Using HPCToolkit with Statically-Linked Programs

The HPCToolkit Team

March 31, 2009

Abstract

HPCToolkit is an integrated suite of tools that supports measurement, analysis, attribution and presentation of application performance for sequential and parallel programs. This paper describes how to link and run HPCToolkit with statically-linked applications.

1 Introduction

Some large-scale parallel systems, for example Compute-Node Linux and BG/P, do not support dynamically-linked executables on the compute nodes and run only fully statically-linked binaries. On these systems, you don't use `hpcrun` to run your program. Instead, you use `hpclink` to build a version of your program with the HPCToolkit profiler linked in.

2 Linking with `hpclink`

In the static case, `hpclink` is used to link the HPCToolkit libraries into your application. This does not require any source-code modifications, but it does involve a small change to the build procedure, namely the final linking step. You continue to make all the object (`.o`) files exactly as before, but in the last step, you use `hpclink` to link in the HPCToolkit code.

First, find out how to build your binary natively, without HPCToolkit, and look for the last step in the build process, the command that produces the single, statically-linked binary. Then, run the same command line, except with `hpclink` in front of it.

For example, suppose your binary is `myprog` and the last step in your `Makefile` combines various object files and libraries as follows.

```
mpicc -o myprog -static file.o ... -l<lib> ...
```

Then, you would build a version with HPCToolkit linked in with the following command line.

```
hpclink mpicc -o myprog -static file.o ... -l<lib> ...
```

In practice, you may want to edit your `Makefile` to always build two versions of your program, perhaps naming them `myprog` and `myprog.hpc`.

3 Running the Statically-Linked Binary

In the dynamic case, the `hpcrun` script sets environment variables to pass the script options to the HPCToolkit library, but `hpcrun` is not used in the static case. Instead, you will need to set these variables yourself in your launch script.

About the only variable you will need to set is `CSPROF_OPT_EVENT`, which controls the sampling events. This should be set to a space-separated list of `EVENT@COUNT` pairs. For example, in a PBS script in Bourne shell or bash syntax, you might write:

```
#!/bin/sh
#PBS -l size=64
#PBS -l walltime=01:00:00
cd $PBS_O_WORKDIR
export CSPROF_OPT_EVENT="PAPI_TOT_CYC@4000000 PAPI_L2_TCM@400000"
aprun -n 64 ./myprog arg ...
```

On the BG/P system, you use the `--env` option to pass environment variables. For example, you might submit a job with:

```
qsub -t 60 -n 64 --env CSPROF_OPT_EVENT="PAPI_TOT_CYC@4000000" \
    /path/to/myprog arg ...
```

4 Troubleshooting

With some compilers you may need to disable the interprocedural optimization. Statically, `hpclink` gets its hooks into a program with the `ld` option `--wrap` (see the ld(1) man page). The interprocedural optimization interferes with the `--wrap` option and prevents `hpclink` from linking the HPCToolkit libraries.