**What is the paper about?**

This paper tells us definition of monitoring distributed systems in details and provides deep comparison of monitoring types.

**What is monitoring?**

The monitoring of distributed systems involves Collecting, processing, aggregating,

and displaying real-time quantitative data about a system, such as

query counts and types, error counts and types, processing times,

and server lifetimes. This information and its display can support the debugging, testing, performance evaluation, and dynamic documentation of distributed systems.

**Why monitor a system in the first place?**

Monitor a system should be in the first place, because if an error occurs, the developer will be notified immediately and can fix the problem at once. Monitoring helps in the following things:

* Analysing long-term trends
* Comparing over time or experiment groups
* Alerting data
* Building dashboards

**Explain the 4 golden signals of monitoring.**

The four golden signals are the following:

1. **Latency**

The time it takes to service a request. It’s important to distinguish between the latency of successful requests and the latency of failed requests. It’s important to track error latency, as opposed to just filtering out errors. For example, For server-side latency, can be a good way to monitor response time.

1. **Traffic**

A measure of how much demand is being placed on your system, measured in a high-level system-specific metric. It’s a key signal because it helps you differentiate capacity problems from improper system configurations that can cause problems even during low traffic. For distributed systems, it can also help you plan capacity ahead to meet upcoming demand.

1. **Errors**

Errors can tell you about misconfigurations in your infrastructure, bugs in your application code, or broken dependencies. The error message will give you more information on the exact problem. Errors can also affect the other metrics by artificially lowering latency or repeated retries that end up saturating other distributed systems.

1. **Saturation**

Saturation defines the load on your network and server resources. Every resource has a limit after which performance will degrade or become unavailable. This applies to resources like CPU utilization, memory usage, disk capacity, and operations per second. It takes understanding of your distributed system design and experience to know which parts of your service could become saturated first. Often, these metrics are leading indicators, so you can adjust capacity before performance degrades.

**According to the paper, how do you do monitoring? What is important?**

According to the paper, in order to perform a useful monitoring a good starting point will be to answer on the following questions: what’s broken, and why?

It’s important to identify which type of monitoring you need to apply for your system. For example, you think about black-box monitoring versus white-box monitoring is that black-box monitoring is symptom-oriented and represents active—not predicted—problems: "The system isn’t working correctly, right now." White-box monitoring depends on the ability to inspect the innards of the system, such as logs or HTTP endpoints, with instrumentation. White-box monitoring therefore allows detection of imminent problems, failures masked by retries, and so forth.

For example, suppose that a database’s performance is slow. Slow database reads are a symptom for the database SRE who detects them. However, for the frontend SRE observing a slow website, the same slow database reads are a cause. Therefore, white-box monitoring is sometimes symptom-oriented, and sometimes cause-oriented.

On the flip-side, black-box monitoring is focused on server metrics like disk space, CPU, memory, load, etc., which are typically thought of as the core monitoring metrics, and understanding performance from the end user’s perspective. Black-box monitoring is used to understand why something within the system is not working correctly.

**What approach would you use for your lab: White-box or Black-box monitoring? Why?**

I thing for my project will be useful the white-bow monitoring. White-box monitoring concerns itself with understanding how your applications run on the server. The metrics measured could be monitoring HTTP requests, response codes, user metrics, etc. Think of white box monitoring as a window into the internal system. White-box monitoring is used to understand or predict why something may fail.

Because my microservices system use Rest API. It will be more informative to choose white-box monitoring system.

**What happened with Bigtable SRE and how did they "fix" the situation?**

The Bigtable service’s SLO was based on a synthetic well-behaved client’s mean performance. Because of problems in Bigtable and lower layers of the storage stack, the mean performance was driven by a "large" tail: the worst 5% of requests were often significantly slower than the rest.

In order to fix situation, the team used a three-pronged approach: while making great efforts to improve the performance of Bigtable, we also temporarily dialled back our SLO target, using the 75th percentile request latency. We also disabled email alerts, as there were so many that spending time diagnosing them was infeasible.