

eclipse shallow mpi project with gcov and gprof

- [Motivation](#)
- [Demo : Shallow water weather model](#)
- [References:](#)

Motivation

A couple basic tools are included with most linux distributions providing code coverage analysis and function profiling: gcov and gprof. Using just these 2 tools, it's easy to gain understanding of how your code is running. With the linuxtools Eclipse incubation project (<http://www.eclipse.org/linuxtools/> , now bundled with [PTP](#)), gprof and gcov are fully integrated into the cdt editor perspective. Let's see how the linuxtools project adds value and puts a fresh look on these classic linux code analysis tools.

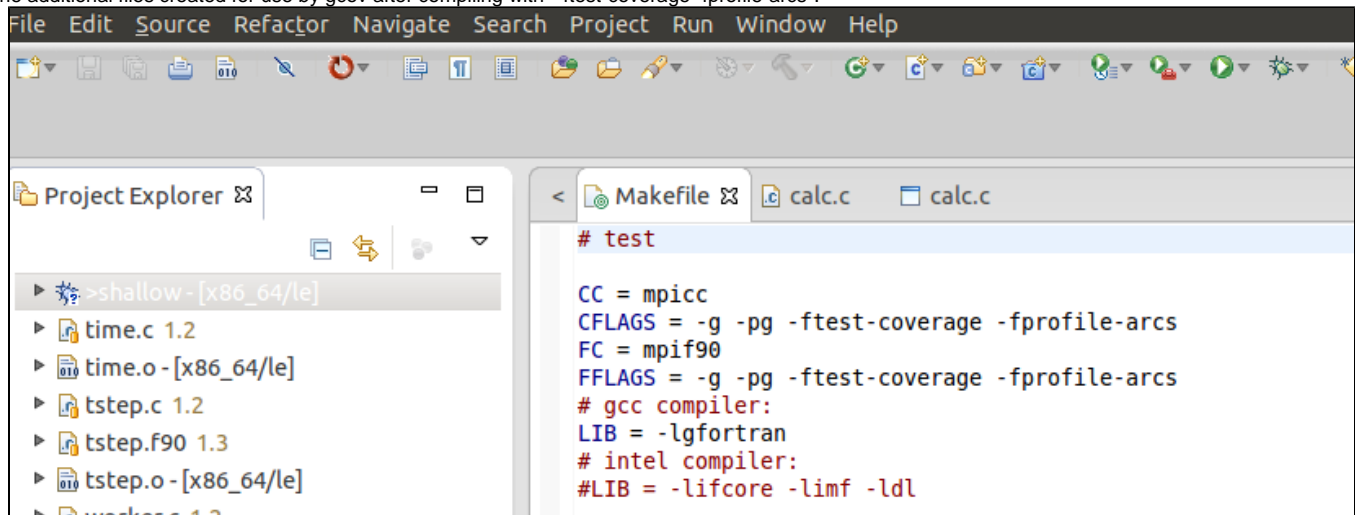
Demo : Shallow water weather model

The Shallow demo project from the eclipse tutorial cvs can showcase gprof and gcov with a few simple adjustments to the Makefile and MPI runtime. Here are the details and some discussion of what can be done with the profiling and coverage output.

First, the source code is from the demo project and the checkout is described in Module 3 of the [PTP SC11 Tutorial](#).

Once checked out, fix the bug in the for() loop of main.c and build the project (Copy Makefile.mk to Makefile and adjust as needed). Then setup a run configuration with openmpi on your favorite system. I've chosen my own laptop but you could also setup a synchronized project and use a remote system of your choice. My laptop runs Ubuntu, the GNU compilers, and OpenMPI which makes working with Eclipse and developing test/demo codes a snap.

To use gprof and gcov, add the appropriate flags to the Makefile. I'll run a single test job to get both profile and coverage data at once by using -pg for gprof and "-ftest-coverage -fprofile-arcs" for gcov (-g is included so that we have source code line numbering available). Do a project .> refresh and note the additional files created for use by gcov after compiling with "-ftest-coverage -fprofile-arcs".



Setup the MPI run configuration with the Environment variable GMON_OUT_PREFIX defined with a name for your individual MPI rank gmon outputs. By default gmon.out is used but MPI doesn't do that well and you end up with a profile that's missing most of the information, so by using GMON_OUT_PREFIX, each MPI rank adds its process id to its gmon output filename.

The gmon output files can be combined in a summary with the gprof -s command as shown. It's interesting to compare the summary gmon output to that from one of the ranks (copy a rank's gmon_* to gmon.out to easily view it with Eclipse linuxtools).

```
galen@lgheron:~/workspace/shallow$ gprof -s shallow gmon_*
galen@lgheron:~/workspace/shallow$ ls
calc.c      decs.h      eclipse.inc  init.o      time.c      worker.c
calc.gcda   diag.c      gmon_shallow.9885  main.c      time.gcda   worker.gcda
calc.gcno   diag.gcda   gmon_shallow.9886  main.gcda   time.gcno   worker.gcno
calc.o      diag.gcno   gmon_shallow.9887  main.gcno   time.o      worker.o
copy.c      diag.o      gmon_shallow.9888  main.o      tstep.c
copy.gcda   dump.c      gmon.sum          Makefile    tstep.f90
copy.gcno   dump.gcda   init.c            Makefile.gem  tstep.gcda
copy.o      dump.gcno   init.gcda         Makefile.mk  tstep.gcno
CVS         dump.o      init.gcno         shallow      tstep.o
galen@lgheron:~/workspace/shallow$
```

<div> <div>Problems</div> <div>Tasks</div> <div>Console</div> <div>Properties</div> <div>Remote Environments</div> <div>gcov</div> <div>gprof</div> <div>gprof</div> </div>					
gmon file: /home/galen/workspace/shallow/gmon.out program file: /home/galen/workspace/shallow/shallow 4 bytes per bucket, each sample counts as 10.000ms					
Name (location)	▲	Samples	Calls	Time/Call	%Time
▼ Summary		8			100.0%
▼ calc.c		0			0.0%
calcuvzh		0	1000	0ns	0.0%
► copy.c		0			0.0%
► diag.c		1			12.5%
► main.c		0			0.0%
► time.c		4			50.0%
▼ tstep.f90		3			37.5%
▼ tstep		3	1000	30.000us	37.5%
► tstep (tstep.f90:64)		1			12.5%
► tstep (tstep.f90:73)		1			12.5%
► tstep (tstep.f90:75)		1			12.5%
► worker.c		0			0.0%

<div> <div>Problems</div> <div>Tasks</div> <div>Console</div> <div>Properties</div> <div>Remote Environments</div> <div>gcov</div> <div>gprof</div> <div>gprof</div> </div>					
gmon file: /home/galen/workspace/shallow/gmon.sum program file: /home/galen/workspace/shallow/shallow 4 bytes per bucket, each sample counts as 10.000ms					
Name (location)	▲	Samples	Calls	Time/Call	%Time
▼ Summary		23			100.0%
▼ calc.c		3			13.04%
▼ calcuvzh		3	3000	10.000us	13.04%
▼ calcuvzh (calc.c:47)		1			4.35%
0x401d00		1			4.35%
▼ calcuvzh (calc.c:49)		2			8.7%
0x401f54		1			4.35%
0x401f64		1			4.35%
► copy.c		0			0.0%
► diag.c		1			4.35%
► init.c		0			0.0%
▼ main.c		0			0.0%
► main		0	0		0.0%
► setup_res		0	4	0ns	0.0%
► update_global_ds		0	5	0ns	0.0%
▼ time.c		5			21.74%
► timetend		5	3000	16.666us	21.74%
▼ tstep.f90		14			60.87%
► tstep		14	3000	46.666us	60.87%
► worker.c		0			0.0%

The gcov view is similar to the gprof view but keep in mind that you're looking at code coverage and not necessarily performance or timing information (though there is a relationship...code not executed is performing quite well !). Also note that multiple executions will accumulate values in the gcov output files until they are removed or truncated to zero-length (2nd run to demonstrate this).

<div> <div>Problems</div> <div>Tasks</div> <div>Console</div> <div>Properties</div> <div>Remote Environments</div> <div>gcov</div> <div>gprof</div> <div>gprof</div> </div>				
<div> <div>program runs = 4</div> <div>program file : /home/galen/workspace/shallow/shallow</div> </div>				
Name	Total Lines	Instrumented	Executed Line	Coverage %
▼ Summary	1166	351	298	84.9%
▼ calc.c	54	12	12	100.0%
calcuvzh		12	12	100.0%
▶ copy.c	92	13	6	46.15%
▶ diag.c	76	25	25	100.0%
▶ dump.c	91	38	0	0.0%
▶ init.c	87	30	30	100.0%
▼ main.c	262	83	75	90.36%
main		67	60	89.55%
setup_res		11	10	90.91%
update_global_ds		5	5	100.0%
▶ time.c	57	14	14	100.0%
▶ timestep.f90	88	42	42	100.0%
▶ worker.c	359	94	94	100.0%

Selecting (double click) a source code line from either the gcov or gprof view and you'll see the file and routine highlighted in the cdt c/c++ perspective. Also notice the support for the .f90 file and its routines.

FileEditSourceRefactorNavigateSearchProjectRunWindowHelp

Project Explorer

>shallow-[x86_64/le]

time.c 1.2

time.o-[x86_64/le]

tstep.c 1.2

tstep.f90 1.3

tstep.o-[x86_64/le]

worker.c 1.2

worker.o-[x86_64/le]

>calc.gcda

>calc.gcno

>copy.gcda

>copy.gcno

>diag.gcda

>diag.gcno

>dump.gcda

>dump.gcno

eclipse.inc 1.1

>gmon_shallow.9885

>gmon_shallow.9886

>gmon_shallow.9887

>gmon_shallow.9888

>gmon.out

>gmon.sum

>init.gcda

>init.gcno

>main.gcda

>main.gcno

>Makefile

Makefile.gem 1.2

Makefile.mk 1.5

>time.gcda

>time.gcno

>tstep.gcda

>tstep.gcno

Makefilecalc.ccalc.c

* Language = BSD c using Argonne NL macros

* O/S = Unix System V

* H/W = Encore Multimax 320

#include "decs.h"

void calcuvzh(jstart,jend,p,u,v,cu,cv,h,z,fsdx,fsdy)

int jstart,jend;

float p[n][m];

float u[n][m];

float v[n][m];

float cu[n][m];

float cv[n][m];

float h[n][m];

float z[n][m];

float fsdx, fsdy;

{

int i,j,ip,jp;

for(j=jstart;j<=jend;j++){

jp=(j+1)%n;

for(i=0;i<m;i++){

ip=(i+1)%m;

cu[j][ip]=0.5*(p[j][ip]+p[j][i])*u[j][ip];

cv[jp][i]=0.5*(p[jp][i]+p[j][i])*v[jp][i];

z[jp][ip]=(fsdx*(v[jp][ip]-v[jp][i])-fsdy*(u[jp][ip]-u[j][ip]))/(p[j][i]+p[j][ip]+p[jp][ip]+p[jp][i]);

h[j][i]=p[j][i]+0.25*(u[j][ip]*u[j][ip]+u[j][i]*u[j][i]+v[jp][i]*v[jp][i]+v[j][i]*v[j][i]);

}

}

}

ProblemsTasksConsolePropertiesRemote Environmentsgcovgprof

program runs = 4

program file : /home/galen/workspace/shallow/shallow

Name	Total Lines	Instrumented	Executed Line	Coverage %
calcuvzh	12	12	100.0%	

The screenshot displays the Eclipse IDE interface. On the left, the 'Project Explorer' shows a project named 'shallow' with various source files and object files. The main editor window shows the 'Coverage' view for the file 'calc.c'. The coverage data indicates a single call to the function 'calcuvzh' at address 0x401d00, which took 4.35% of the total execution time. The source code for 'calcuvzh' is shown, featuring a loop that processes data for 'Encore Multimax 320'.

Coverage Data:

Name (location)	Samples	Calls	Time/Call	%Time
calcuvzh (calc.c:47)	1			4.35%
0x401d00	1			4.35%

Source Code Snippet:

```

* H/W      = Encore Multimax 320
*
*****

#include "decs.h"

void calcuvzh(jstart,jend,p,u,v,cu,cv,h,z,fsdx,fsdy)
int jstart,jend;
float p[n][m];
float u[n][m];
float v[n][m];
float cu[n][m];
float cv[n][m];
float h[n][m];
float z[n][m];
float fsdx, fsdy;
{
    int i,j,ip,jp;

    for(j=jstart;j<=jend;j++) {
        jp = (j+1) % n;
        for (i = 0; i < m; i++){
            ip = (i+1) % m;
            cu[j][ip] = 0.5*(p[j][ip]+p[j][i])*u[j][ip];
            cv[jp][i] = 0.5*(p[jp][i]+p[j][i])*v[jp][i];
            z[jp][ip] = (fsdx*(v[jp][ip]-v[jp][i])-fsdy*(u[jp][ip]-u[j][ip]))/(p[j][i]+p[j][ip]+p[jp][ip]+p[jp][i]);
            h[j][i] = p[j][i]+0.25*(u[j][ip]*u[j][ip]+u[j][ip]+u[j][i]*u[j][i]+v[jp][i]*v[jp][i]+v[j][i]*v[j][i]);
        }
    }
}

```

References:

- http://wiki.eclipse.org/Linux_Tools_Project/GProf/User_Guide
- <http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/SGIAItix/Doc/timing.html>
- <http://archive.ncsa.illinois.edu/lists/perftools/may02/msg00001.html>
- http://wiki.eclipse.org/Linux_Tools_Project/GCov/User_Guide
- https://bw-wiki.ncsa.uiuc.edu/display/~arnoldg/rait_tests