Folding
User guide manual
for version 1.0rc5

tools@bsc.es

May 17, 2014

Contents

List of Figures

List of Tables

Chapter 1

Quick user guide

The Folding framework enables depicting the evolution of the performance counters as well as the source code locations by combining information from instrumented and sampled tracefiles. Before continuing this chapter, please make sure that you have generated a tracefile with both sampling and instrumented information. For further references on how to obtain a tracefile ready to apply folding, see Appendix A.

This chapter is divided different sections, each providing insight on how to execute the framework on different types of tracefiles (including MPI, OpenMP, OmpSs and manually instrumented).a The chapter ends with information on how to visualize the results from the framework on Section 1.2.

1.1 Applying the folding

Attention: No matter the code that is being instrumented (MPI, OpenMP, OmpSs, or even using manual instrumentation), it is imperative that the enclosing events have performance counters associated. If this requirement is not fulfilled, the folding process will abort. For a proper configuration on the instrumentation package to ensure that performance counters are emitted, refer to Extrae user guide.

1.1.1 On MPI tracefiles without any further instrumentation

In event the user does not know the application behavior of the application, or just gathered a tracefile with MPI information, it is strongly suggested to use the clustering tool (`BurstClustering`) from the ClusteringSuite[1] to automatically identify compute regions. Notice that variability in obtained clusters may affect the folding results, thus try to tune the clustering tool so as to generate clusters the more compact as possible.

To launch the folding on a clustered MPI tracefile named `mpi_app.clustered.prv`, just issue the following command:

```
${FOLDING_HOME}/bin/folding mpi_app.clustered.prv "Cluster ID"
```

[1]The ClusteringSuite can be downloaded from the BSC tools web page

1.1.2 On OpenMP tracefiles

Extrae instruments OpenMP outlined routines[2], therefore it is possible to apply the folding on tracefiles generated from OpenMP applications. To apply the folding on tracefiles generated from tracefiles, consider an OpenMP tracefile named `omp_app.prv` and issue the following command:

```
${FOLDING_HOME}/bin/folding omp_app.prv "Parallel function"
```

1.1.3 On OmpSs tracefiles

The OmpSs instrumentation package relies on Extrae to generate tracefiles. In addition to the regular instrumentation, OmpSs instrumentation plugin emits additional information into the tracefile reporting activities such when a taskified routine starts and stops. The folding mechanism can be applied to provide insight on these taskified routines. To do so, consider tracefile obtained through OmpSs is named `ompss_app.prv`, just issue the following command:

```
${FOLDING_HOME}/bin/folding ompss_app.prv "User Function Name"
```

1.1.4 Applying folding on tracefiles with manually added instrumentation

There exists the chance for the user to manually delimit the region of code on which the folding should apply. In this case, the user needs to insert events into the tracefile by either using calls to Extrae as `Extrae_user_function` or `Extrae_eventandcounters`.

In case the user takes benefit of the `Extrae_user_function`, the folding can take advantage of the unique event type identifier associated to user routines. To apply the folding mechanism on a tracefile named `user_functions_app.prv` with routine events, execute the following commands:

```
${FOLDING_HOME}/bin/folding user_functions_app.prv "User Function"
```

On the other hand, if the user emits additional events using `Extrae_eventandcounters`, it is appropriate to use a unique event type for all the instrumented regions but using a different event value for every different region. Consider that the tracefile `own_events.prv` contains an event type / value pair such as ¡ 1000, *loc* ¿ where *loc* determines different location points within the application. In order to apply the folding to this tracefile, the user shall invoke the folding such as:

```
${FOLDING_HOME}/bin/folding own_events.prv 1000
```

1.2 Outputs generated

The folding mechanism generates two types of output inside a directory named as the tracefile given (without the `.prv` suffix). A set of gnuplot files where each of these represents the evolution of the performance counters within the region and a Paraver tracefile with synthetic information derived from the folding mechanism. 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. To explore the gnuplot files, the user may invoke a visualizer named `wxfolding-viewer`, by invoking it from the newly created directory such as:

[2]Several compilers, when face a `#pragma omp parallel` region in the source code, wrap the contents of the region in an additional routine called outlined routine.

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

The user can also inspect the gnuplot files exploring the contents of the directory by using a call to `ls .gnuplot`. The name of the gnuplot files identifies the region and the performance counter analysed, and the user only needs to call gnuplot such as:

```
gnuplot -persist mpi_app.clustered..codeblocks.fused.any.any.any.Cluster_1.Group_0.PAPI_TOT_INS
```

The example given, for instance, would show the evolution of the graduated instructions of the region named Cluster 1.

Appendix A

Get a tracefile

In order to use the folding framework it is imperative to use tracefiles with performance metrics gathered using instrumentation and sampling. This chapter provides some Extrae configuration skeleton files that the analyst may use so as to instruct Extrae to generate tracefiles ready to be processed through the Folding mechanism. Note that these configuration files are extremely simplistic and only focus on setting an appropriate set of performance counters and ensure that the sampling mechanism is enabled (in these examples with a sampling period of 25 milliseconds). For any further information with respect to the Extrae configuration, please refer to the Extrae user guide available at the BSC website.

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. Note also, that an XML configuration file is provided in subsection A.1.3 for other architectures not listed here. 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.

A.1 XML configurations

A.1.1 For Intel SandyBridge processors

```
<?xml version='1.0'?>

<trace enabled="yes"
 home="/gpfs/apps/CEPBATOOLS/extrae/2.2.0"
 initial-mode="detail"
 type="paraver"
 xml-parser-id="Id: xml-parse.c 637 2011-05-30 10:47:47Z harald $"
>
  <mpi enabled="yes">
    <counters enabled="yes" />
  </mpi>
  <callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <pacx enabled="no">1-3</pacx>
    <sampling enabled="yes">1-5</sampling>
  </callers>
  <counters enabled="yes">
    <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,
        PAPI_FP_INS,RESOURCE_STALLS:SB
      </set>
      <set enabled="yes" domain="all" changeat-time="500000us">
        PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_CN,PAPI_BR_UCN,PAPI_BR_MSP,
        PAPI_LD_INS,PAPI_SR_INS,RESOURCE_STALLS:LB
      </set>
      <set enabled="yes" domain="all" changeat-time="500000us">
        PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_VEC_SP,PAPI_VEC_DP,RESOURCE_STALLS,
        RESOURCE_STALLS:RS,RESOURCE_STALLS:ROB
      </set>
    </cpu>
  </counters>
  <buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
  </buffer>
  <sampling enabled="yes" type="default" period="25m" />
  <merge enabled="yes" />
</trace>
```

A.1.2 For Intel Nehalem processors

```
<?xml version='1.0'?>
<trace enabled="yes"
 home="/gpfs/apps/CEPBATOOLS/extrae/2.2.0"
 initial-mode="detail"
 type="paraver"
 xml-parser-id="Id: xml-parse.c 637 2011-05-30 10:47:47Z harald $"
>
  <mpi enabled="yes">
    <counters enabled="yes" />
  </mpi>
  <callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <pacx enabled="no">1-3</pacx>
    <sampling enabled="yes">1-5</sampling>
  </callers>
  <counters enabled="yes">
    <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>
  </counters>
  <buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
  </buffer>
  <sampling enabled="yes" type="default" period="25m" />
  <merge enabled="yes" />
</trace>
```

A.1.3 For the remaining architectures

```
<?xml version='1.0'?>
<trace enabled="yes"
 home="/gpfs/apps/CEPBATOOLS/extrae/2.2.0"
 initial-mode="detail"
 type="paraver"
 xml-parser-id="Id: xml-parse.c 637 2011-05-30 10:47:47Z harald $"
>
  <mpi enabled="yes">
    <counters enabled="yes" />
  </mpi>
  <callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <pacx enabled="no">1-3</pacx>
    <sampling enabled="yes">1-5</sampling>
  </callers>
  <counters enabled="yes">
    <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>
  </counters>
  <buffer enabled="yes">
    <size enabled="yes">500000</size>
    <circular enabled="no" />
  </buffer>
  <sampling enabled="yes" type="default" period="25m" />
  <merge enabled="yes" />
</trace>
```