Automatically Detecting Anomalies and Outliers in Real-Time

Homin Lee (@hominprovement)
Outline

- Monitoring
- Alerting
- Outlier vs. Anomaly Detection
- Outlier Detection Algorithms
- Anomaly Detection Algorithms
Monitor Everything
Monitor Everything
Datadog gathers performance data from all your application components.

AWS  Docker  CoreOS  Chef  Puppet  Github
Pagerduty  Nagios  Go  Postgres  Java  VMware
Redis  MySQL  Apache  Tomcat  MongoDB  New Relic
Monitor Everything
Monitor Everything
Monitor Everything?

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spidly consumer lag seconds by $az</td>
<td>1 minute behind</td>
</tr>
<tr>
<td>Spidly hosts normalized load by AZ</td>
<td></td>
</tr>
<tr>
<td>Query error rate (%)</td>
<td>too many errors</td>
</tr>
<tr>
<td>Avg of spidly.points_per_second over...</td>
<td>4M pps!</td>
</tr>
<tr>
<td>user (blue), iowait (orange), system...</td>
<td></td>
</tr>
<tr>
<td>System load (1, 15)</td>
<td></td>
</tr>
<tr>
<td>nr_dirty (orange), nr_mapped (grey)</td>
<td></td>
</tr>
<tr>
<td>Write/Read IOPs/sec</td>
<td></td>
</tr>
<tr>
<td>Shared Memory Usage</td>
<td>toast (52GB), worry above (35GB)</td>
</tr>
<tr>
<td>dirty pct (nr_dirty / nr_mapped)</td>
<td></td>
</tr>
<tr>
<td>spidly write throughput (points/s)</td>
<td></td>
</tr>
<tr>
<td>spidly.open_stores, spidly.open_files</td>
<td></td>
</tr>
<tr>
<td>spidly write throughput (messages/s)</td>
<td></td>
</tr>
<tr>
<td>postgres Master CPU</td>
<td></td>
</tr>
<tr>
<td>Avg of spidly.persist_queue_len over...</td>
<td></td>
</tr>
<tr>
<td>Avg of system.mem.cached over ro...</td>
<td></td>
</tr>
<tr>
<td>spidly store queries/sec</td>
<td></td>
</tr>
<tr>
<td>Avg of spidly.query_ms.95percentil...</td>
<td></td>
</tr>
<tr>
<td>spidly.write_retries, spidly.sharded...</td>
<td></td>
</tr>
<tr>
<td>query ms 95p by $az</td>
<td></td>
</tr>
</tbody>
</table>
Alerting

things go downhill when we hit this
Alerting?
Alerting?
Outlier and Anomaly Detection
Outlier Detection
Outlier Detection
Outlier Detection
Outlier Detection Algorithms

MAD
median absolute deviation from the median

DBSCAN
density-based spatial clustering of applications with noise
Robust Outlier Detection Algorithms
Median Absolute Deviation

$$\text{MAD}(D) = \text{median}( \{ |d_i - \text{median}(D)| \} )$$
Median Absolute Deviation

\[ \text{MAD}(D) = \text{median} \left( \{ \lvert d_i - \text{median}(D) \rvert \} \right) \]

\[ D = \{ 1, 2, 3, 4, 5, 6, 100 \} \]
Median Absolute Deviation

$$\text{MAD}(D) = \text{median}( \{ |d_i - \text{median}(D)| \} )$$

$$D = \{ 1, 2, 3, 4, 5, 6, 100 \}$$

\[ \text{median} = 4 \]
Median Absolute Deviation

\[ \text{MAD}(D) = \text{median}( \{ |d_i - \text{median}(D)| \} ) \]

\[ D = \{ 1, 2, 3, 4, 5, 6, 100 \} \]
\[ \text{median} = 4 \]
\[ \text{deviations} = \{ -3, -2, -1, 0, 1, 2, 96 \} \]
Median Absolute Deviation

\[ \text{MAD}(D) = \text{median}( \{ |d_i - \text{median}(D)| \} ) \]

\[ D = \{ 1, 2, 3, 4, 5, 6, 100 \} \]
\[ \text{median} = 4 \]
\[ \text{deviations} = \{ -3, -2, -1, 0, 1, 2, 96 \} \]
\[ \text{abs deviations} = \{ 0, 1, 1, 2, 2, 3, 96 \} \]
Median Absolute Deviation

\[
\text{MAD}(D) = \text{median}( \{ |d_i - \text{median}(D)| \} )
\]

\[D = \{ 1, 2, 3, 4, 5, 6, 100 \}\]
median = 4
deviations = \{ -3, -2, -1, 0, 1, 2, 96 \}
abs deviations = \{ 0, 1, 1, 2, 2, 3, 96 \}
MAD = 2
Median Absolute Deviation

$$MAD(D) = \text{median}( \{ |d_i - \text{median}(D)| \} )$$

$$D = \{ 1, 2, 3, 4, 5, 6, 100 \}$$

median = 4

deviations = \{ -3, -2, -1, 0, 1, 2, 96 \}

abs deviations = \{ 0, 1, 1, 2, 2, 3, 96 \}

MAD = 2  (std dev = 33.8)
Median Absolute Deviation

Parameters: Tolerance, Pct

\{ \text{tol.} = 3.0 \}
DBSCAN

Parameters:
epsilon, min_samples
DBSCAN

![DBSCAN Graph]

The graph above illustrates the behavior of DBSCAN for different values of $d/4$, $d/2$, $3d/4$, and $d$. The x-axis represents these values, and the y-axis shows the number of clusters identified. The data shows fluctuations and peaks across different iterations, indicating the dynamic nature of DBSCAN's clustering based on its distance thresholds.
DBSCAN

\[ \epsilon \sim \text{median(dist from median series)} \times \text{tolerance} \]
MAD or DBSCAN?
MAD or DBSCAN?
Some subtleties
Some subtleties
Some subtleties
Anomaly Detection
Anomaly Detection
Alerting

last 15m: 96% anomalous (threshold is > 90%)
Alerting
Alerting

EVALUATION WINDOW
(1 hour)

HISTORICAL CONTEXT: LAST WEEK

HISTORICAL CONTEXT: LAST 7 WEEKS
What's Normal?
What’s Normal?
What’s Normal?
What’s Normal?
Past Performance...

past 30 minutes

past day

past week

past 5 weeks
Anomaly Detection Alg 1: Robust
Decomposition
Decomposition
Decomposition
Decomposition
Stable Features
Robust Anomaly Detection
Robust Anomaly Detection
Robust Anomaly Detection
Robust Anomaly Detection
Robust Anomaly Detection
Anomaly Detection Alg 2: Adaptive
Anomaly Detection Alg 2: Adaptive
Parameters
Anomalies or Noise?
Future Work

Max of dd.rawls_hadoop.interval.finish...
Thanks!

Homin Lee

homin@datadoghq.com

@hominimprovement