Alerting with SLOs and error budgets

Nick Apostolakis

30th of August 2020

What are the objectives of this presentation

At the end of this presentation we should be able to answer the following questions:

- How we can define availability

What are the objectives of this presentation

At the end of this presentation we should be able to answer the following questions:

- How we can define availability
- What is an error budget and how it is connected to an SLO

What are the objectives of this presentation

At the end of this presentation we should be able to answer the following questions:

- How we can define availability
- What is an error budget and how it is connected to an SLO
- How can we use SLOs error budgets to make decisions

What are the objectives of this presentation

At the end of this presentation we should be able to answer the following questions:

- How we can define availability
- What is an error budget and how it is connected to an SLO
- How can we use SLOs error budgets to make decisions
- How can we use error budgets to define alerting strategy

What are the objectives of this presentation

At the end of this presentation we should be able to answer the following questions:

- How we can define availability
- What is an error budget and how it is connected to an SLO
- How can we use SLOs error budgets to make decisions
- How can we use error budgets to define alerting strategy
- What are the most common alerting strategies we can use based on error budgets

Defining availability

In order to use error budgets we need first to define availability. In the SRE world this can only be done by defining an SLI, its SLO and its SLA.

It is a mistake to define an availability goal of 100%. By definition this allows no room for error, therefore no room for change/improvement.

There are two ways to define availability.

Time based availability

Time based availability which can calculated by using the formula:

$$availability = \frac{uptime}{uptime + downtime}$$

Aggregate availability

Or aggregate availability which depends on request success rate and in this case the availability is calculated over a rolli ng window. The formula used for this type of availability is

$$availability = \frac{succesfulRequests}{totalRequests}$$

based on the above formulas, SLIs, SLOs and SLAs are always expressed as a percentage.

A service SLOs are documented in an SLO document that also contains the error budget policy and an example of this can be found here.

This document is then communicated to all the stakeholders and is reviewed every few months to ensure that if meets the needs of our customers.

What is an error budget

When we are defining an SLO, we define an availability target for our service. If we say we want a service to be 99% available this means that we expect our service to be 1% unavailable and this is acceptable.

This remaining margin for error is our error budget. Simply put our error budget is:

$$errorBudget = 100 - SLO$$

Defining and maintaining error budgets

As with the SLO definition, the error budget definition and policy needs to be properly documented, communicated to the key stakeholders (the product manager, the development team, and the SREs) and generally be approved.

This policy will not only define what the margin of error can be but what we do when our service runs out of budget. If that happens then we need to have a plan about how to improve the stability of our service and enforce the error budget.

To make error budget enforcement decisions, you need to start with a written policy. This policy should cover the specific actions that must be taken when a service has consumed its entire error budget for a given period of time, and specify who will take them. Common owners and actions might include:

- The development team gives top priority to bugs relating to reliability issues over the past four weeks.
- The development team focuses exclusively on reliability issues until the system is within SLO. This responsibility comes with

Making decisions using SLOs and Error budgets

Using SLOs and Error budgets can help us make decisions.

In its most basic form, it help us define what happens when we miss our SLOs, meaning when we exhaust our error budget. There are a number of common options such as

- stopping feature launches until the service is once again within SLO or
- devoting some or all engineering time to working on reliability-related bugs.

Apart from using SLOs and error budgets to take decisions about the future development of our project, we can use SLOs and error budgets to define our monitoring and alerting strategy.

We can track several SLIs and monitor several SLOs, but which ones do we alert on, and when do we alert?

This is the main question that we need to answer, and using SLOs and error budgets gives us a powerful tool to analyse the behaviour of our system and assign significance to alert events.

Alerting quality criteria

In order to generate alerts from an SLO and an error budget, we need a way to combine these two elements into a specific rule. Our goal is to be notified for a **significant event: an event that consumes a large fraction of the error budget**.

We need to consider the following attributes when evaluating an alerting strategy:

- *Precision* The proportion of events detected that were significant. Precision is 100% if every alert corresponds to a significant event.

Alerting quality criteria

In order to generate alerts from an SLO and an error budget, we need a way to combine these two elements into a specific rule. Our goal is to be notified for a **significant event: an event that consumes a large fraction of the error budget**.

We need to consider the following attributes when evaluating an alerting strategy:

- *Precision* The proportion of events detected that were significant. Precision is 100% if every alert corresponds to a significant event.
- *Recall* The proportion of significant events detected. Recall is 100% if every significant event results in an alert.

Alerting quality criteria

In order to generate alerts from an SLO and an error budget, we need a way to combine these two elements into a specific rule. Our goal is to be notified for a **significant event: an event that consumes a large fraction of the error budget**.

We need to consider the following attributes when evaluating an alerting strategy:

- *Precision* The proportion of events detected that were significant. Precision is 100% if every alert corresponds to a significant event.
- *Recall* The proportion of significant events detected. Recall is 100% if every significant event results in an alert.
- *Detection time* How long it takes to send notifications in various conditions. Long detection times can negatively impact the error budget.

Alerting quality criteria

In order to generate alerts from an SLO and an error budget, we need a way to combine these two elements into a specific rule. Our goal is to be notified for a **significant event: an event that consumes a large fraction of the error budget**.

We need to consider the following attributes when evaluating an alerting strategy:

- *Precision* The proportion of events detected that were significant. Precision is 100% if every alert corresponds to a significant event.
- *Recall* The proportion of significant events detected. Recall is 100% if every significant event results in an alert.
- *Detection time* How long it takes to send notifications in various conditions. Long detection times can negatively impact the error budget.
- *Reset time* How long alerts fire after an issue is resolved. Long reset times can lead to confusion or to issues being ignored.

Google SRE alerting strategies 1

The most simple alerting strategy is when Error rate exceeds SLO. We can choose a small time window (e.g 10minutes) and alert if the error rate over that window exceeds the SLO.

For example, if the SLO is 99.9% over 30 days, alert if the error rate over the previous 10 minutes is ≥ 0.1:

```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate10m{job="myjob
```

The 10 minute average can be calculated by using the following Recording Rule in Prometheus:

```
record: job:slo_errors_per_request:ratio_rate10m
expr:
  sum(rate(slo_errors[10m])) by (job)
    /
  sum(rate(slo_requests[10m])) by (job)
```

We can calculate the error budget consumed when this alert fires by the following formula:

Google SRE alerting strategies 1 cont

This first approach is fairly simplistic and has the following pros and cons:

Pros:

- Good detection time
- Good recall

Cons:

- Low precision. It fires for all errors, even ones that do not threaten the SLO
- It produces a large volume of alerts increasing the noise in the alert channels and leading to alert fatigue

Using this approach we could have up to 144 alerts per day, do not act on any of them and still meet the SLO. We can do a lot better than this.

Google SRE alerting strategies 2

We can improve the precision of the previous method by trying to increase the window size and capture more errors consuming more error budget and therefore increasing the significance of the event.

Lets say that we notify only if an event consumes 5% of the 30 day error budget in a 36 hour window, with a 100% error rate:

```
- alert: HighErrorRate
   expr: job:slo_errors_per_request:ratio_rate36h{job="myjo
```

In this case the detection time is:

$$detectionTime = \frac{1 - SLO}{errorRatio} * alertingWindowSize$$

In this case the detection time for our example is:

$$detectionTime = \frac{100 - 99.9}{100} * 36 = \frac{0.1}{100} * 36h = 0.036h = 2.16min$$

Google SRE alerting strategies 3

Instead of the previous approach we can alert if our errors remain above our SLO threshold for a specific duration:

```
- alert: HighErrorRate
    expr: job:slo_errors_per_request:ratio_rate1m{job="myjo
    for: 1h
```

This approach has a higher precision, but it does have poor recall and poor detection time.

A 100% outage will be treated with the same significance with a 0.2% outage but the 100% outage will consume our full error budget by the time that the alert is executed.

Also if the service recovers in the duration of the 1 hour then the counter resets itself so you may have periodic outages of 100% during that 1 hour window and the alert never fires.

For the above reasons using duration for alerting should be avoided.

The figure below shows the average error rate over a 5-minute

Google SRE alerting strategies 4

We can improve on all the previous strategies. Instead of depending on the alert window or the duration, we can use the error budget consumption rate as the method to detect which even is significant or not.

We can do that by introducing a burn rate into our calculations. In this case our formulae become:

$$detectionTime = \frac{1 - SLO}{errorRatio} * alertingWindowSize * burnRate$$

and the error budget consumed is

$$consumedErrorBudget = \frac{burnRate * alertingWindowSize}{alertingPeriod}$$

or

$$burnRate = \frac{consumedErrorBudget * alertingPeriod}{alertingPeriod}$$

Google SRE alerting strategies 4 cont

By defining how much error budget we want to consume we can derive the burn Rate and then the detection time. For an SLO of 99.9%, an error ratio of 6%,an alerting window of 1 hour, an alerting period of 30 days (or 720 hours) and a consumed error budget of 5% we have the following:

$$burnRate = \frac{0.05 * 720h}{1h} = 36$$

so for our burn rate of 36 our alert will be triggered in:

$$detectionTime = \frac{100 - 99.9}{0.06} * 1 * 36h \approx 58min$$

The alerting rule for this is:

```
- alert: HighErrorRate
  expr: job:slo_errors_per_request:ratio_rate1h{job="myjob"
```

With this method we have good precision, shorter time window,

Google SRE alerting strategies 5

We can improve our alerting logic by using multiple burn rates and time windows, and fire alerts when burn rates surpass a specified threshold. This option retains the benefits of alerting on burn rates and ensures that you don't overlook lower (but still significant) error rates.

It's also a good idea to set up ticket notifications for incidents that typically go unnoticed but can exhaust your error budget if left unchecked—for example, a 10% budget consumption in three days. This rate of errors catches significant events, but since the rate of budget consumption provides adequate time to address the event, you don't need to page someone.

Google recommends 2% budget consumption in one hour and 5% budget consumption in six hours as reasonable starting numbers for paging, and 10% budget consumption in three days as a good baseline for ticket alerts. The appropriate numbers depend on the service and the baseline page load. For busier services, and depending on on-call responsibilities over weekends and holidays, we

Google SRE alerting strategies 5 cont 1

This alerting strategy can be implemented with the following rules:

```
expr: (
        job:slo_errors_per_request:ratio_rate1h{job="myjob"
      or
        job:slo_errors_per_request:ratio_rate6h{job="myjob"
      )
severity: page

expr: job:slo_errors_per_request:ratio_rate3d{job="myjob"}
severity: ticket
```

Where the various burn rates are combined using logical operators, in this case OR. The detection time according to burn rate can be seen in the following diagram:

Google SRE alerting strategies 5 cont 2

This method has a lot of pros, but also has a few cons.

It is adaptable, can alert quickly, if the error rate is high or low, has good precision has good recall and it has routing to the appropriate alert type based on how quickly someone has to react.

Its cons are, complexity, a long reset time, and it needs alert suppression to avoid multiple alerts firing if more than one conditions are true. We can improve this last one in the next iteration.

Google SRE alerting strategies 6

We can enhance the multi-burn-rate alerts in previous strategy to notify us only when we're still actively burning through the budget—thereby reducing the number of false positives.

To do this, we need to add another parameter: a shorter window to check if the error budget is still being consumed as we trigger the alert. A good guideline is to make the short window 1/12 the duration of the long window.

For example, you can send a page-level alert when you exceed the 14.4x burn rate over both the previous one hour and the previous five minutes. This alert fires only once you've consumed 2% of the budget, but exhibits a better reset time by ceasing to fire five minutes later, rather than one hour later:

```
expr: (
        job:slo_errors_per_request:ratio_rate1h{job="myjob"
      and
        job:slo_errors_per_request:ratio_rate5m{job="myjob"
      )
```

References

- https://landing.google.com/sre/sre-book/chapters/embracing-risk/#id-AnCDFmtB
- https://landing.google.com/sre/workbook/chapters/implementing-slos/
- https://landing.google.com/sre/workbook/chapters/implementing-slos/#decision-making-using-slos-and-error-budgets
- SLO Document example: https://landing.google.com/sre/workbook/chapters/slo-document/
- https://landing.google.com/sre/workbook/chapters/alerting-on-slos/
- https://engineering.bitnami.com/articles/implementing-slos-using-prometheus.html
- https://github.com/confluentinc/training-cao-src/tree/master/solution/mo-gen/prometheus/grafana/dashboards

Questions?