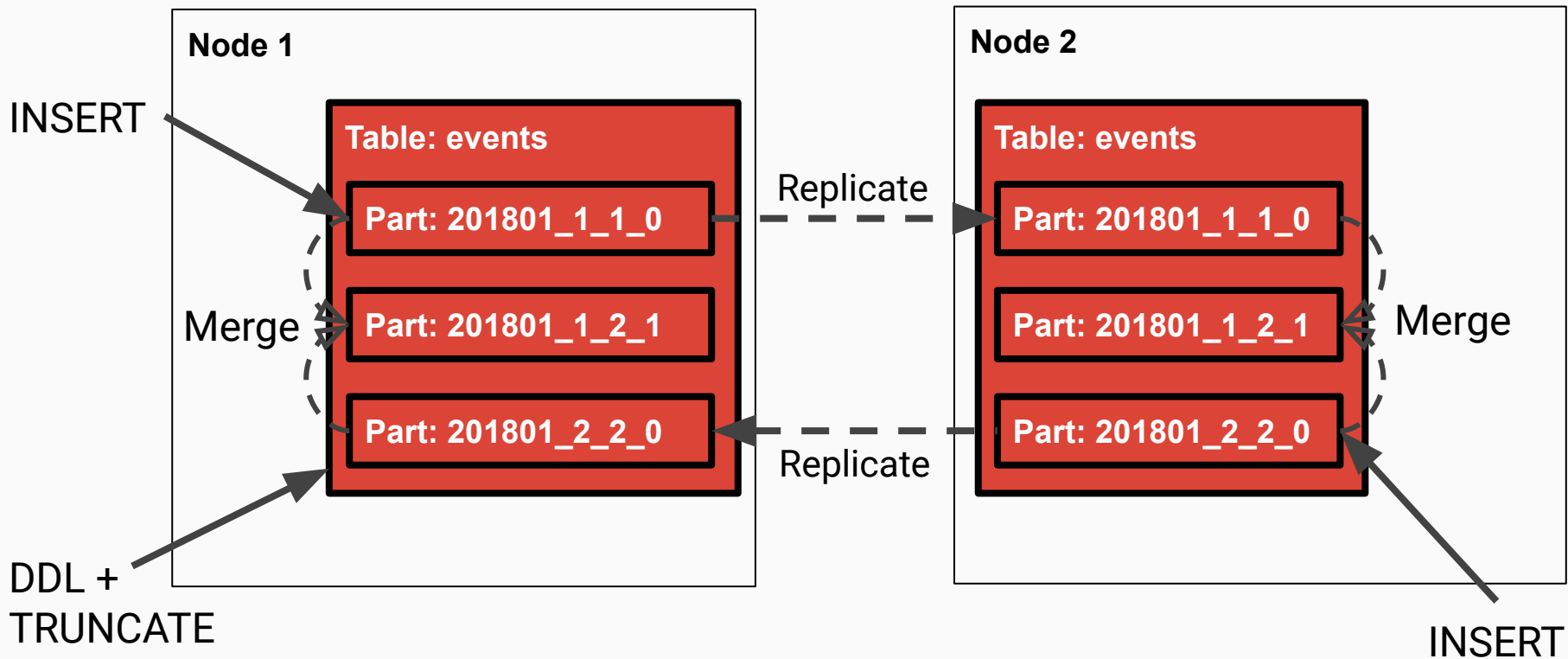
Clickhouse replication extends MergeTree

Non-Replicated vs. Replicated Table Engines



Replication works on per-table basis

Asynchronous multi-master replication



Zookeeper stores state of replicas

:2181

zookeeper-1

```
/clickhouse/.../task_queue/ddl  
/clickhouse/.../metadata  
/clickhouse/.../columns  
/clickhouse/.../leader_election  
/clickhouse/.../log  
/clickhouse/.../replicas  
/clickhouse/.../blocks  
...
```

Distributed DDL queue

Table descriptions

Leader for merges

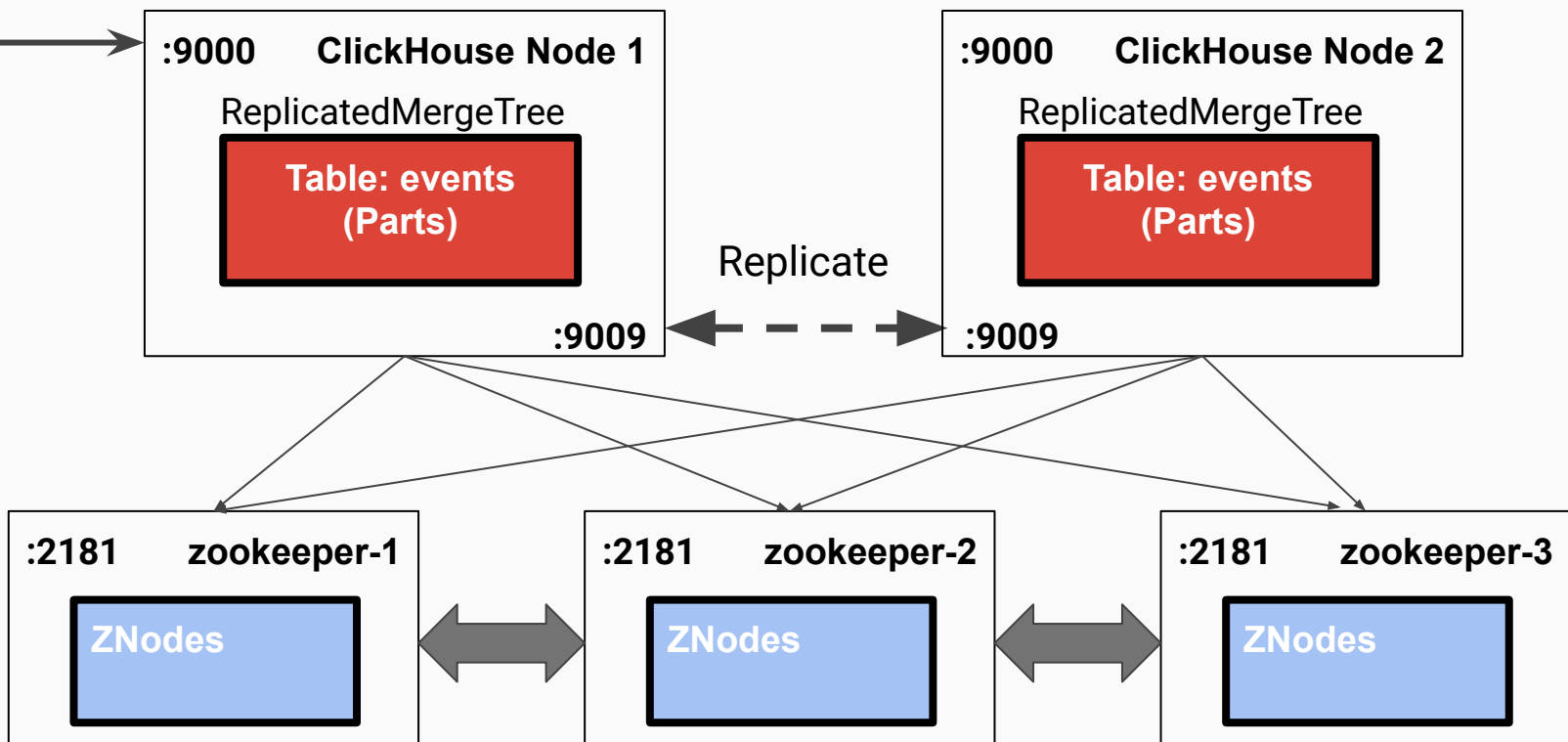
Log of actions

Replicas including parts

Block checksums for deduplication

Clickhouse and Zookeeper implementation

INSERT



Replication Setup

Demo Time

Replication and
sharding in action

Getting started

1. Install Clickhouse servers

```
version="19.11.3.11"  
sudo apt-get install \  
    clickhouse-common-static=$version \  
    clickhouse-client=$version \  
    clickhouse-server=$version
```

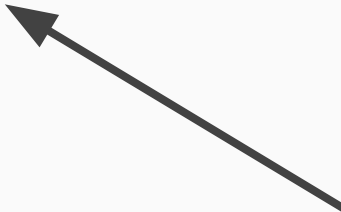
2. Install Zookeeper on at least on 1 additional server (More later)

```
sudo apt-get install zookeeper
```


Clusters define sharding and replication layouts

`/etc/clickhouse-server/config.d/remote_servers.xml:`

```
<yandex>
  <remote_servers>
    <ShardedAndReplicated>
      <shard>
        <replica><host>10.0.0.71</host><port>9000</port></replica>
        <replica><host>10.0.0.72</host><port>9000</port></replica>
        <internal_replication>true</internal_replication>
      </shard>
      <shard>
        . . .
      </shard>
    </ShardedAndReplicated>
  </remote_servers>
</yandex>
```



Use ZK-based internal replication; otherwise, distributed table sends INSERTS to all replicas

Macros enable consistent DDL over a cluster

/etc/clickhouse-server/config.d/macros.xml:

```
<yandex>
  <macros>
    <cluster>ShardedAndReplicated</cluster>
    <shard>01</shard>
    <replica>01</replica>
  </macros>
</yandex>
```

Zookeeper tag defines servers and task queue

/etc/clickhouse-server/config.d/zookeeper.xml:

```
<yandex>
  <zookeeper>
    <node><host>10.0.0.61</host><port>2181</port></node>
    <node><host>10.0.0.62</host><port>2181</port></node>
    <node><host>10.0.0.63</host><port>2181</port></node>
  </zookeeper>
  <distributed_ddl>
    <path>/clickhouse/ShardedAndReplicated/task_queue/ddl</path>
  </distributed_ddl>
</yandex>
```

**Clickhouse restart required after
Zookeeper config changes**

Restart and create tables

```
systemctl restart clickhouse-server # on all servers
```

```
CREATE TABLE events_local ON CLUSTER '{cluster}' (  
    EventDate DateTime, CounterID UInt32, UserID UInt32)  
ENGINE =  
ReplicatedMergeTree('/clickhouse/{cluster}/test/tables/events_local/{shard}',  
    '{replica}')  
PARTITION BY toYYYYMM(EventDate)  
ORDER BY (CounterID, EventDate, intHash32(UserID))
```

```
CREATE TABLE events ON CLUSTER '{cluster}'  
AS test.events_local  
ENGINE = Distributed('{cluster}', 'test', 'events_local', rand())
```


Start adding data!

**INSERT via
distributed table**



```
INSERT INTO events(EventDate, CounterID, UserID) VALUES  
(now(), 1, 2), (now(), 2, 2)
```

```
INSERT INTO events_local(EventDate, CounterID, UserID) VALUES  
(now(), 1, 2), (now(), 2, 2)
```



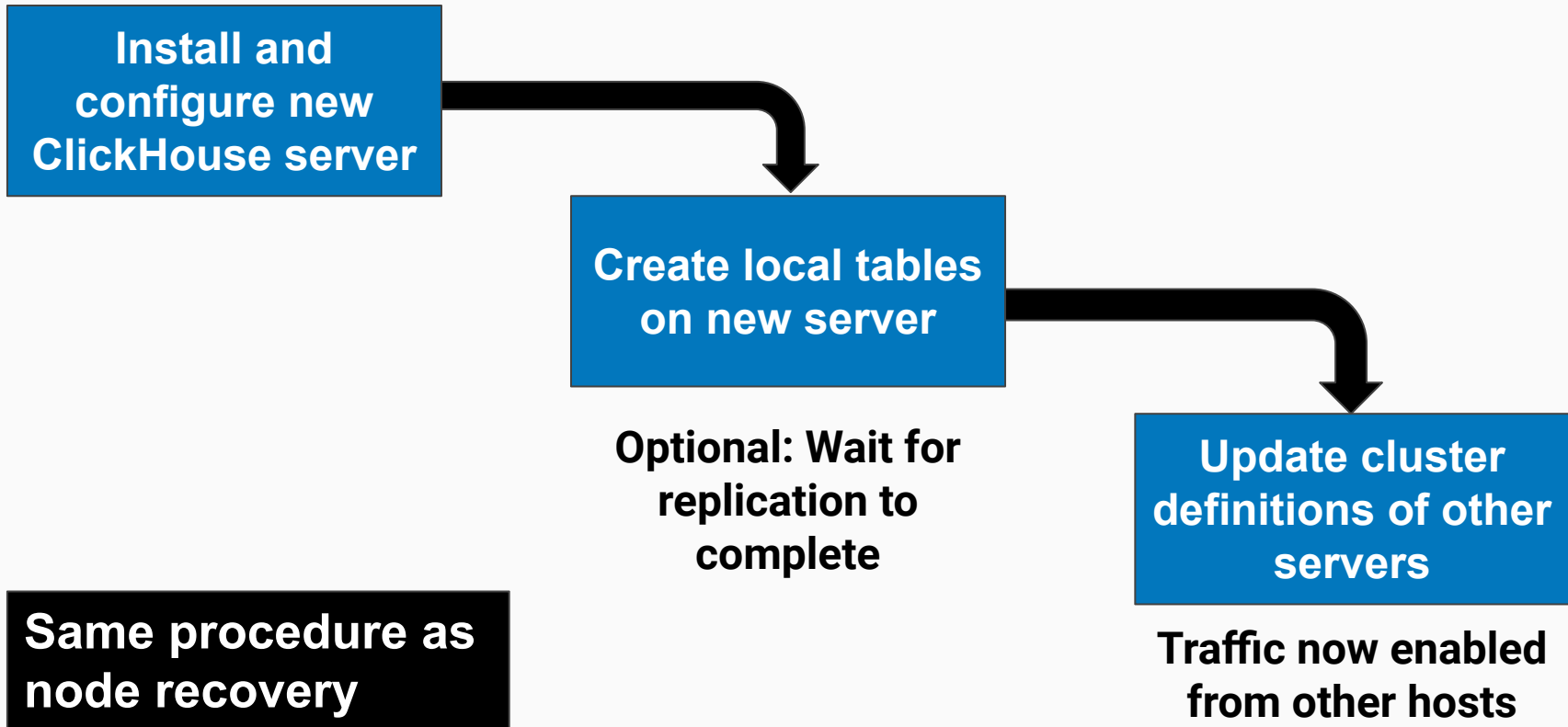
**INSERT directly into
replicated table**

Standard Lifecycle Procedures

SQL Commands and Replication

Command	Replicated?	ON CLUSTER?	Notes
CREATE TABLE	No	Yes	Creates ZK path for table
DROP TABLE	No	Yes	Deletes ZK path for table
SELECT	No	No	Use distributed table to apply across shards
INSERT	Yes	No	Use distributed table to apply across shards
ALTER TABLE	Yes	Yes	
OPTIMIZE TABLE	Yes	Yes	
TRUNCATE TABLE	Yes	Yes	
CREATE DATABASE	No	Yes	
DROP DATABASE	No	Yes	Deletes ZK path for all tables in database

Adding a new node to cluster



Adding a new node, STEP 1

-- Create local replicated table.

```
CREATE TABLE events_local (  
    EventDate DateTime, CounterID UInt32, UserID UInt32)  
ENGINE =  
ReplicatedMergeTree('/clickhouse/{cluster}/test/tables/events_local/{shard}',  
    '{replica}')  
PARTITION BY toYYYYMM(EventDate)  
ORDER BY (CounterID, EventDate, intHash32(UserID))
```

-- Create distributed table.

```
CREATE TABLE events  
AS test.events_local  
ENGINE = Distributed('ShardedAndReplicated', 'test', 'events_local', rand())
```

Adding a new node, STEP 2

/etc/clickhouse-server/config.d/remote_servers.xml:

```
<yandex>
  <remote_servers>
    <ShardedAndReplicated>
      <shard>
        <replica><host>10.0.0.71</host><port>9000</port></replica>
        <replica><host>10.0.0.72</host><port>9000</port></replica>
      </shard>
      <shard> . . . </shard>
    </ShardedAndReplicated>
    <Sharded>

<shard><replica><host>10.0.0.71</host><port>9000</port></replica></shard>
<shard><replica><host>10.0.0.72</host><port>9000</port></replica></shard>
    <Sharded>
  </remote_servers>
</yandex>
```


Removing a node from cluster

**Remove node from
other cluster
configs**

**Stops traffic from
other nodes**

```
drop table test.events_local  
(OR)  
drop database test
```

**Delete replicated
tables**

**Removes
Zookeeper data**

**Discard node/stop
ClickHouse**

(Optional)



**Do not clean up
Zookeeper by hand!**

Changing table schema

Use replicated cluster



```
-- Change a replicated table  
ALTER TABLE events_local ON CLUSTER ShardedAndReplicated  
    ADD COLUMN Extra String DEFAULT 'nothing'
```

```
-- Change a distributed table  
ALTER TABLE events ON CLUSTER Sharded ADD COLUMN Extra String
```

Use non-replicated cluster



Managing data and partitions

Clear all data in a base table across cluster.

```
TRUNCATE TABLE events_local ON CLUSTER ShardedAndReplicated
```

Drop partition for a single shard or for entire cluster.

```
-- Drop only on this shard (removes on all replicas)
```

```
ALTER TABLE events_local  
    DROP PARTITION 201907
```

```
-- Drop on cluster (removes on all shards and all replicas)
```

```
ALTER TABLE events_local ON CLUSTER ShardedAndReplicated  
    DROP PARTITION 201907
```

Resync/recover replicated table contents

(Run SHOW CREATE TABLE locally or on another node)

-- Drop local table and clean up Zookeeper

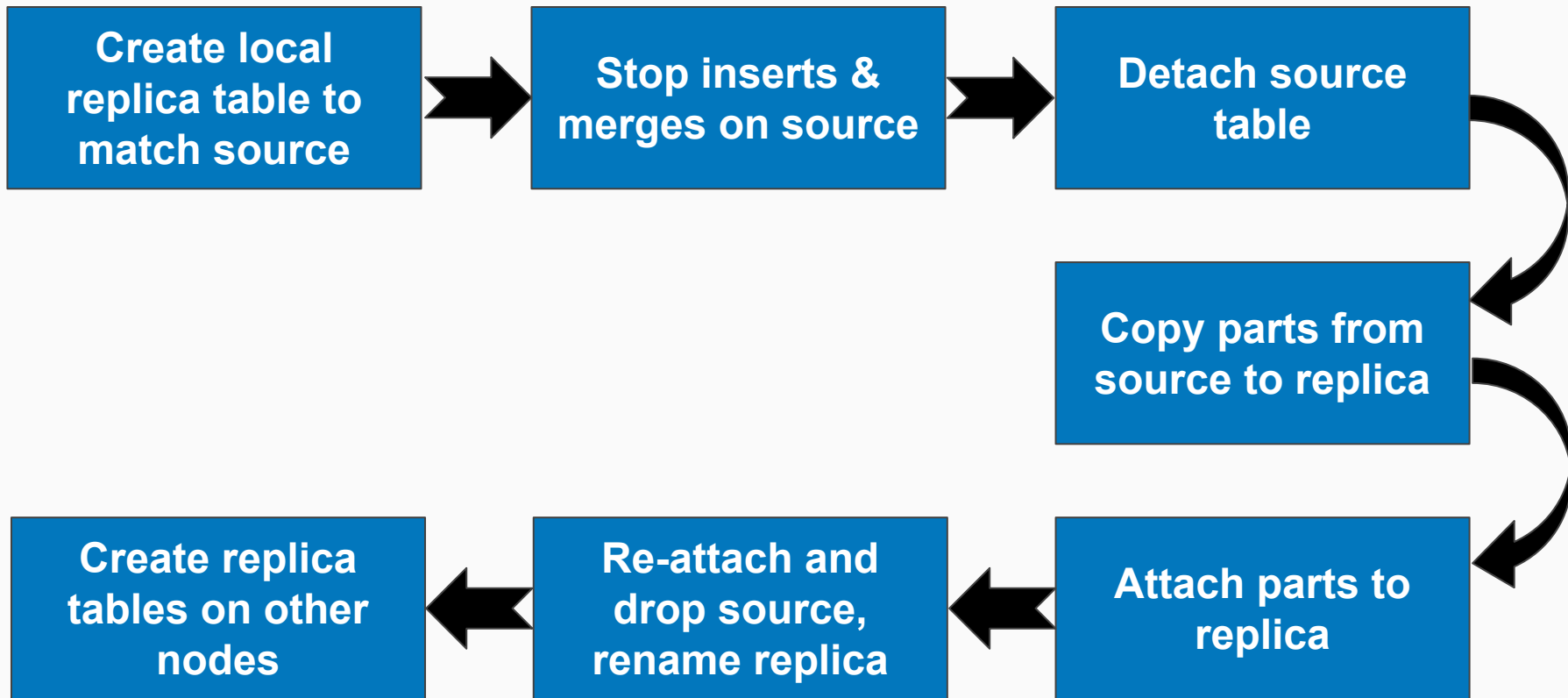
```
DROP TABLE events_local
```

-- Recreate, which automatically fetches data from replica.

```
CREATE TABLE test.events_local (  
    `EventDate` DateTime,  
    `CounterID` UInt32,  
    `UserID` UInt32)  
ENGINE =  
ReplicatedMergeTree('/clickhouse/{cluster}/test/tables/events/{shard}',  
    '{replica}')
```

```
PARTITION BY toYYYYMM(EventDate)  
ORDER BY (CounterID, EventDate, intHash32(UserID))  
SETTINGS index_granularity = 8192
```

Adding an unreplicated table to replication



Adding existing table to replication, STEP 1

-- Create a replicated version of source merge_tree table

```
CREATE TABLE migration_demo_replicated
AS test.migration_demo
ENGINE =
ReplicatedMergeTree('/clickhouse/{cluster}/test/tables/migration_demo/{shard}
', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY (CounterID, EventDate, intHash32(UserID))
```

-- Check for and stop on-going merges

```
SELECT * FROM system.processes
SELECT * FROM system.merges
SYSTEM STOP MERGES migration_demo
```



**Path name cannot
be changed later!**

Adding existing table to replication, STEP 2

-- Generate mv commands to transfer parts from source to replica table

```
SELECT concat('sudo mv /var/lib/clickhouse/data/', database,  
             '/migration_demo/', name, ' /var/lib/clickhouse/data/', database,  
             '/migration_demo_replicated/detached') AS mv  
FROM system.parts WHERE table='migration_demo' AND active
```

-- Generate ALTER TABLE commands to reattach parts to replica

```
SELECT DISTINCT  
    concat('alter table migration_demo_replicated attach partition ',  
          partition, ';')  
FROM system.parts  
WHERE (table = 'migration_demo') AND active
```

Adding existing table to replication, STEP 3

-- Detach table and move the parts over

```
DETACH TABLE migration_demo
```

1. Run generated mv commands to transfer parts to replicated table.
2. Run generated ALTER TABLE ATTACH PARTITION commands to add to new table.
3. (Optional but good to do) Ensure that parts are all properly attached.

```
SELECT * FROM system.parts WHERE table = 'migration_demo_replicated'
```

-- Reattach and drop source, then rename replica table

```
ATTACH TABLE migration_demo
```

```
DROP TABLE migration_demo
```

```
RENAME TABLE migration_demo_replicated TO migration_demo
```

-- Create replica table on remaining shards/replicas

Zookeeper Configuration

Setting up Zookeeper (quick version)

```
# apt-get install zookeeper
```

```
# echo "1" > /var/lib/zookeeper/myid
```

```
# vi /etc/zookeeper/conf/zoo.cfg # set server(s) and purge task interval
```

```
# sudo -u zookeeper /usr/share/zookeeper/bin/zkServer.sh start
```

You MUST set purge interval in zoo.cfg!!
autopurge.purgeInterval=1

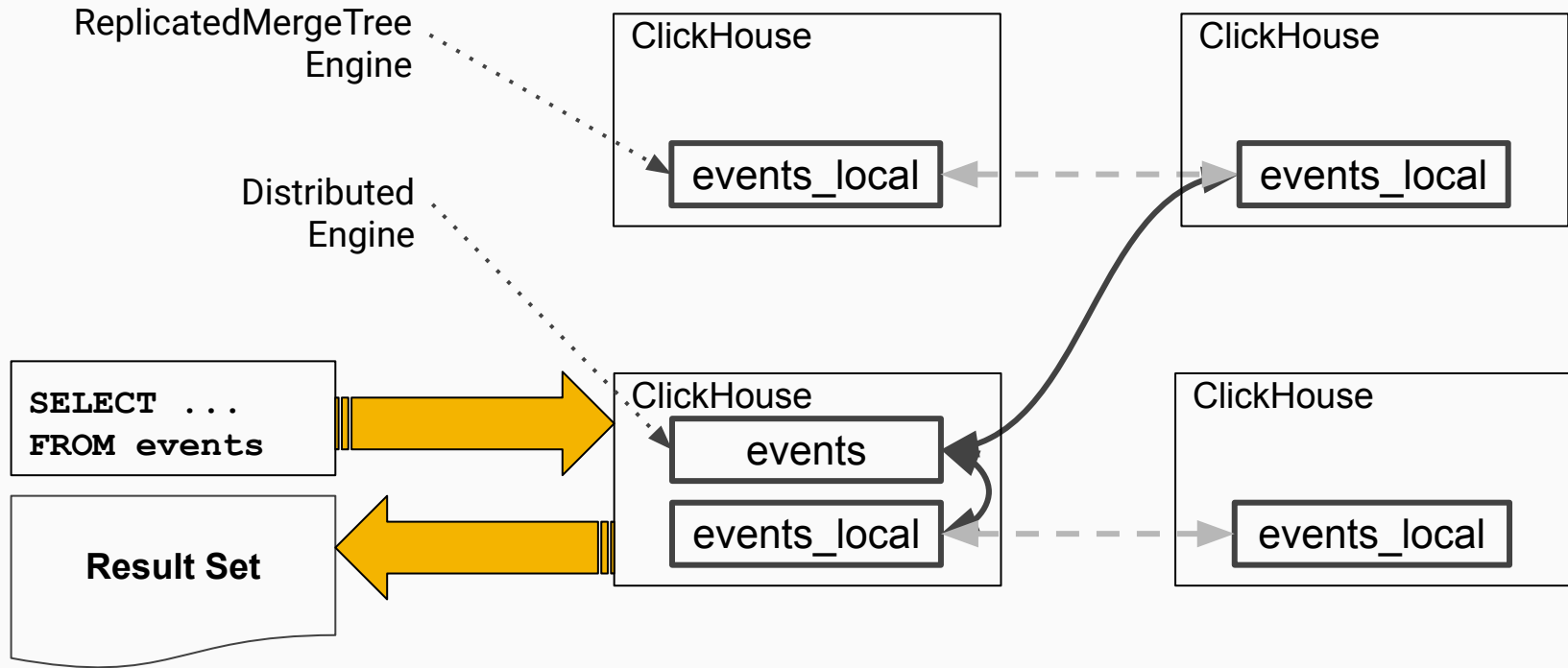
How to keep Zookeeper happy

- Install Zookeeper on separate server(s) from ClickHouse
- Store Zookeeper logs on fast storage (SSD recommended)
- Use 3 or 5 Zookeeper nodes for production installations
 - Never use 2; it's less reliable than 1 node
- Always configure autopurge (autopurge.purgeInterval=1)
- Do not share ClickHouse Zookeeper servers with other systems like Kafka
- Do not clean up Zookeeper data manually
- Use snapshots for Zookeeper backups
- Recovering dead Zookeeper clusters is a manual process--avoid it!

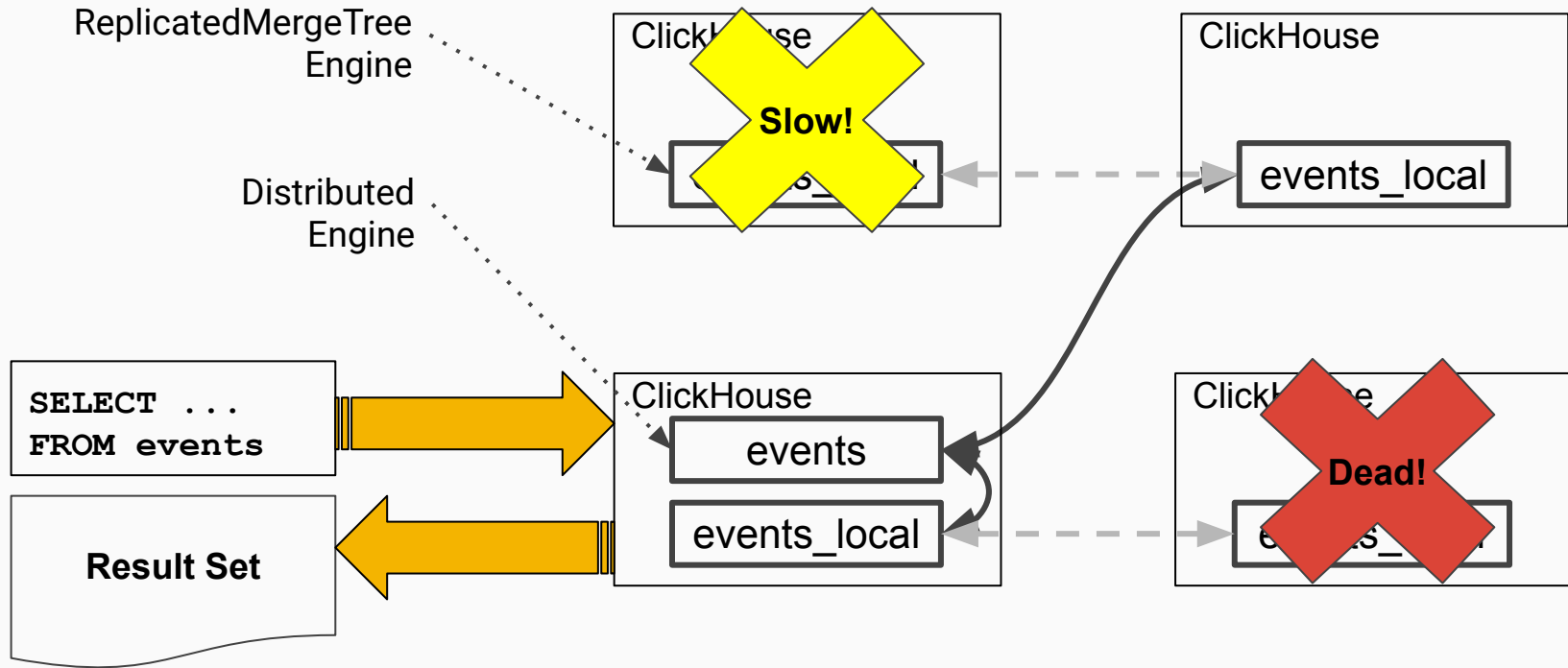
See <https://clickhouse.yandex/docs/en/operations/tips/#zookeeper> for more

Applying Replication and Sharding

Distributed tables and SELECT



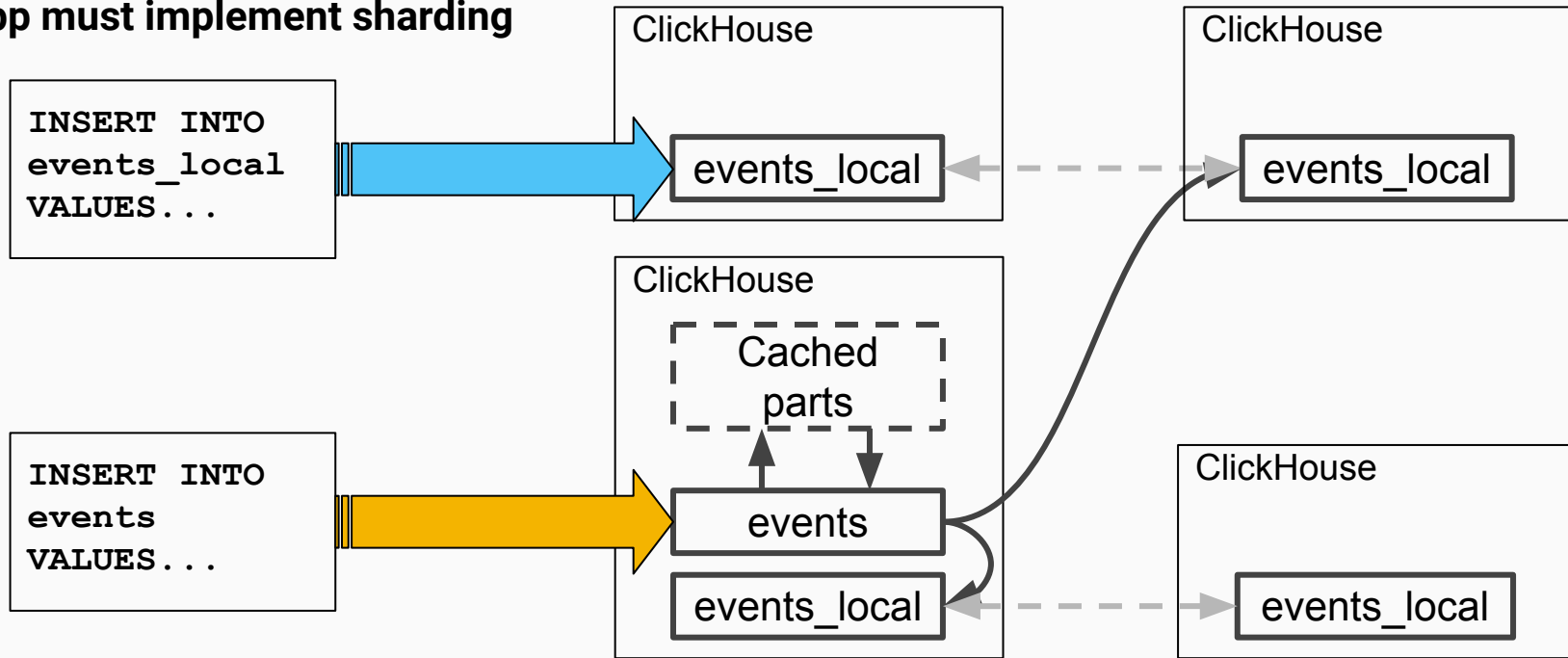
Distributed tables and SELECT



Is it better to load distributed or local tables?

(+) Faster; synchronous, larger parts

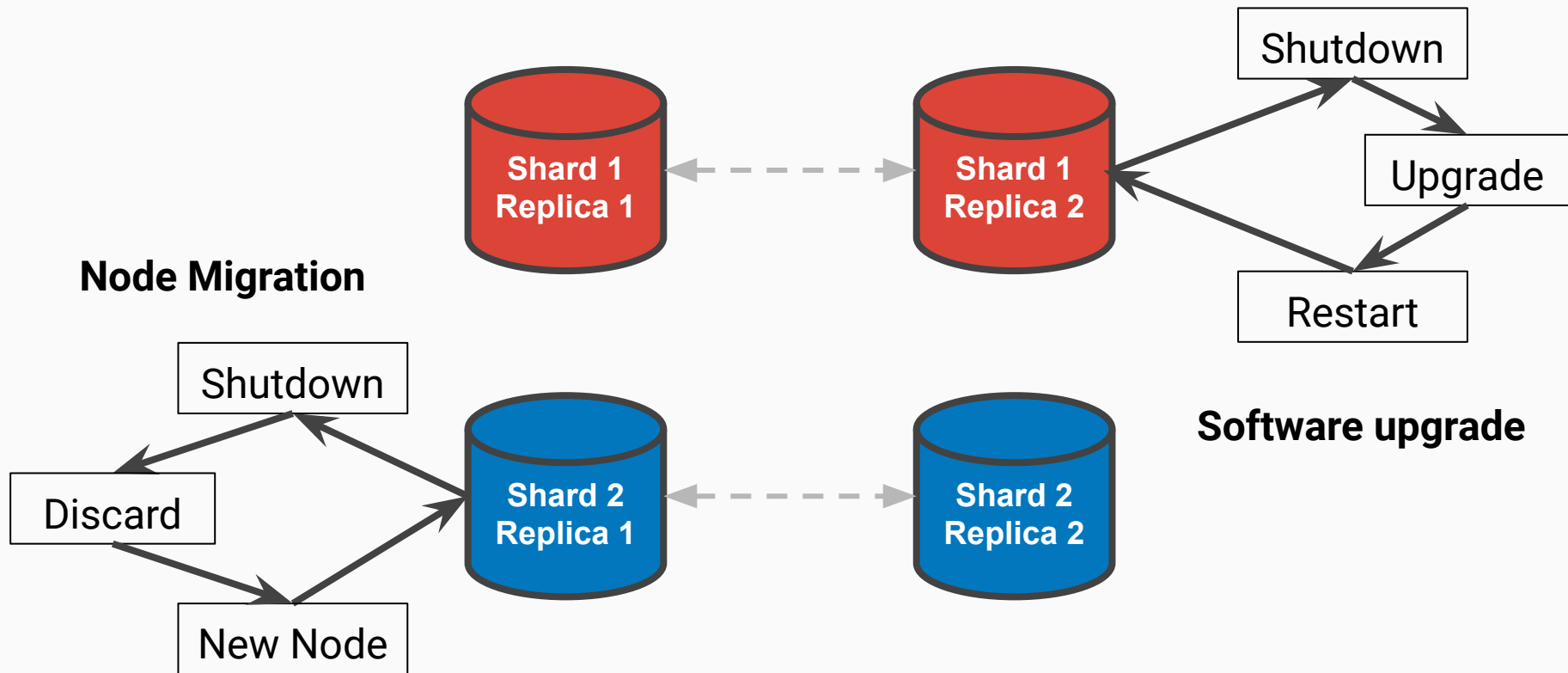
(-) App must implement sharding



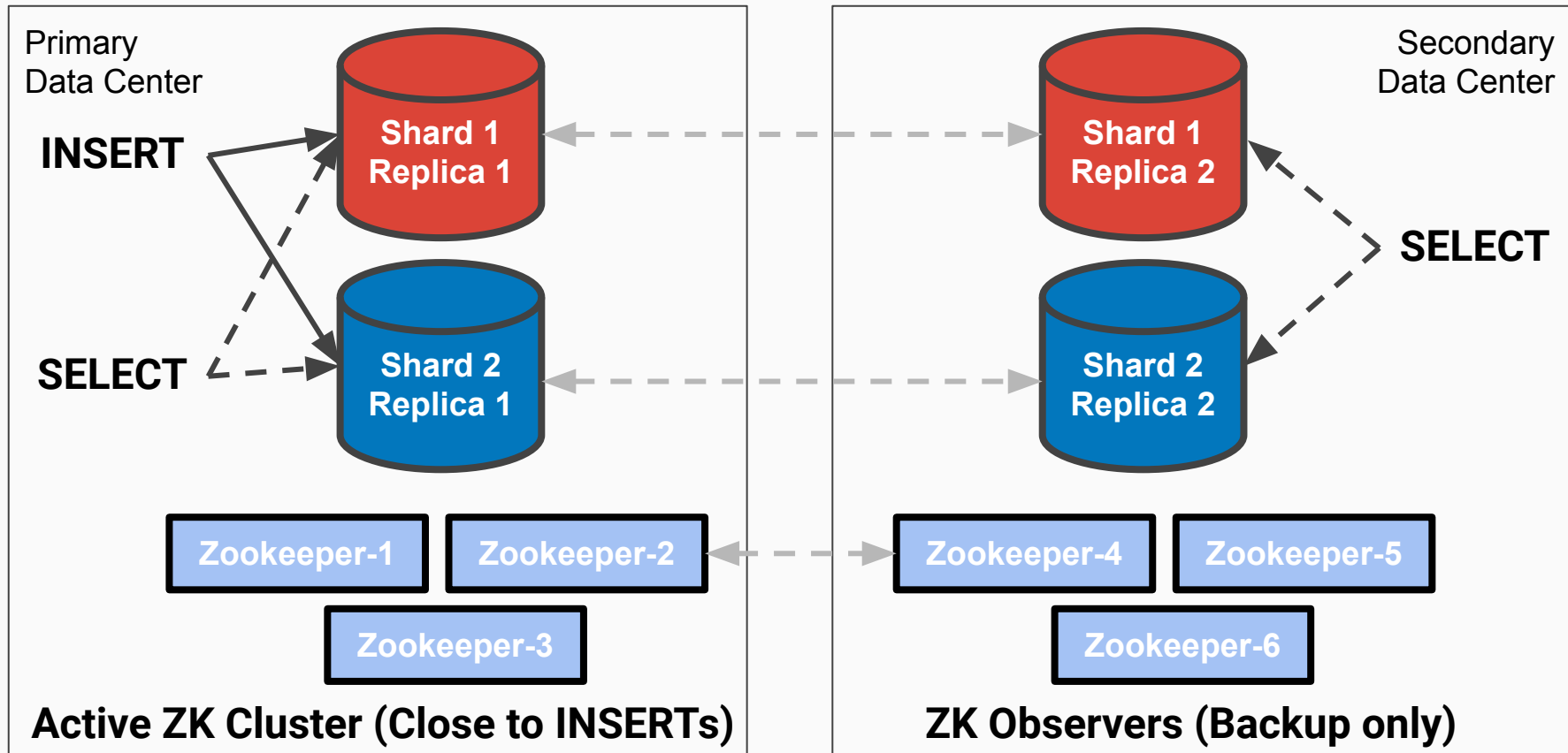
(+) Clickhouse handles sharding

(-) Double writes of data, async insert, smaller parts

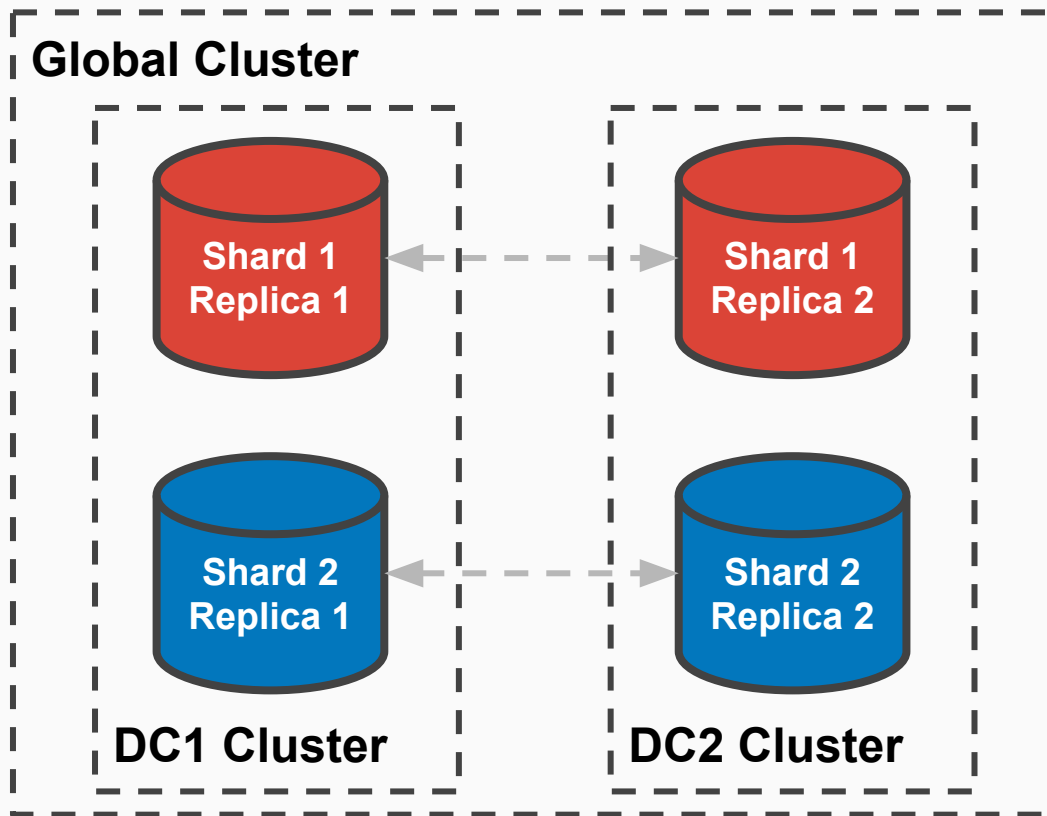
How can replication help migration/upgrade?



How does multi-region replication work?



How can I refer flexibly to sets of nodes?



Are there more tricks and strategies?

Yes! Many more!

- Layered replication
 - Replicated shards with own clusters plus a global cluster
- Cross replication
 - Multiple shards replicated between pairs of hosts
- More load balancing tricks
 - The <remote-servers> tag has a number of options for picking replicas within shards

Diagnostics and Tuning

Key system tables for distributed data

Name	Description
system.clusters	Hosts and clusters (used by Distributed engine)
system.macros	Current macro values
system.merge_tree_settings	Table engine defaults for MergeTree family
system.merges	Pending merges on MergeTree tables
system.parts	Parts belonging to MergeTree tables
system.replicas	Status of replicated tables on the server
system.settings	Settings applicable to current session
system.zookeeper	Reads data from zookeeper paths

Useful tuning parameters

MergeTree Settings

Values that affect
replication and use of
zookeeper

```
SELECT *  
  FROM system.settings  
 WHERE name LIKE '%replica%'
```

Session Settings

Values that affect
distributed queries and
replicated DDL

```
SELECT *  
  FROM system.settings  
 WHERE (name LIKE '%distributed%')  
        OR (name LIKE '%replica%')
```

Remote function is handy to check data

```
SELECT * FROM
(
    SELECT hostName(), *
    FROM remote('10.0.0.71', 'test', 'events_local')
    UNION ALL
    SELECT hostName(), *
    FROM remote('10.0.0.72', 'test', 'events_local')
    UNION ALL
)
```


Conclusion

Mysteries and solutions

- Replication solves HA, scaling, and migration
- Clickhouse can handle many sharding and replication layouts (“clusters”)
- Clickhouse replication works on tables
- Clickhouse replicates INSERTs as well as DDL
- Distributed tables spread load across shards
- Keep Zookeeper healthy to ensure replication health
- Do not manually change Zookeeper data

Is there a better way to manage replication?

Clickhouse Kubernetes Operator!

<https://github.com/Altinity/clickhouse-operator>

Thank you!

Special Offer:

Contact us for a
1-hour consultation!

Contacts:

info@altinity.com

Visit us at:

<https://www.altinity.com>

Free Consultation:

<https://blog.altinity.com/offer>