HBench - a GUI-based platform for performance benchmarking

HBench provides a simple one-form GUI for managing performance-oriented research of algorithms. It runs a specified engine over a set of benchmarks with multiple parameter sets. It collects statistics into a csv file, based on output emitted by the tested program to stdout in a standard format:

“### <field name> <numeric value>.

For example, to collect run-time and the number of iterations, make your program emit to stdout, e.g.,

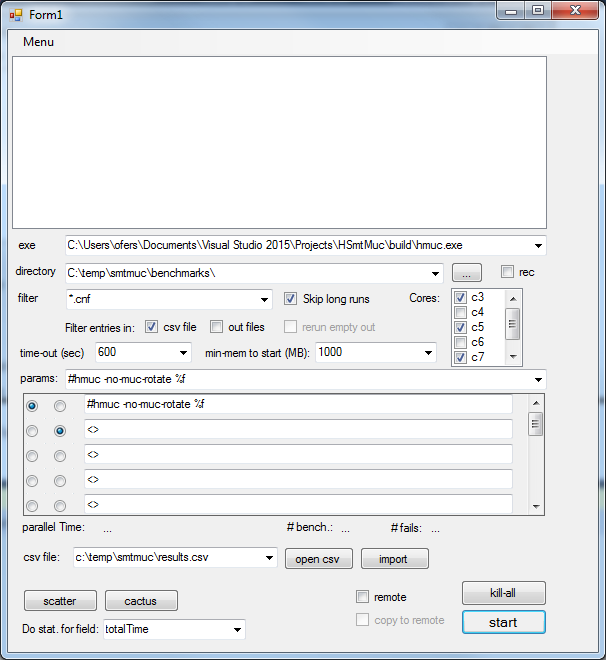
### time 20.2

### iterations 349

The csv file can then be opened in Excel for analysis.

HBench has a collection of utilities for controlling the process such as smart filtering, automatic generation of plots (in latex/pdf, via a tool called cpbm), control over the affinity of the processes (in windows machines), remote running on Linux-based clusters, and more.

The form looks as follows:



Each of the buttons/menu items has a tooltip which explains what it does. Here is a summary of those, top-down (other than the menu items, which are described last):

| **Control** | **Description** | **Available in ‘remote’ mode** |
| --- | --- | --- |
| exe | Absolute path to the executable. | No |
| directory | Benchmarks directory | Yes |
| rec | Read the benchmarks directory recursively | Yes |
| filter | Filters the benchmarks in ‘directory’ | Yes |
| filter/csv-file | Exclude benchmarks that have an entry in the csv file (see ‘csv file’ below). | Yes |
| filter/out-file | Exclude benchmarks that have a corresponding ‘.out’ file (see explanation below the table) | No |
| filter/ rerun empty-out | If Filter/out-file is enabled, with this option we ask to disregard empty ‘.out’ files. Such files typically indicate a failure owing to time-out/mem-out/crash… | No |
| skip long-runs | Once a benchmark times-out, it won’t be ran again with other parameters in this session. | yes |
| time-out | Time-out in sec. | No |
| min-mem to start | A lower-bound on the free memory to begin the next benchmark run. | No |
| cores | When running locally, you can determine the active cores. Bench recognizes the number of (logical) cores on your machine. It preserves cores 1,2 (if hyperthreading is enabled on your machine and the ‘hyperthreading’ key is set to true in the config file) for all other processes on your computer (e.g. your anti-virus), and retrieves the affinity to normal in the end of the run. If you have hyperthreading, your run-time results will be more repeatable if you use only the odd (or only the even) cores (because e.g., logical cores 3,4 are the same physical core). | No |
| params | The set of parameters to be executed.  In the comboBox a set of parameters is given as a comma-separated list of strings.  %f is swapped with the benchmark-name.  You may add a label preceded by # (e.g., “#afterBugFix”) as an extra parameter which will not be forwarded to the executable, but will still be part of the identifier of the benchmark (in the csv file and in the name of the .out file). This is useful when you make a change in the code and want to compare the “before” and “after” versions (i.e., the parameters are the same but the code has changed). | yes |
| csv file | The csv file that accumulates the results | Yes |
| open csv | Opens the csv file in Excel | Yes |
| import | Populates the csv file based on existing .out files. | Yes |
| scatter | Generate tex/pdf file with a scatter plot comparing the two parameters that are marked with the radio buttons on the left. This is based on the cpbm package (see installation instructions). | Yes |
| cactus | Generate tex file with a cactus plot comparing all parameters. This is based on the cpbm package (see installation instructions). | Yes |
| stat field | The field that is used for the Scatter / Cactus plots, e.g. “TotalRunTime”. | Yes |
| remote | Run on a remote machine (e.g., a linux-based cluster). Most of the configuration of this option is done in the config file (bench.exe.config) – see installation instructions. | Yes |
| copy to remote | Copy the benchmark file to the remote destination (which is determined by the field remote\_bench\_dir in the config file). You can leave it unchecked and hence save a lot of time if you already have all benchmarks on the remote destination. | Yes |
| kill-all | On windows this kills all processes with the exe name equal to the ‘exe’ field above. When running remotely on a linux machine, it sends a kill command to all processes under the user name specified in the field ‘remote\_user’ in the config file. | Yes |
| start | Start execution (locally / on a remote machine, depending on the ‘remote’ checkbox). | Yes |
| menu / cleanup / del fails from csv | Delete lines from the csv file, which correspond to a benchmark that was not solved by ALL parameters. | Yes |
| menu / cleanup / del shorts from csv | Delete from csv benchmarks that their runtime is < 1 sec. in at least one of the parameters | Yes |
| menu / cleanup / del all-fail benchmarks | delete \*benchmark files\* for which exe fails under all parameters according to the data in the csv files | Yes |
| menu / history file | History file maintains the history of the field-entries in the various comboBoxes/checkbox. If you edit it manually then either restart, or press Menu / refresh menus to apply the changes. | Yes |
| menu / advanced config | Opens the xml-based config file for editing. You may need to restart to apply your changes. | yes |

Some basic concepts:

1. The csv file accumulates the results of the benchmarks. It adds automatically the directory, benchmark file and an indication whether the benchmark ran to completion. Pressing ‘open csv’ opens it in Excel. Note that if you add pivot tables, graphs, etc. you need to save it as a .xlsx file.
2. Each run is recorded in a file with extension “.out”, with the parameters as part of the file name. For example running the file c:\temp\benchmarks\try.cnf with parameters –flat –exp 3 will produce the file c:\temp\benchmarks\try.cnf.flatexp3.out
3. Cores: HBench pushes all processes (those that windows allows) in your machine to cores 1,2 in order to free the others for benchmarking. If there is no hyper-threading then only to core 1. It retrieves the original affinity at the end of the run. You can choose from the remaining cores which ones to activate. On a machine with hyper-threading, the results are likely to be more reliable if you only activate the odd or even cores but not both, e.g., cores 3,5,7 if you have 8 logical cores.

Running on a remote machine.

1. You need to be able to access the remote machine via ssh without password (hence you need to set a private/public key access).

2. Change the fields prefixed with remote\_ in the config file. The complicated part is remote\_ssh\_cmd, which depends on the exact process you are running on the remote machine.

The example configuration assumes there is a script on the remote directory called hmuc.sh in the hmuc subdirectory, which takes two arguments ‘arg’ (the command-line parameters of the executables) and ‘out’ (the name of the output file).

Furthermore, it assumes that this benchmark is ran via PBS, which means that jobs are submitted via the qsub command. Here is how the command is specified:

<add key="remote\_ssh\_cmd"

value="cd ~/hmuc; qsub -v arg=\&quot;%p\&quot;,out=\&quot;%o\&quot; hmuc.sh"/>

The \&quot; sequence (including the semicolon) translates into \”

%p translates into the parameters

%o translates into the full output path

So the actual full ssh command can be e.g.,

ssh [ofers@tamnun.technion.ac.il](mailto:ofers@tamnun.technion.ac.il) ”cd ~/hmuc; qsub -v arg=\”-param1 –param2 try.cnf \”,out=\”$HOME/ofers/test/try.cnf.param1param2.out\” hmuc.sh”

Inside hmuc.sh we run the executable (hmuc in this case) and redirect its output into `out’:

./hmuc $arg | grep "###" > $out

The grep command is not mandatory, but it makes the output contain only the lines that are necessary and hence saves communication ime. These output files are later copied into the local machine for reading into the csv file.