

Using PAPI in an Educational Cluster

Vince Weaver <vincent.weaver@maine.edu>

1 PAPI Background

The Performance API (PAPI) is a library used for gathering performance information on various high-performance computing (HPC) systems. While originally designed for HPC, it can be used fine on desktop and laptop systems too.

You can find more information on the PAPI website: <https://icl.utk.edu/papi/>

2 Installing PAPI

Before using PAPI you will have to install it. Note that PAPI is only really supported on Linux systems. Also note that PAPI works best on real hardware. While it will run inside of a virtualized container, various features might not be supported when running in a VM.

2.1 Already Installed

If PAPI is already installed, you are all set! One way to tell is try running `papi_component_avail` if it runs it is installed and you can look at the output to see what hardware is supported.

2.2 Installing your Linux Distribution's Package

If you're lucky you are running on a Linux Distribution which has PAPI available as an installable package.

- Redhat (and related operating systems) does
- Debian and related have libpapi (libpapi-dev)
- Other operating systems might have support too?

2.3 Installing from Source

If your OS doesn't have support, or if you want to run a version newer than what your OS provides that you can compile from source. You can follow PAPI development here: <https://github.com/icl-utk-edu/papi>. I will describe building the most recent development version, there are also stable releases available

- Use git to download the newest version
- cd into the src direction
- run `./configure`

- If you want to have components enabled beyond just the CPU you will need to pass the argument BLAH to configure
- run make
- You can run tests to see if things are good
- run make install which by default installs to /usr/local you can change this when configuring setting prefix

3 Using the PAPI command line utilities

3.1 papi_component_avail

This tool lists all the components installed with PAPI and lists if they are working.

3.1.1 perf_event_paranoid

Note, if CPU events are not working most likely the `perf_event` paranoid value is not set properly. On Linux for security reasons performance counters are disabled by default. You need to set the value in `/proc` to ?

This requires root permissions and your sysadmin might need to set this up.

To have this happen automatically at boot you can create a file in `blah` called `blah` with these contents

3.2 papi_avail

This tool lists the pre-defined events available on this system.

Every processor architecture, and sometimes every processor version, has different sets of events available. PAPI tries to abstract this away by having a core set of pre-defined events that you can add that attempt to give you the same results no matter what system you run on. This includes things like `PAPI_TOT_CYC` which measures total CPU cycles, or `PAPI_TOT_INS` which measures total retired instructions.

Some of the pre-defined events aren't available on all processors. This sometimes frustratingly includes events you might want, like floating point event counts. Some of the events are also derived events, meaning that PAPI measures multiple events and combines them to give a result.

Be aware that your system only has so many event slots and so the pre-defined events might use them up more quickly by requiring multiple events.

3.3 papi_native_avail

This tool lists all “native” events on a system. These are the actual raw events that your system supports. You can use these instead of the pre-defined events, but be aware these events might not be portable across different systems.

If your system reports no events here it means PAPI (and the underlying libpfm4 library it uses for providing event lists) doesn't have support for your current processor. If you are running on new hardware you might check to see if a more recent version of PAPI might have better support.

4 Instrumenting Code with PAPI

5 Using PAPI with pthreads

TODO:

You need to notify PAPI you are running on pthreads

You might need to have PAPI record events in each thread separately.

In theory PAPI supports “inherited” events where the events are totaled for you, but I have not tested that.

6 Using PAPI with OpenMP

In theory you can use PAPI on OpenMP systems just like pthreads, but it possibly only works accidentally (because OpenMP uses pthreads behind the scenes).

7 Using PAPI with MPI

With MPI your code might be running across a whole cluster. PAPI has no built-in MPI support, you’ll have to use PAPI on each thread separately and if you want overall combined results you’ll have to gather and add them up yourself.

8 Using PAPI on GPUs

PAPI has support for various GPUs these days. Assuming PAPI was compiled with GPU support you can use it to measure GPU results.

8.1 NVIDIA/CUDA

TODO: CUDA example

8.2 AMD/CUPTI

TODO: AMD/CUPTI Example

8.3 Intel

TODO: Intel Example