HPCToolkit Quickstart Guide

HPCToolkit Project Team

July 19, 2010

- *Documentation.* Man pages and user guides are available from the `share/doc/guides` directory of the release or at `http://hpctoolkit.org/documentation.html`.

- *Command-line help.* Help with the HPCToolkit performance tools `hpcrun`, `hpcstruct`, or `hpcprof` can be found by executing the command with the argument `--help`.

- *Measuring application performance.*

  For a sequential or multithreaded code:
  ```
  hpcrun [hpcrun options] myprogram [program arguments]
  ```
  For an MPI code:
  ```
  mpiexec -n 32 hpcrun [hpcrun options] myprogram [program arguments]
  ```
  [1]

  This command will produce a directory with the name `hpctoolkit-myprogram-measurements`. If an `hpcrun` command is run under control of PBS or GridEngine batch job schedulers, the name of the measurements directory will have its batch job id appended. For single-threaded programs, this directory will contain a single file with the suffix `myprogram-`*`nodeid`*`.hpcrun`. *Nodeid* is a hex value representing the host name for the node on which the measurements were collected. For parallel codes, performance measurements will be saved in files within the measurements directory with names that have the MPI rank (0 if none) and the thread id embedded.

- *Running statically-linked binaries.*
  On systems such as Compute-Node Linux and BG/P that run only statically-linked binaries on the compute nodes, use `hpclink` to build a statically-linked version of your application with the HPCToolkit library linked in. For example,
  ```
  hpclink mpicc -o myprog file.o ... -l<lib> ...
  ```
  Then, set the `HPCRUN_EVENT_LIST` environment variable in the launch script before running the application.
  ```
  export HPCRUN_EVENT_LIST="PAPI_TOT_CYC@4000000"
  aprun -n 64 myprog arg ...
  ```
  See the *Using* HPCToolkit *with Statically-Linked Programs* guide for more information.

- *Recovering program structure information.*
  ```
  hpcstruct myprogram
  ```
  This command will produce a file `myprogram.hpcstruct` that should be passed to `hpcprof` with the `-S` argument.

- *Analyzing measurements and correlating them to source code.*
  ```
  hpcprof     -S myprogram.hpcstruct -I path-to-myprogram-src/'*' \
     hpctoolkit-myprogram-measurements
  ```
  or
  ```
  mpi-launcher...  \
  hpcprof-mpi -S myprogram.hpcstruct -I path-to-myprogram-src/'*' \
     hpctoolkit-myprogram-measurements
  ```
  The difference between `hpcprof` and `hpcprof-mpi` is that the latter is designed to process (in parallel) measurements from large-scale executions. Either command will produce a directory with the name `hpctoolkit-myprogram-database`. If this database directory already exists, `hpcprof/hpcprof-mpi` will append its process id at the end of the aforementioned directory name to yield a unique name.

[1] Here, we assume that your MPI launcher is named `mpiexec`. It may be `mpirun` or something else depending on the MPI implementation you are using.

Another potentially important option, especially for machines that require executing from special file systems, is the `-R/--replace-path` option for substituting instances of *old-path* with *new-path*: `-R 'old-path=new-path'`.

- *Viewing application performance measurements.*
  ```
  hpcviewer hpctoolkit-myprogram-database
  ```
  Help for using `hpcviewer` can be found at `http://hpctoolkit.org/documentation.html` or in a help pane available within the viewer. The help pan can be brought up from `hpcviewer`'s `Help` menu.

- *Performance analysis techniques.* These strategies are described in the user guide *Effective Strategies for Analyzing Program Performance with* HPCToolkit, which is available as mentioned above in the documentation bullet.

  – A waste metric, which represents the difference between achieved performance and potential peak performance is a good way of understanding the potential for tuning the node performance of codes. `hpcviewer` supports synthesis of derived metrics to aid analysis. Derived metrics are specified within `hpcviewer` using spreadsheet-like formula. See the `hpcviewer` help pane for details about how to specify derived metrics.

  – Scalability bottlenecks in parallel codes can be pinpointed by differential analysis of two profiles with different degrees of parallelism.