Analysis: So Far

July 22, 2013

1 What's New

Basic idea #1 is the straightforward observation that when looking for time series changepoints, we actually know exactly where to look - the time when the config change occurred. Previously, I was computing changepoints of each time series and attempting to use some window after a config change to find "related" changes. This is a bit clumsy: we don't have an indication of the "magnitude" of the change.

Looking slightly deeper at changepoint detection algorithms, it turns out some first compute a probability of a changepoint at each timestep, then pick points from the resulting time series (e.g., by thresholding). That is, for each metric, we can compute $P_{commit,node,metric}$. Currently, it is summed over a window, which may or may not make sense depending upon the test.

Basic idea #2 is a heuristic: if a time series change is indeed related to a config change, we expect to see time series changes in the metrics on nodes where the change had an effect. Initially I thought of using Jaccard similarity between the set of changed nodes (with a 1 for each time series on that node: $C_{commit,node}$) and metrics with changes, but P is continuous! I'm currently using Pearson's r instead (abusing a mix of notation):

$$r_{commit} = \text{cor.test}(P_{commit,node,metric}, \{C_{commit,node} \forall metric\})$$

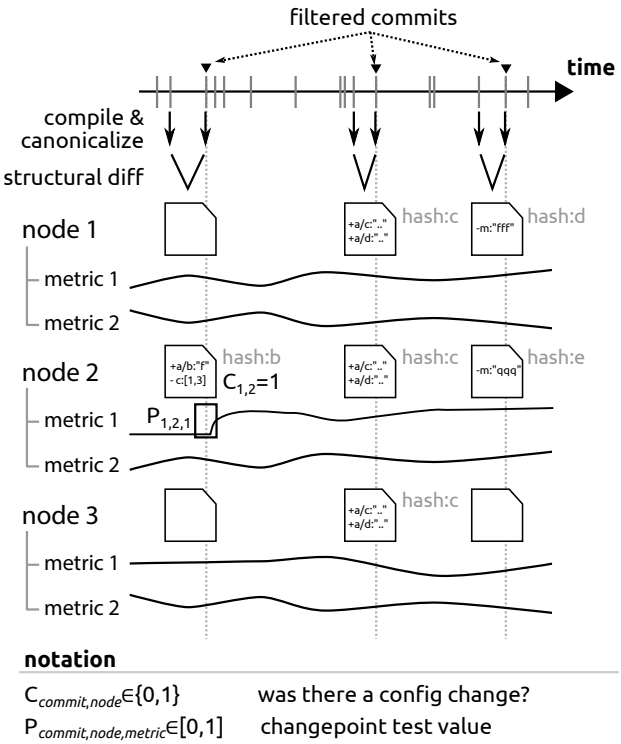

Figure 1: Where we're at: analysis flow

2 The result

This output is similar to before: a ranked list of config changes, with a ranked list within each of time series metrics that are most strongly correlated with that configuration change.

3 Knobs to twiddle

- What to use as the changepoint probability metric
- What to use for accumulating P (e.g., choice of window size)
- What to use for correlation between config changes and metric change strength
  - The technique itself
  - Going beyond change presence/absence and distinguishing between different change signatures (see hash:* example in Figure 1)