Resolving Out-of-Memory Issues

Occasionally larger Gatsby sites, or moderately-sized sites with unusual characteristics, run into out-of memory errors during `gatsby build` or `gatsby develop` processes.

Depending on the frequency of errors, out-of-memory errors can make it difficult or impossible to deploy new versions of your site.

This document will explain what causes out-of-memory (OOM) issues in Gatsby sites and how to fix or mitigate them.

How process memory usage works

Gatsby build and develop are Node.js processes, which are allocated some fixed amount of memory by the operating system. Your build has to stay below this limit at all times, otherwise the process will crash.

The amount of memory you're using is the sum of your baseline memory usage, which is proportional to your site size, and memory usage "spikes" which happen during certain memory-intensive parts of the build process. So the maximum amount of memory you'll use is equivalent to your baseline usage plus the max spike, and your goal is to keep:

Allocated memory < baseline usage + max(spike)

(A good analogy for this is an electrical power grid; the amount of energy available on the grid at any time is the sum of "baseline" power sources like nuclear power plants that product a fairly constant amount of power at all times, plus "spiky" power sources like wind and solar that produce a variable amount of power depending on environmental conditions like wind speeds, time of day, cloud cover, etc).

A taxonomy of out-of-memory issues

There are three types of out-of-memory issues that we see occur in Gatsby sites:

- First, image processing with gatsby-plugin-image (Generating image thumbnails). This is related loosely to the number of images. but more related to the dimensions of individual images; overly large images (eg a 3,000 x 3,000 pixel JPG) can cause memory spikes.

- Second, Javascript bundle creation (Building production JavaScript and CSS bundles) and page serialization (Building static HTML for pages). Depending on the size of your templates, serializing them during the page build (SSR) process can cause memory spikes.
- Third, memory usage from the number of "nodes" stored as data in the system. Gatsby represents data you source as "nodes" and keeps all the nodes in memory. This establishes a "baseline" level of memory usage.

Profiling your memory usage

If you've either experienced, or are worried about OOM memory crashes, we strongly recommend that you profile build memory usage to establish a baseline.

It's a good practice to profile memory periodically once your site crosses 10k or 20k pages, or if you've experienced out-of-memory challenges on smaller sites.

The Gatsby team's recommendation is to run `gatsby` within a profiling tool like process-top. Run your site like the following:

```
npx process-top ./node_modules/.bin/gatsby build
```

This will output every second profile information like the following:

```
cpu: 180.5% | rss: 431 MB (5.0%) | heap: 163 MB / 269 MB (60.7%)
| ext: 280 MB | delay: 792 ms | 00:00:09 | loadavg: 1.71, 1.65,
1.78
```

("RSS" is the total memory being used).

Options to reduce "out-of-memory" risk

There are a number of options to fix OOM issues; some are specific to OOMs occurring at certain stages in the build process, while others are general solutions that will work at any stage. In addition, while some of these options are universal best practices, others involve tradeoffs against build speed, site performance, site debuggability, or cost.

1. General solutions

Upgrade Gatsby & gatsby source plugins The Gatsby team is continually shipping upgrades that reduce the amount of memory footprint. The simplest and first solution should be to upgrade your version of Gatsby and source plugins to the latest version and see what impact that has.

Try using Gatsby Cloud Paid tiers of Gatsby Cloud parallelize image processing and run other parts of the build process in ways that are less memory intensive than comparably sized hardware.

Try reducing the number of cores Gatsby defaults to using the number of physical cores on your machine to speed up builds, and parallelizes during certain steps of the process (image processing, Javascript bundle creation, and HTML generation).

So if you're experiencing out of memory issues during any of those steps, you can try setting the environment variable `GATSBY_CPU_COUNT` to a lower number, like 2. Note that this will slow your builds down!

Increase allocated memory and/or upgrade your hardware. The default node process in node >= 12 has a 2GB "heap size" allocated for memory.

It's plausible that you could increase memory usage significantly. Increasing the max heap size to 4GB or 8GB is quite common (using `NODE_OPTIONS=--max-old-space-size` set to 4096 or 8192).

The theoretical limit on a 64-bit machine is 16 terabytes, but of course the practical limit is much lower. Netflix experimented with raising this limit to 32GB back in 2014; the Gatsby team is aware of a few sites that run this at 16GB (on machines with 16GB of RAM).

So:

- Some Gatsby build hosts, like Gatsby Cloud, will automatically set the memory limit to the maximum available in memory. Others may require you to use the `NODE_OPTIONS` key if you want to increase the memory limit.
- if you're at the memory limit for your plan tier, and your build host offers more memory at the next tier, you may want to consider upgrading plan tiers.
- Alternatively, if you self-host your CI/CD pipeline, you may need to upgrade to a beefier machine, VM, or container that has the desired amount of memory.

In addition, if you do these in production, when running gatsby develop locally, you'll probably also want to limit the number of nodes used in the development process and/or allocate additional memory locally as well.

Finally, you can't allocate more memory than is available on your machine, so be careful not to set `NODE_OPTIONS=--max-old-space-size` to a value greater than your hardware's RAM, as that may negatively impact the Node process's ability to garbage collect properly.

2. Reducing crashes due to gatsby-plugin-image

Pre-optimize images by downsampling Overly large images being used by Gatsby image can cause memory spikes. For example, if your image will only ever be 1000 pixels wide, and you have a 2000 x 3000 pixel source image stored in your repository or headless CMS, it may be worthwhile to convert that source

image to a 667 x 1000 pixel version. This piece on pre-optimizing images can give some guidance.

Disable AVIF Generating AVIFs often causes increased memory usage by gatsby-plugin-image. This option is turned off by default in gatsby-plugin-image; if you've turned it on and are seeing issues, try turning it back off.

3. Reducing crashes in generating Javascript bundles & serializing HTML pages

Turn off source maps generation Use Gatsby plugin that disables sourcemap generation by modifying webpack config: gatsby-plugin-no-sourcemaps (this can help build speed as well).

The side effect is that this will make any errors happening in production significantly harder to debug by obscuring the stacktrace, so if your team regularly inspects production errors (perhaps using eg Sentry, Rollbar, or comparable tools), you may want to consider other options.

Inspect and reduce bundle sizes Serialization memory spikes are proportional to template sizes. Optimizations to improve bundle size, especially of your largest bundle sizes, should have benefits to this.

When you're scanning bundle sizes to see what might be causing out-of-memory issues, rather than focusing on critical paths, focus on anomalously large bundles. If one of your bundles is 3MB and the rest are 300KB, you might have, eg, accidentally imported an 2.5MB image directly into a bundle.

4. Reducing crashes due to large number of nodes

Examine use of headless CMS For similar amounts of pages and content, some source plugins cause your site to build more slowly than others. Those same source plugins also have greater memory usage. (The underlying reason, which causes both behaviors, is that these source plugins create a larger number of nodes).

Similar to build speeds, if sourcing and processing content from your CMS is causing these spikes, and there are significantly lower-memory alternative(s), you may want to consider migrating some or all content there.

Plan for site growth by examining the number of nodes The number of nodes being generated is output in the raw logs of every gatsby build, eg:

```
22:50:32 PM: info Total nodes: 29, SitePage nodes: 1 (use --verbose
for breakdown)
```

Because the amount of baseline memory usage is roughly proportional to the number of nodes, and the number of nodes is roughly proportional to the number

of pages (assuming the CMS setup stays the same), this can be useful for planning around future growth.

Disable cache persistence When the amount of nodes approaches 1.5 million (typically a 100k page Contentful site, or a 300k page Sanity site), the way Gatsby serializes internal state can cause out of memory issues. If your site is near this, and you're experiencing issues, try setting the environment variable `GATSBY_DISABLE_CACHE_PERSISTENCE=1`.

Note that for sites at this scale, build times may also become a workflow issue for your team, so be sure to spend time thinking about how to improve build times as well.