

Howto run Moore

...for various purposes

Stephan Nies
snies@cern.ch

Howto run Moore ...

- ... just run it
- ... with custom compiler flags
- ... with valgrind (to search for leaks or profiling)
- ... with google perftools tcmalloc (for faster execution)
- ... with google perftools CPUProfiler (for profiling)
- ... configured by a TCK (to try it)

Howto run Moore (just run it):

Get it: latest nightly:

```
SetupProject Moore HEAD --build-env --nightly lhcb2 Tue  
getpack Hlt/Moore head  
SetupProject Moore HEAD –nightly lhcb2 Tue
```

latest release:

```
SetupProject Moore v7r1 –build-env  
getpack Hlt/Moore v7r1
```

Build it:

```
cd $User_release_area/Moore_<version>/Hlt/Moore/cmt  
cmt config  
source setup.csh  
cmt make
```

Run it:

```
$MOOREROOT/options/Moore.py
```

Howto run Moore with custom compiler flags:

Well known tags for production and debugging code:

slc4_amd64_gcc34
slc4_amd64_gcc34_db

There are also hidden tags like (old example):

tag slc3_ia32_gcc344_cov (code coverage)
tag slc3_ia32_gcc344_pro (profiling)

Most important is where all this stuff is set:

GaudiPolicy/cmt/requirements

Howto run Moore with custom compiler flags:

Best Way: CMTUSERCONTEXT

Create a custom requirements file in a random folder

```
setenv CMTUSERCONTEXT <that folder>
```

Example:

```
macro_append componentshr_linkopts " -lprofiler "
macro_append cppflags " -g -DNODEBUG "
macro_remove_regexp componentshr_linkopts "\,\-\s"
```

Valgrind is a great tool:

Benefits:

- no need to recompile
- call graph information
- memory leak detection
- nice gui kachegrind

Usage:

```
source /afs/cern.ch/lhcgroup/rich/vol4/jonrob/scripts/new-valgrind.csh
valgrind --tool=callgrind application
```

Documentation:

<https://twiki.cern.ch/twiki/bin/view/LHCb/CodeAnalysisTools>

<https://twiki.cern.ch/twiki/bin/view/Atlas/SoftwareDevelopmentWorkBookValgrind>

<http://docs.kde.org/kde3/de/kdesdk/kcachegrind/index.html>

Howto run Moore with Google perftools tcmalloc

we got 25% for Moore

- Benefits:**
- faster execution 15 – 25 %
 - especially when a lot of memory allocation
 - dead simple

- Procedure:**
- Only runs under SLC5 with gcc 4.3 (SLC4 possible)
 - Just type this command before you run:

```
setenv LD_PRELOAD /afs/cern.ch/lhcb/group/trigger/vol3/snies/  
perftools_slc5_gcc_43/lib/libtcmalloc_minimal.so
```

Howto run Moore with Google perftools CPUProfiler

- Benefits:**
- Callgraph support
 - Only recompile one Lib
 - Fast with low sample rate

- Procedure:**
- Only runs under SLC5
 - Setup environment
 - Place calls to start and stop the profiler
 - Compile just the Lib that contains the Start and Stop, link it against the libprofile.so
 - Run Moore

Setting up the environment

```
ssh -X user@lx64slc5.cern.ch
source /afs/cern.ch/lhc/b/group/trigger/vol3/snies/
    perftools slc5_gcc_43/scripts/perftool_env.csh
SetupProject Moore HEAD --build-env --nightly lhcb2 Tue
getpack Hlt/Moore head
getpack Kernel/LHCbAlgs head #or any other package
SetupProject Moore HEAD --nightly lhcb2 Tue
Cd $User_release_area/Moore_HEAD/Hlt/Moore/cmt

# Edit some file and place calls to profiler (next page)

cmt config
source setup.csh
cmt br cmt conig
cmt br cmt make
```

Placing calls to pertools CPUProfiler

Example: Kernel/LHCbAlgs/src/createODIN.cpp

```
#include <google/profiler.h>

createODIN::createODIN( const std::string& name,
                        ISvcLocator* pSvcLocator)
    : GaudiAlgorithm ( name , pSvcLocator ),
m_decoder(0)
{
    ProfilerStart(getenv("PERFTOOLSLOG"));
}

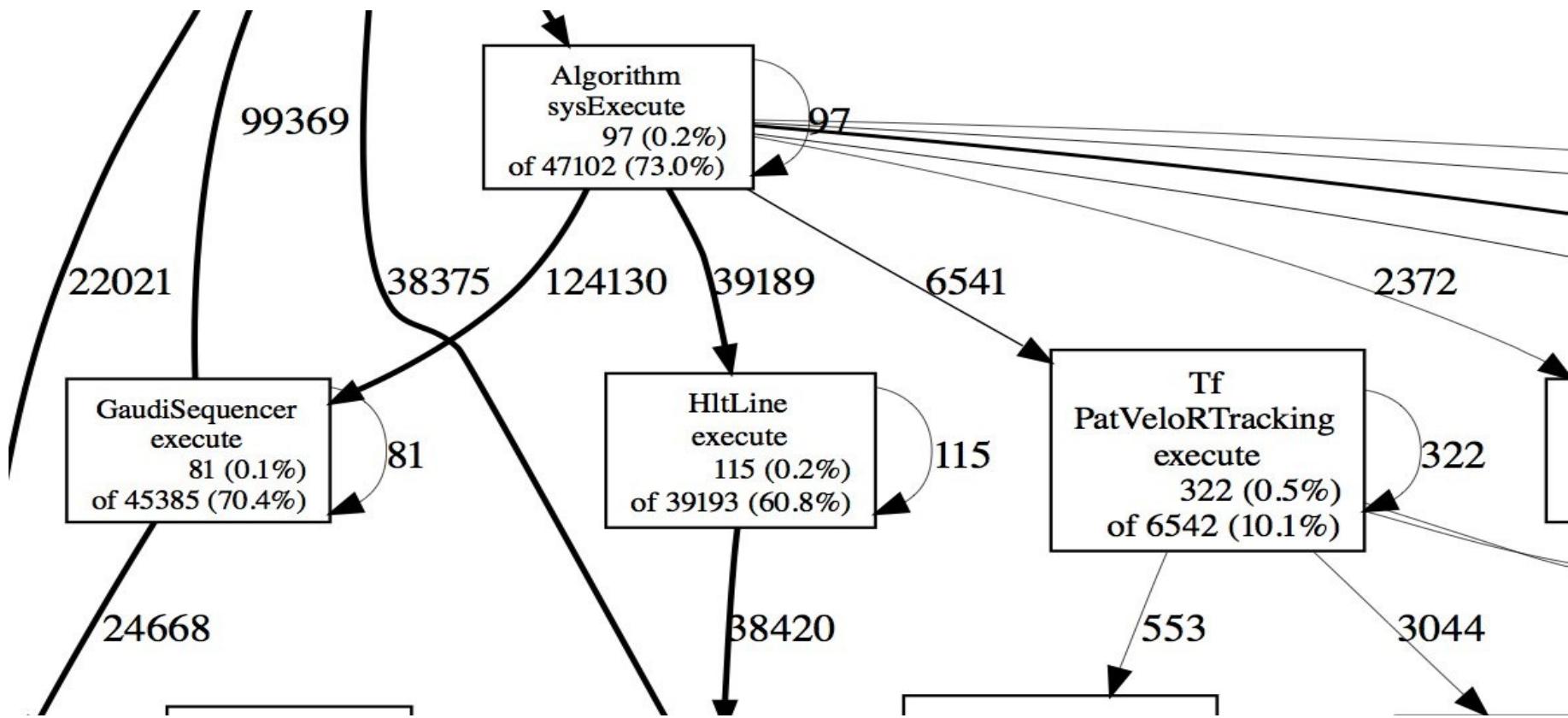
createODIN::~createODIN()
{
    ProfilerStop();
}
```

Run program and analyse profile:

```
setenv PERFTOOLSLOG /tmp/mylog  
  
$MOOREROOT/opt/Moore.py  
  
# for text output:  
pprof --text /afs/cern.ch/sw/lcg/external/Python/2.5.4/x86_64-  
    slc5-gcc43-opt/bin/python /tmp/mylog  
  
# for a postscript graph:  
Pprof --gv /afs/cern.ch/sw/lcg/external/Python/2.5.4/x86_64-  
    slc5-gcc43-opt/bin/python /tmp/mylog
```

Docs: <http://code.google.com/p/google-perftools/>

pertools CPUProfiler sample output:



Howto run Moore configured by a TCK

```
#!/usr/bin/env gaudirun.py
#
# Minimal file for running Moore from python prompt
# Syntax is:
#   gaudirun.py .../options/Moore.py
# or just
#   ./options/Moore.py
#
from Moore.Configuration import *

Moore().Simulation = True
Moore().DataType    = '2009'
Moore().DDDBtag     = "default"
Moore().CondDBtag   = "default"
Moore().inputFiles  = [ '/data/bfys/lhcbl/data/2009/RAW/FULL/FEST/FEST/
%d/0%d_000000001.raw'%(i,i) for i in range(50627,50630) ]

Moore().Verbose = True
Moore().UseTCK = True
Moore().InitialTCK = '0x804d0000'
```

Thanks to Gerhard

Conclusion:

- The testing, profiling and optimization of Moore is running full steam ahead
In the last 3 weeks we found ways to improve the speed by 35%
- Various tools are available
- You are invited to make use of them
contact me if you have questions