The perf dashboard is unmaintained and currently not active

Introduction

Performance Dashboard does continuous monitoring of performance characteristics of the Go implementation. It notifies codereview threads about any significant changes caused by the commit, allows to see performance changes caused by recent commits, allows to investigate changes in detail .

Builders

The dashboard uses two builders: linux-amd64 running Ubuntu 14.04 and windows-amd64 running Windows 8.1. Both builders has the same hardware: 2 x Intel Xeon E5620 @ 2.4GHz, 8 HT cores, 12GB RAM.

Benchmarks

The builders run benchmarks from the x/benchmarks repo:

- `json` : marshals and unmarshals large json object, in several goroutines independently.
- `http` : http client and server serving "hello world", uses persistent connections and read/write timeouts.
- `garbage` : parses net package using go/parser, in a loop in several goroutines; half of packages are instantly discarded, the other half is preserved indefinitely; this creates significant pressure on the garbage collector.
- `build` : does 'go build -a std'.

Metrics

Metrics collected are:

- `allocated` : amount of memory allocated, per iteration, in bytes
- `allocs` : number of memory allocations, per iteration
- `cputime` : total CPU time (user+sys from time Unix utility output), can be larger than time when GOMAXPROCS>1, per iteration, in ns
- `gc-pause-one` : duration of a single garbage collector pause, in ns
- `gc-pause-total` : total duration of garbage collector pauses, per iteration, ns
- `latency-50/95/99` : request latency percentile, in ns
- `rss` : max memory consumption as reported by OS, in bytes
- `sys-gc` : memory consumed by garbage collector metadata (`MemStats.GCSys`), in bytes
- `sys-heap` : memory consumed by heap (`MemStats.HeapSys`), in bytes
- `sys-other` : unclassified memory consumption (`MemStats.OtherSys`), in bytes
- `sys-stack` : memory consumed by stacks (`MemStats.StackSys`), in bytes
- `sys-total` : total memory allocated from OS (`MemStats.Sys`), in bytes
- `time` : real time (essentially the same as std Go benchmarks output), per iteration, in ns
- `virtual-mem` : virtual memory consumption as reported by OS, in bytes

And for build benchmark:

- `binary-size` : size of the go command, in bytes
- `build-cputime` : CPU time spent on the build, in ns

- `build-rss` : max memory consumption of the build process as reported by OS, in bytes
- `build-time` : real time of the build, in ns

Profiles

The dashboard also collects a set of profiles for every commit, they are available from the details page. For usual benchmarks CPU and memory profiles are collected. For build benchmark - perf profile, per-process split of CPU time and per-section size.

Perf Changes View

The view allows to see aggregate information about significant performance changes caused by recent commits.

Rows:

- The first row shows difference between the latest release and tip.
- The rest of the rows show deltas caused by individual commits.

Columns:

- The first column is commit hash.
- Second - number of benchmarks that were executed for the commit to far.
- Third - metric name, or the special 'failure' metric for build/runtime crashes.
- Fourth - negative deltas.
- Fifth - positive deltas.
- The rest describe commit.

You can click on any positive/negative delta to see details about the change.