The benchmark-run tool

Umut Acar, Arthur Charguéraud, Mike Rainey

17 October 2014

Synopsis

prun [*MODE*] [*PRUN_OPTIONS*] [*PROG_OPTIONS*]

or

prun [-mode *MODE*] [*PRUN_OPTIONS*] -args "[*PROG_OPTIONS*]"

If not specified, the *MODE* is set to "default". Other valid modes include the `speedup` mode.

Remark The later form with the explicity `-args` should be used if you want stability through updates of pbench that may add new options.

Description

The benchmark-run tool is a program that takes a specification of a set of runs of a given program, issues the specified runs to the system, and collects the output of the runs in a textual format. The tool relies on a certain formatting to be followed by the program that is specified to be run. The exact format is described below. The result of every run is recorded in the folder `_results/`.

Options

The benchmark-run options

The benchmark-run options selects the behavior of the program that collects data on benchmark runs of a specified program. The *PRUN_OPTIONS* can be zero or more of the following:

`-mode` ***MODE*** Specifies the mode *MODE*, among `default` or `speedup`. Defaults to `default`.

`--verbose` Print to stdout, for each run of the program, the exact command that was issued.

`--virtual` Only print the commands that would be issued.

`--dummy` Before performing the specified runs, issue a "dummy run" of the program.

`-runs` ***n*** Specifies for each specified combination of arguments a number of times n to run with the corresponding arguments (default is 1).

`-timeout` ***t*** Maximum amount of time to wait for each run to complete (in seconds, must be int > 0; or use -1 for no timeout (-1 is default)).

`-output` ***f*** Name of the filename f to receive the output printed by the runs (default "results.txt").

`-attempts` ***n*** Maximum number of attempts when runs fail (default is 1).

`-args` ***PROG_OPTIONS*** Specifies the combinations of arguments *PROG_OPTIONS* to issue to the specified program. It typically includes a '-prog' option.

`--normal` Clear the target results file before performing the runs.

`--append` Append the results of the run to the target results file, if the specified file exists; otherwise, writes results to a fresh results file.

`--complete` Perform only those runs that have not already been reported in the target results file (currently incompatible with `-runs`).

`--replace` Overwrite in the target results file only the run that are requested (currently incompatible with `-runs`).

`-output-mode [normal|append|complete|replace]` Like one of the options above.

Program options

The program options *PRUN_OPTIONS* defines the behavior of the programs to be run by the tool. A program run is specified by a string of program options, which should be in the form, e.g.:

```
prog1,prog2  -n 34,35  -m 3.2,4.5
```

or

```
-prog prog1,prog2  -n 34,35  -m 3.2,4.5
```

The cross product of all comma separated values is considered by the `prun` tool.

Remark The later form with the explicit `-prog` should be used if you want stability through updates of pbench that may add new options.

Limitation If your program expects arguments whose name conflicts with some of the names reserved by `prun`, such as `-prog` or `-timeout` or `run`, you will need to either change these names or set up a wrapper script around your program to rename the argument on the fly.

Examples

The following commands specify an experiment on the Fibonacci program. In the experiment, the program is run on two inputs, namely 39 and 40. Two algorithms are considered by the experiment, namely `recursive` and `cached` versions. For each distinct configuration, two runs are performed by the experiment.

```
make prun
make -C examples/basic fib
prun -prog examples/basic/fib -algo recursive,cached -n 39,40 -runs 2
```

The following command specifies an experiment where a program is run multiple times varying number of processors. The run where zero is passed as the number of processors by convention indicates a run of the sequential program which is used to get the baseline run time.

```
prun -prog examples/others/speedup.sh -proc 0,1,2,3,4
```

The following command adds an extra dimension to the speedup experiment: in this case, the program under consideration is run using two alternative algorithms.

```
prun -prog examples/others/speedup.sh -proc 0,1,2,3,4 -algo foo,bar
```

Speedup mode

The speedup mode is to be used for preparing data for `pplot speedup`.

Usage

prun speedup [*PRUN_OPTIONS*] [-baseline *COMMAND*] [-parallel *COMMAND*] -args "[*PROG_OPTIONS*]"

where *COMMAND* includes the name of a binary possibly with arguments.

`-baseline` **"[*COMMAND*]"** Provide a binary to be used for the baseline program; the binary is then passed the same "args" option as the parallel program.

`-parallel` **"[*PROG_OPTIONS*]"** Provide a binary to be used for the parallel program, and a combination of options to be used for the various parallel programs to benchmark.

`-args` "[***PROG_OPTIONS***]" Provide a combination of options; for each combination, the data for one speedup curve will be generated.

`-proc` ***proc1,proc2,...,procN*** Provide the list of processors to use.

`-baseline-runs` ***n*** Specify a number of runs n specific to the baseline evaluation.

`-baseline-timeout n` Specify a timeout specific to the baseline evaluation.

Example

```
make prun
prun speedup -baseline "examples/others/speedup.sh -algo foo -proc 0" -baseline-runs 1  -par
```