CRAY
THE SUPERCOMPUTER COMPANY

A very short HOWTO for using CRAYPAT for the first time

Workshop at UDE
June 9-10
Stefan Andersson

- CRAYPAT is CRAYs tool for generating different profiles of your executable
- It is very powerful and the number of options and arguments can be overwhelming for a user starting working with it
- To address this issue, craypat offers a fast and easy path to get first results, the automatic profiling analysis (APA), which is explained in the following slides
- More information is located under ‘man pat’ and the command pat_help (after loading the xt-craypat module).

Sampling or Tracing

- First it's important to know that craypat can collect profiling data either by "sampling" or by "tracing".
- Sampling (fast, might miss something)
  - pat samples the program counter at a given time interval
- Tracing (slower, more accurate)
  - selected function entry points are traced and produce a data record in the runtime experiment data file each time the function is executed.
- Both approaches have advantages and disadvantages.

Step 1 : Create an instrumented version

- Create an instrumented executable using the sampling method
  - module load xt-craypat
  - make clean
  - make
  - pat_build -f -Oapa a.out
- This will load the craypat module, set the correct path to the needed libries and create an executable using a fast ‘sampling’ method of collecting the profiling data.

Step 2 : Run the executable

- **Run the executable created by craypat.**
  If your run script is complicated you can also rename the instrumented executable to a.out and run without any changes to your script.
  Even if you change the names, you shouldn't delete the original a.out created in step 1. It is needed for the following 'apa step'.
  After the job finishes successfully, you should see one *.xf file (or a new directory containing *.xf files) in your working directory.

Step 4 : Using the apa file

- The ascii apa file pat_report created in the last step can be used to create an instrumented executable where only the 'hotspot' routines get traced. The advantage is that the run with the traced routines contains more accurate information but still runs in an acceptable time (compared to trace all routines)

- **pat_build -O <apa file>**
- It will create a new a.out+apa
  Repeat step 2 and 3 with this new executable (you will of course not get a new apa file).

Step 3 : Create a pat report

- **pat_report <your xf file> > report.txt**

  This will create a simple pat_report output file and at the same time create a *.ap2 file, which contains the same information as the *.xf file but is more portable and is used by app2 (apprentice). At the same time pat_report will also create a trace option file *.apa (as –Oapa was specified with pat_build in step 2d)

  You can delete the *xf file at this point and continue to work with the .ap2 file.
  Do "man pat_report" to check for more options for the report output (calltree, callers ...).