Folding

User's guide manual

tools@bsc.es
Barcelona, 2015.

U User guide

U.1 Quick start guide

The Folding is a mechanism that provides instantaneous performance metrics, source code references and memory references[1]. This mechanism receives a trace-file (currently generated by Extrae - see further details on generating a trace-file for the Folding in Appendix G) and generates plots and an additional trace-file depicting the fine evolution of the performance. The Folding uses information captured through instrumentation and sampling mechanisms and smartly combines them. In this context, the samples are gathered from the computing into a synthetic region by preserving their relative time within their original region so that the sampled information determines how the performance evolves within the region. Consequently, the folded samples represent the progression in shorter periods of time no matter the monitoring sampling frequency, and also, the longer the runs the more samples get mapped into the synthetic instance. The framework has shown mean differences up to 5% when comparing results obtained sampling frequencies that are two orders of magnitude more frequent.

U.1.1 Decompressing the package

The Folding package is distributed in a .tar.bz2 file that can be uncompressed in the working directory by executing the following command:

```
tar xvz folding-1.0rc8-x86_64.tar.bz2
```

where `folding-1.0rc8-x86_64.tar.bz2` refers to the Folding package.

U.1.2 Contents of the package

After decompressing the package, the working directory should be populated with the directories (and corresponding descriptions) as listed in Table U.1.

[1]This last option is experimental at the moment of writing this document

Table U.1
Contents of the folding package.

Directory	Contents
bin/	Binary packages
etc/	
extrae-configurations/	Minimal configuration files for Extrae
models/	Configuration files to calculate performance models
basic/	
ibm-power5/	
intel-nehalem/	
intel-sandybridge/	
include/	Header files for the development of 3rd party tools
lib/	Libraries for the folding
share/	Miscellaneous files
cfg/	Configuration files for Paraver
doc/	Documentation
html	
examples/	
folding-writer/	Example on how to generate data for the folding
user-functions/	Sample tracefile with manually instrumented regions
clusters/	Sample tracefile with automatically detected regions

U.1.3 Quick run

This section provides examples of three types of execution of the Folding tool. These examples take benefit of the included sample trace-files from the package. For further information on how to generate trace-files for the Folding tool, check Appendix G.

U.1.3.1 Applied to manually instrumented regions

This first example uses a trace-file from the 444.namd SPEC benchmark that contains manually instrumented information that is located in

```
${FOLDING_HOME}/etc/share/examples/user-functions
```

This trace-file was generated by Extrae and delimiting the main loop using the Extrae API[2], more precisely the `Extrae_user_function` which emits events with label `User function` (or event type 60000019). To apply the Folding process to this trace-file, simply execute the following commands:

```
# cd ${FOLDING_HOME}/etc/share/examples/user-functions
# ${FOLDING_HOME}/bin/folding 444.namd.prv "User function"
```

U.1.3.2 Applied to automatically characterized regions

This example consists of a trace-file for the Nemo application when executed in MareNostrum3. This trace-file contains information regarding automatically characterized regions. This characterization has been done using the Clustering

[2]Please refer to `http://www.bsc.es/computer-sciences/performance-tools/documentation` for the latest Extrae User's Guide.

tool[3] This tool enriches the trace-file by adding events labeled as `Cluster ID` (and event type 90000001) into the trace-file. In this context, these events identify similar computation regions based on the event value. To apply the Folding process to this trace-file, simply execute the following commands:

```
# cd ${FOLDING_HOME}/etc/share/examples/user-functions
# ${FOLDING_HOME}/bin/folding \
  nemo.exe.128tasks.chop1.clustered.prv "Cluster ID"
```

This trace-file also contains all the necessary performance counters in order to take benefit of several performance models based on performance counters. Simply add the `-model intel-sandybridge` option to the Folding script to generate the plots with information of the models instead of providing each performance counter individually. The commands to execute should look like this:

```
# cd ${FOLDING_HOME}/etc/share/examples/user-functions
# ${FOLDING_HOME}/bin/folding -model intel-sandybridge \
  nemo.exe.128tasks.chop1.clustered.prv "Cluster ID"
```

U.1.4 Exploring the results

The Folding mechanism generates two types of output inside a directory named as the trace-file given (without the `.prv` suffix). The first type of results include a set of gnuplot files where each of these represents the evolution of the performance counters within the region. The tool also generates a Paraver trace-file with synthetic information derived from the Folding mechanism.

U.1.4.1 Using gnuplot

With respect to the gnuplot files, the Folding mechanism generates as many files as the combination of analyzed regions (clusters, OpenMP outlined routines, taskified OmpSs routines, or manually delimited regions) and the counters gathered during the application execution. The user can easily list the generated gnuplot files by calling `ls .gnuplot` within the directory created. The name of the gnuplot files contain the trace-file prefix, the identification of the region folded, and the performance counter shown. For instance, the example described in Section U.1.3.1 generates output files that can be explored by executing the command:

```
gnuplot -persist \
  444.namd.codeblocks.fused.any.any.any.main.Group_0.PAPI_TOT_INS.gnuplot
```

This file refers to the user routine `main` (which was manually instrumented) of the trace-file 444.namd.prv and provides information of the total graduated instructions (`PAPI_TOT_INS`). The user will notice that there are additional files for the different performance counters and they can explore them individually. The Folding also generates an additional plot that combines the metrics of all the counters into a single plot. This plot mainly provides information with respect to the MIPS rate (referenced on the right Y-axis), and ratio of the remaining performance counters per instruction (referenced on the left Y-axis). For the particular case of the example from Section U.1.3.1, this plot can be explored calling:

[3] Please refer to `http://www.bsc.es/computer-sciences/performance-tools/documentation` for the latest documentation with respect to the Clustering tool.

```
gnuplot -persist \
  444.namd.codeblocks.fused.any.any.any.main.Group_0.PAPI_TOT_INS.gnuplot
```

The aforementioned instructions also apply to the automatically delimited example described in Section U.1.3.2. In this case, the region names are numbered as `Cluster_1` to `Cluster_11`, but they also contain the trace-file prefix and the performance counters to explore them individually. If the user requested the performance models, then additional gnuplot files are created to provide information regarding these models. For the particular case of the Intel SandyBridge model, it generates three models that always generate the MIPS rate and add different metrics:

- *instruction-mix*
  Gives insight of the type of instructions executed along the region.
- *architecture-impact*
  Provides information regarding to the cache misses at different levels and the branch mispredicts along the region.
- *stall-distribution*
  This plot shows information regarding on which components of the processor are stalling the processor pipeline.

For instance, to open the instruction mix for the region labeled as Cluster 1 of the Nemo application executed in Section U.1.3.2, the user needs to open the plot invoking:

```
gnuplot -persist \
  nemo.exe.128tasks.chop1.clustered.codeblocks.fused.any.any.any.\
  Cluster_1.Group_0.instructionmix.gnuplot
```

The tool also provides a GUI-based tool to explore the plots. the user may invoke a visualizer named `wxfolding-viewer`, by invoking it from the newly created directory such as:

```
${FOLDING_HOME}/bin/wxfolding-viewer *.wxfolding
```

U.1.4.2 Using Paraver

The Folding process generates a trace-file with a suffix `.folded.prv` that lets Paraver to display some parts of the folded results. The Folding package includes several configuration files for Paraver to help analysing the results. We outline the following configuration files:

- *views/*
  - `win_folded_type.cfg`
    Generates a time-line that shows in which instances the Folding results have been integrated. This helps correlating the original trace-file and its contents with the folded trace-file. Notice that only one instance per type (where type refers to function, cluster, *etc.*..) is folded.

  - `win_folded_mips.cfg`
    Generates a time-line showing a signal of the MIPS rate within the folded instances.
  - `win_folded_processed_call-stack_caller.cfg`
    Generates a time-line showing the most time-dominant routines as they have been executed within the folded instances.
  - `win_folded_processed_call-stack_callerline.cfg`
    Generates a time-line showing the most time-dominant source code references (as pair of line and file) as they have been executed within the folded instances.

- *histograms/*
  - `3dh_folded_mips_per_caller.cfg`
    Generates a histogram that shows the achieved MIPS rate depending on the caller (columns) for a particular region folded.
  - `3dh_folded_mips_per_callerline.cfg`
    Generates a histogram that shows the achieved MIPS rate depending on the source code references (pair of line and file in columns) for a particular region folded.

G Generate a trace-file for the Folding

This chapter covers the minimum and necessary steps so as to configure Extrae [1] in order to use its resulting trace-files for the Folding process. There are three requirements when monitoring an application with Extrae in order to take the most benefit from the Folding tool. First, it is necessary to enable the sampling mechanism in addition to the instrumentation mechanism (see Section G.1). Second, it is convenient to collect the appropriate performance counters for the underlying processor (see Section G.2). Finally, Extrae needs to capture a segment of the call-stack in order to allow the Folding to provide information regarding the progression of the executed routines. The forthcoming sections provide information on how to enable these functionalities through the XML tags for the Extrae configuration file.

G.1 Enabling the sampling mechanism

Extrae is an instrumentation package that by default collects information from different parallel runtimes, including but not limited to: MPI, OpenMP, pthreads, CUDA and OpenCL (and even combinations of them). Extrae can be configured so that it also uses sampling mechanisms to capture performance metrics on a periodic basis. There are currently two alternatives to enable sampling in Extrae : using alarm signals and using performance counters. For the sake of simplicity, this document only covers the alarm-based sampling. However, if the reader would like to enable the sampling using the performance counters they must look at section 4.9 in the Extrae User's Manual for more details.

Listing G.1
Enable default time-based sampling in Extrae.

```
<sampling enabled="yes" type="default" period="50m" variability="10m"/>
```

[1]Please refer to `http://www.bsc.es/computer-sciences/performance-tools/documentation` for the latest Extrae User's Guide.

The XML statements in Listing G.1 need to be included in the Extrae configuration file. These statements indicate Extrae that sampling is enabled (`enabled="yes"`). They also tell Extrae to capture samples every 50 milliseconds (ms) with a random variability of 10 ms, that means that in practice samples will be randomly collected with a periodicity of 50±10 ms. With respect to type, it determines which timer domain is used (see `man 2 setitimer` or `man 3p setitimer` for further information on time domains). Available options are: `real` (which is also the `default` value, `virtual` and `prof` (which use the SIGALRM, SIGVTALRM and SIGPROF respectively). The default timing accumulates real time, but only issues samples at master thread. To let all the threads collect samples, the type must be set to either `virtual` or `prof`.

Additionally, the Folding mechanism is able to combine several performance models and generate summarized results that simplify understanding the behavior of the node-level performance. Since these performance models are heavily-tighted with the performance counters available on each processor architecture and family, the following sections provide Extrae XML configuration files ready to use on several architectures. Since each architecture has different characteristics, the user may need to tune the XML presented there to make sure that all the list performance counters are gathered appropriately.

G.2 Collecting the appropriate performance counters

The Folding mechanism provides, among other type of information, the progression of performance metrics along a delimited region through instrumentation points. These performance metrics include the progression of performance counters of every performance counter by default. To generate these kind of reports, Extrae must collect the performance counters during the application execution and this is achieved by defining counter sets into the `<counters>` section of the Extrae configuration file (see Section 4.19 of the Extrae User's guide for more information).

We have developed some performance models based on performance counters ratios among performance counters in order to ease the analysis of the reports. Each of these performance models aims at providing insight of different aspects of the application and system during the execution. Since the availability of the performance counters changes from processor to processor (even in the same processor family), the following sections describe the performance counters that are meant to be collected in order to calculate these performance models. These sections include the minimal `<counters>` sections to be added in a previously existing Extrae configuration file, but the Folding package also includes full Extrae configuration files in `${FOLDING_HOME}/etc/extrae-configurations`.

G.2.1 Intel SandyBridge processors

The listing G.2 indicates Extrae to prepare five performance counter sets with performance counters that are available on Intel SandyBridge processors. The collection of these performance counters allows the Folding to apply the models contained in the `${FOLDING_HOME}/etc/models/intel-sandybridge` that include: instruction mix, architecture impact and stall distribution.

Listing G.2
Counter definition sets for the Extrae configuration file when used on Intel SandyBridge procesors.

```
<cpu enabled="yes" starting-set-distribution="cyclic">
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L2_DCM,PAPI_L3_TCM
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_MSP,PAPI_BR_UCN,PAPI_BR_CN,RESOURCE_STALLS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_VEC_DP,PAPI_VEC_SP,PAPI_FP_INS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_LD_INS,PAPI_SR_INS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,RESOURCE_STALLS:LOAD,RESOURCE_STALLS:STORE,
    RESOURCE_STALLS:ROB_FULL,RESOURCE_STALLS:RS_FULL
  </set>
</cpu>
```

Listing G.3
Counter definition sets for the Extrae configuration file when used on Intel Nehalem processors.

```
<cpu enabled="yes" starting-set-distribution="cyclic">
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L2_DCM,PAPI_L3_TCM,
    RESOURCE_STALLS:SB,RESOURCE_STALLS:ROB
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_SR_INS,PAPI_BR_CN,PAPI_BR_UCN,PAPI_BR_MSP,
    PAPI_FP_INS,RESOURCE_STALLS:LB
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_LD_INS,PAPI_VEC_SP,PAPI_VEC_DP,
    RESOURCE_STALLS,RESOURCE_STALLS:RS
  </set>
</cpu>
```

G.2.2 Intel Nehalem processors

The listing G.3 indicates Extrae to prepare three performance counter sets with performance counters that are available on Intel Nehalem processors. The collection of these performance counters allows the Folding to apply the models contained in the `${FOLDING_HOME}/etc/models/intel-nehalem` that include: instruction mix, architecture impact and stall distribution.

Listing G.4
Counter definition sets for the Extrae configuration file when used on IBM Power5 procesors.

```
<cpu enabled="yes" starting-set-distribution="cyclic">
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_GCT_EMPTY_CYC,PM_LSU_LMQ_SRQ_EMPTY_CYC,
    PM_HV_CYC,PM_1PLUS_PPC_CMPL,PM_GRP_CMPL,PM_TB_BIT_TRANS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_FLUSH_BR_MPRED,PM_BR_MPRED_TA,
    PM_GCT_EMPTY_IC_MISS,PM_GCT_EMPTY_BR_MPRED,PM_L1_WRITE_CYC
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_LSU_FLUSH,PM_FLUSH_LSU_BR_MPRED,PM_CMPLU_STALL_LSU,
    PM_CMPLU_STALL_ERAT_MISS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_GCT_EMPTY_SRQ_FULL,PM_FXU_FIN,PM_FPU_FIN,
    PM_CMPLU_STALL_FXU,PM_FXU_BUSY,PM_CMPLU_STALL_DIV
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_IOPS_CMPL,PM_CMPLU_STALL_FDIV,PM_FPU_FSQRT,
    PM_CMPLU_STALL_FPU,PM_FPU_FDIV,PM_FPU_FMA
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PM_INST_CMPL,PM_CYC,PM_CMPLU_STALL_OTHER,PM_CMPLU_STALL_DCACHE_MISS,
    PM_LSU_DERAT_MISS,PM_CMPLU_STALL_REJECT,PM_LD_MISS_L1,PM_LD_REF_L1
  </set>
</cpu>
```

G.2.3 IBM Power5 processors

The listing G.4 indicates Extrae to prepare six performance counter sets with performance counters that are available on IBM Power5 (and similar) processors. The collection of these performance counters allows the Folding to calculate the CPIStack model for the IBM Power5 processor which is contained in `${FOLDING_HOME}/etc/models/powerpc-p5`.

G.2.4 Other architectures

The previous definitions of counter sets included performance counters that are available on the specific stated machines. Since these performance counters may not be available on all the systems, we provide also a group of counter sets that may be available on a variety of systems. Listing G.5 defines three Extrae counter sets that may be available on many systems (caveat here, not all systems may provide them). With the use of these counter sets, the Folding can apply the models contained in the `${FOLDING_HOME}/etc/models/basic` that include: instruction mix and architecture impact.

Listing G.5
Basic counter definition sets for other processors not stated before.

```
<cpu enabled="yes" starting-set-distribution="cyclic">
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L2_DCM,PAPI_L3_TCM
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_CN,PAPI_BR_UCN,PAPI_LD_INS,PAPI_SR_INS
  </set>
  <set enabled="yes" domain="all" changeat-time="500000us">
    PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_VEC_SP,PAPI_VEC_DP,PAPI_FP_INS,PAPI_BR_MSP
  </set>
</cpu>
```

Listing G.6
Collect call-stack information at sample points.

```
1 <callers enabled="yes">
2   <mpi enabled="yes">1-3</mpi>
3   <pacx enabled="no">1-3</pacx>
4   <sampling enabled="yes">1-5</sampling>
5 </callers>
```

G.3 Capturing the call-stack at sample points

By default, the sampling mechanism captures the performance counters indicated in the `counters` section and the Program Counter interrupted at the sample point. The Folding provides the instantaneous progression of the routines that last at least a minimum given duration. To enable this type of analysis, it is necessary to instruct Extrae to capture a portion of the call-stack during its execution. Listing G.6 shows how to enable the collection of the call-stack at the sample points in the Extrae configuration file. The mandatory lines to capture the call-stack at sample points are lines 1 and 4. Line 1 indicates that this section must be processed and Line 4 tells Extrae to capture levels 1 to 5 from the call-stack (where 1 refers to the level below to the top of the callstack).