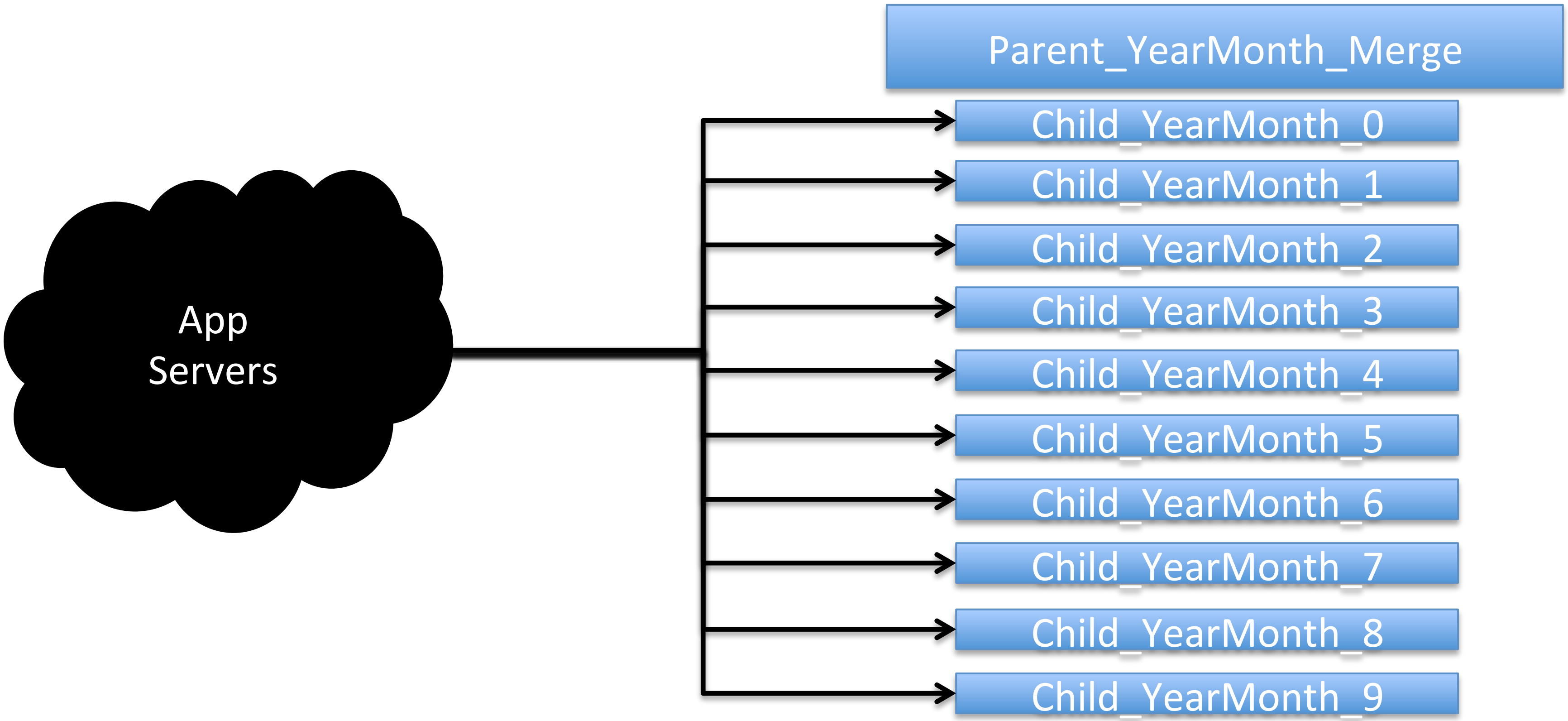
Sqooping 50 Million Rows a Day from MySQL

Eric Hernandez
Database Administrator

SELLING SOURCE®
Marketing. Technology. Data.



App Servers

Parent_YearMonth_Merge

Child_YearMonth_0

Child_YearMonth_1

Child_YearMonth_2

Child_YearMonth_3

Child_YearMonth_4

Child_YearMonth_5

Child_YearMonth_6

Child_YearMonth_7

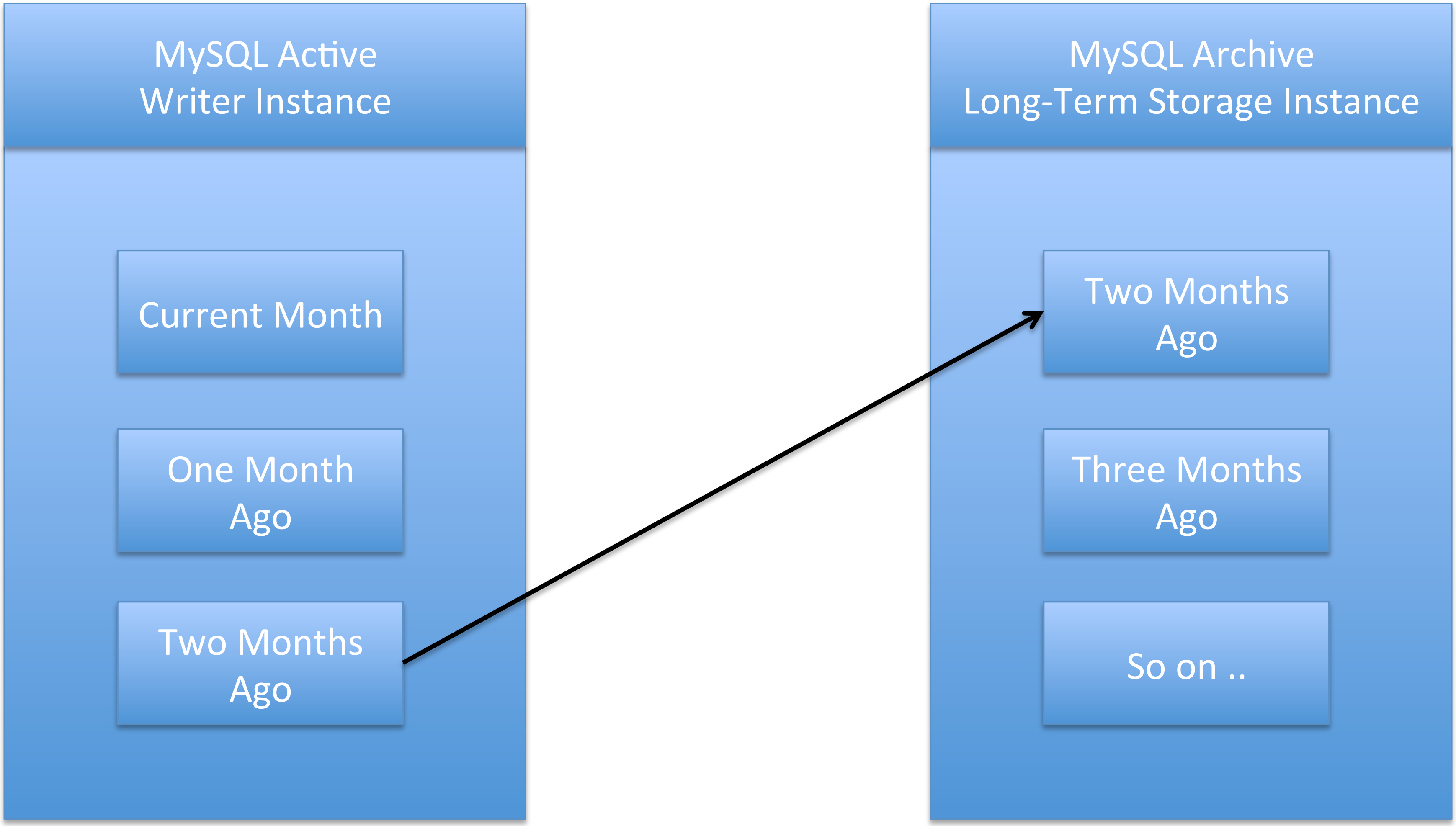
Child_YearMonth_8

Child_YearMonth_9

+50 Million New Rows a Day

+1.5 Billion Rows a Month

3 Month Rotational Life Cycle



MySQL Active
Writer Instance

Current Month

One Month
Ago

Two Months
Ago

MySQL Archive
Long-Term Storage Instance

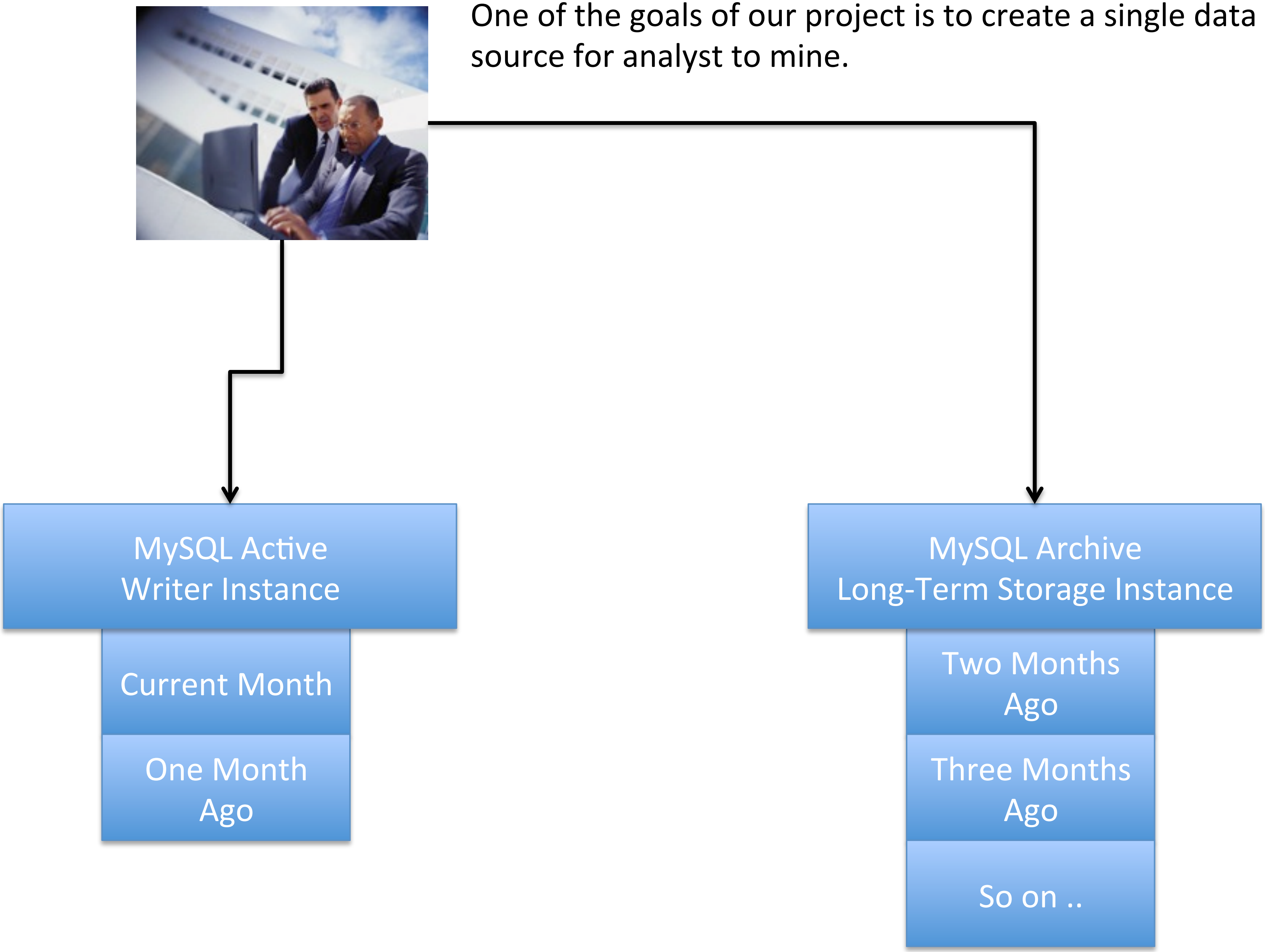
Two Months
Ago

Three Months
Ago

So on ..

Problem: Data Analyst have to pull data from two different sources.

One of the goals of our project is to create a single data source for analyst to mine.



MySQL Active Writer Instance

Current Month

One Month Ago

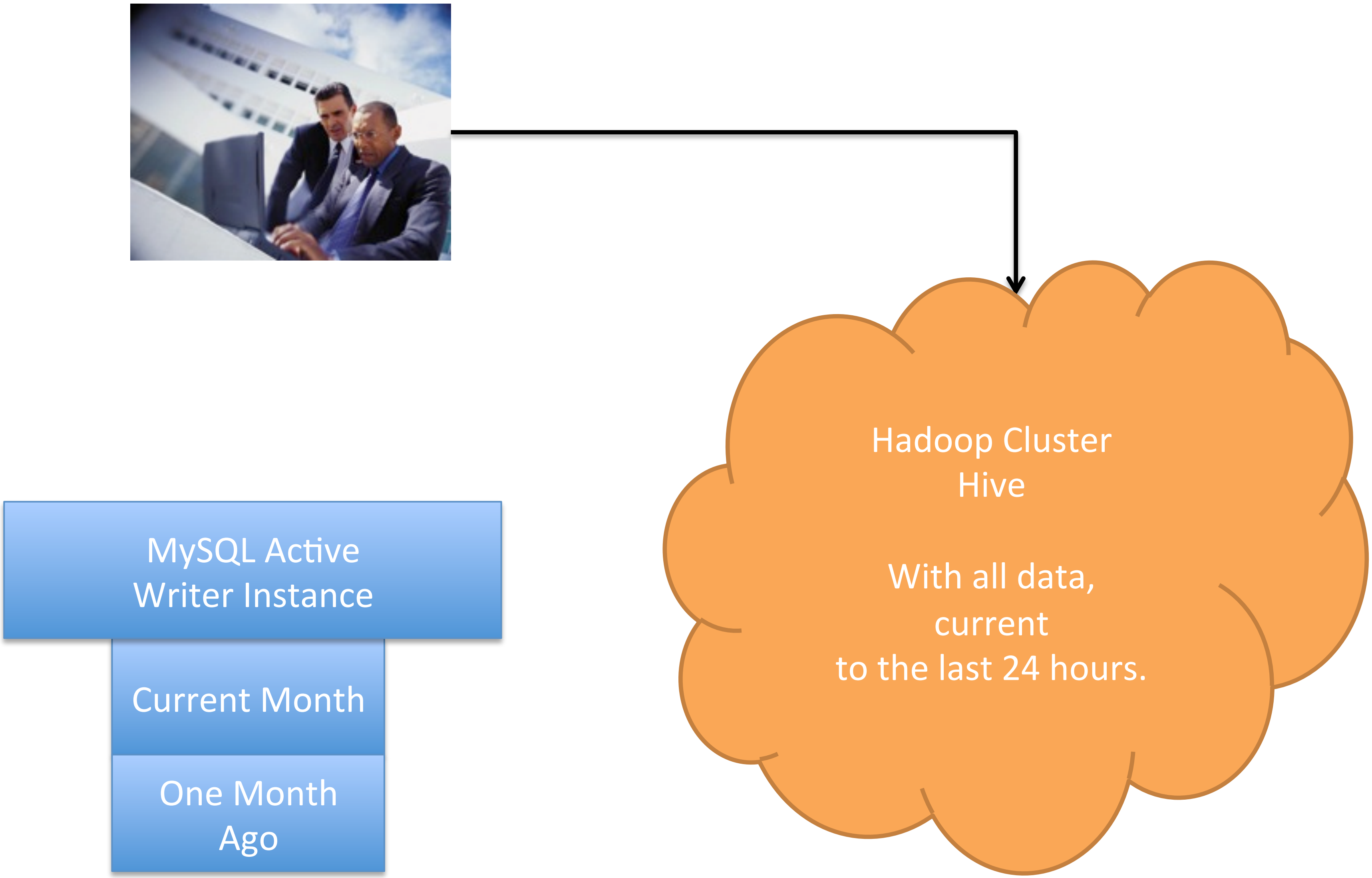
MySQL Archive Long-Term Storage Instance

Two Months Ago

Three Months Ago

So on ..

Data Analyst with Hadoop only have to pull from one data source.



Hadoop Cluster
Hive

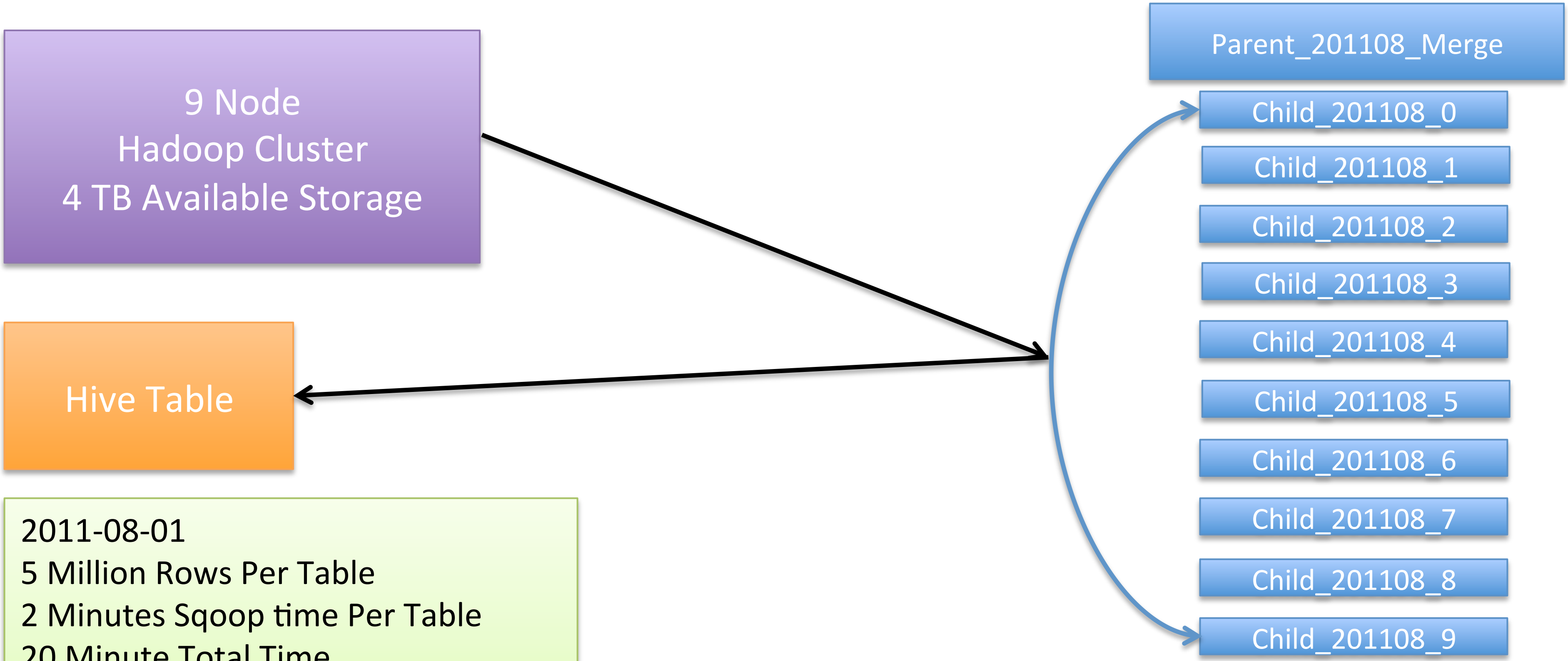
With all data,
current
to the last 24 hours.

MySQL Active
Writer Instance

Current Month

One Month
Ago

Attempt 1.0 Sqooping in Data from MySQL
Sqoop entire table into hive every day at 0030



9 Node
Hadoop Cluster
4 TB Available Storage

Hive Table

Parent_201108_Merge

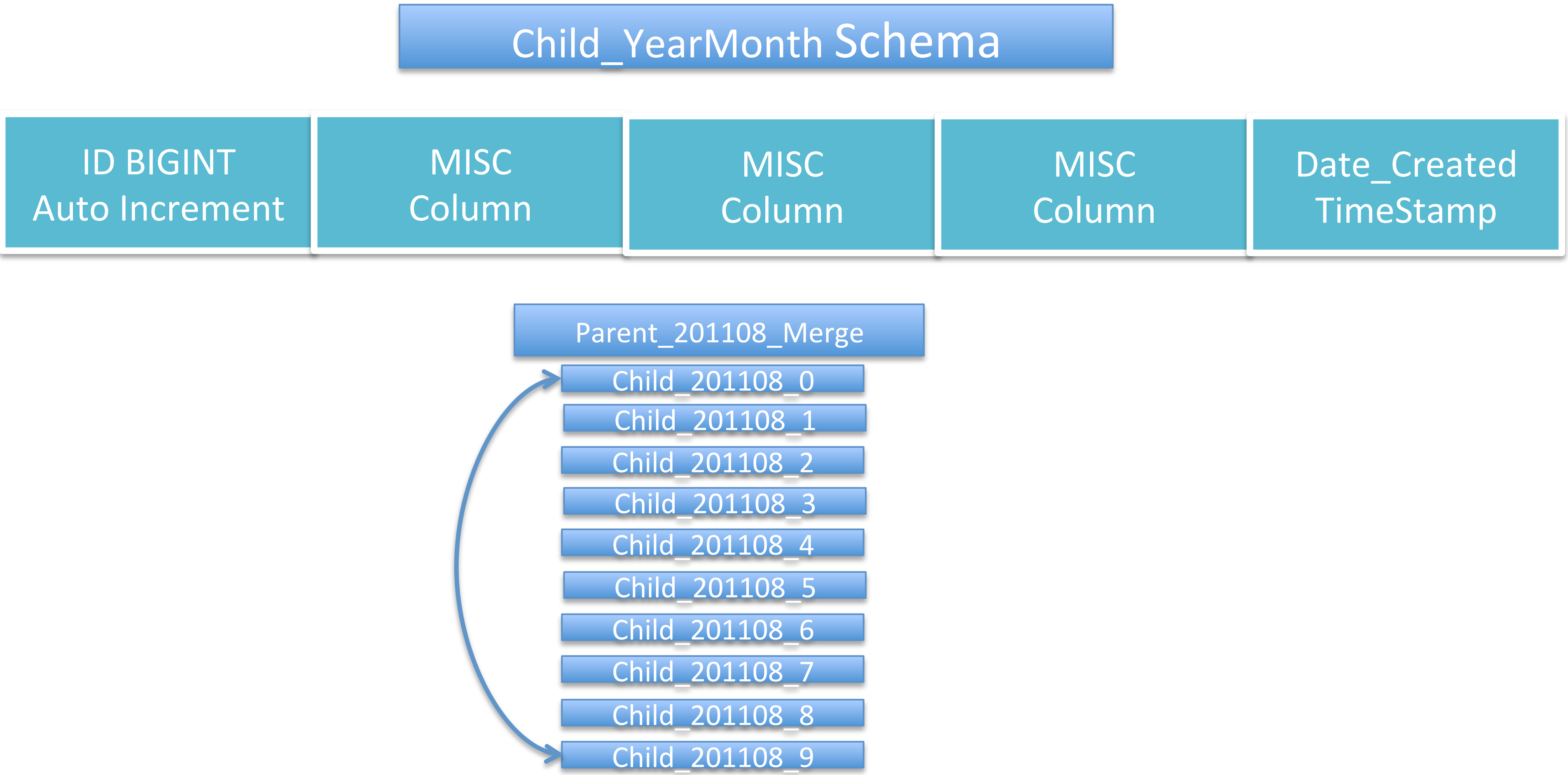
Child_201108_0
Child_201108_1
Child_201108_2
Child_201108_3
Child_201108_4
Child_201108_5
Child_201108_6
Child_201108_7
Child_201108_8
Child_201108_9

2011-08-01
5 Million Rows Per Table
2 Minutes Sqoop time Per Table
20 Minute Total Time
Total 50 Million Rows into Hive Table

2011-08-02
10 Million Rows Per Table
4 Minutes Sqoop time Per Table
40 Minutes Total Time
Total 100 Million Rows into Hive Table

2011-08-10
50 Million Rows Per Table
20 Minutes Sqoop time Per Table
200 Minutes Total Time
Total 500 Million Rows into Hive Table

Attempt 2.0 Incremental Sqoop of Data from MySQL



Child_YearMonth Schema

ID BIGINT Auto Increment

MISC Column

MISC Column

MISC Column

Date_Created TimeStamp

Parent_201108_Merge

Child_201108_0

Child_201108_1

Child_201108_2

Child_201108_3

Child_201108_4

Child_201108_5

Child_201108_6

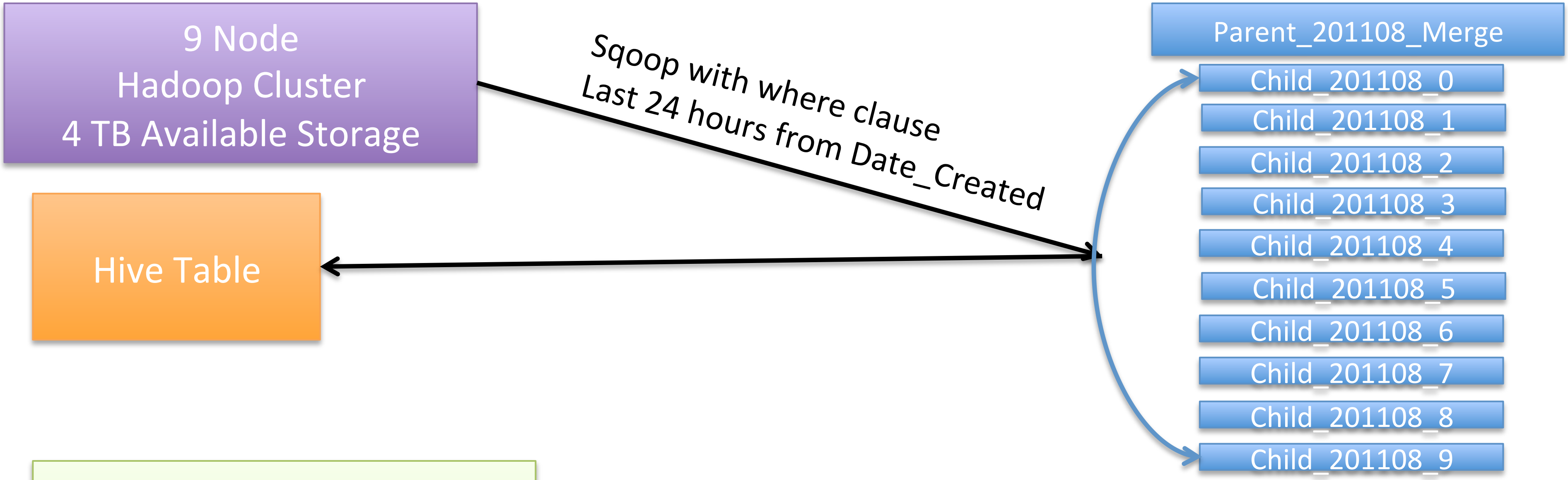
Child_201108_7

Child_201108_8

Child_201108_9

```
sqoop import --where "date_created between '${DATE} 00:00:00' and '${DATE} 23:59:59’”
```

Attempt 2.0 Incremental Sqoop of Data from MySQL



9 Node
Hadoop Cluster
4 TB Available Storage

Hive Table

Sqoop with where clause
Last 24 hours from Date_Created

Parent_201108_Merge

Child_201108_0
Child_201108_1
Child_201108_2
Child_201108_3
Child_201108_4
Child_201108_5
Child_201108_6
Child_201108_7
Child_201108_8
Child_201108_9

2011-08-01
5 Million Rows Per Table
2 Minutes Sqoop time Per Table
10 Minute Total Time
Total 50 Million Rows into Hive Table

2011-08-02
5 Million Rows Per Table
2 Minutes Sqoop time Per Table
10 Minute Total Time
Total 50 Million Rows into Hive Table

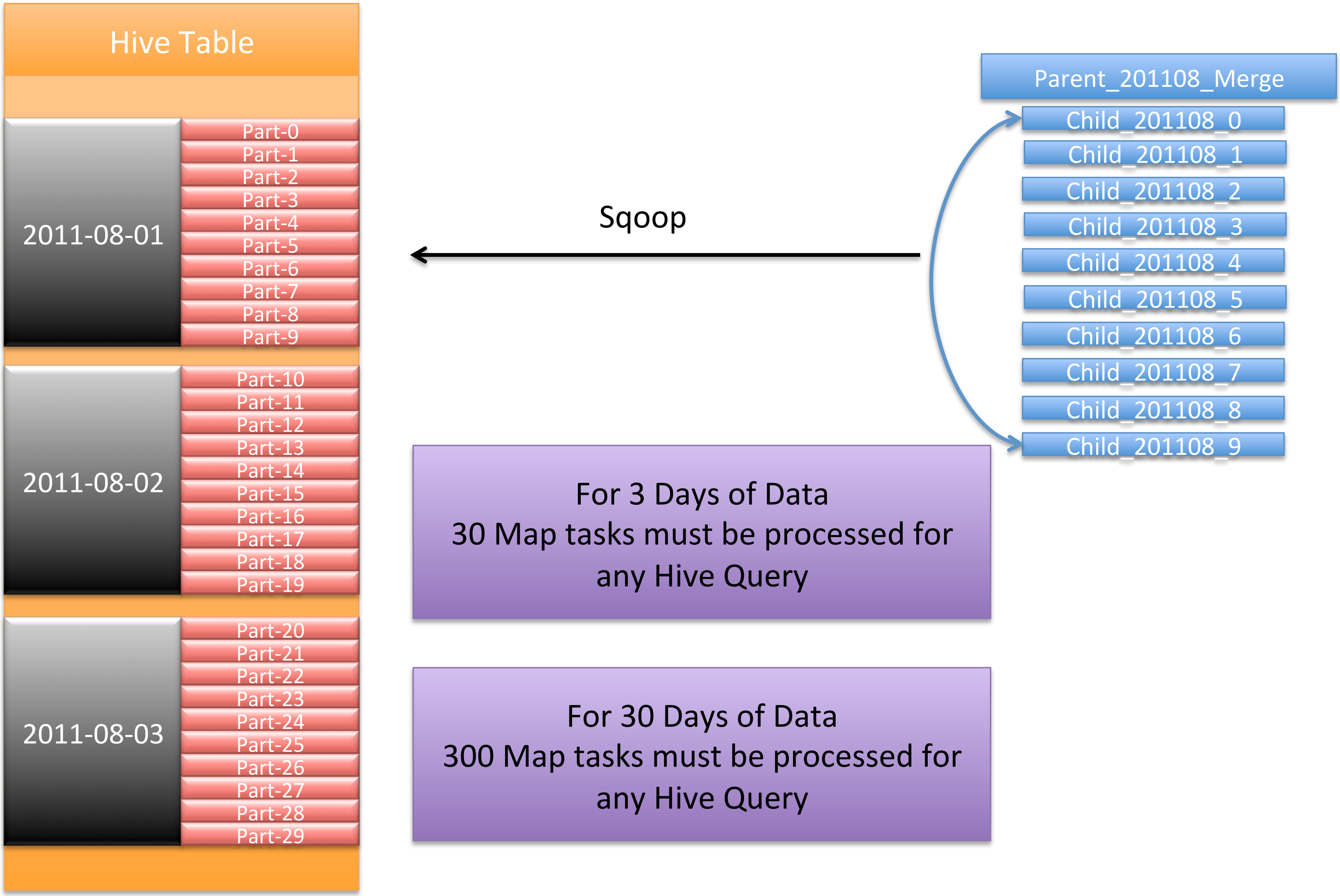
2011-08-10
5 Million Rows Per Table
2 Minutes Sqoop time Per Table
10 Minute Total Time
Total 50 Million Rows into Hive Table

Consistent run times for sqoop jobs achieved

After our 2.0 Incremental Process we had achieved consistent run times however, two new problems surfaced.

1) Each day 10 new parts would be added to the Hive table which caused 10 more map tasks per hive query.
2) Space consumption on hadoop cluster.

Too many parts and map tasks per query.



Hive Table

2011-08-01

Part-0
Part-1
Part-2
Part-3
Part-4
Part-5
Part-6
Part-7
Part-8
Part-9

2011-08-02

Part-10
Part-11
Part-12
Part-13
Part-14
Part-15
Part-16
Part-17
Part-18
Part-19

2011-08-03

Part-20
Part-21
Part-22
Part-23
Part-24
Part-25
Part-26
Part-27
Part-28
Part-29

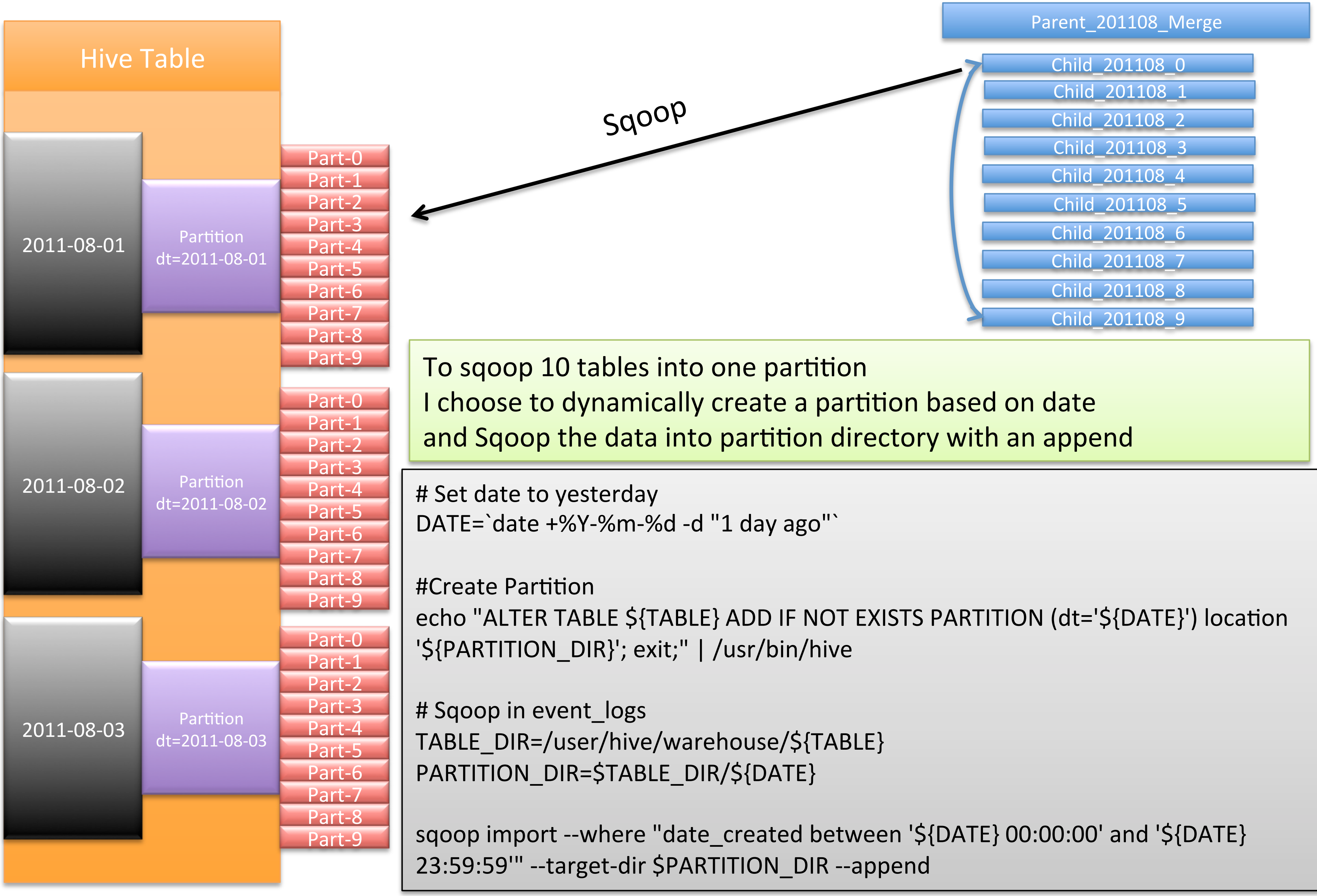
Sqoop

Parent_201108_Merge

Child_201108_0
Child_201108_1
Child_201108_2
Child_201108_3
Child_201108_4
Child_201108_5
Child_201108_6
Child_201108_7
Child_201108_8
Child_201108_9

For 3 Days of Data
30 Map tasks must be processed for
any Hive Query

For 30 Days of Data
300 Map tasks must be processed for
any Hive Query



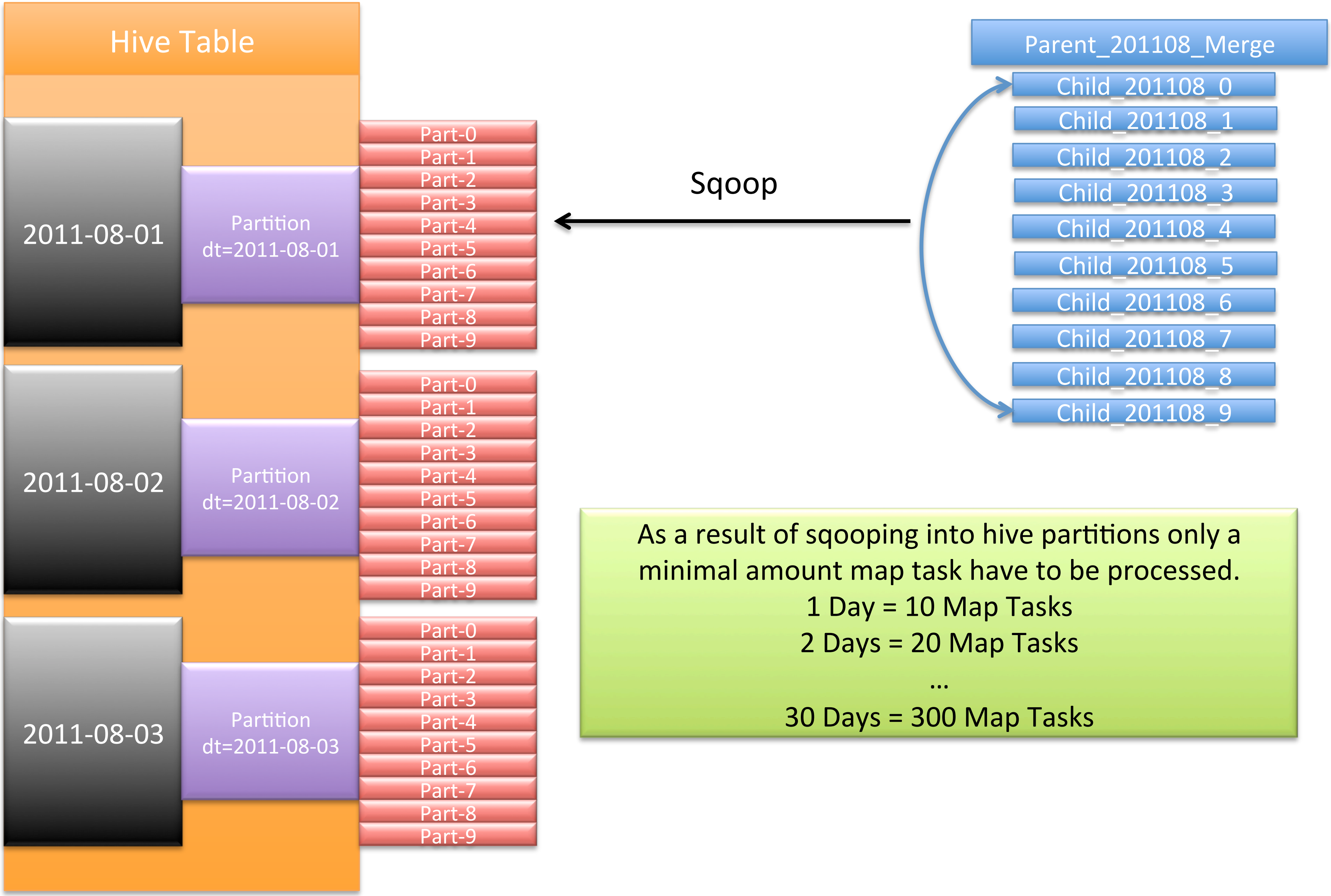
To sqoop 10 tables into one partition
I choose to dynamically create a partition based on date
and Sqoop the data into partition directory with an append

```
# Set date to yesterday
DATE=`date +%Y-%m-%d -d "1 day ago"`

#Create Partition
echo "ALTER TABLE ${TABLE} ADD IF NOT EXISTS PARTITION (dt='${DATE}') location
'${PARTITION_DIR}'; exit;" | /usr/bin/hive

# Sqoop in event_logs
TABLE_DIR=/user/hive/warehouse/${TABLE}
PARTITION_DIR=$TABLE_DIR/${DATE}

sqoop import --where "date_created between '${DATE} 00:00:00' and '${DATE}
23:59:59'" --target-dir $PARTITION_DIR --append
```



Hive Table

2011-08-01

Partition dt=2011-08-01

Part-0
Part-1
Part-2
Part-3
Part-4
Part-5
Part-6
Part-7
Part-8
Part-9

2011-08-02

Partition dt=2011-08-02

Part-0
Part-1
Part-2
Part-3
Part-4
Part-5
Part-6
Part-7
Part-8
Part-9

2011-08-03

Partition dt=2011-08-03

Part-0
Part-1
Part-2
Part-3
Part-4
Part-5
Part-6
Part-7
Part-8
Part-9

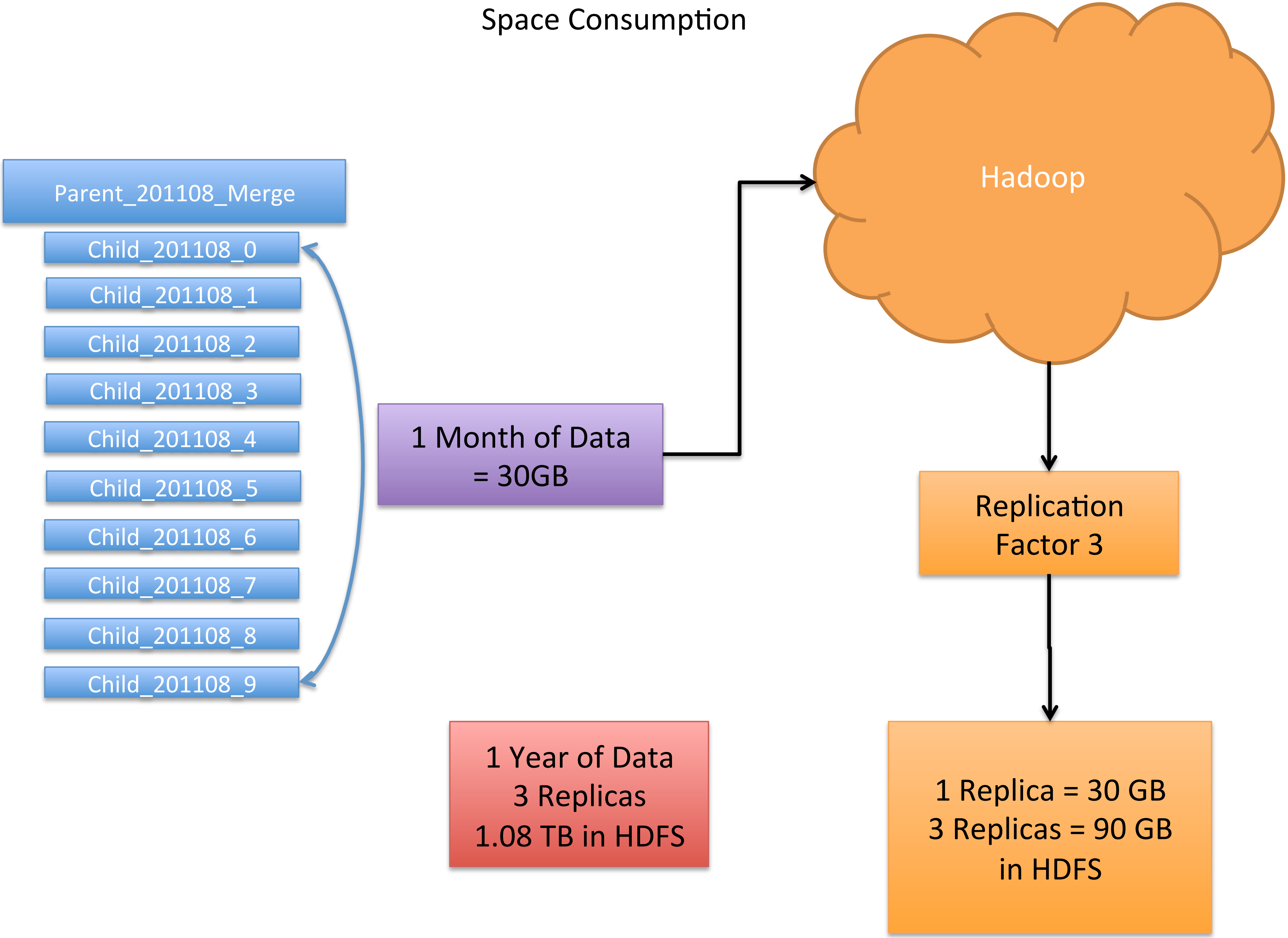
Sqoop

Parent_201108_Merge

Child_201108_0
Child_201108_1
Child_201108_2
Child_201108_3
Child_201108_4
Child_201108_5
Child_201108_6
Child_201108_7
Child_201108_8
Child_201108_9

As a result of sqooping into hive partitions only a minimal amount map task have to be processed.
1 Day = 10 Map Tasks
2 Days = 20 Map Tasks
…
30 Days = 300 Map Tasks

Space Consumption



Parent_201108_Merge

Child_201108_0

Child_201108_1

Child_201108_2

Child_201108_3

Child_201108_4

Child_201108_5

Child_201108_6

Child_201108_7

Child_201108_8

Child_201108_9

1 Month of Data
= 30GB

Hadoop

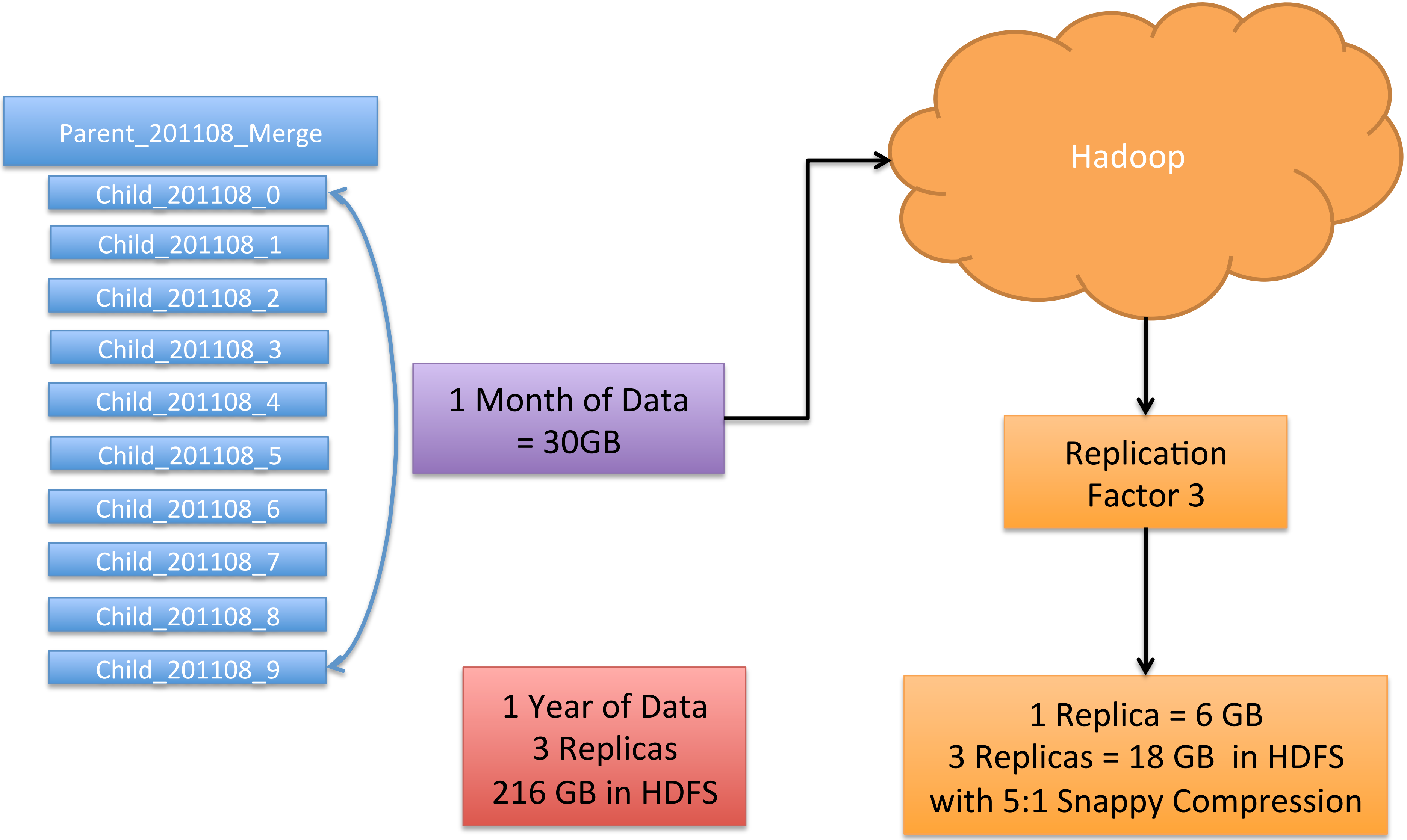
Replication
Factor 3

1 Replica = 30 GB
3 Replicas = 90 GB
in HDFS

1 Year of Data
3 Replicas
1.08 TB in HDFS

Sqooping with Snappy

```
sqoop import --compression-codec org.apache.hadoop.io.compress.SnappyCodec -z
```



Parent_201108_Merge

Child_201108_0

Child_201108_1

Child_201108_2

Child_201108_3

Child_201108_4

Child_201108_5

Child_201108_6

Child_201108_7

Child_201108_8

Child_201108_9

1 Month of Data
= 30GB

Hadoop

Replication
Factor 3

1 Replica = 6 GB
3 Replicas = 18 GB in HDFS
with 5:1 Snappy Compression

1 Year of Data
3 Replicas
216 GB in HDFS

Summary

1) Develop some kind of incremental import when sqooping in large active tables. If you do not, your sqoop jobs will take longer and longer as the data grows from the RDBMS.
2) Limit the amount of parts that will be stored in HDFS, this translates into time consuming map tasks, use partitioning if possible.
3) Compress data in HDFS. You will save space in HDFS as your replication factor makes multiple copies of your data. You may also benefit in processing as your Map/Reduce jobs have less data to transfer and hadoop becomes less I/O bound.

?