The RED Method

Patterns for instrumentation & monitoring.

Tom Wilkie
tom@kausal.co  @tom_wilkie
github.com/tomwilkie
• Founder **Kausal**, “transforming observability”

• Prometheus developer

• Home brewer

Previously:

• Worked on Kubernetes & Prometheus at Weaveworks

• SRE for Google Analytics

• Founder/CTO at Acunu, worked on Cassandra
Introduction
Why does this matter?

The USE Method
Utilisation, Saturation, Errors

The RED Method
Requests Rate, Errors, Duration..

The Four Golden Signals
RED + Saturation
Introduction
The USE Method
For every resource, monitor:

- **Utilisation**: % time that the resource was busy
- **Saturation**: amount of work resource has to do, often queue length
- **Errors**: the count of error events

http://www.brendangregg.com/usemethod.html
<table>
<thead>
<tr>
<th></th>
<th>Utilisation</th>
<th>Saturation</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Memory</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Disk</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Network</td>
<td>✔</td>
<td>✔</td>
<td>✖</td>
</tr>
</tbody>
</table>
CPU Utilisation:

1 - \( \text{avg(rate(node_cpu\{\text{job="default/node-exporter",mode="idle"}\}[1m])))} \)

CPU Saturation:

\[
\frac{\text{sum(node_load1\{\text{job="default/node-exporter"}\})}}{\text{sum(node:node_num_cpu:sum)}}
\]
Memory Utilisation:

\[
1 - \frac{\sum(\text{node\_memory\_MemFree}\{\text{job}="\ldots"\} + \text{node\_memory\_Cached}\{\text{job}="\ldots"\} + \text{node\_memory\_Buffers}\{\text{job}="\ldots"\})}{\sum(\text{node\_memory\_MemTotal}\{\text{job}="\ldots"\})}
\]

Memory Saturation:

\[
1e3 * \frac{\sum(\text{rate}(\text{node\_vmstat\_pgpgin}\{\text{job}="\ldots"\}[1m]) + \text{rate}(\text{node\_vmstat\_pgpgout}\{\text{job}="\ldots"\}[1m]))}{\sum(\text{node\_memory\_MemTotal}\{\text{job}="\ldots"\})}
\]
• CPU Errors, Memory Errors
• Hard Disk Errors!
• Disk Capacity vs Disk IO
• Network Utilisation
• Interconnects
Demo Time
• “The USE Method” - Brendan Gregg

• KLUMPS - Kubernetes/Linux USE Method with Prometheus

  https://github.com/kausalco/public
The RED Method
For every service, monitor request:

- **Rate** - number of requests per second
- **Errors** - the number of those requests that are failing
- **Duration** - the amount of time those requests take
Ah, here's the controversial bit. RED as an alternative to @brendangregg's USE method @tom_wilkie @weaveworks
import (  
    "github.com/prometheus/client_golang/prometheus"
)

var requestDuration = prometheus.NewHistogramVec(prometheus.HistogramOpts{  
    Name: "request_duration_seconds",
    Help: "Time (in seconds) spent serving HTTP requests.",
    Buckets: prometheus.DefBuckets,
}, []string{"method", "route", "status_code"})

func init() {
    prometheus.MustRegister(requestDuration)
}
func wrap(h http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        m := httpsnoop.CaptureMetrics(h, w, r)
        requestDuration.WithLabelValues(r.Method, r.URL.Path,
            strconv.Itoa(m.Code)).Observe(m.Duration.Seconds())
    })
}

func server(addr string) {
    http.Handle("/metrics", prometheus.Handler())


    ...}))
}
Rate:
\[
\text{sum(rate(request\_duration\_seconds\_count\{job=\ldots\}\[1m]\))}
\]

Errors:
\[
\text{sum(rate(request\_duration\_seconds\_count\{job=\ldots\}, status\_code!~\"2..\"\}\[1m]\))}
\]

Duration:
\[
\text{histogram\_quantile(0.99, sum(rate(request\_duration\_seconds\_bucket\{job=\ldots\}\[1m]\)) by (le))}
\]
DAG of Services
Latencies & Averages
• “Monitoring Microservices” - Weaveworks (slides)
• “The RED Method: key metrics for microservices architecture” - Weaveworks
• “Monitoring and Observability with USE and RED” - VividCortex
• “RED Method for Prometheus – 3 Key Metrics for Monitoring” - Rancher Labs
• “Logs and Metrics” - Cindy Sridharan
• “Logging v. instrumentation”, “Go best practices, six years in” - Peter Bourgon
The Four Golden Signals
Golden Signal
Boost Signal
Up To the Maximum Limit

Improved New Technology
Made in USA
For each service, monitor:

- **Latency** - time taken to serve a request
- **Traffic** - how much demand is placed on your system
- **Errors** - rate of requests that are failing
- **Saturation** - how “full” your services is
• **Saturation** - how “full” your services is
Demo
Time
• “The Four Golden Signals” - The Google SRE Book
• “How to Monitor the SRE Golden Signals” - Steve Mushero
Summary
Thanks! Questions?