Genomics Benchmarking Suite Specification

Purpose

The purpose of the Sanger Genomics Benchmarking Suite is to provide a domain specific benchmarking suite which demonstrates the types of compute used in large genomics institutes. Specifically, the focus is on High Throughput Computing, the IO heavy tasks such as mapping and calling of variants which tend to be found in the early stages of genomics compute pipelines and are not well represented by conventional synthetic benchmarks used in High Performance Computing. The suite will be flexible enough to add new and updated tooling which is constantly evolving in this field.

Additionally, it can be used in conjunction with continuous integration systems to benchmark versions of a program, allowing developers to keep track of the impact of their changes on performance.

Process flow

Figure 1 below shows the primary process flow of the Benchmarking executable.

The first step is the reading of the configuration file of which a default is provided. This file contains:

- The location of the resource server from which the resource pack containing the data files for benchmarking are downloaded.
- The S3 bucket to which benchmarking results will be uploaded.
- The configuration for the tests as detailed in the Tests section of this document.

This file will not contain local configuration such as:

- Working directory for executables
- Working directory for data files
- iPerf server for networking tests

These will be provided as mandatory command line parameters or in a separate configuration file.

Next the test programs and data are downloaded, and the test programs compiled for the system they are being benchmarked on.

Following this each test specified in the configuration file are run and the results serialised into JSON format. The tests are run serially so as not to interfere with each other. Root access will be required to allow us to purge caches and reset state between runs.

The collected data will then be uploaded to the S3 server for collation and comparison with other results. Additionally, to provide immediate feedback to the person running the benchmark, local plots with comparison to older publicly available systems will be created at the end of the run.

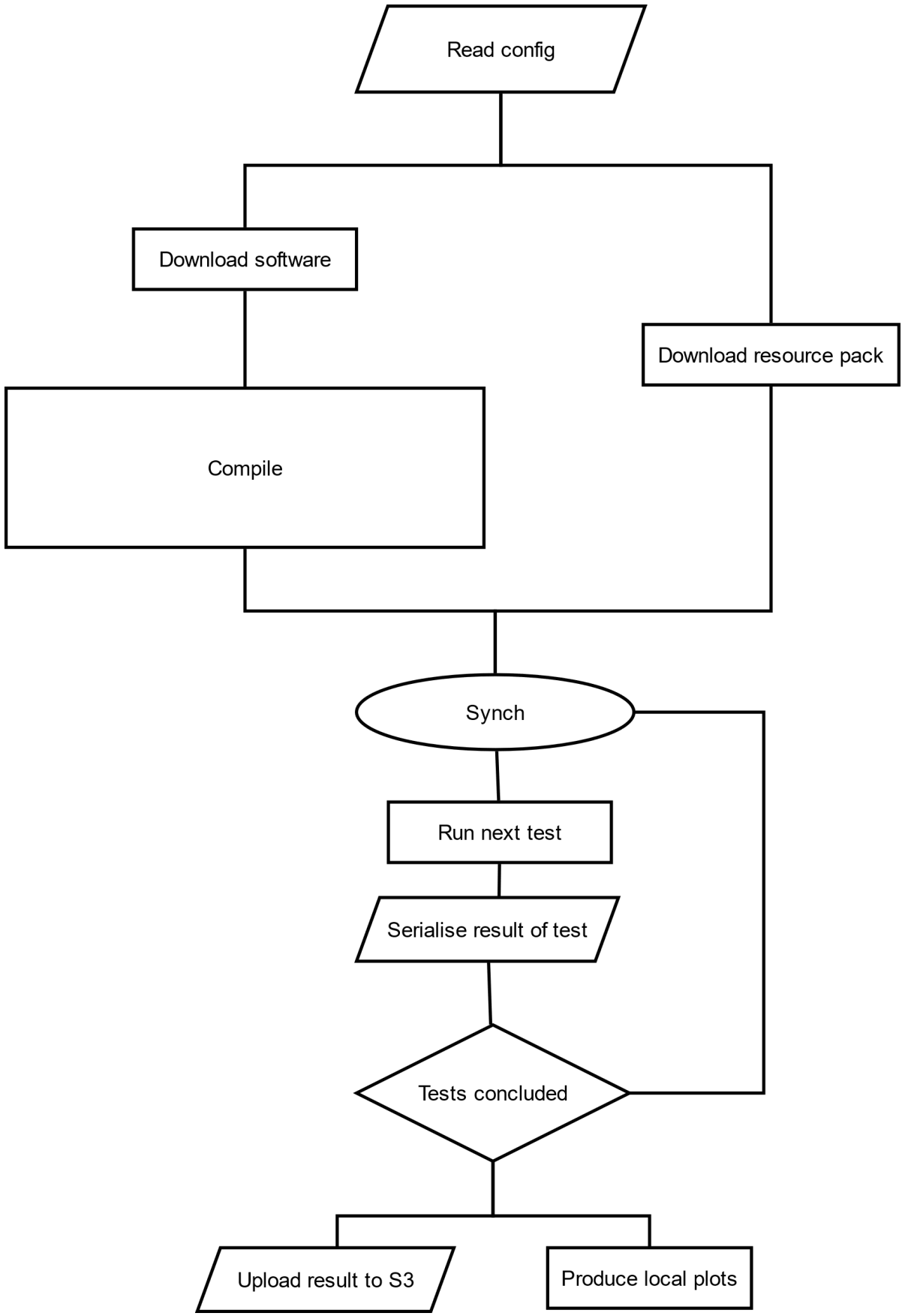

Figure 1 - Process flow of benchmarking process

Tests

Runnable program

This is the standard configurable domain specific informatics tasks that we want to benchmark.

Parameters

- Runtime
    - Command line
      Command line with appropriate threads parameter placeholder for interpolation
    - Threads * Processes configuration
      Default: 1*1
    - NUMA configuration
      Whether to apply NUMA pinning to process
      Default: Unconfigured system default
    - Replications
      How many times we repeat the test
- Build time
    - Package location
    - Compilation script

Results

Derived from /usr/bin/time output.

- CPU usage
    - Wall time (seconds)
      Elapsed real (wall clock) time used by the process, in seconds.
    - CPU User time (seconds)
      Total number of CPU-seconds that the process used directly (in user mode), in seconds.
    - CPU Sys time (seconds)
      Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
    - CPU %
      Percentage of the CPU that this job got. This is just user + system times divided by the total running time.
- Memory
    - RSS
      Average resident set size of the process, in Kilobytes
- Process stats
    - Involuntary context switches
      Number of times the process was context-switched involuntarily (because the time slice expired).
    - Voluntary context switches
      Number of times that the program was context-switched voluntarily, for instance while waiting for an I/O operation to complete.
    - Major faults
      Number of major, or I/O-requiring, page faults that occurred while the process was running. These are faults where the page has actually migrated out of primary memory.

    - Minor faults
      Number of minor, or recoverable, page faults. These are pages that are not valid (so they fault) but which have not yet been claimed by other virtual pages. Thus the data in the page is still valid but the system tables must be updated.
- IO
    - Number of file system inputs by the process.
    - Number of file system outputs by the process.
- Exit status

IO test

This test will be run on all “data” drives. Based on the IOzone benchmark run in automatic mode

Parameters

Results

This produces a 3-dimensional matrix of output per test in a form like kB/sec * kB file size * kB Req size.

Networking test

This test will be run on all primary networking interfaces drives. Based on iperf3 using JSON output.

Parameters

- Server to test against
- Server port
- UDP
  If true then use UDP instead of TCP
- Parallel connections to use
- Time in seconds
  How long to run the test over

Results

- sum_sent -> bytes
- sum_sent -> bits per second
- sum_sent -> retransmits
- sum_recieved -> bytes
- sum_recieved -> bytes

Upload to S3

Results from benchmarks will be uploaded to a Sanger public write-only S3 bucket as JSON. Uploads which pass validation will then be loaded into a JSON database for results rendering.

Security

Uploads will be checked to ensure they are:

- Appropriately sized (restricted by s3)
- Rate limited from host of origin
- Conform to our results schema (upload complete will trigger a validator which checks they conform to our schema and deletes files which do not).

Local plots

Results will be plotted against older systems for a sanity check at the end of the run.