Gaining efficiency with time series in ELK.

Velocity San Jose
THE DISTRIBUTED REAL TIME TELEMETRY CHALLENGE.
The Problem

- Globally distributed infrastructure spanning bare metal systems, hosting providers, and third party integrations
- Dynamically scales in both directions
- Emits logs, time series data, packet captures, sflow, etc
- Nominal throughput of roughly 200k data points per second
- Spikes reaching 5-700k data points per second
- Data must be collected and analyzed in as close to real time as possible
- Need fine grained control over what data to keep or reduce and when that happens
Initial Solution

OpenTSDB cluster setup on thirteen servers.
Used Tcollector for handling collection of:
  • Server and OS metrics
  • Application metrics

Elasticsearch cluster setup on three servers.
Used Rsyslogd + Logstash for handling collection of:
  • Log data
  • General unstructured telemetry
Worked great... Until it didn’t

- External scaling pressure caused large increases in throughput and general telemetry load on systems that were not easy to scale.
- External scaling pressure + increased organizational need for telemetry data compounded problems.
- Our telemetry systems started requiring more time to manage than the infrastructure it was intended to monitor.
Elasticsearch
- Large diverse community
- Suite of management tools for the cluster and data
- Easier to manage overall cluster
- Easy to manage data retention
- Easy to scale

OpenTSDB
- Prolific but polarized community
- Powerful but very complex tools
- Difficult to manage overall cluster
- Difficult to manage data retention
- Difficult to scale

A Glimmer of Hope
THE DECISION TO REFACTOR.
• Requirements

• One single telemetry system to handle all collected data
• Capable of granularly determining what data to remove or reduce and when to do so
• Easily scale to meet demands as the infrastructure it monitors changes
• Has a large diverse community for support
• Offers enough analysis capabilities to satisfy our requirements
The comparison

Elasticsearch
- Statistical Analysis ✔
- Designed for scaling ✔
- Handles any data type ✔
- Has Rollups* ━
- Can remove data based on time or content ✔
- Large diverse community ✔

OpenTSDB
- Statistical Analysis ✔
- Designed for scaling ━
- Handles any data type ✘
- Has Map-Reduce ✔
- Can remove data based on time or content* ━
- Large diverse community ━
• The Choice

• We ended up going all in on Elasticsearch and the Elastic stack in general
• Specifically picked up the beats suite
  • metricbeat
  • filebeat
  • packetbeat
• Leveraged the well known Kibana UI and Logstash data processor
THE ELASTIC ADVANTAGE.
Community

- Large and diverse community
- Active discussion boards like discuss.elastic.co
- Tons of custom open source tooling like elastalert
- Vast array of Elastic sponsored plugins and applications
  - Logstash/Kibana/Beats
  - Curator
  - Vega/Canvas
Clustering

- Automatic and manual clustering options
  - Multicast
  - Unicast
- Robust fault tolerance
- Automatic failure resolution
- Zero downtime maintenance and upgrades
- Scaling both up and down can be done with ease*
Data Introspection

- Scripted queries, fields, and aggregations
- Query for ip, numeric, and time ranges
- Combine related and unrelated telemetry
- Robust text matching from exact match and token matches to full regex support

```python
query_string: "ipv4:10.0.0.0/8 AND cores:[4 TO 12] AND request:'v1/api/zones/*" 
```
Data Structure Support

- Can handle parent/child relationships
- Full IP address support
- Geo location data types
- Array data types
- Rich document structure, meaning no more line data format

```json
{
    "@timestamp": 1528200457,
    "load": {
        "1m": 0.12,
        "5m": 0.09,
        "15m": 0.08
    },
    "cores": 4,
    "host": "foo.example.com",
    "ipv4": "10.0.0.1"
}
```

VS.

```system
system.load.1m 1528200457 0.12 host=foo.example.com ipv4=10.0.0.1
system.load.5m 1528200457 0.09 host=foo.example.com ipv4=10.0.0.1
system.load.15m 1528200457 0.08 host=foo.example.com ipv4=10.0.0.1
system.load.cores 1528200457 4 host=foo.example.com ipv4=10.0.0.1
```
WHERE ELASTIC FALLS SHORT.
• **Monotonic Counters**

  - No graceful handling of counter resets
  - No ability to do a rate calculation on the raw counter value
  - Causes false positive alerts
  - Causes spikes/dips on visualizations
Downsampling during a query is the act of “pre”aggregating pieces of the individual matched time series together before aggregating the various matched time series into one series.

- Elasticsearch only supports bucketing.
- Can cause skewed results if not planned for.
- Breaks certain types of aggregation regardless of planning.
Learning Curve

- Tough initial operational learning curve
  - Need to worry about index and shard sizing
  - Mappings are critical but complex
    - A lot of knobs to turn, thankfully mostly well documented
- Complex query language is extremely powerful but cumbersome to get used to
- Scripting can be a performance trap if not well monitored
- Only saving grace is one query language to understand to for all telemetry
THE MOVE.

STRUGGLES AND TRIUMPHS
The Struggles

- Index and shard sizing
  - Keep index and shard counts low, even when not in active use they require non-negligible resources to maintain
  - Keep each shard of an index roughly around 50GB for best throughput
- Leading wild card searches
  - These are incredibly heavy queries on elasticsearch, disable them if at all possible
  - Add “indices.query.query_string.allowLeadingWildcard: false” to your elasticsearch.yml
- Index Mappings
  - Ensure the fields of your documents are properly managed, and avoid full “text” fields whenever possible
- Data structure matters
  - Documents should have a reasonable amount of fields, but attempt to reduce unique documents in favor of larger documents
• **The Triumphs**

  • Biggest triumph is a single cluster to manage
    • Saves the operations team money and time
  • Ability to combine multiple unrelated metrics in a single query to bring about new insights
    • For instance CPU metrics related directly to network utilization in the same query
  • Single query language to analyze application logs and the metrics that those same applications emit
• Conclusion

• Effective solution for NS1
• Elasticsearch is great if your team lacks deep understanding of hadoop and things like hbase/hdfs/zookeeper
• General solution coming with the standard problems general solutions have
  • Can’t beat specialized solutions
  • Gets you about 98% of the way there
• Comes down to use case
• Would do it again but likely attack the problem differently
THANK YOU.