Logging and Analytics Overview
How Much Data

- **Syndication Partners**
  - 3.3 Billion Records/Day (June 2018)
  - Avg ~900 Bytes/Message for TC
  - Resulting in ~970GB/Day Raw

- **Comcast - OTT**
  - 11.7 Billion Records/Day (June 2018)
  - Avg ~700 Bytes/Message for NetStats
  - Resulting in ~3.5TB/Day Raw

- **Comcast - Title VI**
  - 60 Billion Records/Day (June 2018)
  - Avg ~700 Bytes/Message for NetStats
  - Resulting in ~17.9TB/Day Raw
What Has Been Built
The Tech

- Filebeat
  - Grabbing and forwarding of logs
- Kafka
  - Aggregating and Queing of logs
- Spark
  - Parsing, Transforming, Enhancement, and delivery of messages
- ElasticSearch
  - Log search and statistics gathering
What is TrafficLogs

- Think Logstash but better
- Uses Spark for scalability
  - Attempted Beam, more about that later
- Yaml based configs
  - Pipeline config
  - Complex parse, transform, and enrichment tree designs
- Any source
  - Kafka only currently
- Any destination
  - ES, Kafka, S3 currently available
The Challenges So Far
Apache Beam

● Issues
  ○ No support for backpressure
  ○ No support for Kafka offset tracking outside of filesystem
  ○ Introduced innate slowness due to limitations of Spark integration and workflow optimizations
  ○ Adaption layer resulted in re-deserialization and instantiation for each batch
  ○ Beam wasn’t mature enough

● Solution
  ○ Replaced Beam with direct Spark SDKs
ElasticSearch using Ceph Block Storage

- **Issues**
  - Slow to read/write.
  - Lots of congestion from neighbors and ourselves
  - Added 2nd layer of replication (3x) outside of ElasticSearch’s own

- **Solution**
  - Moved to lots VMs (120) using ephemeral storage
Spark to ElasticSearch

- **Issues**
  - Constant Flushing to Disk
  - Slow Spark batch times even when no data

- **Solutions**
  - Increased batch send limits
  - Disabled refresh after each batch submit (batch.write.refresh=false)
  - Node state improvements
    - Currently refreshed on every Task that writes to ES
    - More ES nodes means more nodes to get state about, increasing
    - Working with Elastic on improvements to allow caching and background fetching of node state
  - Failed events stay in same ES writer as successful ones
ElasticSearch Index Performance

● Issues
  ○ Constant Flushing to Disk
  ○ High CPU load

● Solutions
  ○ Field map all the things
  ■ Templates are your friend
  ■ Use primitives
  ■ Only use Keyword indexing where FullText isn’t needed
    ● Default string type indexing results in FullText and Keyword being applied
  ○ Increase Shards and Decrease replication
    ■ Use Curator to increase replication later
  ○ Moving to ephemeral helped greatly here as well
S3 Uploading

● Issues
  ○ Minimum File Size for Multipart Upload (5MB/part)
  ○ Max File Size for Single part Upload (5GB)
  ○ Built-In File System support via Hadoop creates tons of very small files

● Solutions
  ○ LZO Compression done on all executores
  ○ Aggregate LZO parts within each batch to 1 task for upload
  ○ Use S3 SDK directly
Future Plans
Central User Interface

- Combined Portal
  - Kibana
  - Turnilo
    - Formulary called Swiv
    - Opensource fork of Pivot from Imply
  - Grafana
  - Zeppelin
  - ElasticHQ
  - KafkaManager
  - Potentially Superset

- Spark Management
  - Job Management
  - Job Status
    - Active Date Ranges
    - Active Offset Ranges
    - Backlog

- Data Availability
- Elastic Curator Management
- Parser Config Generator
Report Generation

Growth Forecasting

- Infrastructure per mille requests
- Data Volume per mille requests
- Bandwidth forecasting
- Arbitrary Aggregation Grouping
  - Per Device
  - Per Customer

Tenant Usage & Billing

- Per CDN, Tenant, Service, and Delivery Service
- General Usage metrics
  - Bandwidth
  - Requests at each layer
- Billing
  - Multiple billing points
# Filebeat Replacement

<table>
<thead>
<tr>
<th>Filebeat</th>
<th>LogForwarder</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Uses Sarama for Kafka Client</td>
<td>● Written in GO</td>
</tr>
<tr>
<td>● Small broker outages would pause sending completely until all brokers where online</td>
<td>● Uses LibRDKafka</td>
</tr>
<tr>
<td>● No and/or limited control on throughput</td>
<td>○ Supported by Confluence</td>
</tr>
<tr>
<td></td>
<td>○ Better handling of broker outages</td>
</tr>
<tr>
<td></td>
<td>● Support for pipe and file based logs</td>
</tr>
</tbody>
</table>
Apache Druid

- Rolled up stats as opposed to individual search records
  - Faster queries
  - Retain longer windows of time
  - Query much larger windows of time
  - Power many dashboards and monitoring systems
- Allows you to pivot data in multiple dimensions. Allowing you to see more and do more with your data.
- Opens up more possibilities within reporting, managing, and day to day operations