# Profiling

**Contents**

MPE, HPM, mp_profiler, Peekperf, and Xprofiler are part of the IBM High Performance Computing Toolkit (HPCT) which is described briefly here.

## MPE

The latest version of MPE (the MPI profiling library from ANL) is located in `/contrib/bgl/mpe2`.

To compile your program with MPE, use the following flags:

| | |
|---|---|
| include flags | `-I /contrib/bgl/mpe2/include` |
| link flags | `-L /contrib/bgl/mpe2/lib -llmpe -lmpe` |

**MPE Compile Example**

```
mpixlc -g -I /contrib/bgl/mpe2/include -c -o test_mpe.o test_mpe.c
mpixlc -o test_mpe test_mpe.o -g -L /contrib/bgl/mpe2/lib -llmpe -lmpe
```

## TAU

TAU (Tuning and Analysis Utilities) is a portable profiling and tracing toolkit. Full documentation is available here.

The TAU compilation scripts and analysis tools are available at `/contrib/bgl/tau/bin`. The TAU makefiles which control the instrumentation configuration are available under `/contrib/bgl/tau/lib`.

The most common method of instrumentation is to use TAU's compiler wrappers to automatically insert calls into the source code. To set this up, the appropriate Makefile and options must be specified, either in environment variables or as command-line options to the compiler wrappers.

**TAU configuration**

```
export TAU_MAKEFILE = /contrib/bgl/tau/lib/Makefile.tau-bgltimers-mpi-pdt
export TAU_OPTIONS = '-optVerbose -optNoRevert'
```

– or –

```
tau_f90.sh -tau_makefile=[path and makefile] tau_options=[options] example.F90
```

Specifying a Makefile controls what is profiled and how.

| Available Makefiles | |
|---|---|
| Makefile.tau-bgltimers-mpi-pdt | MPI profiling, light-weight BG/L timers (**recommended**) |
| Makefile.tau-mpi-pdt | MPI profiling, default timer |
| Makefile.tau-mpi-pdt-mpitrace | MPI tracing |
| Makefile.tau-papi-mpi-pdt | MPI profiling with PAPI counters |

## gprof

- Use the flags `-p -g` when compiling **and** linking your code. This produces gprof style files (`gmon.out.<proc num>`), which can be used with GNU gprof to get a call graph profile and see how long each subroutine is taking in the code Build:

```
/contrib/bgl/bin/mpxlc -p -g -c ring-hello.c
/contrib/bgl/bin/mpxlc -p -g -o ring-hello ring-hello.o
```

After running:

```
voran@fr0101ge:~> ls gmon.out.*
gmon.out.0  gmon.out.1  gmon.out.2  gmon.out.3  gmon.out.4
```

Run gprof:

```
gprof ring-hello gmon.out.0
```

## mpi_trace - MPI timing wrappers for BGL

- Installed in `/contrib/bgl/mpi_trace`
  Instructions for usage are in `/contrib/bgl/mpi_trace/README.mpi_trace`

### Counter Limitations

The reason the mpi_trace output in `bgl_counter_stats.txt`mentions that the "MFlops/GFlops values are suspect due to counter limitations", is that the ppc440 processors in the compute nodes cannot count a full set of FPU events. There are 4 sets of FPU operations that are counted, and **only one set can be counted at a time per processor**: add_subtract, mult_div, trinary (fma), and oedipus (SIMD ops). The FPU events are distributed cyclically among the MPI tasks, like this:

| MPI task | event |
|---|---|
| 0 | add_subtract |
| 1 | mult_div |
| 2 | trinary (fmadd) |
| 3 | oedipus (fpmadd) |
| 4 | add_subtract |
| 5 | mult_div |
| 6 | trinary |
| ... | |

Each event is averaged among the processors that counted that event, and then the total number of flops is calculated as `(add_sub) + (mult_div) + 2*trinary + 4*oedipus`. If you want to count one event on all processors, set the environment variable HPM_GROUP to 0, 1, 2, or 3.

## IBM's High Performance Computing Toolkit (HPCT)

### HPM

Located in `/contrib/bgl/hpm`, brief instructions are at `/contrib/bgl/hpm/doc/README`, and the event sets are listed in `/contrib/bgl/hpm/doc/event_sets.txt`
There are examples that you can copy to your own directory and try (complete with Makefiles) in `/contrib/bgl/hpm/example`:

- `test1` is in C
- `test2` is in fortran

Building test1:

```
voran@fr0101ge:~> cp /contrib/bgl/hpm/examples/test1 .
voran@fr0101ge:~> cd test1/
voran@fr0101ge:~> make clean default
rm -rf .o  *~ *core
rm -rf *.rts
rm -f .viz perfhpm*
/opt/ibmcmp/vac/7.0/bin/blrts_xlc -O3 -qarch=440 -qtune=440 -qhot \
  -I/bgl/BlueLight/ppcfloor/bglsys/include -I/contrib/bgl/hpm/include/ \
  -c -o sanity.rts.o sanity.c
/opt/ibmcmp/vac/7.0/bin/blrts_xlc -O3 -qarch=440 -qtune=440 -qhot \
  -I/bgl/BlueLight/ppcfloor/bglsys/include -I/contrib/bgl/hpm/include/ \
  -o sanity.rts sanity.rts.o -L/bgl/BlueLight/ppcfloor/bglsys/lib \
  -L/contrib/bgl/hpm/lib/ -lbgl_perfctr.rts -lmpich.rts -lmsglayer.rts \
  -ldevices.rts -lrts.rts -ldevices.rts -lrts.rts -lhpm.rts -lm
```

Running test1, counting event set 6:

```
cqsub -n 2 -t 10 -e HPM_EVENT_SET=6 sanity.rts
```

After it runs you will have these files in your directory:

```
hpm0001_sanity program_00000.viz  perfhpm00001.0
hpm0001_sanity program_00001.viz  perfhpm00001.1
```

The perfhpm* files are text files for each process that show the values of the counters for the event set specified by HPM_EVENT_SET, and the .viz files are for use with Peekperf.

### mp_profiler

- An MPI profiler and trace tool, installed in `/contrib/bgl/mp_profiler`. Instructions for usage are at `/contrib/bgl/mp_profiler/doc/README`
  There are examples in `/contrib/bgl/mp_profiler/examples`, complete with Makefiles.

### Peekperf

- Installed in `/contrib/fe_tools/peekperf`

### Xprofiler

- Installed in `/contrib/fe_tools/xprofiler`