Optimize Unicorn Workers in a Ruby on Rails App

If your Rails app is leaking memory - Unicorn will make it worse.

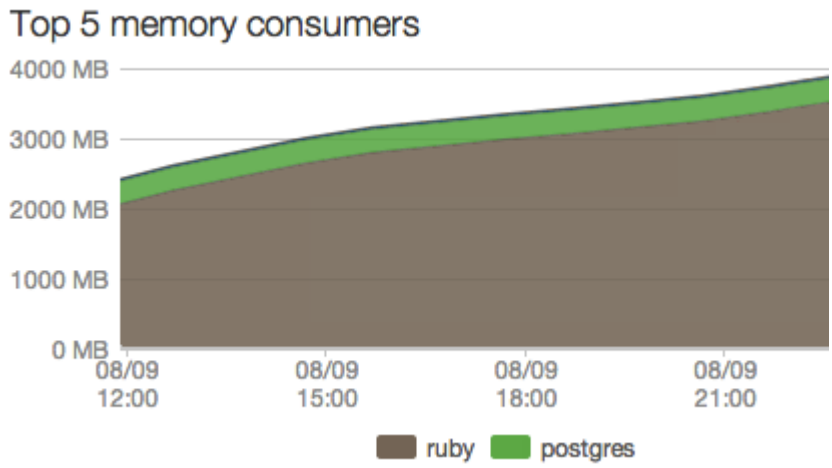
Unicorn uses forked processes to achieve concurrency. Forked processes are essentially copies of each other. Rails apps running on Unicorn tend to consume much more memory.

When a child process is forked, it is the exact same copy as the parent process. However, the actual physical memory copied need not be made. Since they are exact copies, both child and parent processes can share the same physical memory. Only when a write is made-- then we copy the child process into physical memory.

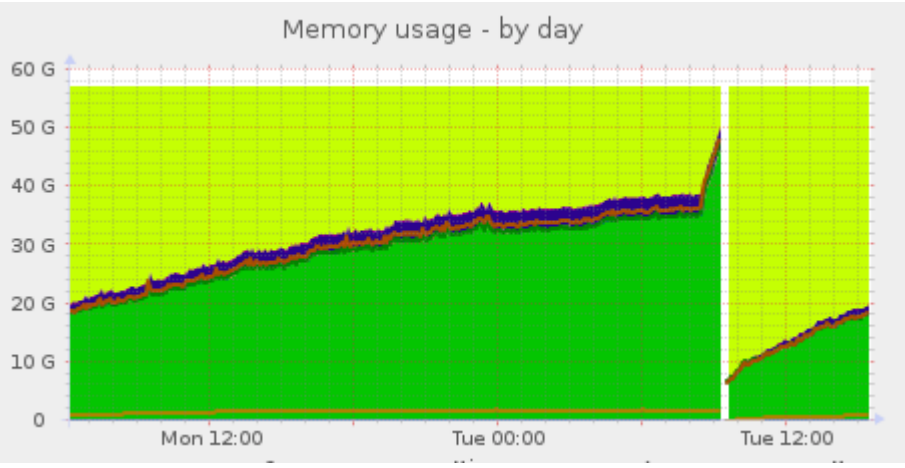
Each of these fork processes consumes memory because they are copies of the Rails application. While having more workers would mean that our application could handle more incoming requests, but we are tied to the amount of physical memory in our system.

It is easy for a rails app to leak memory.

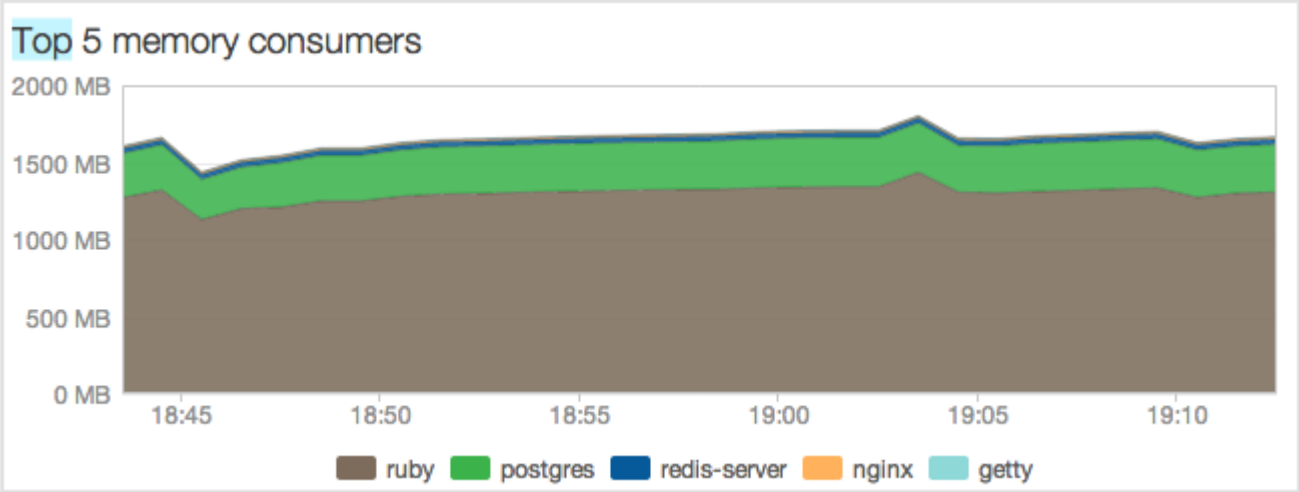
This is an example of a Rails application running Unicorn with memory leaks:



This is our memory usage:



Should be like:



Solution: gem unicorn-worker-killer

unicorn-worker-killer gem provides automatic restart of Unicorn workers based on max number of requests, and process memory size (RSS), without affecting any requests. This will greatly improves site's stability by avoiding unexpected memory exhaustion at the application nodes.

With the unicorn-worker-killer gem, we can also make entries in the logfile every time a worker is restarted.

https://github.com/kzk/unicorn-worker-killer