September 2021

Automated Software Rollouts Based On Confidence Intervals

Joe Turner

Leila Al-Muhtasib

Automated Software Rollouts Based On Confidence Intervals

ABSTRACT

Canarying is a technique for software rollout that attempts to ensure that performance metrics do not regress below a threshold due to the rollout. The updated software version is rolled out to a fixed percentage of users or data centers, and metrics are checked to ensure that deviations in performance are below the fixed threshold. While the current mechanisms can eliminate major regressions, they are not flexible enough to prevent medium-sized regressions from occurring. This disclosure describes techniques for creating smarter canaries for software rollouts using confidence intervals and feedback loops. The techniques described herein enable dynamic rollouts and ensure that performance metrics do not regress to even a small degree.

KEYWORDS

- Software rollout
- Automated Rollout
- Confidence Interval
- Reliability Engineering
- Software version
- Software release
- Performance regression
- Canarying

BACKGROUND

When performing software rollouts/releases, an important objective is to ensure that performance metrics do not regress below a threshold due to the rollout. Generally, rollouts may be "canaried", where the updated software version is rolled out to a specific percentage of users or data centers (depending on the type of software being updated). Subsequent to initiating the rollout, metrics are checked to ensure that deviations in performance are below the threshold. However, at present, the threshold and the canary percentage (% of users/data centers that receive the updated version) are fixed. This leaves the possibility for medium-sized regressions with eventual rollout. While such a practice ensures that major regressions do not occur due to the rollout, such rollouts are not sufficiently accurate or flexible to prevent medium or small regressions.

DESCRIPTION

This disclosure describes techniques for creating smarter canaries for software rollouts using confidence intervals and feedback loops. The techniques described herein can be used to perform dynamic canaried rollouts and ensure that performance metrics do not regress to even a small degree. Existing canarying technologies just verify that the raw difference between a canary version and a control version is below a threshold (as per the specified risk tolerance). However, this measured value is ever-changing, and therefore, whether the metric value is good or bad completely depends on the time window chosen for comparison.

Per techniques of this disclosure, the concept of confidence intervals is used to reduce the possibility of the chosen metric regressing beyond the acceptable threshold during rollout. Confidence intervals are computed using the variability of the observed data. These indicate a range where the true effect on the metric is supposed to lie (with some percent confidence,

typically 95%). The confidence interval can be used to determine the worst-case scenario for a given metric with a high (typically 95%) degree of confidence. A feedback loop is used to automate rollout of the canaried version to larger fractions of users/servers based on the upper confidence interval of the regression in the chosen performance metric.

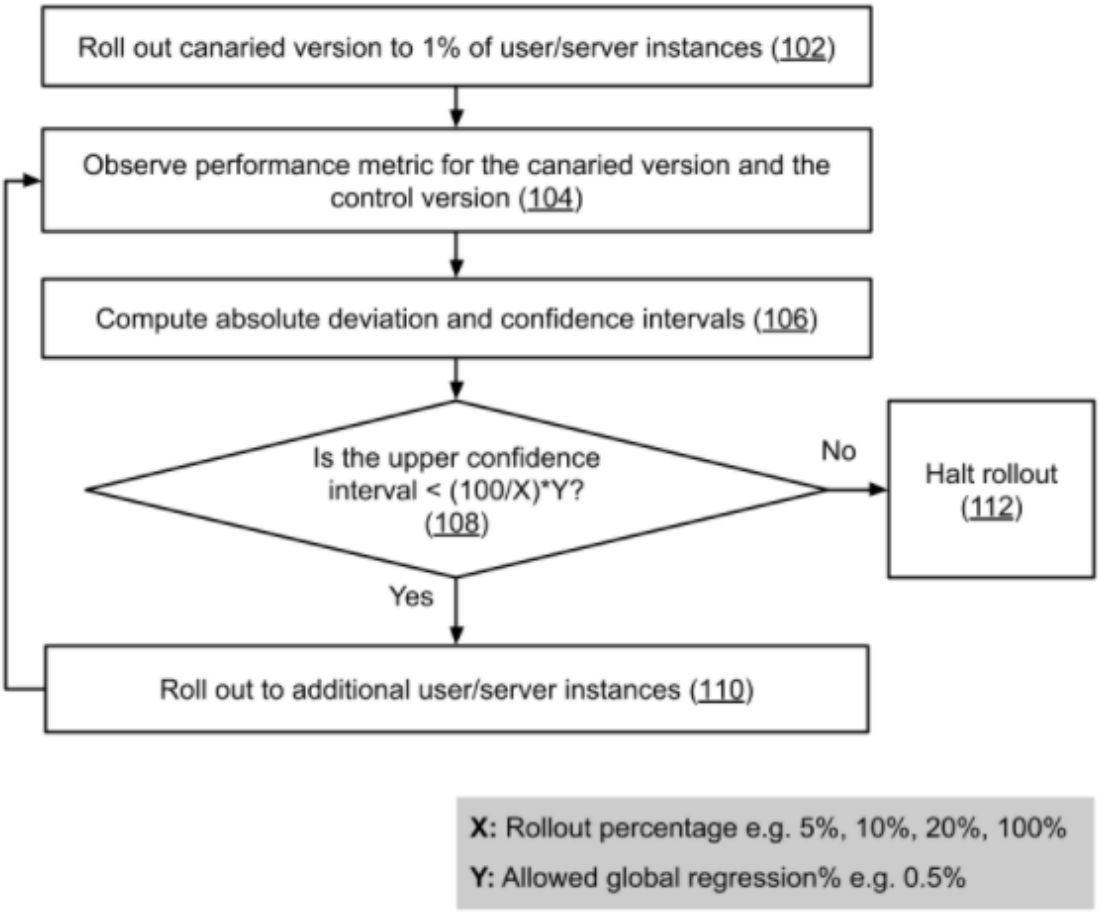

Fig. 1: Example process for automated rollout without performance regression

Fig. 1 illustrates an example process for automated rollouts as per the techniques described in this disclosure. The process is set up with a chosen allowed regression percentage-Y% - as the threshold that is utilized for automated rollout decisions. At the start of the process, the canaried version is rolled out to 1% (or other suitably low value) of users or server instances

(102). The chosen performance metric for the canaried version and the control version is observed (104).

The absolute deviation between performance of the canaried version and the control version, and corresponding confidence intervals are computed (106). To roll out the canaried version to X% of the server/user instances, the upper limit of the confidence interval is compared with (100/X)*Y (108). If the upper confidence interval is strictly less than (100/X)*Y, the canaried version is rolled out to X% of server/user instances (110), and its performance is monitored. For instance, at X=100%, this is trivially Y%, because Y was the allowed global regression percentage. For X=10%, this would be 10*Y, because when only 10% of users are affected, a regression of Z % on that 10% equates to a global regression of 10*Z. If the condition is not met, the rollout is halted (112), optionally reverting to the control version. Using this formula, the rollout can then proceed dynamically. Instead of using some fixed time interval for the canary phase, we wait until the desired confidence interval width is found.

While the foregoing discussion refers to a fully continuous rollout, it is possible to utilize multiple steps that the rollout ramps to, with each step corresponding to a specific percentage along the way to 100%. The specific step sizes can be configured as appropriate. For example. A large number of steps (e.g. 20+) may be suitable in many rollout contexts. In stepped rollout, the proportion of additional user/server instances (in step 110 of Fig. 1) is selected based on the next step.

A key advantage of the automated rollout mechanism as proposed herein is that the rollout can be managed with high precision. Per the described mechanism, rollouts provide a better guarantee of stability but are not associated with a fixed total time over which the updated version becomes available. Whereas previous rollout techniques may need to use a somewhat

high threshold for "error rate" increase (e.g., 50%), the described mechanism can use a smaller variance threshold (e.g. 10%). While the automated rollout may take additional time, stability is guaranteed. This tradeoff between speed and reliability is particularly useful for critical systems. The described techniques are suitable for rolling out updated binaries to client devices or to servers.

CONCLUSION

This disclosure describes techniques for creating smarter canaries for software rollouts using confidence intervals and feedback loops. The techniques described herein enable dynamic rollouts and ensure that performance metrics do not regress to even a small degree.