OpenTSDB

2.4 and 3.0 Update

YAHOO!

Who Am I?

Chris Larsen

- Maintainer and author for OpenTSDB since 2013
- Software Engineer @ Yahoo
- Central Monitoring Team

Who I’m not:

- A marketer
- A sales person

What Is OpenTSDB?

- Open Source Time Series Database
- Scales to 10s of millions of writes per second
- Sucks up ***all*** data and keeps going
- Never lose precision (if you have space)
- Scales using HBase or Bigtable

What are Time Series?

- **Time Series**: A sequence of discrete data points (values) ordered and indexed by time associated with an identity.

E.g.:

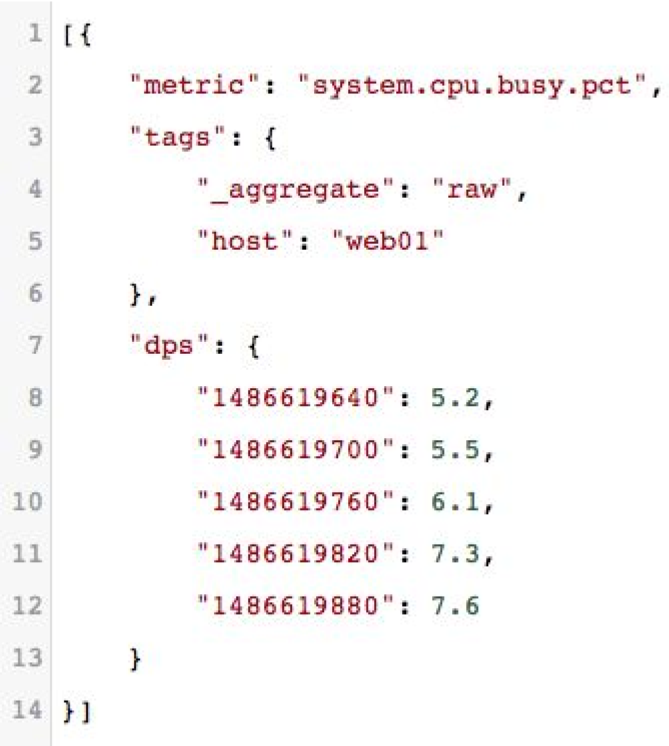
web01.sys.cpu.busy.pct 45% 1/1/207 12:01:00

web01.sys.cpu.busy.pct 52% 1/1/207 12:02:00

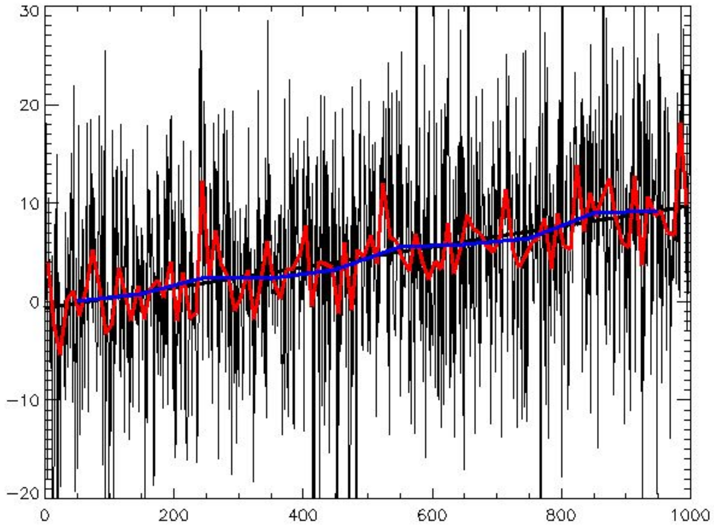
web01.sys.cpu.busy.pct 35% 1/1/207 12:03:00

^ Identity ^ Value ^ Timestamp

What are Time Series?



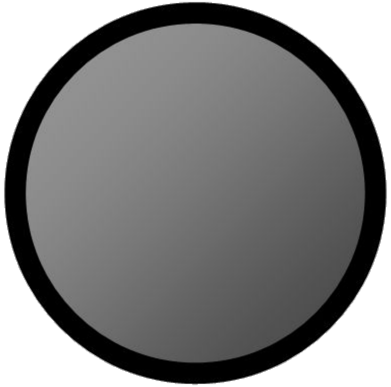
```
[{
    "metric": "system.cpu.busy.pct",
    "tags": {
        "_aggregate": "raw",
        "host": "web01"
    },
    "dps": {
        "1486619640": 5.2,
        "1486619700": 5.5,
        "1486619760": 6.1,
        "1486619820": 7.3,
        "1486619880": 7.6
    }
}]
```



What are Time Series?

Data Point:

- Metric + Tags
- + Value: 42
- + Timestamp: 1234567890



^ a data point ^

sys.cpu.user 1234567890 42 host=web01 cpu=0

- Payload could also be a string, a blob, a histogram, etc.

What are HBase and Bigtable?

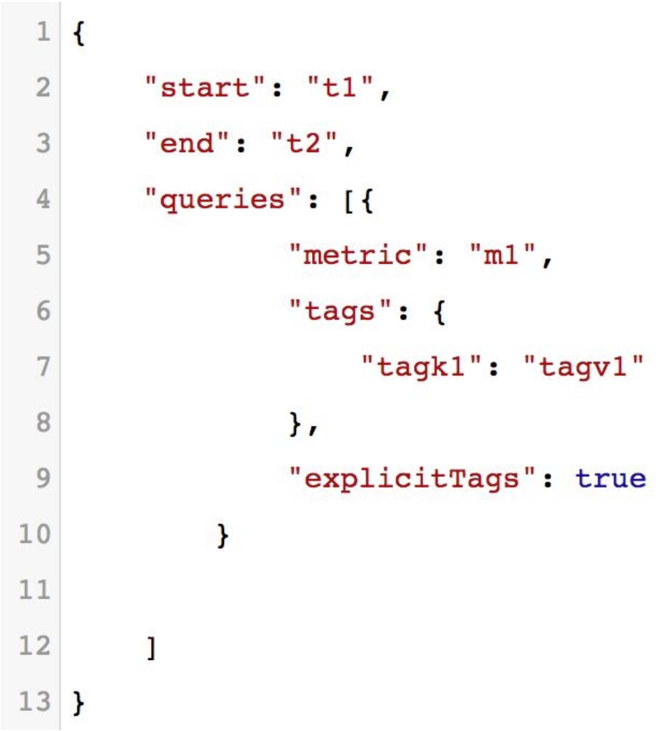
- HBase is an OSS distributed LSM backed hash table based on Google’s Bigtable.
- Key value, row based column store.
- Sorted by row, columns and cell versions.
- Supports:
  - Scans across rows with filters.
  - Get specific row and/or columns.
  - Atomic operations.
- CP from CAP theorem.

OpenTSDB Schema

- Row key is a concatenation of UIDs and time:
  - salt + metric + timestamp + tagk1 + tagv1… + tagkN + tagvN
- sys.cpu.user 1234567890 42 host=web01 cpu=0
  \x01\x00\x00\x01\x49\x95\xFB\x70\x00\x00\x01\x00\x00\x01\x00\x00\x02\x00\x00\x02
- Timestamp normalized on hour or daily boundaries.
- All data points for an hour or day are stored in one row.
- Data: VLE 64 bit signed integers or single/double precision signed floats, Strings and raw histograms.
- Saves storage space but requires UID conversion.

OpenTSDB Schema

Row Key	Columns (qualifier/value)
m t1 tagk1 tagv1	o1/v1 o2/v2 o3/v3
m t1 tagk1 tagv2	o1/v1 o2/v2
m t1 tagk1 tagv1 tagk2 tagv3	o1/v1 o2/v2 o3/v3
m t1 tagk1 tagv2 tagk2 tagv4	o1/v1 o3/v3
m t1 tagk3 tagv5	o1/v1 o2/v2 o3/v3
m t1 tagk3 tagv6	o2/v2
m t2 tagk1 tagv1	o1/v1 o3/v3
m t2 tagk1 tagv2	o1/v1 o2/v2



```
{
    "start": "t1",
    "end": "t2",
    "queries": [{
            "metric": "m1",
            "tags": {
                "tagk1": "tagv1"
            },
            "explicitTags": true
        }

    ]
}
```

OpenTSDB Use Cases

- Backing store for Argus: Open source monitoring and alerting system.
- 50M writes per minute.
- ~4M writes per TSD per minute.
- 23k queries per minute.
- https://github.com/salesforce/Argus

OpenTSDB Use Cases

- Monitoring system, network and application performance and statistics.
- Single cluster: 10M to 18M writes/s ~ 3PB.
- Multi-tenant and Kerberos secure HBase.
- ~200k writes per second per TSD.
- Central monitoring for all Yahoo properties.
- Over 1 billion active time series served.
- Leading committer to OpenTSDB.

Other Users

EMC^2®

StackExchange

Bloomberg

Pinterest

pepperdata®

box

Limelight NETWORKS

ticketmaster®

tumblr.

rocketfuel

Artificial intelligence. Real results.

ebay™

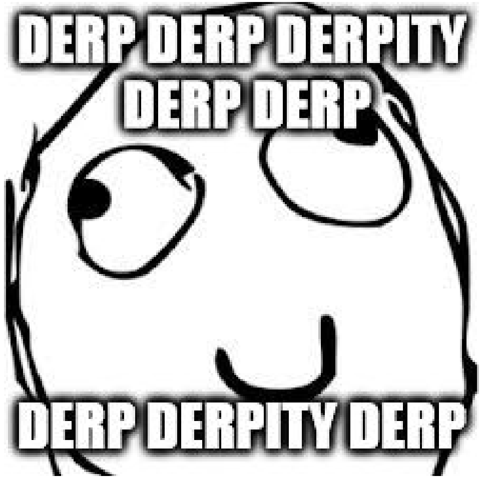
New for OpenTSDB 2.4

- Rollup / Pre-Aggregated storage and querying
  - Improves query speed
  - Allows for high-resolution data to be TTL’d out
- Histogram/Digests/Sketches
  - Accurate percentile calculations on distributed measurements such as latencies.
- Date Tiered Compaction support
- Authentication/Authorization plugin

The Problem of Percentiles

- Aggregating percentiles ==

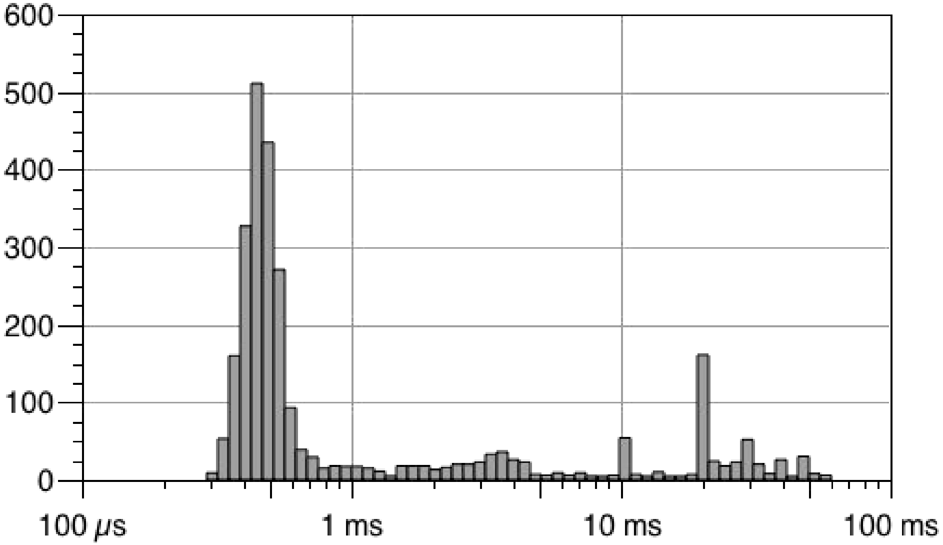
```
latency.p99.9  42.50  host=web01
latency.p99.9  58.98  host=web02
latency.p99.9  41.28  host=web03
latency.p99.9  41.94  host=web04
```



- Averaging percentiles is in accurate.
  E.g. **46.175** hides the bad host, **web02**
- Max is more useful for finding bad hosts
- But there are better ways...

Histograms

- Distribution of frequency of measurements over a time period
- Simplest form: latency measurement buckets storing counts falling within those buckets. E.g.



```
latency.histogram  0,15.0=0:15.0,30.0=1:30.0,45.0=4:45.0,60.0=0   host=web01
latency.histogram  0,15.0=1:15.0,30.0=0:30.0,45.0=2:45.0,60.0=4   host=web02
latency.histogram  0,15.0=2:15.0,30.0=0:30.0,45.0=4:45.4,60.0=0   host=web03
latency.histogram  0,15.0=0:15.0,30.0=1:30.0,45.0=4:45.0,60.0=0   host=web04
```

Histograms

Histogram	p99	p85	p50
latency.histogram 0,15.0=0:15.0,30.0=1:30.0,45.0=4:45.0,60.0=0 host=web01	37.5	37.5	37.5
latency.histogram 0,15.0=1:15.0,30.0=0:30.0,45.0=2:45.0,60.0=4 host=web02	52.5	52.5	52.5
latency.histogram 0,15.0=2:15.0,30.0=0:30.0,45.0=4:45.4,60.0=0 host=web03	37.5	37.5	37.5
latency.histogram 0,15.0=0:15.0,30.0=1:30.0,45.0=4:45.0,60.0=0 host=web04	37.5	37.5	37.5
Averaged Percentiles:	**41.25**	**41.25**	**41.25**
Summed Histograms:			
latency.histogram 0,15.0=3:15.0,30.0=2:30.0,45.0=14:45.0,60.0=4	**52.5**	**52.5**	**37.5**

Histograms

- Pros:
  - Fixed size (877 bytes for 97 buckets per data point)
  - Richer analysis (probability distribution, etc)
  - Mergable via group by and downsampling
  - Fixed rank error, variable value error
- Cons:
  - Much more network/storage space required
  - Loss of accuracy (somewhere within the bucket) but precise
  - Common metrics libraries lack support

Pluggable Implementations

Yahoo’s Data Sketches

- Collection of approximation algorithms with mergability and configurable accuracy v. size (~26k for 2M measurements)
- Deterministic rank error
- Tapering log size with N measurements per sketch
- Good for median percentiles
- https://datasketches.github.io/

Sketches Library from YAHOO!

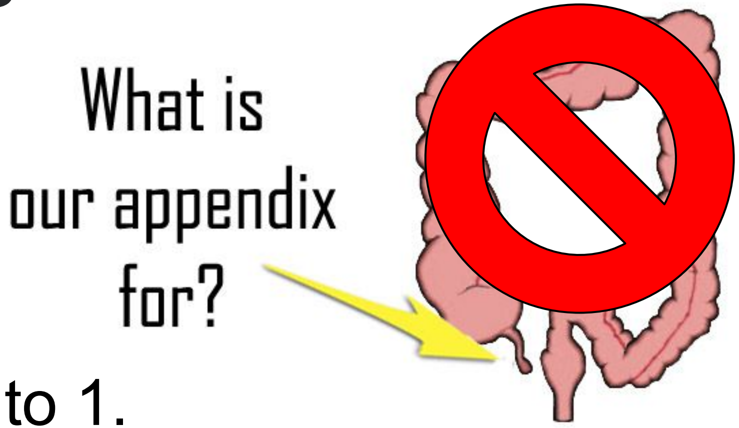
A Java software library of *stochastic streaming algorithms*

Pluggable Implementations

T-Digest

- Offshoot of Q-Digest K-means clustering quantile approximations
- Small error at top and bottom of the quantile range
- Mergable
- Able to store floating point as well as integers
- https://github.com/tdunning/t-digest

The Problem of Appends



- 2.2 Introduced appends to move away from TSD compactions.
- 1 second resolution = 3600 columns per row => compact into 1.
- But with appends, HBase:
  - Reads the column (from memstore or disk)
  - Appends the data and writes back to memstore (and possibly block cache)
  - Send full data back to the client

The Problem of Appends

- Negatives:
  - Possible disk thrashing if columns have been compacted out of the memstore
  - Higher CPU utilization on the region servers
  - Longer wait time on the client side
- Future Solution:
  - Yahoo’s HBase developers (Francis, Thiruvel) working on an optimization using coprocessors.
  - Trials underway, details in August

The Problem of Compactions

- HBase compaction merges multiple store files into one, saving space.
- But if we assume the data is time series, with older data immutable and updates only to new data…
- ...we can avoid re-compacting old files that won’t change and skip them at scan time.
- HBASE-15181 from Yahoo and Flurry supports organizing store files by date and time.
- PR #990 from Karan at SalesForce allows TSDB to write HBase timestamps

AsyncHBase 1.8

- AsyncHBase is a fully asynchronous, multi-threaded HBase client
- Supports HBase 0.90 to 1.x
- Faster and less resource intensive than the native HBase client
- Support for scanner filters, META prefetch, “fail-fast” RPCs

AsyncHBase 1.8

- Batched GetRequests thanks to Tian-Ying at Pinterest and Bizu at Yahoo
- Reverse scanning support thanks to Jiayun at Harvard
- HBase 1.3.x+ support thanks to Karan at SalesForce
- MultipleColumnPrefixFilter
- Skip WAL with increments
- AtomicIncrements with multiple columns per request

OpenTSDB on Bigtable

- Bigtable
  - Hosted Google Service
  - Client uses HTTP2 and GRPC for communication
- OpenTSDB heads home
  - Based on a time series store on Bigtable at Google
  - Identical schema as HBase
  - Same filter support (fuzzy filters are coming)

OpenTSDB 3.0

- **Problem**: Queries are slow and the order of operations is immutable
- **Solutions**: *(This part is ready for testing!)*
  - New composable query layer allowing operations in any order
  - Support for querying multiple sources and merging the results (e.g. use Facebook’s Berengi as a write-cache and Redis as a query cache)
  - Support for multi-cluster queries for active-active, high-availability setups

OpenTSDB 3.0

- **Problem**: Storing other types of data or using other backends is a pain.
- **Solutions**: *(In progress)*
  - Pluggable storage interface allowing for various schemas and implementations
    (e.g. native HTable client, AsyncHBase, native Bigtable client, etc)
  - Abstracted data types for pluggable implementations of time series (e.g. raw binary, histograms, SCADA data)

OpenTSDB 3.0

- **Problem**: What about anomaly detection, forecasting, etc?
- **Solutions**: *(In progress)*
  - Integration with Yahoo’s EGADS time series functions library
  - Period-over-period analysis with model caching
  - Clustering algorithms for detecting outliers
  - https://github.com/yahoo/egads

OpenTSDB 3.0

- New Java APIs
- Servlet for standard deployment using your favorite server
- Tracing with Zipkin and OpenTracing
- New debugging UI
- Improved Docker support

Alternative TSDBs

InfluxDB

druid

DalmatinerDB

citusdata

Timely

TIMESCALE

AXIBASE

riakTS

SiriDB

Warp 10

Prometheus

BTrDB: Berkeley Tree Database

https://misfra.me/2016/04/09/tsdb-list/

More Info and Credits

- Thanks to the Monitoring and HBase teams at Yahoo, Pythian for Bigtable support and our OSS contributors!
- Contribute at github.com/OpenTSDB/opentsdb
- Website: opentsdb.net
- Mailing List: groups.google.com/group/opentsdb

Images

- https://commons.wikimedia.org/wiki/File:Programmer_writing_code_with_Unit_Tests.jpg
- http://www.doncio.navy.mil/CHIPS/ArticleDetails.aspx?ID=8098
- https://i.imgflip.com/t96s8.jpg
- https://commons.wikimedia.org/wiki/File:Twemoji_1f626.svg
- https://xkcd.com/1425/
- https://commons.wikimedia.org/wiki/Emoji#/media/File:Twemoji_1f623.svg
- https://c1.staticflickr.com/1/508/32307332875_40e73bf750_b.jpg
- http://code.flickr.net/2008/10/27/counting-timing/
- http://3.bp.blogspot.com/-tTXEI5IiQh4/VQqaJz4LtSI/AAAAAAAAEL8/n5AwTVNI-Us/s1600/Introduction%2Bto%2BSQL.png