The use of proxy indicators in service management

javacodegeeks.com/2017/06/use-proxy-indicators-service-management.html

Operating any non-trivial IT-service is a complex exercise. There are multiple aspects one needs to bear in mind while building a good operations team. One of such aspects is transparency in the service level quality. In order to avoid confusion and blame games, every good operations team has agreed upon certain quality level the service operated must be matching.

The quality is measured using different indicators and compared against expectations. When the measured value deviates from the expectation, you will have a strong signal that you need to take action upon. To give you an idea, the indicators used to express the operational quality tend to take forms similar to the following examples:

- Invoice generator must be able to generate 10,000 invoices/hour 99% of the time
- End user interactions in the e-shop must complete without errors 99.9% of the time
- Product search must respond in under 1,500ms 99% of the time and in under 5,000ms 99.95% of the time.

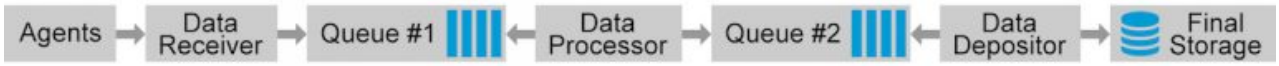
The three examples above express expectations on service quality in three different terms. Each term sets an expectation on a specific indicator: first one focuses on throughput, second expresses the accepted error rate and third is a requirement to latency.

However, there are situations where direct measurement of the desired indicator is hard or impossible. In such cases, **proxy indicators** are used as an indirect measure that represents a phenomenon in the absence of a direct measure.

In this post I am going to use a real world situation from our own operations and will use this as an example to explain the concept of a **proxy indicator** in IT operations.

Proxy indicators in action: an example

To understand the example, let's start with some background of the problem we faced. The problem originates from how Plumbr is accepting and processing the monitoring data received from our Agents. The simplified process of this processing is visible below:



Our data processing pipeline is divided into multiple microservices, accepting, processing and storing the data in different phases. Each of the microservices is an independent batch process with single responsibility.

The problem at hand was related to monitoring the processing pipeline. We needed to monitor how quickly are we able to make the data arriving to our servers available for querying from the storage.

The requirement to the process was expressed as "99% of the transactions arriving to Data Receiver must be processed in under 1 minute. 99.9% of the transactions must be processed in under 5 minutes."

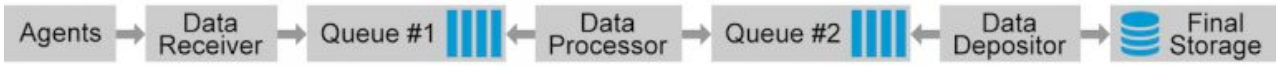
The requirement is straightforward and should be thus usable as the indicator of whether we are fulfilling the requirement. As it turned out, the way the microservices were built, we could not directly measure the latency of the entire processing stream.

The problem originated in data being processed in batches, making tracking of the individual packets of monitoring

data impossible within the current solution. So we needed a different indicator to make sure our data processing pipeline is not stalling and failing to deliver according to expected requirements.

Analyzing the incidents we had to solve during the past months, we discovered a possibility to monitor the pipeline differently and use **proxy indicators to capture the signal of processing delays being present**.

Looking again at the processing pipeline, you can notice the two queues decoupling the different microservices:



What we found was that we can monitor the length of the queues to capture a signal when either the Queue #1 or #2 has became too long. For example, whenever the Data Processor service would not able to keep up with the pace the data is arriving to Queue #1, the length of the Queue #1 starts growing, indicating a bottleneck.

Introducing the monitoring on queue length already covered 90% of the situations where our processing was lagging behind. But as you might have already noticed, this would not signal us whenever the bottleneck is occurring after the last queue, so as the last line of defence we also introduced monitoring on the amount of data we stored in the final storage, to make sure the Data Depositor microservice is not the source of the problems.

So we ended up with following proxy indicators being used instead of the original end-to-end latency requirement:

- The length of the queues must not exceed 50,000 for 99% of the time and 500,000 for 99.95% of the time.
- The final storage must store between 250,000 and 1,000,000 events in any minute for 99.95% of the time

Monitoring for these simple indicators ended up signaling us correctly on 98%+ situations where the data processing was not performing according to the latency requirements.

What we also ended up discovering was that the problems in this front most often occurred when some of our customers suddenly started sending us unexpectedly large amounts of data. So based on the signals we were able to automatically spawn a dedicated service to process data from such customers. This isolated the impact, meaning that the sudden surge in traffic would not introduce a delay in processing of other customers' data. It also introduced additional processing power to digest the data from the problematic customer, thus resolving the (temporary) processing delay faster for the particular customer.

Take-away

The example above described a specific use case you might not face anytime soon. But the concept of proxy indicators is more generic and can be deployed in numerous situations. For example, you might face a requirement related to client-side latency, but it might only be possible to measure latency at the server. Monitoring the server-side latency would now work as a proxy indicator to expose the situations where the server-side component is the source of performance issues.

Do not get me wrong – whenever a direct indicator is measurable, you should always prefer this approach and avoid proxy indicators as the source for signal or quality measurement altogether. However, in the situations where you cannot directly monitor the actual indicator, understanding and using the proxy indicators is a valuable concept to keep in your toolbelt.

Reference: The use of proxy indicators in service management from our JCG partner Ivo Mägi at the Plumbr Blog blog.