Network DVR Meets Big Data

Stephen Kraiman
Solutions Architect, ARRIS
Introduction

Stephen Kraiman
Solutions Architect

ARRIS

• 2015 Revenue: 4.79B US$
• Projecting 2016 revenue of 6.6-6.8 billion US$.
• Worlds largest set-top box supplier
• 2016 acquired Pace Plc
• 2013 acquired Motorola Home assets
• Provides infrastructure for cable, telco, and satellite TV service providers
Agenda

• Copyright, Fair Use, and nDVR
• Problem Statement
• Why Hadoop (or My Journey Through the Land of Hadoop)
• Model/Modelling Environment
• Traffic Model and Networking Analysis
• nDVR Recorder Architecture and Optimizations
• Lessons Learned
Copyright, Fair Use, and nDVR

- **Copyright** is the legal right created by the law of a country that grants the creator of an original work exclusive rights for its use and distribution. (Wikipedia)
  - Duration – life + 70 years!
- **Fair Use** is the legal doctrine that permits limited use of copyrighted material without acquiring permission from the right user. (Wikipedia)
- **The “Betamax case”**
    - The broader consequence of the Supreme Court’s decision was the establishment of a general test for determining whether a device with copying or recording capabilities ran afoul of copyright law. (Wikipedia)
  - Cartoon Network, LP v. CSC Holdings, Inc., 536 F.3d 121 (2d Cir. 2008) (Cablevision)
    - **Home DVR** - Each user captures their own personal recording
    - **Network DVR** - Each user captures their own personal recording

*** Caveat – I am not a lawyer
Problem Statement

• With a set-top box based DVR the processing and content storage is fully distributed

• When moving to a network based DVR all of the processing and content storage is centralized

• New York city could have 1,000,000 nDVR subscribers

• Peak recording concurrency rate is 100%

• A single program typically generates almost 10 Mb per second

• During peak recording time **1,000 Gbps** needs to be written to the storage system

• Typical network DVR recorders have integrated data fan out and storage
  – Strength: keeps all of the traffic local to the bus of the recorder
  – Weakness: The recorder is a single point of failure

• **The Solution:** Separate the recorder and storage. Use a high availability storage system
  – Challenges:
    • Fan out of recordings
    • Overhead of replication
## Recorder Architecture

<table>
<thead>
<tr>
<th>Integrated Recorder and Storage</th>
<th>Network Load</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>Stream</td>
<td>• Fan out takes place across local machine</td>
<td>• Single point of failure</td>
</tr>
</tbody>
</table>

![Diagram](image.png)
Why Hadoop?

Hadoop is the platform that natively supports resilient distributed data storage and computing.

Components of interest:
- HDFS
- Apache Storm
- Apache Kafka
Kafka is a pub-sub messaging system that is:

- It is durable. Messages are persisted on disk to prevent data loss
- It is distributed
- It is scalable
- It provides a rich set of accessed modes

- Message feeds are called Topics
- Messages are published by a Producer to a Topic
- Messages for a Topic are split into Partitions and appended to a commit log
- Each Partition contains an ordered immutable sequence of messages
- Published messages are retained for a period of time regardless of whether they have been consumed
- Consumers can get the messages in a queuing or pub-sub model
Apache Storm

• Apache Storm is a distributed real-time computation system
  – It reliably processes unbounded streams of data, doing for real-time processing what MapReduce did for batch processing

• A Storm “application” is superficially similar to a MapReduce application
  – A MapReduce job eventually finishes
  – Storm provides “topologies” - A topology processes messages forever (or until you kill it)
Storm Topology

• A Storm topology consists of Spouts and Bolts

• Spouts
  – Accept data from an external source. For example:
    • Kafka or Kestral queue
    • Twitter feed
  – Pass data to 0 or more bolts
  – Sophisticated rules to control how the data is emitted

• Bolts connect to a spout
  – Bolts process or transform the data, and may emit it
  – A bolt can connect to another bolt (chaining)
HDFS, MapReduce, Yarn

• **HDFS (Hadoop Distributed File System)**
  - High performance, high availability distributed file system
  - Based upon a white paper from Google titled “The Google File System.” The system is optimized to:
    - Store VERY large files (100’s of megabytes, gigabytes, to terabytes.
    - Write once and read many times
    - Append to files only
    - Run on commodity hardware
  - It is not good for
    - Low latency data access
    - Lots of small files
    - Multiple writers, random writes

• **Key Concepts**
  - Files are broken into blocks; default is 64 MB
  - Blocks are replicated across storage nodes; default replication value is 3
  - When a block fails, Hadoop automatically creates a new replica
  - There is a concept location. Local, in rack, in data center, remote

• **Summary**
  - High availability distributed file storage system
  - Uses simple replication for redundancy
Goals

• Need to have repeatable tests and results
• Need to be able to run the tests and gather the results autonomously
• It needed to be cheap

Solution?

• Dedicated cluster was not an option
  – Cost
  – Administrative challenges
• What about the cloud?
  – Used Google Cloud Platform
• Why Google Cloud Platform
  – It was really easy!
    https://cloud.google.com/hadoop/bdutil
What is the Cost?

- It takes 15 minutes to:
  - Create the cluster
  - Download git and other tools
  - Pull the code from git and build it
- Each test cases task ~30 minutes to run
- There are 6 use cases in the test suite
- Virtual Machine Config:
  - 4 virtual CPUs
  - 15GB RAM
  - 10GB + 100GB HDs
- Cost per Machine
  - CPU: $0.20/HR (range is $0.060 to $0.20 depending on usage parameters)
  - Storage: $0.04/GB per month. 110GB/Hr is $0.00611!

- 5 nodes + master = 6 machines
- Cost is $1.24 per hour
- Test runs takes 3.25 hours
- Total cost per run is 4.03 US$
The Development Cycle

Toolset

- Git and GitHub
  - Manages source code. Also eases code builds in a distributed environment
  - Manages the scripts to create and destroy the Google Cloud execution environment

- Apache Maven for builds

- Eclipse is the IDE
nDVR Recorder Architecture

MPD VideoChunks

DASH Client

Kafka Broker

Kafka Broker

Kafka Broker

Kafka Topic Feed

Kafka Topic Feed

Kafka Topic Feed

Kafka Topic Feed

Storm Cluster

Hadoop Name Node

Hadoop Data Node

Recordings

Storm nDVRRecorder

Hadoop Data Node

Recordings

Storm nDVRRecorder

Hadoop Data Node

Recordings

Storm nDVRRecorder

Hadoop Data Node

Recordings

Storm nDVRRecorder

Hadoop Data Node

Recordings
Use Cases

- USE_CASE_NAME="cdvr-2mxr1xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=1
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-2mxr10xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=10
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-2mxr50xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=50
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-2mxr1xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=1
  - NUM_EXECUTORS=5

- USE_CASE_NAME="cdvr-2mxr10xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=10
  - NUM_EXECUTORS=5

- USE_CASE_NAME="cdvr-2mxr50xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=50
  - NUM_EXECUTORS=5

- USE_CASE_NAME="cdvr-5par2mxr1xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=1
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-5par2mxr10xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=10
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-5par2mxr50xe1"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=50
  - NUM_EXECUTORS=1

- USE_CASE_NAME="cdvr-5par2mxr1xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=1
  - NUM_EXECUTORS=5

- USE_CASE_NAME="cdvr-5par2mxr10xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=10
  - NUM_EXECUTORS=5

- USE_CASE_NAME="cdvr-5par2mxr50xe5"
  - BYTES_PER_2_SEC=2000000
  - NUM_RECORDINGS=50
  - NUM_EXECUTORS=5
Use Case: Three Replicas

- Use case: cdvr-5par2mxr10e1.pdf
- Bps = 1,000,000
- nUsers = 10
- nReplicas = 3
The Traffic Model

\[ \text{Traffic}_{\text{in}} \approx \text{Traffic}_{\text{out}} \approx \text{MBps} \times n\text{Users} \times (n\text{Replicas} - 1) \]

**Example #1**  MBps = 1, nUsers = 10, nReplicas = 3
- \( 1 \times 10 \times (3 - 1) = 20 \text{ MBps} \)

**Analysis**
- The simulated source runs on hadoop-m
- Kafka runs on hadoop-m
- Storm topology runs on all nodes
  - The system seems to route all of the data to the Storm instance on hadoop-m
- Hadoop-m, and hadoop-w-0 through hadoop-w-4 host HDFS
  - It appears as though all of the data is written to HDFS on hadoop-m and then replicated across the other nodes
# Recorder Architectures (HDFS)

## Network Load

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Stream</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Recorder and Storage</td>
<td>Stream</td>
<td>• Fan out takes place across local machine</td>
<td>• Single point of failure</td>
</tr>
<tr>
<td>Recorder + HDFS</td>
<td>Stream + (stream * nUsers * nreplicas – 1)</td>
<td>• High availability • Fan out occurs on local machine</td>
<td>• Replication occurs over the network • Uses more storage</td>
</tr>
</tbody>
</table>
DVR Recording Usage from April 2013

- Note that the daily recording peak occurs between 7PM and 10PM.
Recorder Architectures (HDFS w/Single Replica)

<table>
<thead>
<tr>
<th>Network Load</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Recorder and Storage</td>
<td>Stream</td>
<td>• Fan out takes place across local machine</td>
</tr>
<tr>
<td>Recorder + HDFS</td>
<td>Stream + (stream * nUsers * nreplicas - 1)</td>
<td>• High availability</td>
</tr>
<tr>
<td>Recorder + HDFS w/Single replica</td>
<td>Stream</td>
<td>• Fan out occurs on local machine</td>
</tr>
</tbody>
</table>
# Recorder Architectures (HDFS w/Single Replica)

<table>
<thead>
<tr>
<th>Recorder Architectures (HDFS w/Single Replica)</th>
<th>Network Load</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Recorder and Storage</td>
<td>Stream</td>
<td>• Fan out takes place across local machine</td>
<td>• Single point of failure</td>
</tr>
<tr>
<td>Recorder + HDFS</td>
<td>Stream + (stream * nUsers * nreplicas – 1)</td>
<td>• High availability • Fan out occurs on local machine</td>
<td>• Replication occurs over the network • Uses more storage</td>
</tr>
<tr>
<td>Recorder + HDFS w/Single replica</td>
<td>Stream</td>
<td>• Fan out occurs on local machine</td>
<td>• Single point of failure</td>
</tr>
<tr>
<td>Recorder + HDFS w/controlled Replication</td>
<td>(stream * nUsers * nreplicas – 1)</td>
<td>• High availability most of the time • Fan out occurs on local machine • Replication occurs on demand (after peak load)</td>
<td>• Single point of failure until replicated • Uses more storage</td>
</tr>
</tbody>
</table>
Use Case: One Replica w/Deferred Replication

- Use case: cdvr-2mxr10xe1-repl-1-to-3.pdf
- Bps = 1,000,000
- nUsers = 10
- nReplicas = 1
- Two minutes after recording completes change to nReplicas to 3
Lessons Learned
Use Case Architecture (Theoretical)

- Feed Simulator
- Kafka Broker (hadoop-m)
  - Kafka Topic Feed
    - Kafka Partitions 1 or 5
      - Kafka Topic Feed
        - Storm Cluster
          - Storm nDVR Recorder (hadoop-m)
            - Hadoop Name Node
              - Recordings
        - Kafka Topic Feed
          - Storm nDVR Recorder (hadoop-w-0)
            - Hadoop Data Node
              - Recordings
        - Kafka Topic Feed
          - Storm nDVR Recorder (hadoop-w-1)
            - Hadoop Data Node
              - Recordings
        - Kafka Topic Feed
          - Storm nDVR Recorder (hadoop-w-2)
            - Hadoop Data Node
              - Recordings
        - Kafka Topic Feed
          - Storm nDVR Recorder (hadoop-w-3)
            - Hadoop Data Node
              - Recordings
        - Kafka Topic Feed
          - Storm nDVR Recorder (hadoop-w-4)
            - Hadoop Data Node
              - Recordings

Copyright 2016 – ARRIS Group, Inc. All rights reserved.
Use Case Architecture (Observed)

Feed Simulator -> Kafka Broker (hadoop-m) -> Kafka Topic Feed

Kafka Partitions 1 or 5

Storm Cluster

Storm nDVR Recorder (hadoop-m)

Hadoop Name Node

Hadoop Data Node

Recordings

Storm nDVR Recorder (hadoop-w-0)

Hadoop Data Node

Recordings

Storm nDVR Recorder (hadoop-w-1)

Hadoop Data Node

Recordings

Storm nDVR Recorder (hadoop-w-2)

Hadoop Data Node

Recordings

Storm nDVR Recorder (hadoop-w-3)

Hadoop Data Node

Recordings

Storm nDVR Recorder (hadoop-w-4)

Hadoop Data Node

Recordings

Hadoop Data Node

Recordings

Hadoop Data Node

Recordings

Hadoop Data Node

Recordings
Lessons Learned

- Racers, start your engines: The velocity of change is overwhelming
- Most examples are out of date
- Maven is your frenemy
- Much of the documentation is incomplete or wrong
Thanks!

Stephen Kraiman

Solutions Architect @ ARRIS
skraiman@arris.com
https://www.linkedin.com/in/stephenkraiman