Gatsby Memory Benchmark

The goal of this benchmark is to test Gatsby's memory usage and look for potential optimizations.

The Docker Container

The docker container used in these tests sets up a Debian instance with node 14 installed (as well as npm/yarn/etc). It has ports 9000 (for hosting gatsby) and 9229 (for debugging) exposed.

Within the container, two points to your local filesystem are mounted:

- /usr/src/gatsby : Your local gatsby repo
- /usr/src/site : The memory benchmark gatsby site

If you'd like to configure `jemalloc` to run within the container, set the `JEMALLOC=1` env var when building the docker container.

Commands

Tests

yarn test --memory X --num-nodes Y --node-size Z

Runs a test build within a docker container with the given memory allotment. Within our gatsby-node, we'll create X nodes with a string property of size Y.

Example: running a build with 1000 nodes of 1mb each, in a docker container with 8gb of memory.

```
$ yarn test --memory 8g --num-nodes 500 --node-size 1m
```

yarn test-suite --name some-name --suite [incremental|exhaustive]

Runs through test suites defined in `scripts/test-suite.js` and outputs results to `output/some-name`. Output includes a `results.csv` with a summary of all builds, as well as breakdowns for each memory configuration.

incremental
Incremental tests run builds with a `node-size` of 1m. For each memory allotment, it will start with 100 nodes in the build and increment by 100 on each success. The test will stop when all builds in a given configuration fail. See `incrementalConfig` in `scripts/test-suite.js` to customize test sets.

exhaustive
Exhaustive tests are just that, exhaustive. It will measure the time/success of every combination given. See `exhaustiveConfig` in `scripts/test-suite.js` to customize test sets.

Docker

These commands are used for interfacing with docker and have built-in utilities for managing the docker container.

yarn docker:build

Builds the container used for testing. If you'd like to configure `jemalloc` to run within the container, set the `JEMALLOC=1` env var.

Example:

```
$ JEMALLOC=1 yarn docker:build
```

yarn docker:remove

Removes the docker image.

yarn docker:rebuild

Shorthand for remove + build.

yarn docker:start

Starts the container built by `yarn docker:build`.

yarn docker:connect

Connects to the container started by `yarn docker:start`.

yarn docker:start-and-connect

A shorthand for start + connect.

yarn docker:stop

Stop the container used for testing.

yarn docker:stats

Show a polling display of the container's docker stats.

Gatsby

These commands are used for interfacing with gatsby.

yarn gatsby:build

Simply an alias to `yarn gatsby build`.

yarn gatsby:serve

Starts `gatsby serve` on port 9000 and sets the host properly to work inside docker.

yarn gatsby:develop

Starts `gatsby develop` on port 9000 and sets the host properly to work inside docker.

yarn gatsby:build:debug

Runs `gatsby build` with `inspect-brk` set to start the debugging process on port 9229.

yarn gatsby:develop:debug

Runs `gatsby develop` with `inspect-brk` set to start the debugging process on port 9229.

Setup

Currently we can reproduce builds crashing with out default settings

- Docker container running with 2GB limit
- 300 nodes x ~2MB each = ~600MB of "just" nodes data in each process (number of nodes can be controlled with NUM_NODES env var)
- 3 workers + main process (`GATSBY_CPU_COUNT` set to 4 in docker image, but you can specify different value with env var - for example `GATSBY_CPU_COUNT=6 yarn gatsby:build`)
- `eq_field` template using fast filters (single `eq` specifically)

Goal is to make `eq_field` template to not cause crashes, then add next template (different operator) that cause crashes and repeat until all queries can be handled with set memory limits.

Workflow

While `gatsby-dev` command is available inside docker, from my testing it seems like it doesn't pick up file changes when run there. Workflow that seems to work reliably:

When starting working with this benchmark:

- start `yarn watch` (possibly with `--scope`) in monorepo
- start `gatsby-dev` outside of docker in benchmark directory (just like with regular site)
- `yarn test --memory 8g --num-nodes 1000 --node-size 1m`

And repeat as many times as you want:

- make changes to `gatsby` source code as you normally would
- run your `yarn test` command again