Table of Contents

Google Perftools Usage for LHCb Software Optimization

About

gperftools contain a set of measurement tools and an optimised memory allocation library called tcmallc.

Installing

Download both gperftools and libunwind, you will need to compile libunwind before gperftools. The INSTALL instructions for gperftools-2.0 recommend version 0.99-beta of libunwind. It is best to install both to your home directory so unpack libunwind and use:

```
./configure --prefix=$HOME
make
make install
```

To compile gperftools use:

```
./configure --prefix=$HOME LIBS="-Wl,--rpath -Wl,"$HOME"/lib -L"$HOME"/lib" CPPFLAGS=-I$HOME/incl
make LIBS="-Wl,--rpath -Wl,"$HOME"/lib -L"$HOME"/lib" CPPFLAGS=-I$HOME/include LDFLAGS=-L$HOME/li
make install LIBS="-Wl,--rpath -Wl,"$HOME"/lib -L"$HOME"/lib" CPPFLAGS=-I$HOME/include LDFLAGS=-L
```

CPU Profiling

general documentation: http://google-perftools.googlecode.com/svn/trunk/doc/cpuprofile.html

Linking against libprofiler.so

You want to link against libprofile.so and use the API calls in your code section after most of the dynamic libraries have been loaded. More details on why can be found in the "64-BIT ISSUES" section of the gperftools README.

You need to place the ProfilerStart() and ProfilerStop() calls somewhere where they only get executed once. A good place are the initialize() and finalize() methods of the Sequencer which runs your algorithms. If you experience segfaults when running with profiling enabled try placing your ProfilerStart()/ProfilerStop() in slightly different places, probably later/earlier (respectively) in the code. This seems more like an art than a science.

In order to add the gperftools include directory and link to libprofile.so edit the cmt/requirements file of your package like so:

```
macro_append YourPackageName_linkopts "-lprofiler -L/location/in/which/you/installed/gperftools/l
macro_append YourPackageName_cppflags "-I/location/in/which/you/installed/gperftools/include"
```

Preloading libprofiler.so

env LD_PRELOAD="/usr/lib/libprofiler.so"

WARNING: Gaudi dynamically loads a lot of libs at runtime. If the CPU Profiler does a sampling interupt while Gaudi loads a lib you will get a crash. So preloading libprofile is not feasble for real world gaudi apps.

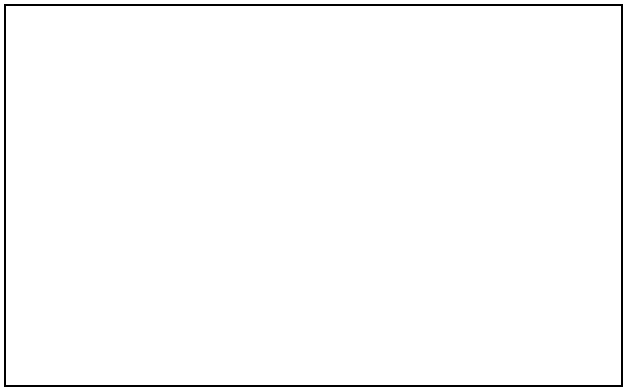
Usage and output

When starting the profiler you need to pass a string as argument which will be used as the file in which to store the profiling information. You will also need to include gperftools/profiler.h in the file containing the ProfilerStart() and ProfilerStop() calls.

Analyse the output file with pprof which will be installed in ~/bin if you installed gperftools to your home directory.

Text Output

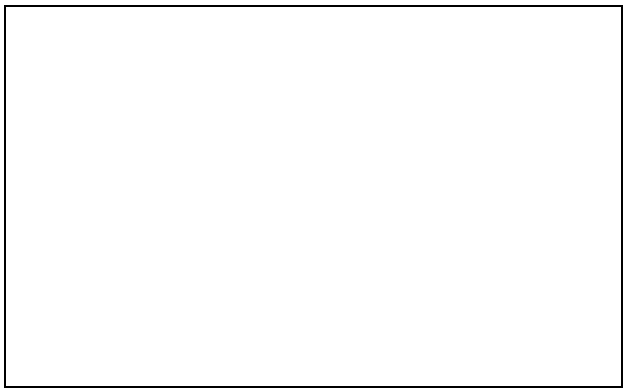
Callgraph



Heap Profiling (aka Memory Profiling)

Text Output

Callgraph



Heap Checking (aka Memory Leak Detection)

tcmalloc

This topic: LHCb > Google-perftools
Topic revision: r3 - 2013-04-19 - TimHead