Energy Aware Runtime (EAR) documentation

Performance and Power models validation

This document is part of the Energy Aware Runtime (EAR) framework. It has been created in the context of the BSC-Lenovo Cooperation project.

Contact

BSC: Julita Corbalan julita.corbalan@bsc.es, ear-support@bsc.es

LENOVO: hpchelp@lenovo.com

1 USE CASE DESCRIPTION

This document describes the main EAR steps to create the performance and power models coefficients used by EAR models for accurate projections. We will use a use case as reference. Assume a cluster with the following nodes:

Architecture	Cores per node	Max. frequency	Min. Frequency
Intel® Xeon® Gold 6148 Processor	40	2400000	1000000
Intel® Xeon® Gold 6126 Processor	12	2600000	1000000
Intel® Xeon® Gold 6130 Processor	16	2100000	1000000

2 CONFIGURING THE ENVIRONMENT

We must create null coefficients for each type of node and add it as default coefficients at ear.conf. EAR distribution includes a basic tool to automatically create null coefficients for a given range of frequency. It is not needed to consider all the frequencies since the idea is to provide a NULL reference to be used by EARL specific per architecture. This tool is located at src/tools.

```
./coeffs_null 6148 2400000 1000000
./coeffs_null 6126 2600000 1000000
./coeffs_null 6130 2100000 1000000
```

That will create three files coeffs.6148.default, coeffs.6126.default, and coeffs.6130.default. We must copy these files at the corresponding island folder, in our case $EAR_ETC/ear/coeffs/island0.

Special default coefficients per groups must be provided, for instance:

```
NodeName=cmp26[35-37] CPUS=32 Island=0 DefCoefficientsFile=coeffs.6130.default
```

With this definition, we will avoid using invalid coefficients for different architecture.

3 LEARNING PHASE EXECUTION

3.1 Preparing kernels for execution

The first step is the kernel compilation. In case you want to compile one by one manually, open the script `bin/scripts/learning/helpers/kernels_executor.sh` and look at the function `learning_phase()`. Here you will find all the paths and compile instructions (make) and install (move) each kernel when `$BENCH_MODE` variable is set to `compile`.

As you can see there are some confusing letters and numbers. This characters are related with the kernel customization, looking for maximize the nodes stress to obtain the maximum quality coefficients.

This customization will take the number of processes to fill the total number of CPUS (not counting hyperthreading). Also a class letter defining the stress level of the benchmark could be required. The class letter goes from lighter levels of stress (A) to heavier levels (D, E...).

To simplify the compilation just run the script `bin/scripts/learning/learning_compile.sh` after open it with an editor to set the number of cores, sockets and cores per socket.

3.2 Kernels execution

To launch the kernels the cluster queue manager SLURM will be used. To complete the learning phase all the kernels will have to be launched in every node of your cluster at the different selected frequencies. You can also maximize the precision of the gathered data by repeating the execution more than 1 time.

It is required the `srun` SLURM's command or `mpirun` bootstraping with SLURM. The `--ear-learning=$P_STATE` parameter next to the selected P_STATE, allows the EAR SLURM's plugin to execute a kernel in learning phase mode.

These are two examples of the `srun` and `mpirun` commands for a node of 40 CPUs:

```
`srun -N 1 -n 40 -J "bt-mz" -w node1001 --ear-policy=MONITORING_ONLY --ear-learning=1
/installation.path/bin/kernels/bt-mz.C.40`
```

```
`mpirun -n 40 -bootstrap slurm -bootstrap-exec-args="-J 'bt-mz' -w node1001
--ear-verbose=1 --ear-learning=1 /installation.path/bin/kernels/bt-mz.C.40`
```

It is required to use the same name for a kernel launched at different P_STATEs, because the binary which calculates the coefficients employs the name to classify the kernels.

To make the things easy, like the compiling phase, a script to do all the work is provided: `bin/scripts/learning/learning_compile.sh`. Run it after editing the same options and also setting the minimum to maximum P_STATE to take into account during the computation. As an example, if you want to compute from the P_STATE 1 to 6, set the minimum and maximum P_STATE options with 1 and 6 respectively.

3.3 Kernel adjustment

If after the launching of a learning phase kernel at P_STATE 1 the elapsed (in seconds) is between 60 and 120, then it is good quality kernel. In case some benchmarks are not between these times, you can increase or decrease the class letter mentioned in the point 3.1. If you want more information about kernel customization, please check the EAR documentation.

For the ease of use, the script `bin/scripts/tools/learning_show.sh` shows the results of the learning phase per node and per kernel. It receives as the first parameter a file containing a list of nodes separated by line breaks. Also you could add two parameters more to search in the output a particular data. For example you could look for the execution of 'bt-mz' kernel for the frequency '2400000'. In that case it will show you in a visually colored manner, the execution of that kernel at that frequency in all the nodes specified by your list, and easily localize the executions over 120 or under 60 seconds.

3.4 Learning data cleaning

If you detected that the generated data is not useful enough to produce good quality coefficients, a script to clean that data is included in `bin/scripts/tools/learning_delete.sh`. This script receives as the first parameter a file containing a list of nodes separated by line breaks. You will have to open it first to edit the MySQL password.

4 LEARNING PHASE VALIDATION

A tool for analyzing the learning phase is included in the package. This tool evaluates the odd values gathered during the learning phase and warns the user if, in example, too much power was consumed or time passed until the execution of a kernel was completed.

Next to this tool, a script is provided to use it with a list of nodes, resulting in a easily readable table to quickly identify the possible learning phase malfunctioning in some nodes. You can find this script in `bin/scripts/tools/learning_validate.sh`. This script requires as the first parameter, the path of the file containing a list of nodes splitted by line breaks.

5 COEFFICIENTS COMPUTATION AND ANALYSIS

Once coefficients have been computed, it is recommended to evaluate its quality. The first step is consist in the analysis of the coefficients over the training set (kernels). The algorithm will take the recently computed coefficients and the gathered data of the kernels execution to guess the time spent and energy consumed of the node.

For the ease of use, a script is provided to perform the analysis and compare the errors: `bin/scripts/tools/coeffs_quality.sh`. This scripts requires two parameters, the file containing a list of nodes separated by line breaks and also the frequency (in KHz) from which it is going to be projected. You will get the error percent between the real time and energy values that took the kernels to compute and the guessed. An error between 1 and 5% is considered good enough.

This script uses a binary named the same ‘coeffs_quality’ under the hood. You can add arguments to the script or just use that to perform a fine grained analysis. In example, you could just thresh the error per application per frequency per node and discover a failed kernel execution which messed up the weight of a coefficient.