Fun With Graphite

What is Graphite?

- Time-series based metrics DB
- Graphing engine for stored metrics

Time-Series Metrics

- Metric data points are expected at a given frequency
- Insufficient metrics in an interval mean nothing is stored
  - This can be adjusted

How Are Metrics Stored?

- Whisper files
- Each metric is a separate file
- Whisper files are pre-allocated
- Files can contain multiple aggregations

Whisper File Location

- /opt/graphite/storage/whisper
- Convert ‘.’ to ‘/’ in path
  - servers.hostname.loadavg.01
  - /opt/graphite/storage/whisper/servers/hostname/loadavg/01.wsp

Whisper File Contents

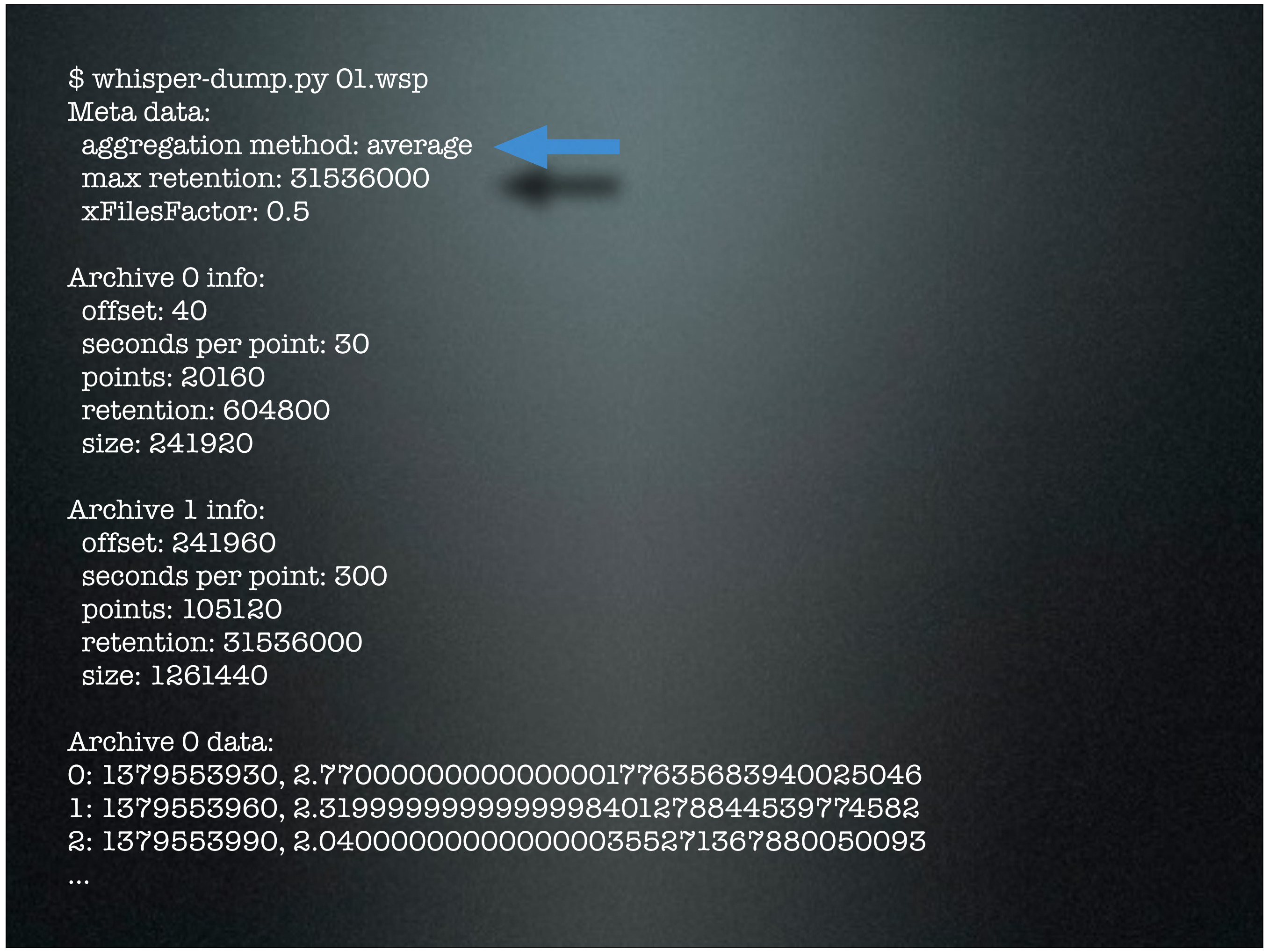
- Pre-allocated
  - Size doesn’t change
  - All timestamps pre-written
  - Only stores a timestamp & a metric result

```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.77000000000000017763568394002504б
1: 1379553960, 2.31999999999999984012788445397745582
2: 1379553990, 2.04000000000000003552713678800500093
...
```



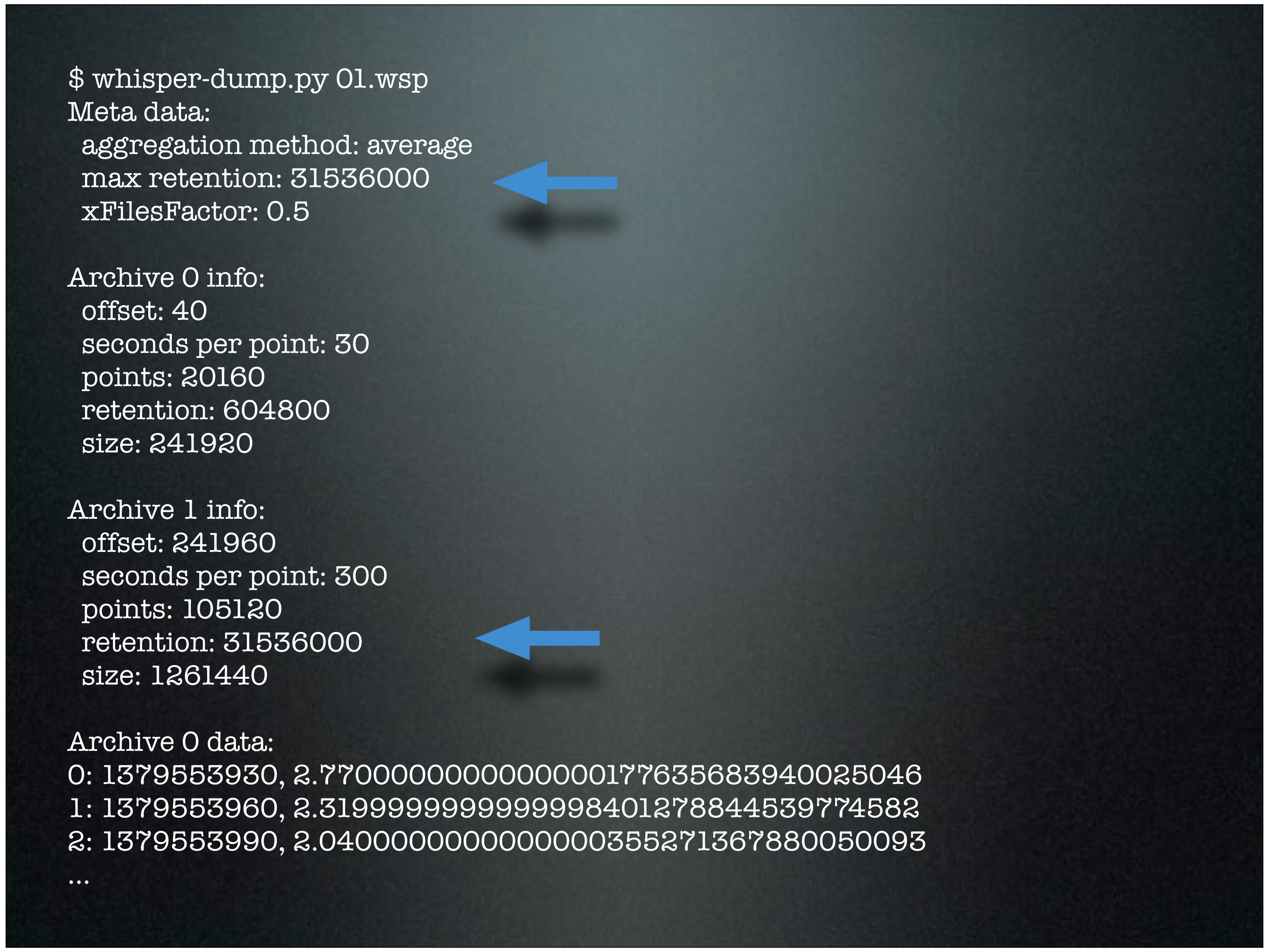
```
$ whisper-dump.py 01.wsp
Meta data:
 aggregation method: average
 max retention: 31536000
 xFilesFactor: 0.5

Archive 0 info:
 offset: 40
 seconds per point: 30
 points: 20160
 retention: 604800
 size: 241920

Archive 1 info:
 offset: 241960
 seconds per point: 300
 points: 105120
 retention: 31536000
 size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Aggregation method:
 * Average (default) – all metrics received in an interval are averaged into a single metric value for the interval
 * Maximum, minimum – only the highest or lowest value is used – all others are discarded
 * Sum – All metrics received during the interval are summed up and the total is stored for the interval



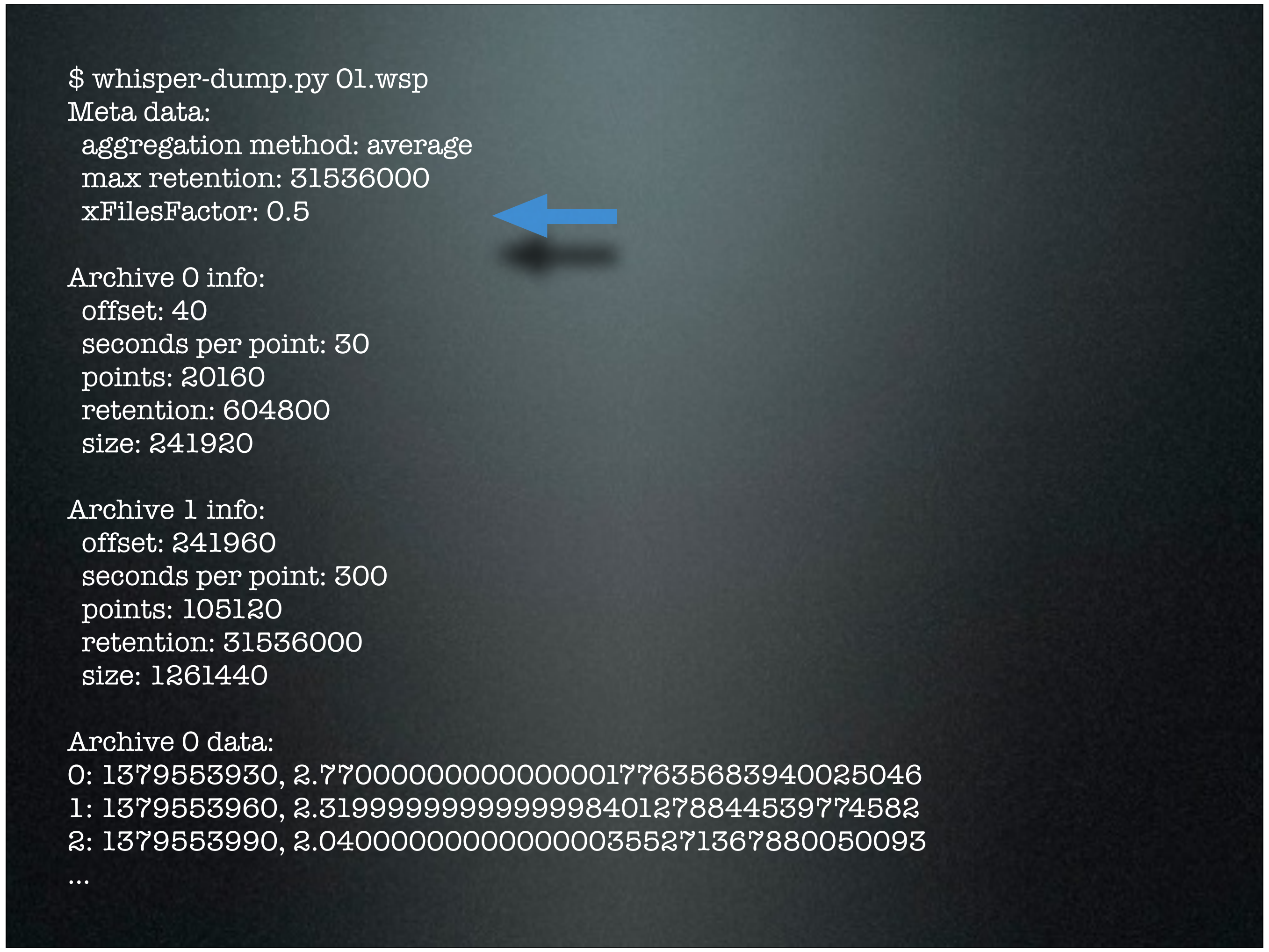
```
$ whisper-dump.py 01.wsp
Meta data:
 aggregation method: average
 max retention: 31536000
 xFilesFactor: 0.5

Archive 0 info:
 offset: 40
 seconds per point: 30
 points: 20160
 retention: 604800
 size: 241920

Archive 1 info:
 offset: 241960
 seconds per point: 300
 points: 105120
 retention: 31536000
 size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Max Retention:
 * Matches the longest retention of all of the archives included in this metric



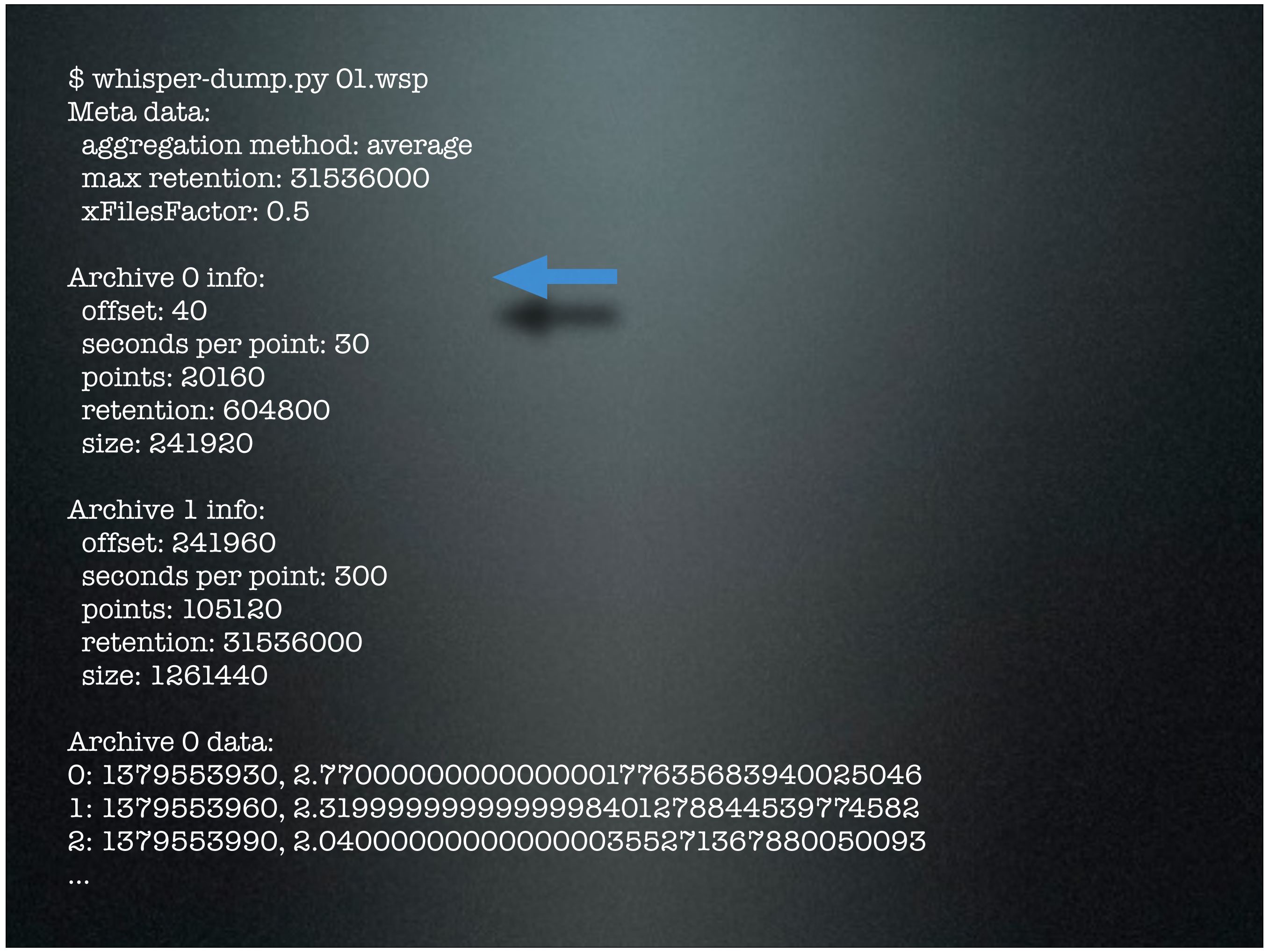
```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

xFilesFactor:
 * Percentage of metrics per interval that must be non-null in order for the interval to be considered valid.
 * If this is set to 0.5, the highest resolution retention is 30 seconds & metrics are injected every 3 seconds, at least 8 metrics per 30 seconds must be non-null values in order for that 30 second interval to have a value attached to it. If less than 8 metrics are received in that 30 second interval, the interval is given a null value (NaN - Not a Number).



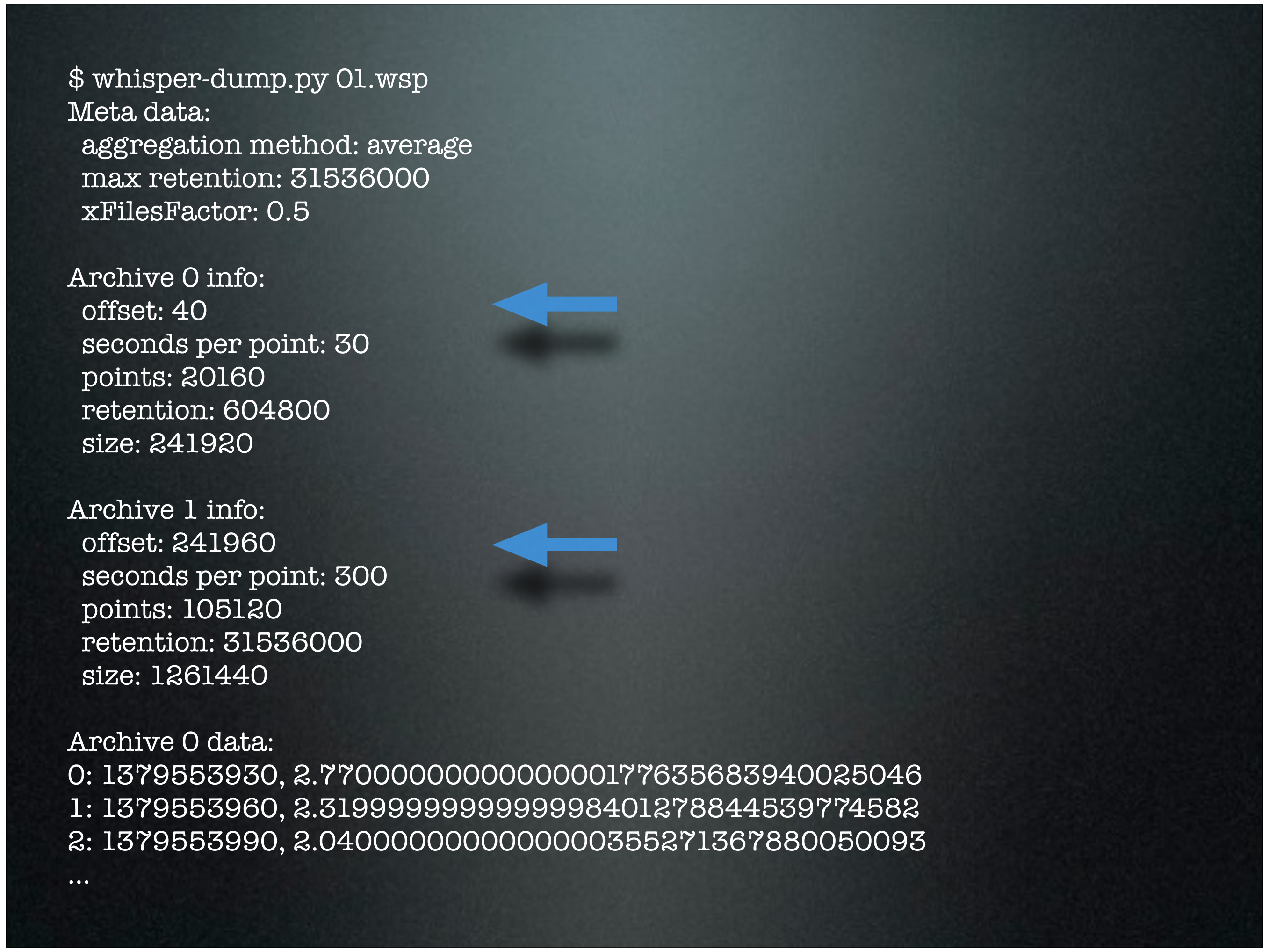
```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Archives:
 * Archives are synonymous with retention schemas or retentions
 * A metric can have multiple retentions
 * This particular metric has 2 retentions defined:
   - Every 30 seconds for 7 days
   - Every 5 minutes for 1 year



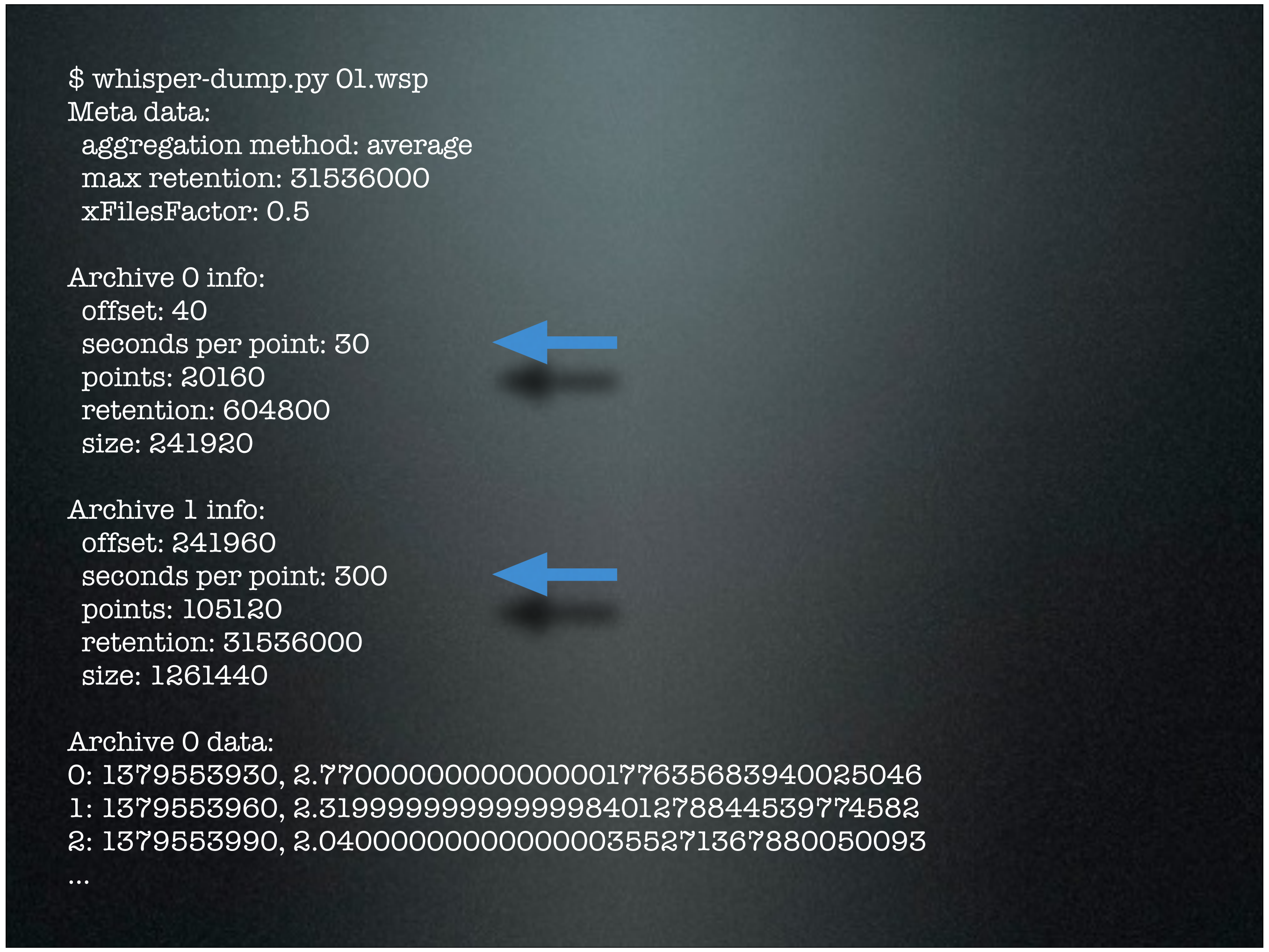
```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.77000000000000017763568394002504 6
1: 1379553960, 2.31999999999999984012788445397745 82
2: 1379553990, 2.04000000000000003552713678800500 93
...
```

Offset:
 * Why byte in the Whisper file the archive (retention) data begins to be stored at



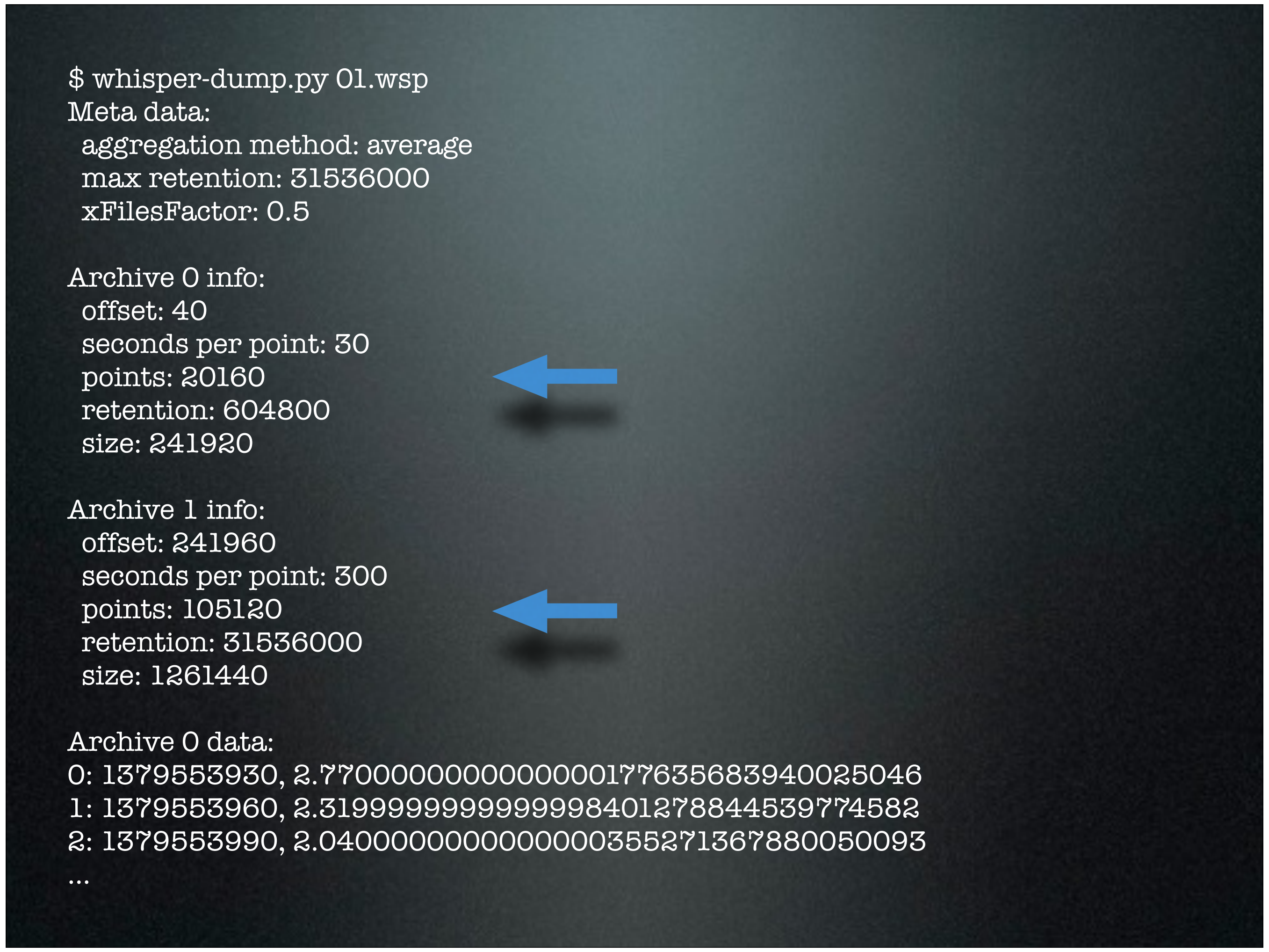
```
$ whisper-dump.py 01.wsp
Meta data:
 aggregation method: average
 max retention: 31536000
 xFilesFactor: 0.5

Archive 0 info:
 offset: 40
 seconds per point: 30
 points: 20160
 retention: 604800
 size: 241920

Archive 1 info:
 offset: 241960
 seconds per point: 300
 points: 105120
 retention: 31536000
 size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.319999999999999840127884453977458
2: 1379553990, 2.040000000000000035527136788005009
...
```

Seconds per point:
 * How long of an interval each metric point stored represents
 * In this file each interval takes the average of all the metrics injected during the interval and stores the result as a single value for the interval defined in each archive (or retention)



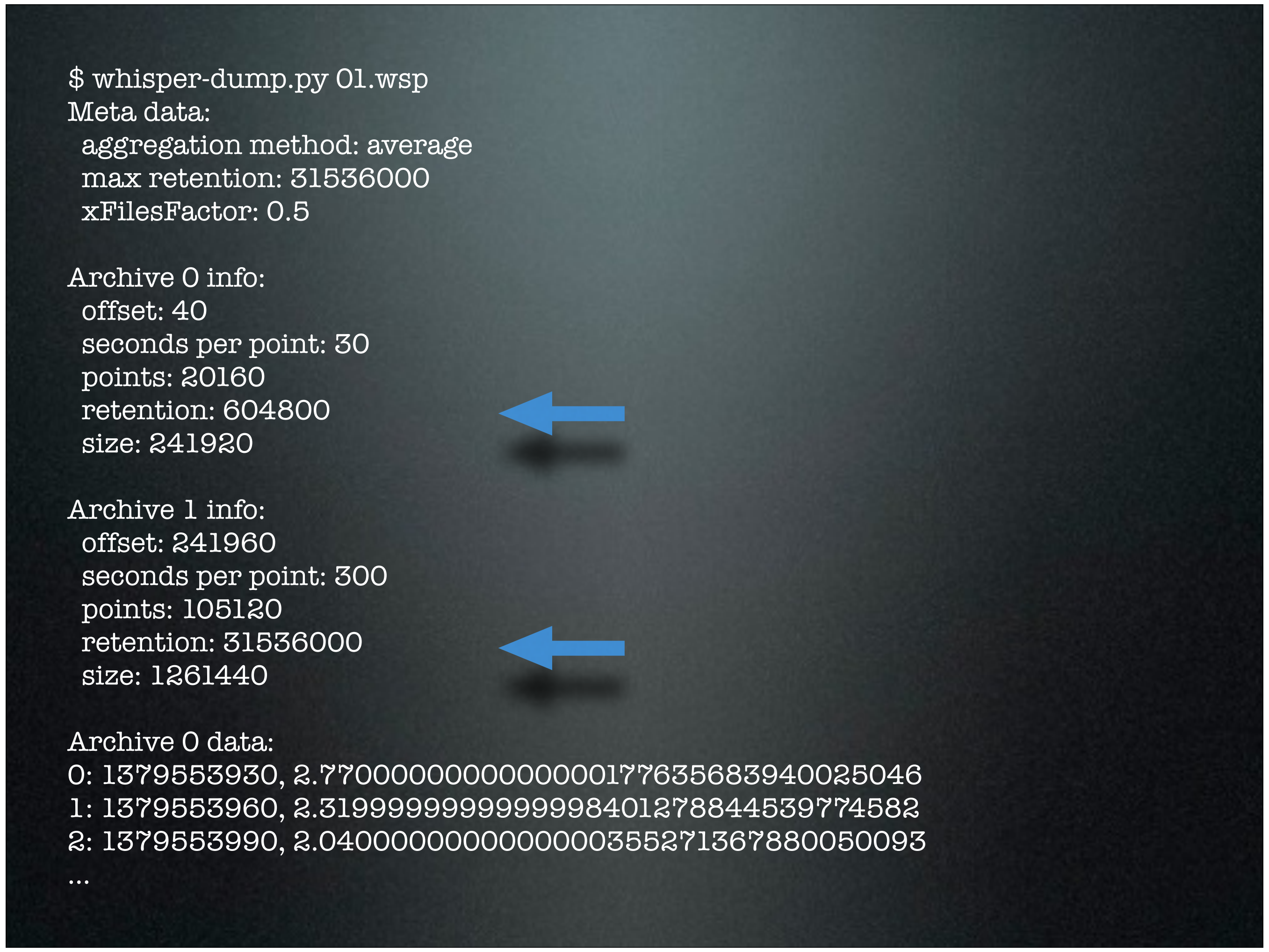
```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Points:
 * The total number of points comprising the retention period
 * 30 seconds * 604,800 seconds (7 days) = 20,160 points
 * 300 seconds (5 minutes) * 31,536,000 (365 days) = 105,120 points



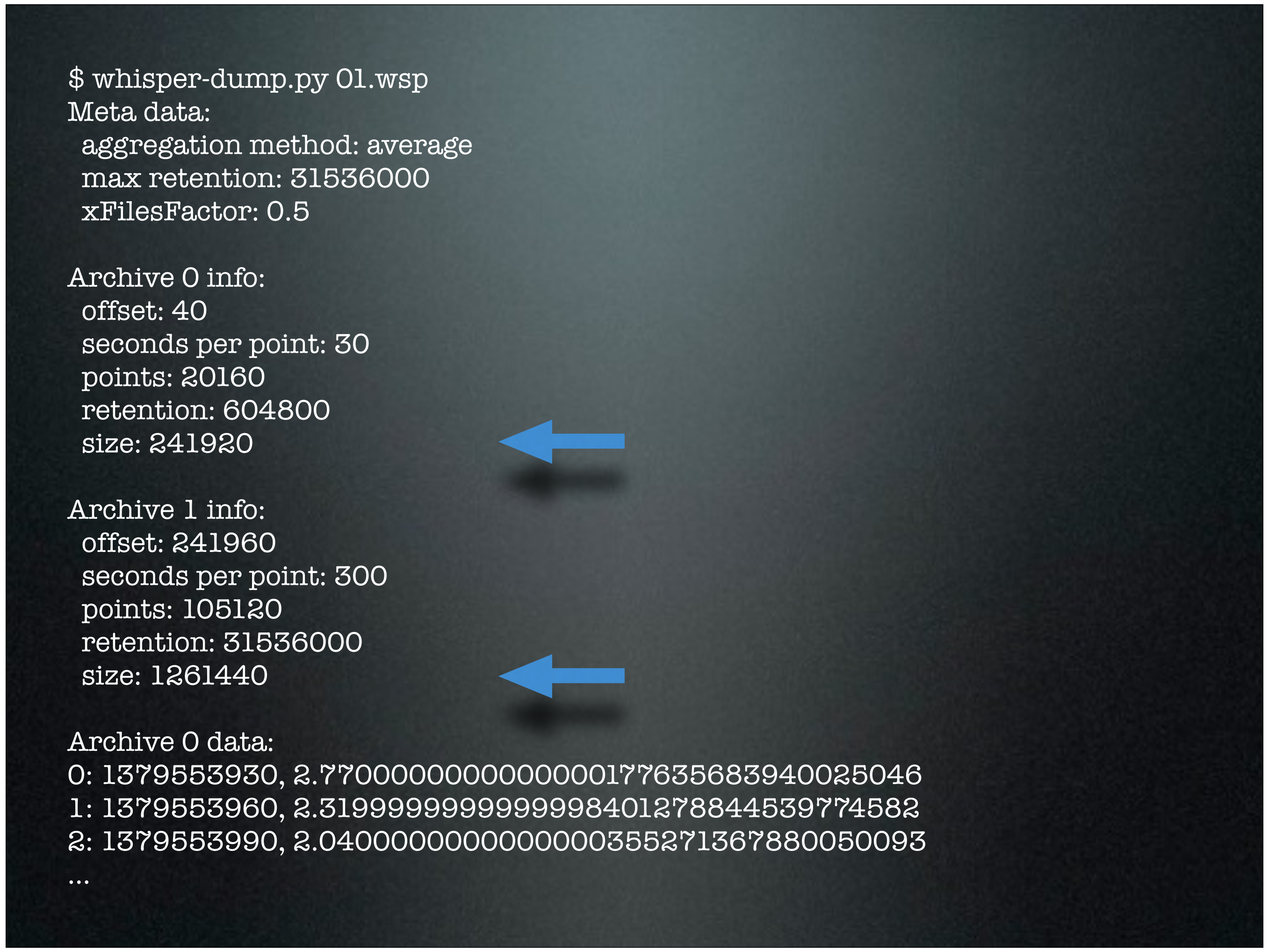
```
$ whisper-dump.py 01.wsp
Meta data:
 aggregation method: average
 max retention: 31536000
 xFilesFactor: 0.5

Archive 0 info:
 offset: 40
 seconds per point: 30
 points: 20160
 retention: 604800
 size: 241920

Archive 1 info:
 offset: 241960
 seconds per point: 300
 points: 105120
 retention: 31536000
 size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Retention:
 * Length of time to keep storing metrics for (in seconds)
  - 604,800 seconds = 7 days
  - 31,536,000 seconds = 1 year



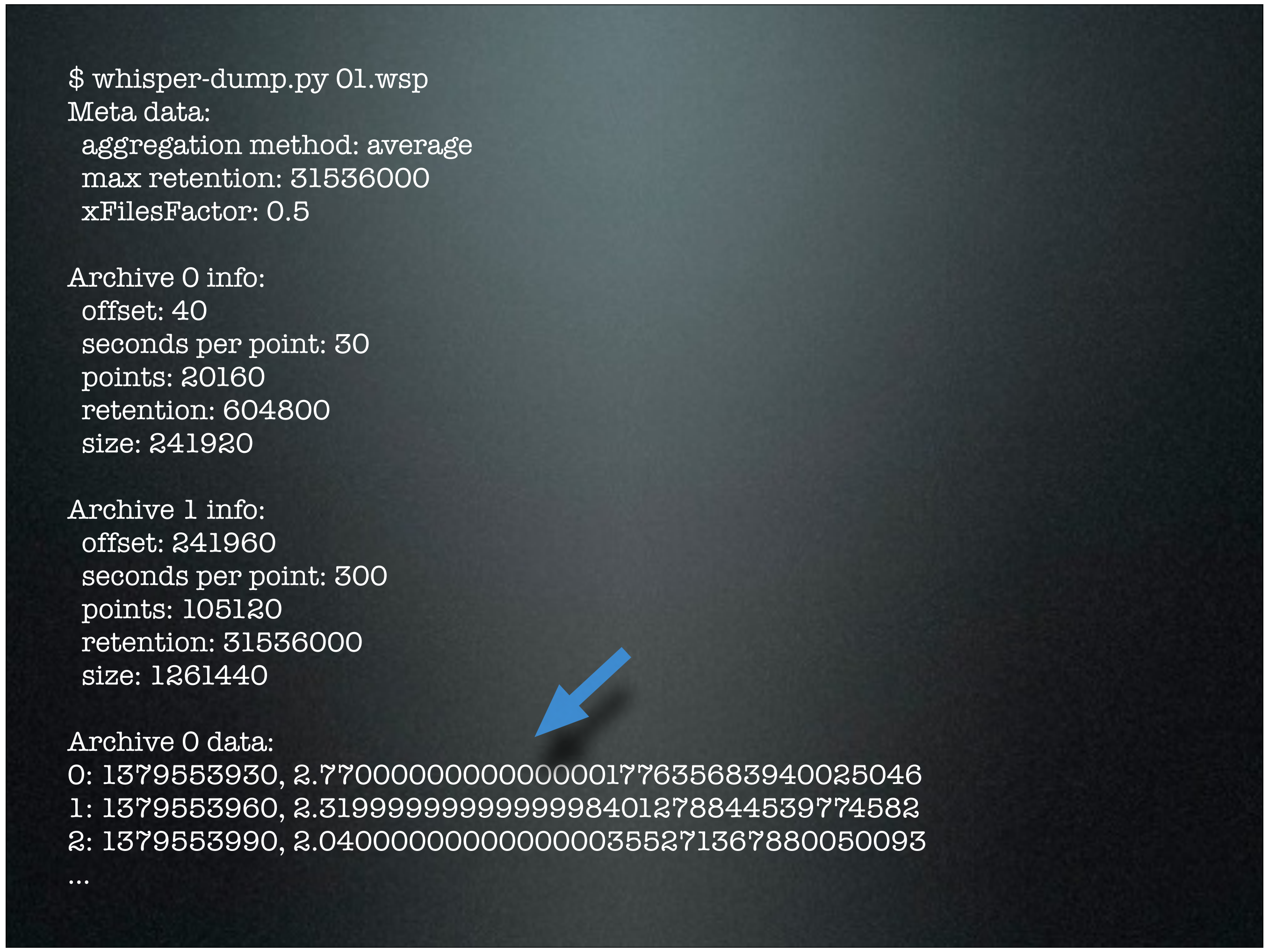
```
$ whisper-dump.py 01.wsp
Meta data:
 aggregation method: average
 max retention: 31536000
 xFilesFactor: 0.5

Archive 0 info:
 offset: 40
 seconds per point: 30
 points: 20160
 retention: 604800
 size: 241920

Archive 1 info:
 offset: 241960
 seconds per point: 300
 points: 105120
 retention: 31536000
 size: 1261440

Archive 0 data:
0: 1379553930, 2.770000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Size:
 * Total size of the archive (retention) in bytes
 * Each metric stored = ~12 bytes



```
$ whisper-dump.py 01.wsp
Meta data:
  aggregation method: average
  max retention: 31536000
  xFilesFactor: 0.5

Archive 0 info:
  offset: 40
  seconds per point: 30
  points: 20160
  retention: 604800
  size: 241920

Archive 1 info:
  offset: 241960
  seconds per point: 300
  points: 105120
  retention: 31536000
  size: 1261440

Archive 0 data:
0: 1379553930, 2.7700000000000000177635683940025046
1: 1379553960, 2.3199999999999998401278844539774582
2: 1379553990, 2.0400000000000000355271367880050093
...
```

Archive data:
 * 3 fields:
  - ID/counter
  - Timestamp (epoch)
  - Metric value

Graphite Daemons

- carbon-relay
- carbon-cache
- carbon-aggregator

carbon-relay

carbon-relay:
 * Like a router for metrics
 * Directs metrics to one or more carbon-cache daemons on one or more servers
 * Can direct metrics based on rules or use consistent hashing with configurable number of replicas
  - Rule-based reads a configuration file with regexes defining which metrics should be sent to which carbon-caches
  - Consistent hashing will write each metric to a pool of carbon-caches replicating each metric N number of times based on what the replication is configured to be

carbon-cache

carbon-cache:
 * Receives the metrics and handles writing them to disk
 * Uses memory to cache recent metrics and asynchronously (by default) writes them to disk to balance IO
 * Follows schema configurations for retentions in storage-schema.conf to define the structure of Whisper files & determine how to store the metric results.
 * Responsible for returning the desired metrics when requested by the API
 * Multiple carbon-caches can run on a given server to improve IO utilization across multiple processes

carbon-aggregator

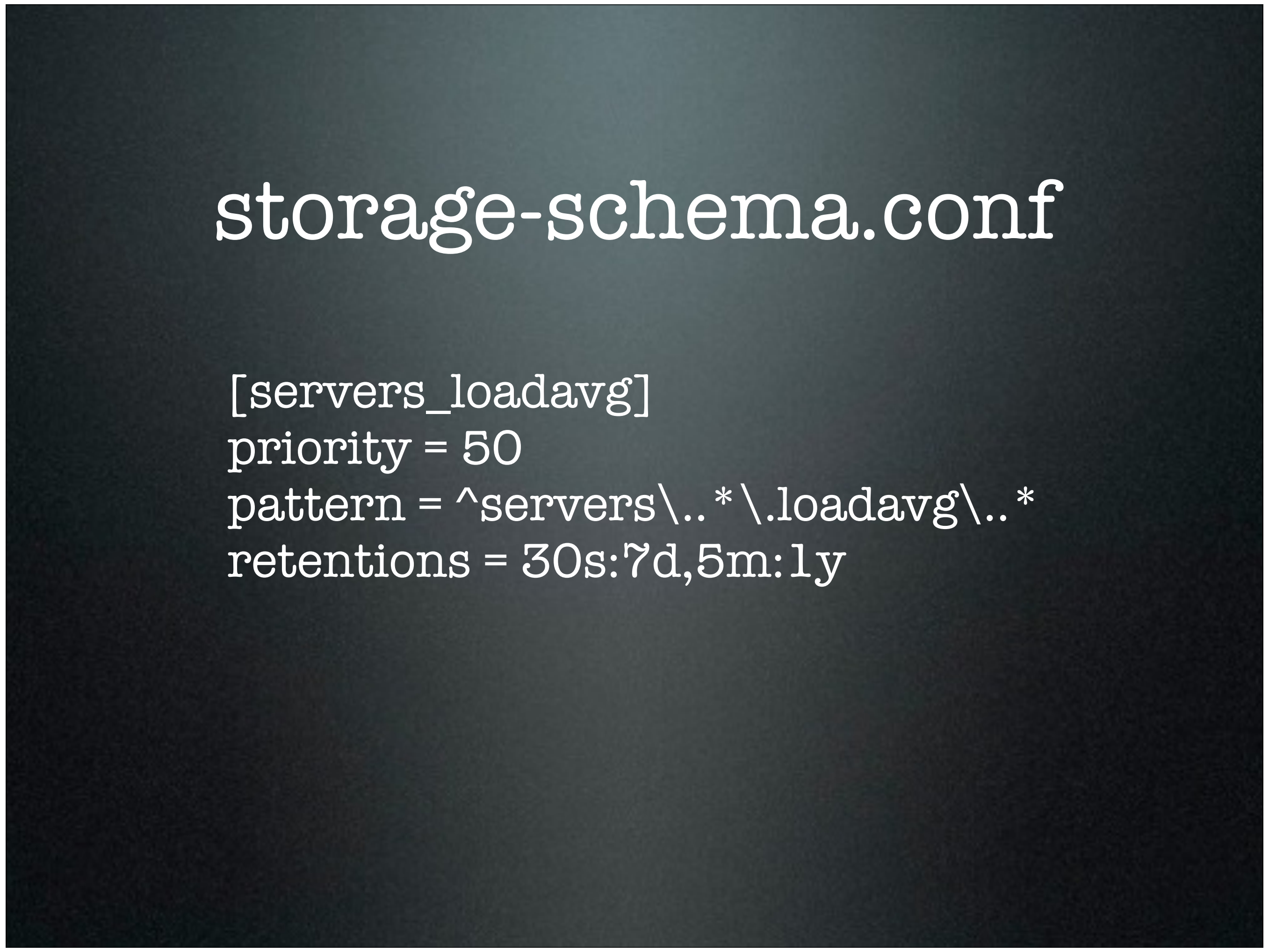
carbon-aggregator:
 * Not currently used at AWeber
 * Pre-processes aggregations before sending them to carbon-cache
 * Because results are buffered & pre-processed they aren’t available in “real-time” like they are if you’re using carbon-cache directly
 * Can help improve IO load

Configuration

- carbon.conf
- storage-schema.conf
- storage-aggregation.conf

carbon.conf

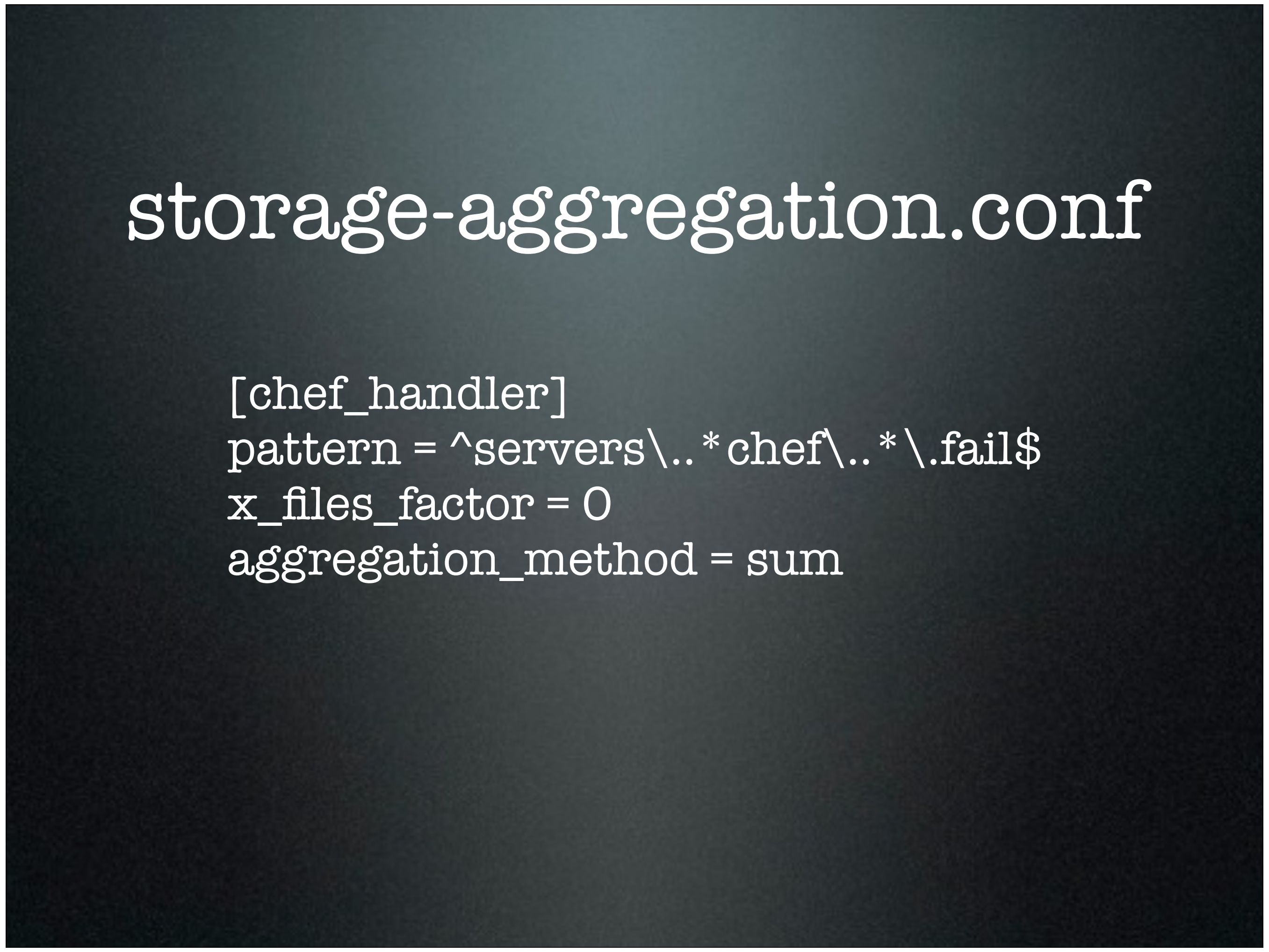
* Configuration file for carbon-relay & carbon-cache daemons
* Defines ports & IP addresses each daemon binds to (listens on)
* Defines how carbon-cache creates & writes to Whisper files
* Defines how carbon-cache uses memory to cache metrics & how aggressively to write to disk
* Defines how carbon-relay is configured (rules-based or consistent hashing)
* Defines which carbon-cache daemons/hosts carbon-relay should send metrics to
* Defines how many metrics carbon-relay can queue waiting for carbon-cache to accept before either dropping new metrics injected or rejecting metrics
  - Because metric injection is via TCP by default, setting carbon-relay to reject metrics vs. dropping them can provide the ability for the client to retransmit metrics (pros/cons to each)



storage-schema.conf

```
[servers_loadavg]
priority = 50
pattern = ^servers\..*\.loadavg\..*
retentions = 30s:7d,5m:1y
```

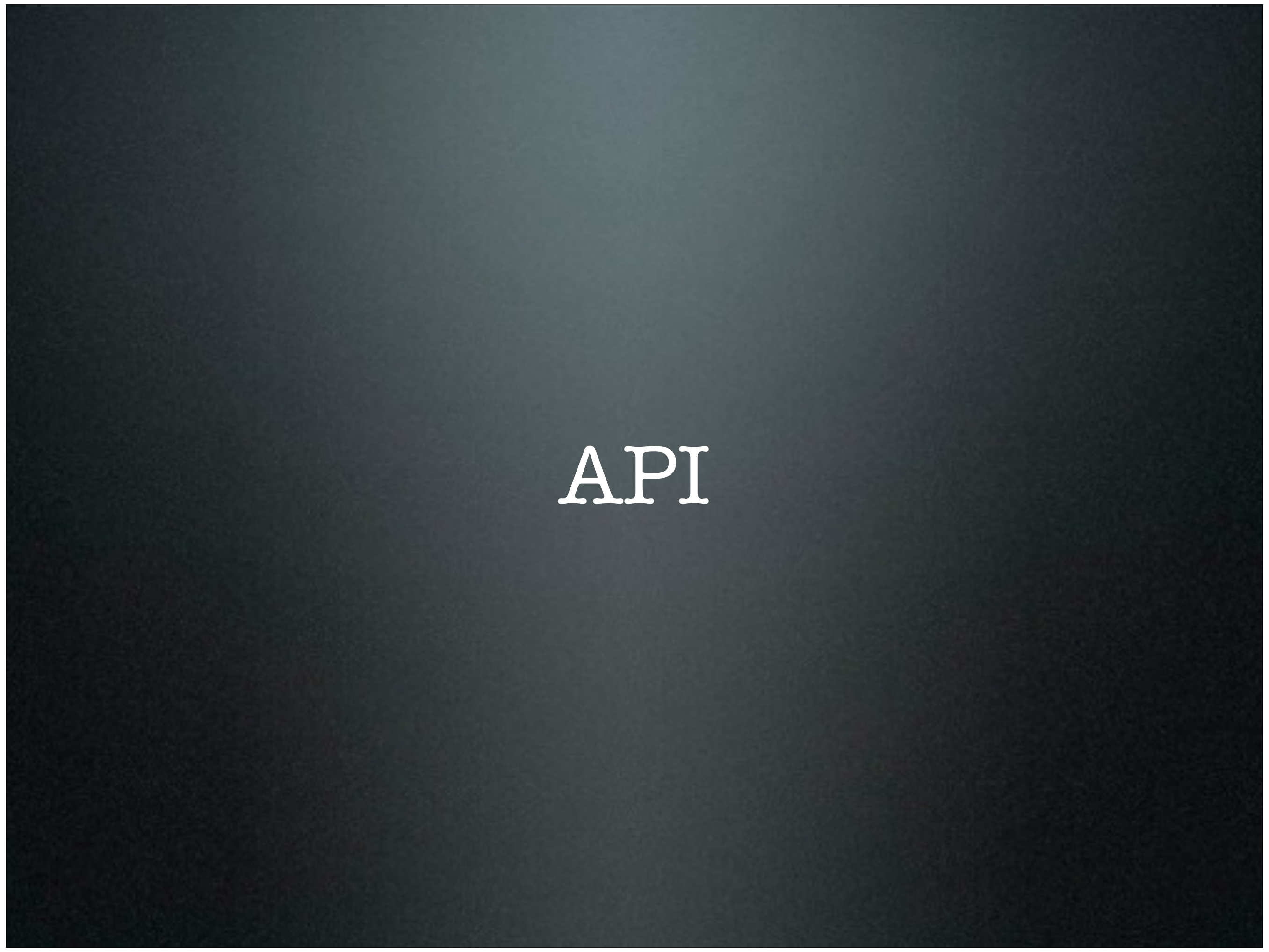
* Defines the retentions (archives) each metric file will include
* Uses a regex to determine which metric matches the rule defined
* Priorities can be used to define higher granularity of specific metrics within a broader scope of metrics of similar names
* Retentions are defined as interval:retention period
 - Interval is how often metrics are stored
 - Retention period is the length of time a particular metric is stored for



storage-aggregation.conf

```
[chef_handler]
pattern = ^servers\..*chef\..*\.fail$
x_files_factor = 0
aggregation_method = sum
```

 * Defines how metrics received over a particular interval are aggregated & stored for that interval
  - In this example, the metric has an interval of 1 minute, so multiple injections of this metric within that 1 minute interval are summed up before being stored
 * Sets the xFilesFactor setting for the Whisper file (defining how many results per interval are required for a valid result)
  - Because this is set to zero, *any* result is sufficient to be considered a valid metric for the interval. Only when *no* results are received for the interval (1 minute) will the interval be given a null value.
  - Because chef-client can be run at any time arbitrarily, a high resolution is used for this metric so that any chef-client run is recorded. The downside to this is that there is a lot of gaps in the file where 'null' values will be written wasting space.
    a. Supposedly the next storage mechanism (Ceres) will allow for only storing actual metrics in expanding/contracting DB files rather than using fixed-size DB files


API

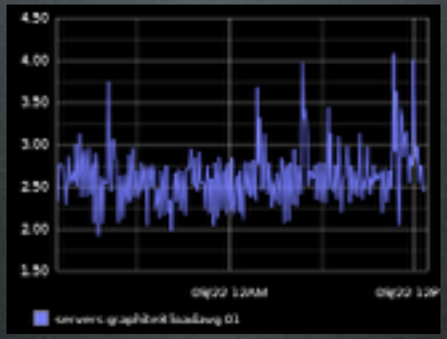
carbon-aggregator:
 * Not currently used at AWeber
 * Pre-processes aggregations before sending them to carbon-cache
 * Because results are buffered & pre-processed they aren’t available in “real-time” like they are if you’re using carbon-cache directly
 * Can help improve IO load

API Features

- URL-based
- Globbing
- Functions (nested)

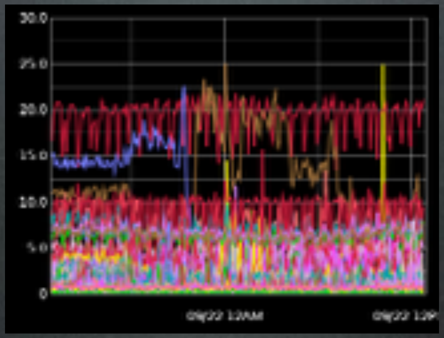
URL-based API

- Query-strings used to access metrics
- http://graphite.colo.lair/render/?target=servers.graphite3.loadavg.01



Globbing/Regexes for Multiple Metrics

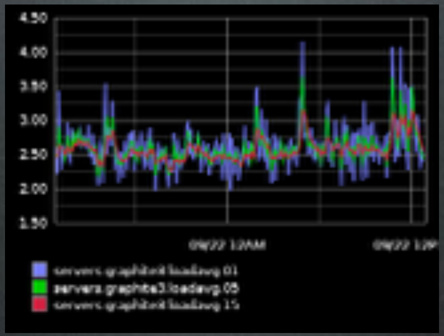
- Globbing for multiple metrics
  - http://graphite.colo.lair/render/?target=servers.*.loadavg.01



Graph represents 1-minute load average across all hosts
 * Note that the legend automatically disappears when there’s >10 metrics plotted at the same time

Globbing/Regexes for Multiple Metrics

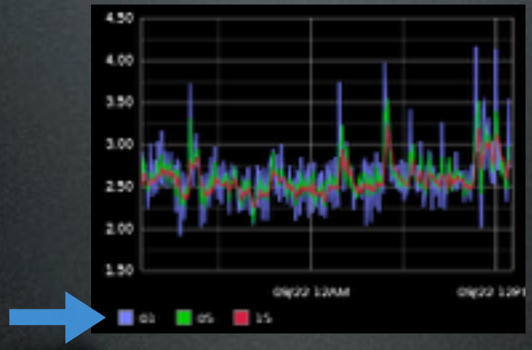
- Simple regex for multiple metrics
  - http://graphite.colo.lair/render/?target=servers.graphite3.loadavg.{01,05,15}



This graph shows 1-minute, 5-minute & 15-minute load average for the graphite3 host.

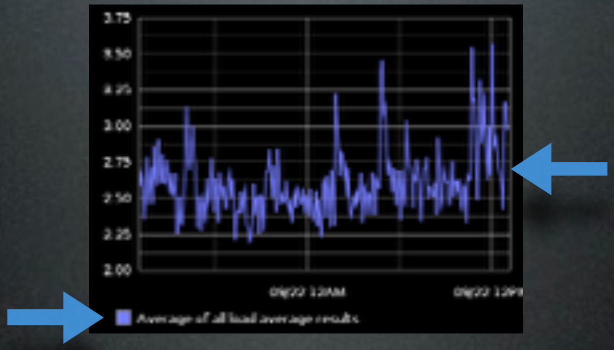
Using Functions

- Simple function for setting the legend names
  - http://graphite.colo.lair/render/?target=aliasByNode(servers.graphite3.loadavg.{01,05,15},3)



Using Functions

- Nesting multiple functions
  - http://graphite.colo.lair/render/?target=alias(averageSeries(servers.graphite3.loadavg.{01,05,15}),"Average+of+all+load+average+results")

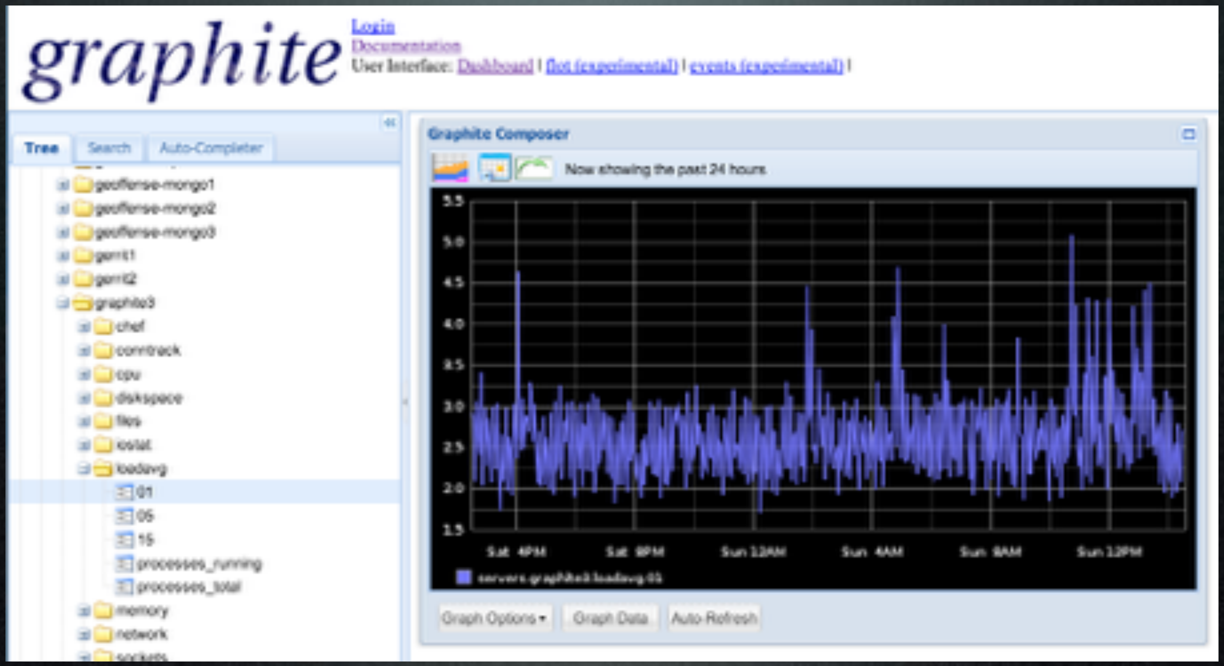


An average of all 3 metric results for loadavg (1, 5 & 15 minute) is plotted & the legend alias is modified to be more friendly.

Web UI

- Graph Generator
- Dashboard Generator

Graph Generator



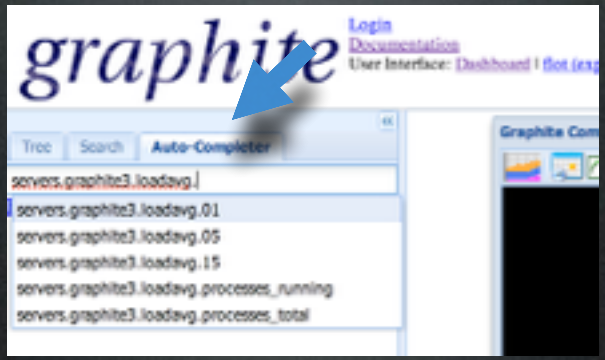
Graph Generator

- Pros:
  - Useful for quickly building complex graphs
  - Apply & remove functions on metrics quickly
- Cons:
  - Only allows for creating a single graph
  - Can’t really save created graphs*

* It is possible to save graphs, but currently only via the ‘root’ user (not the system’s root user).
 - Not a feature we use at AWeber - use the Dashboards instead
 - There is LDAP integration, but other dashboards are probably preferred over spending time on this

Graph Generator Tip

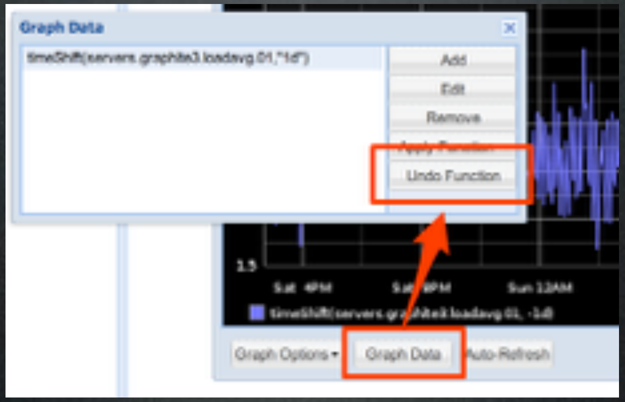
- Auto-Completer tab allows for "finding" metrics
  - Start typing metric to get list
  - Must start from the beginning of the metric name (can't search for partial metric names)



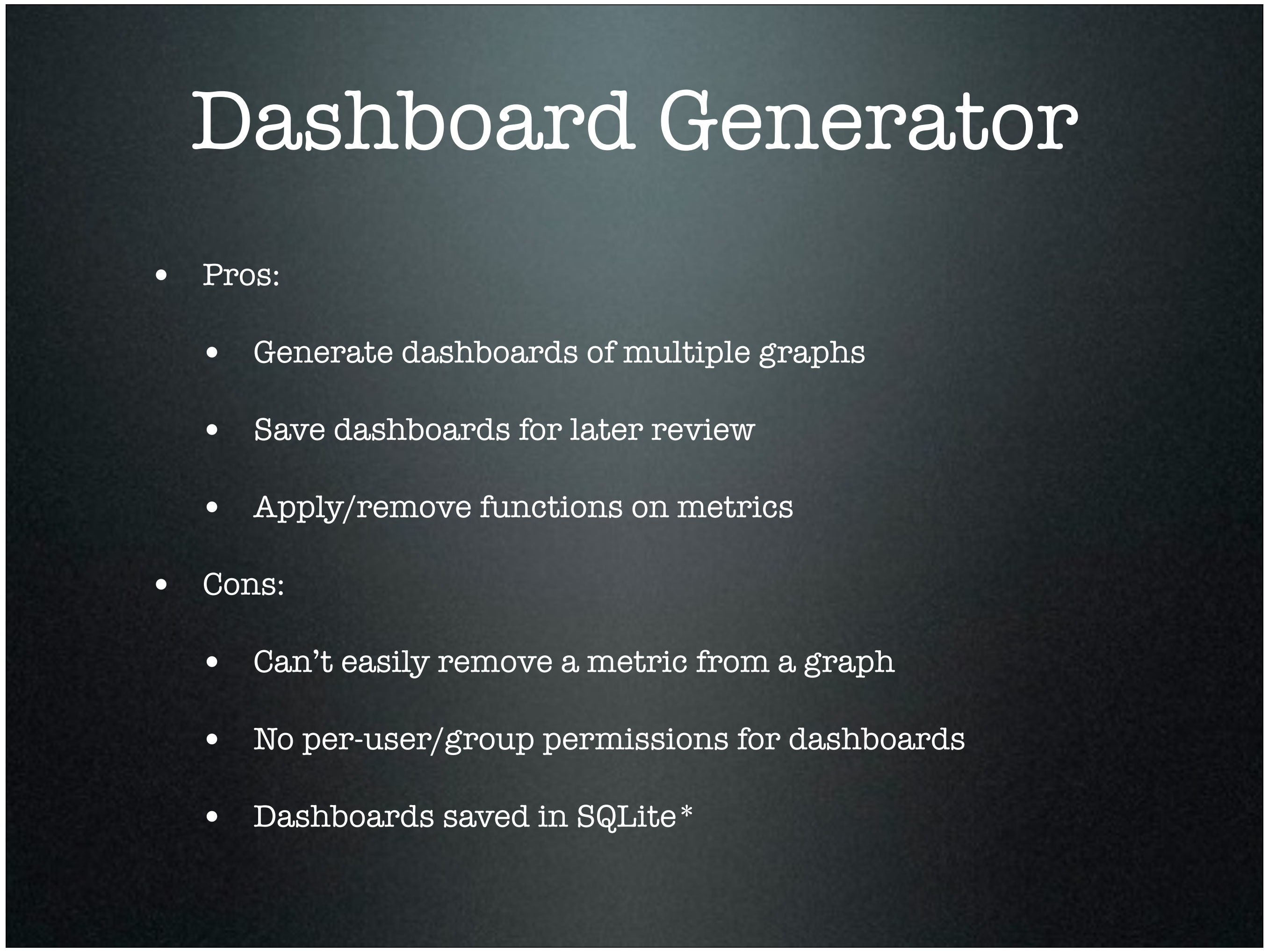
```
servers.graphite3.loadavg.
servers.graphite3.loadavg.01
servers.graphite3.loadavg.05
servers.graphite3.loadavg.15
servers.graphite3.loadavg.processes_running
servers.graphite3.loadavg.processes_total
```

Graph Generator Tip

- Undoing Functions
  - Select a metric in the Graph Data
  - Click Undo Function one or more times to remove outer-most function(s) (like peeling an onion)



```
timeShift(servers.graphite3.loadavg.01,"1d")
```

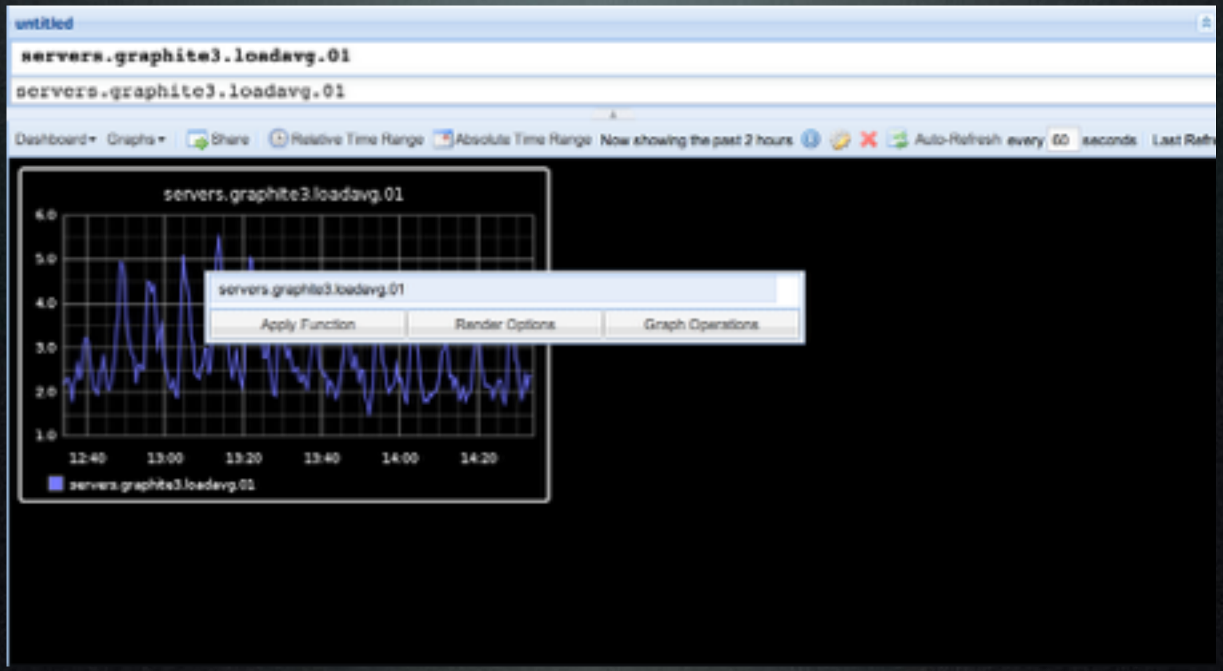


Dashboard Generator

- Pros:
  - Generate dashboards of multiple graphs
  - Save dashboards for later review
  - Apply/remove functions on metrics
- Cons:
  - Can’t easily remove a metric from a graph
  - No per-user/group permissions for dashboards
  - Dashboards saved in SQLite*

* There is support for storing dashboards in PostgreSQL, but it currently isn’t supported at AWeber.
 - Important when setting up clusters of Graphite servers so that dashboards are shared across servers

Dashboard Generator



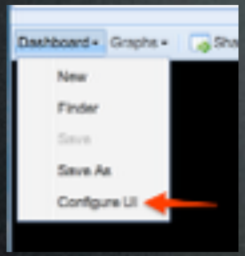
untitled

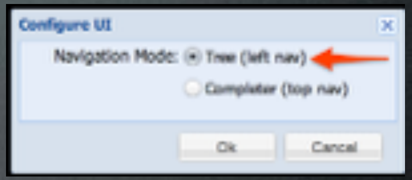
servers.graphite3.loadavg.01

servers.graphite3.loadavg.01

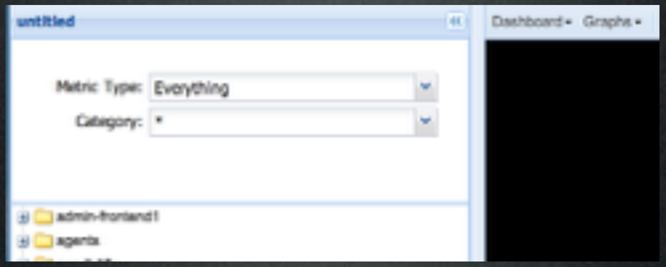
Dashboard Generator Tip

- Change UI to Tree view for easier navigation




Configure UI

Navigation Mode: [✓ Tree (left nav)] Completer (top nav)


untitled

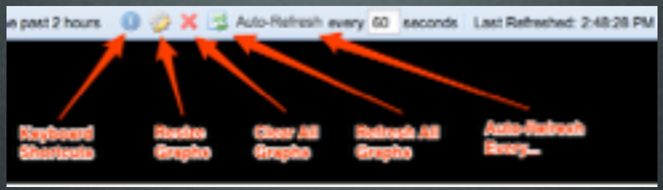
Metric Type: Everything

Category: *

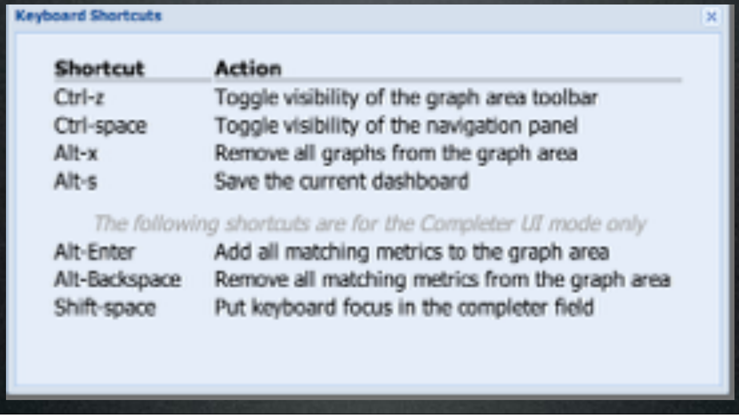
Need to remember to refresh the page after changing the view (did you remember to save everything you were working on??)

Dashboard Generator Tip

- Configuration & auto-refresh buttons



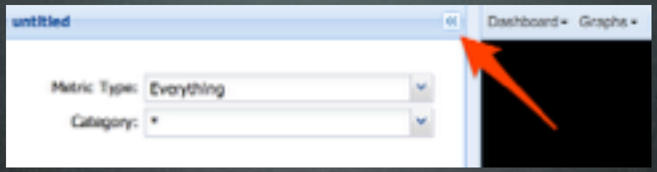
- Keyboard shortcuts


Keyboard Shortcuts

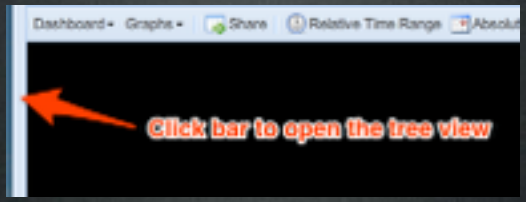
Shortcut	Action
Ctrl-z	Toggle visibility of the graph area toolbar
Ctrl-space	Toggle visibility of the navigation panel
Alt-x	Remove all graphs from the graph area
Alt-s	Save the current dashboard
	The following shortcuts are for the Completer UI mode only
Alt-Enter	Add all matching metrics to the graph area
Alt-Backspace	Remove all matching metrics from the graph area
Shift-space	Put keyboard focus in the completer field

Dashboard Generator Tip

- Hide the tree view



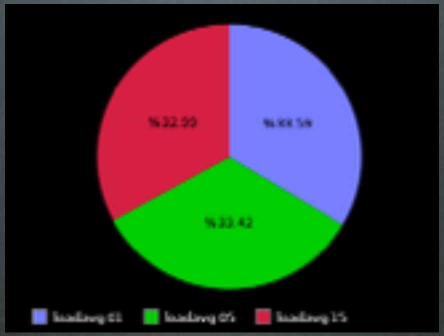
- Open the tree view



Cool! Now What?

Quick Gallery

Pie Charts!?

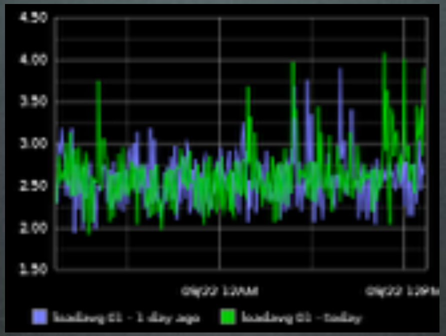


http://graphite.colo.lair/render/?target=aliasByNode(servers.graphite3.loadavg.{01,05,15},2,3)&graphType=pie

* Uses the ‘aliasByNode’ function to define the legend aliases
 - Note that both the 2nd & 3rd position of the metric name are used to create the alias

* ‘graphType’ is used to make it a pie chart
 - ‘line’ & ‘pie’ are the only options available currently
 - A ‘bar’ option has been added as a patch, but not sure it’s merged into production yet upstream

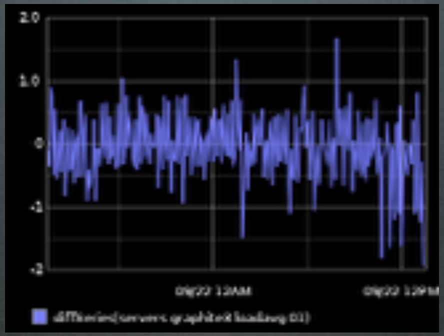
Time Shifted Comparisons



http://graphite.colo.lair/render/?
target=alias(timeShift(servers.graphite3.loadavg.01%2C%221d%22),
%22loadavg%2001%20-%201%20day%20ago
%22)&target=alias(servers.graphite3.loadavg.01,%22loadavg
%2001%20-%20today%22)

* Uses the ‘timeShift’ function to overlay a metric from a day ago with the same metric for today
* Great for comparing historic data from a comparative time period with current data

Difference Between Series

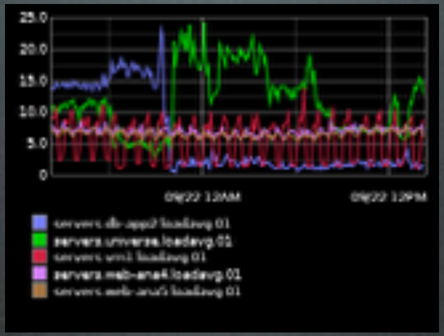


http://graphite3.colo.lair/render/?target=diffSeries(timeShift(servers.graphite3.loadavg.01,%221d%22),servers.graphite3.loadavg.01)

Shows the difference in 1-minute load on graphite3 today vs. the same times yesterday.

 * Potentially useful in defining checks that are more adaptive
 * Allows for monitoring amount of change from historic values rather than setting specific thresholds

Most Deviant



http://graphite3.colo.lair/render/?target=mostDeviant(5,servers.*.loadavg.01)

Shows the 5 most deviant (outlier) metrics from a grouping.

In this case it's the top 5 1-minute load averages across all servers.

Resources

- Jason Dixon’s blog (tips, tricks, tools)
  - http://obfuscurity.com/Tags/Graphite
- Graphite API Reference
  - https://graphite.readthedocs.org/en/latest/functions.html
- Clustering Graphite
  - http://bitprophet.org/blog/2013/03/07/graphite/

Questions?