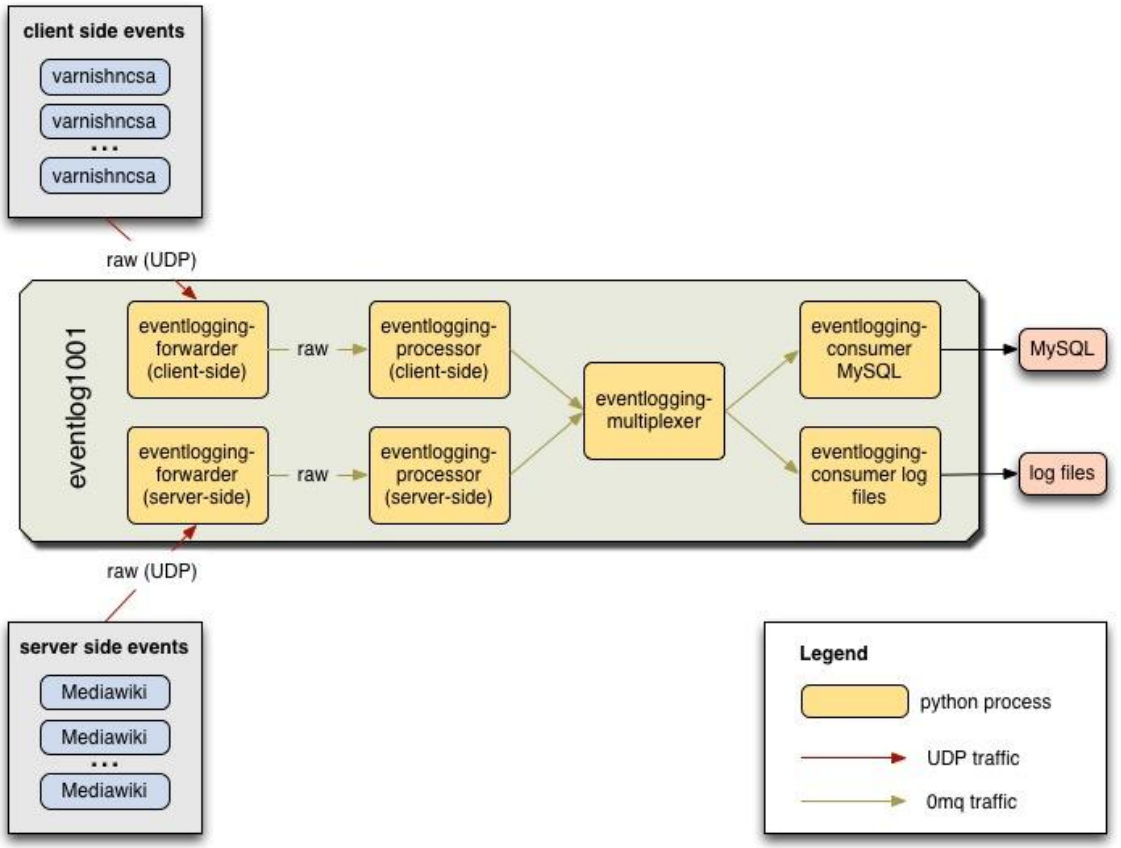
EventLogging

+

EventLogging on ZeroMQ

EventLogging was originally based on ZeroMQ.

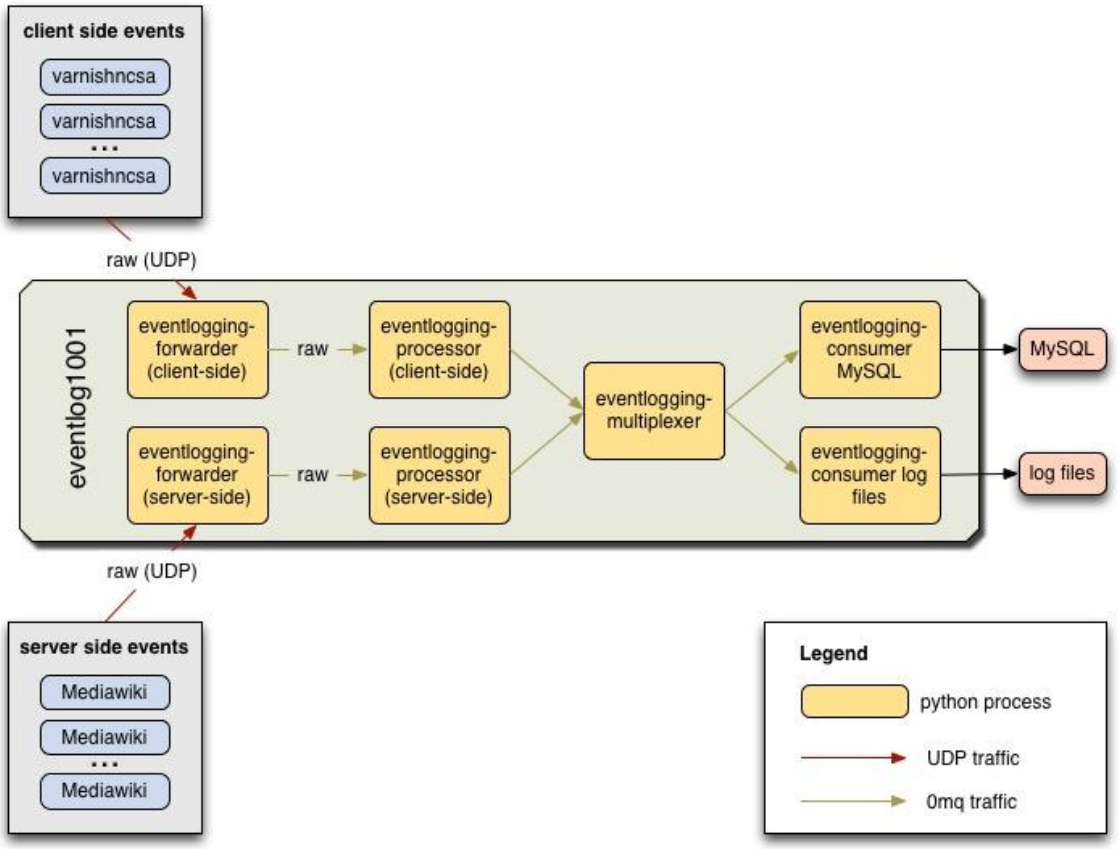


EventLogging on ZeroMQ

Worked great, but was **unreliable**.

Messages originate from potentially **lossy** UDP source.

No buffering. Restart/downtime loses messages. (SPOF)

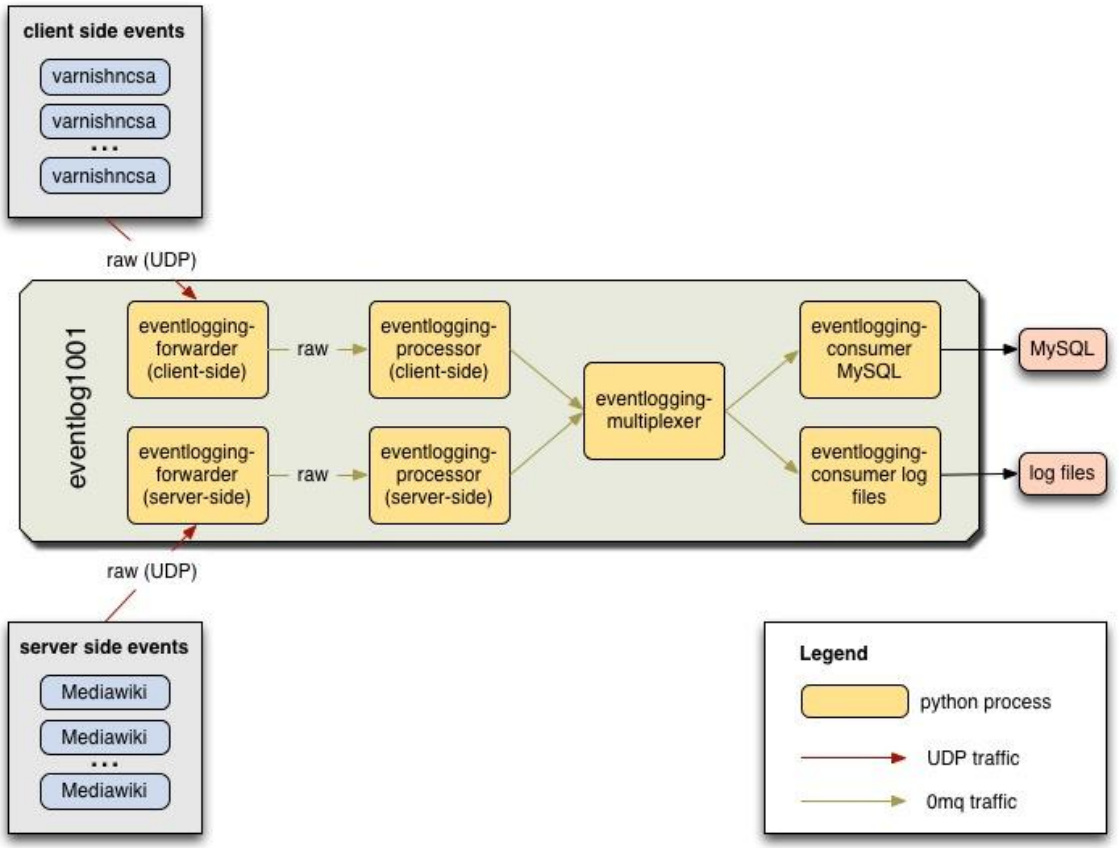


EventLogging on ZeroMQ

This system was **difficult to scale**.

Server runs many processes on **one node**. Difficult to scale. (SPOF)

Could only support 500-1000 messages / second.

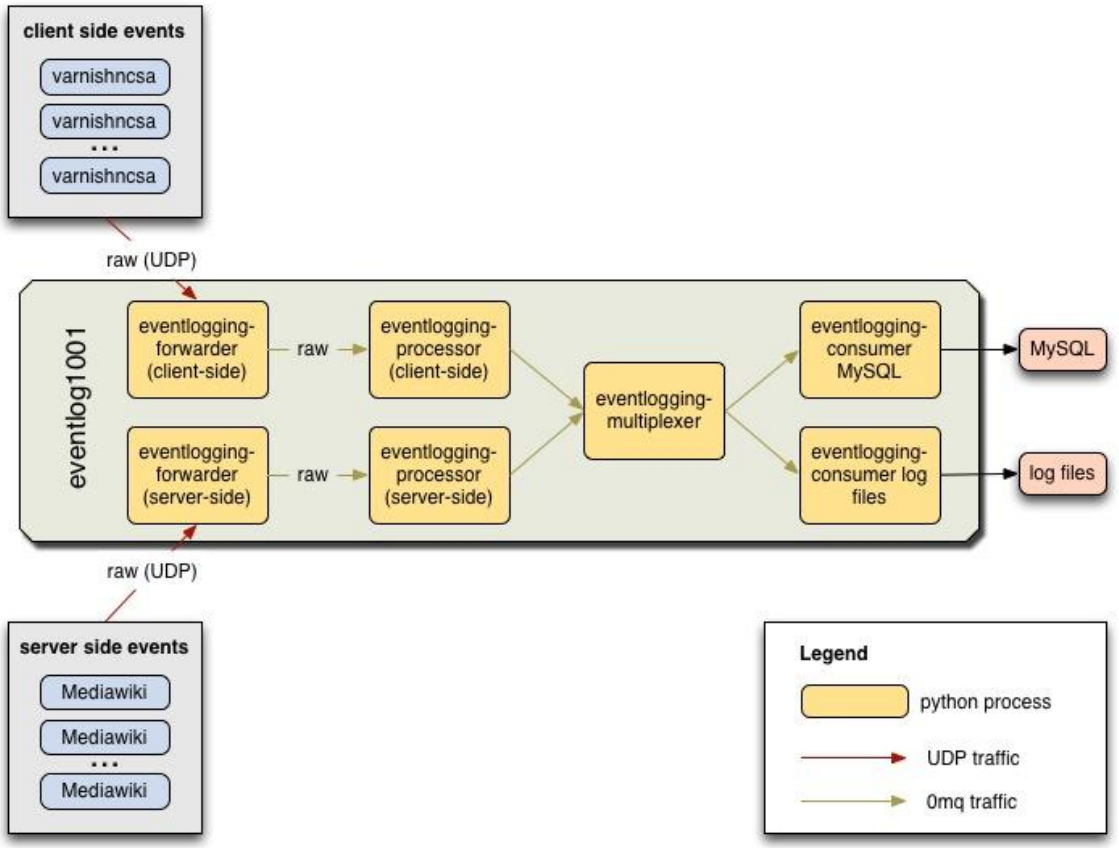


EventLogging on ZeroMQ

MySQL-only **analysis is painful** at scale

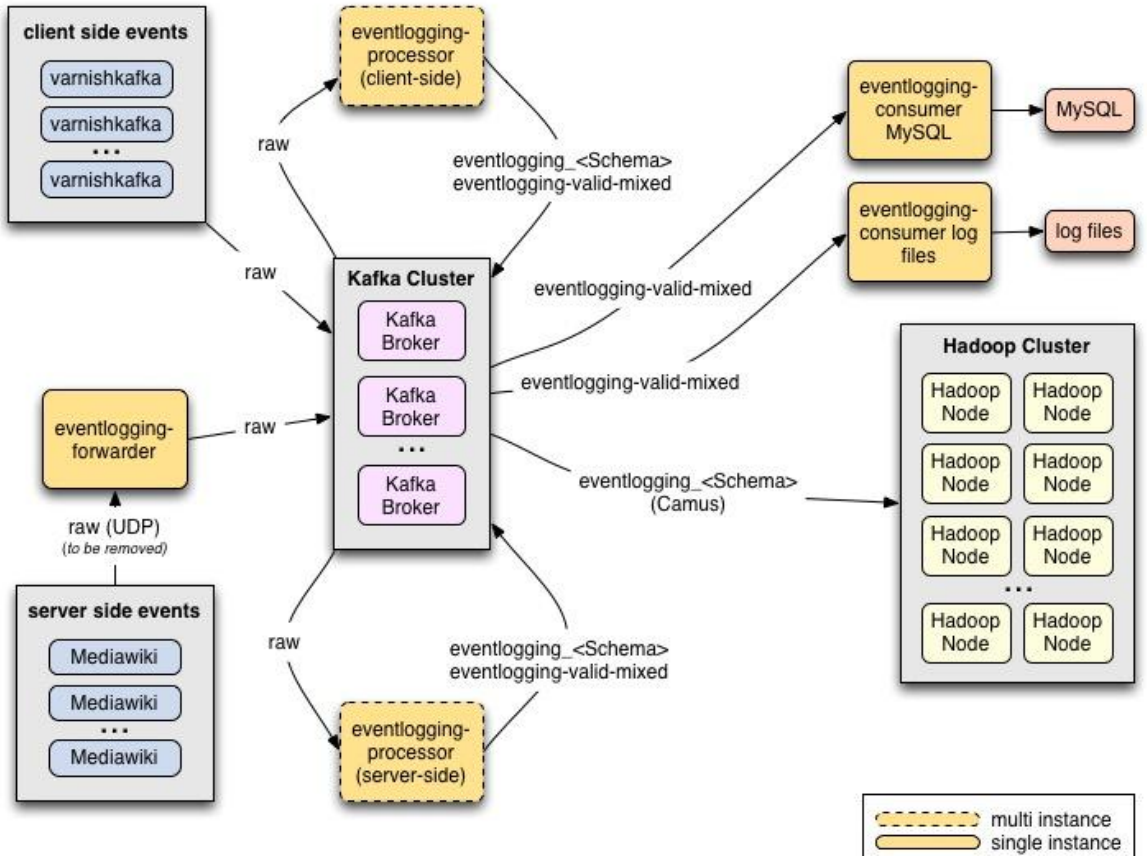
`Schema_Revision` named MySQL tables make for difficult historical analysis.

MySQL not good with **large historical data**.



EventLogging on Kafka

ZeroMQ has been replaced by Kafka.



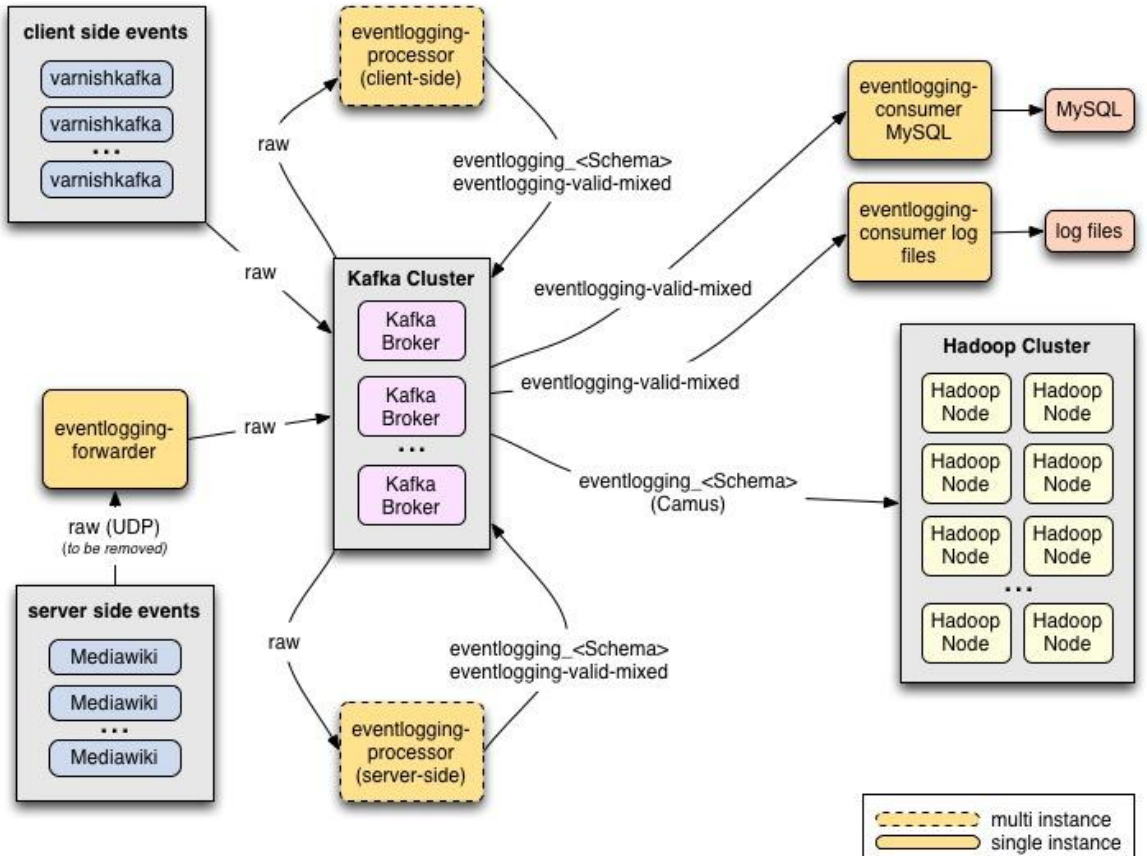
EventLogging on Kafka

Kafka for message transport is **more reliable.**

Messages are sent via TCP based Kafka protocol.

Kafka **buffers** all events for 7 days.

Consumer offsets are committed to Kafka, so restarts or downed processes lose minimal to no data.

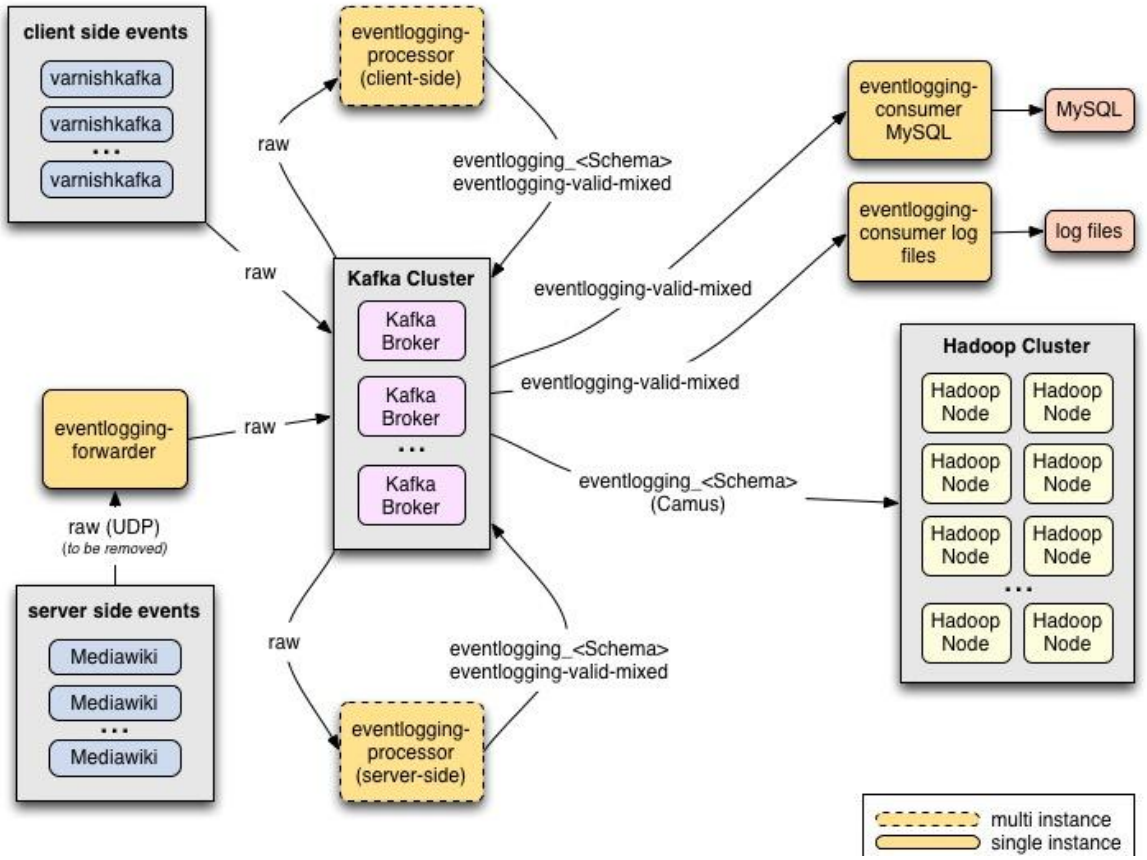


EventLogging on Kafka

Easier to **scale horizontally**.

7000-10000 messages / second on existing single server.

Now ~linearly **scalable**. Able to handle more messages by adding more hardware.

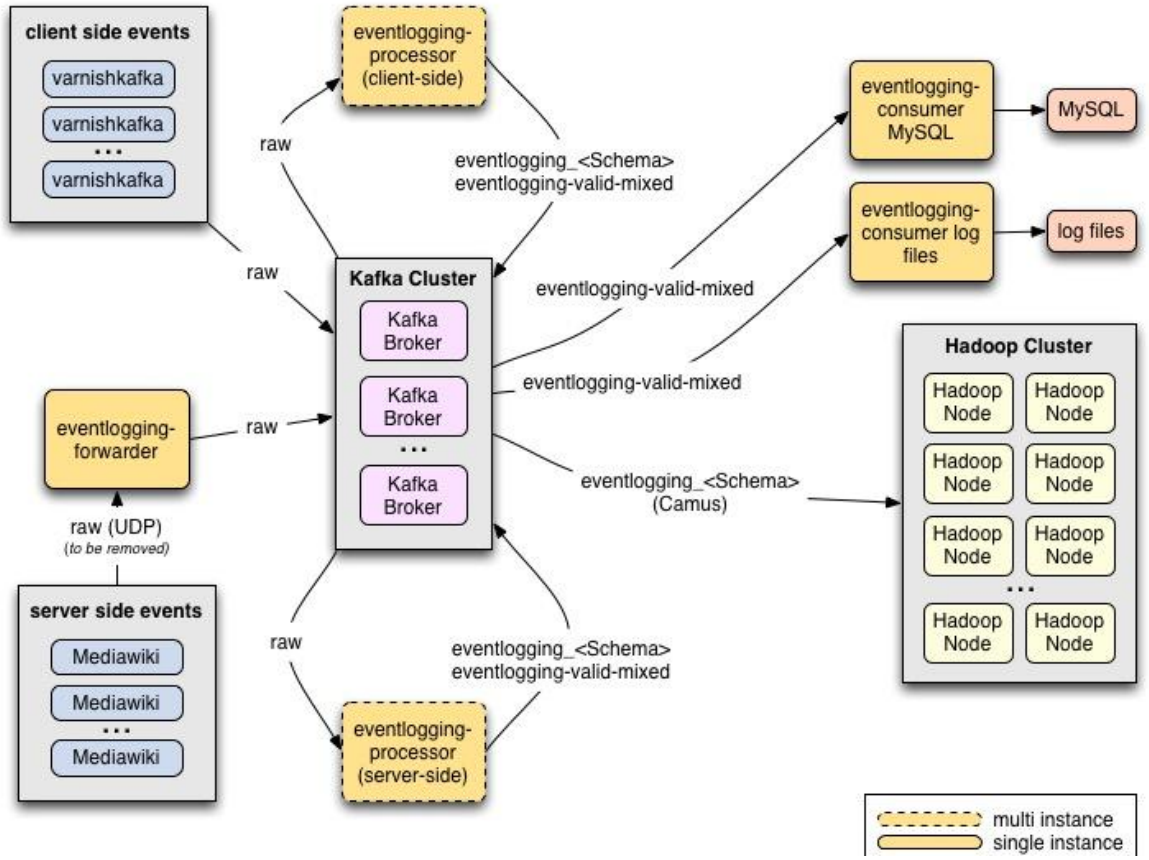


EventLogging on Kafka

Large data **analysis less painful** in Hadoop.

Events in Hadoop ease **cross schema-revision** processing.

Processing in Hadoop can handle way **more data** than MySQL.



EventLogging on Hadoop

EventLogging data is being imported into HDFS at

/wmf/data/raw/eventlogging/eventlogging_SCHEMA/hourly/YEAR/MONTH/DAY/HOUR

e.g.

/wmf/data/raw/eventlogging/eventlogging_Edit/hourly/2015/10/21/16

This is JSON stored in specially compressed files, just like raw webrequest data. View files using hdfs CLI:

```
hdfs dfs -text /wmf/data/raw/eventlogging/eventlogging_Edit/hourly/2015/10/21/16/*
```

EventLogging on Hadoop - Hive

```
ADD JAR file:///usr/lib/hive-hcatalog/share/hcatalog/hive-hcatalog-core.jar;
-- Make sure you don't create tables in the default Hive database.
USE otto;

-- Create a table with a single string field
CREATE EXTERNAL TABLE `Edit` (
  `json_string` string
)
PARTITIONED BY (
  year int,
  month int,
  day int,
  hour int
)
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.SequenceFileInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  '/wmf/data/raw/eventlogging/eventlogging_Edit';

-- Add a partition
ALTER TABLE Edit
  ADD PARTITION (year=2015, month=10, day=21, hour=16)
  LOCATION '/wmf/data/raw/eventlogging/eventlogging_Edit/hourly/2015/10/21/16';

-- Parse the single string field as JSON and get the 'wiki' field.
-- This gets the top 10 most edit wikis in 2015 Oct 21 16:00.
SELECT get_json_object(json_string, '$.wiki') AS wiki, count(*) as cnt
FROM Edit
WHERE year=2015 and month=10 and day=21 and hour=16
GROUP BY get_json_object(json_string, '$.wiki')
ORDER BY cnt DESC
LIMIT 10;
```

wiki	cnt
enwiki	32595
eswiki	12495
ruwiki	10774
jawiki	8669
itwiki	6675
frwiki	6228
ptwiki	5401
dewiki	5383
trwiki	4034
zhwiki	3813

EventLogging on Hadoop - Spark (Scala)

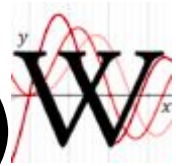
```scala
// Load the JSON string values out of the compressed sequence file
val edit_data = sc.sequenceFile[Long, String](
    "/wmf/data/raw/eventlogging/eventlogging_Edit/hourly/2015/10/21/16"
).map(_._2)

// parse the JSON strings into a DataFrame
val edits = sqlContext.jsonRDD(edit_data)
// Register this DataFrame as a temp table so we can use SparkSQL.
edits.registerTempTable("edits")

// SELECT top 10 edited wikis
val top_k_edits = sqlContext.sql(
    """SELECT wiki, count(*) AS cnt
    FROM edits
    GROUP BY wiki
    ORDER BY cnt DESC
    LIMIT 10"""
)
// Print them out
top_k_edits.foreach(println)
```

```
[enwiki,32595]
[eswiki,12495]
[ruwiki,10774]
[jawiki,8669]
[itwiki,6675]
[frwiki,6228]
[ptwiki,5401]
[dewiki,5383]
[trwiki,4034]
[zhwiki,3813]
```

EventLogging on Hadoop - Spark (Python)



```
# Load the JSON string values out of the compressed sequence file
edit_data = sc.sequenceFile(
    "/wmf/data/raw/eventlogging/eventlogging_Edit/hourly/2015/10/21/16"
).map(lambda x: x[1])

# parse the JSON strings into a DataFrame
edits = sqlCtx.jsonRDD(edit_data)
# Register this DataFrame as a temp table so we can use SparkSQL.
edits.registerTempTable("edits")

# SELECT top 10 edited wikis
top_k_edits = sqlCtx.sql(
"""SELECT wiki, count(*) AS cnt
    FROM edits
    GROUP BY wiki
    ORDER BY cnt DESC
    LIMIT 10"""
)
for r in top_k_edits.collect():
    print "%s: %s" % (r.wiki, r.cnt)
```

```
enwiki: 32595
eswiki: 12495
ruwiki: 10774
jawiki: 8669
itwiki: 6675
frwiki: 6228
ptwiki: 5401
dewiki: 5383
trwiki: 4034
zhwiki: 3813
```

EventLogging consumption from kafka

```
# Uses kafkacat CLI to print window ($1)
# seconds of data from $topic ($2)
function kafka_timed_subscribe {
    timeout $1 kafkacat -C -b kafka1012 -t $2
}

# Prints the top K most frequently
# occuring values from stdin.
function top_k {
    sort      |
    uniq -c   |
    sort -nr  |
    head -n $1
}

while true; do
    date; echo '-----------------------------'
    # Subscribe to eventlogging_Edit topic for 5 seconds
    kafka_timed_subscribe 5 eventlogging_Edit |
    # Filter for the "wiki" field
    jq .wiki |
    # Count the top 10 wikis that had the most edits
    top_k 10
    echo ''
done
```

(live demo)

Links

What is EventLogging?

https://www.mediawiki.org/wiki/Extension:EventLogging/Guide#What_is_EventLogging.3F

Top K Edits examples

https://gist.github.com/ottomata/16e51defbfe5d81bb822

EventLogging docs

https://wikitech.wikimedia.org/wiki/Analytics/EventLogging

EventLogging Throughput Dashboard

https://grafana.wikimedia.org/dashboard/db/eventlogging