opari2

2.0 (revision 381)

Generated by Doxygen 1.7.3

Tue Jun 14 2011 16:39:33

Contents

1 opari2 **1**
- 1.1 USAGE . . . 1
- 1.2 CTC string decoding . . . 3
- 1.3 LINKING (startup initialization only) . . . 3
- 1.4 EXAMPLE . . . 4
- 1.5 NEWS . . . 4
  - 1.5.1 LINK STEP . . . 4
  - 1.5.2 POMP2 . . . 4
  - 1.5.3 POMP2_Parallel_fork . . . 4
  - 1.5.4 pomp_tpd . . . 5
- 1.6 SUMMARY . . . 5

2 Data Structure Index **7**
- 2.1 Data Structures . . . 7

3 File Index **9**
- 3.1 File List . . . 9

4 Data Structure Documentation **11**
- 4.1 POMP2_Region_info Struct Reference . . . 11
  - 4.1.1 Detailed Description . . . 12
  - 4.1.2 Field Documentation . . . 12
    - 4.1.2.1 mCriticalName . . . 12
    - 4.1.2.2 mEndFileName . . . 12
    - 4.1.2.3 mEndLine1 . . . 12
    - 4.1.2.4 mEndLine2 . . . 12
    - 4.1.2.5 mHasCopyIn . . . 12
    - 4.1.2.6 mHasCopyPrivate . . . 12
    - 4.1.2.7 mHasFirstPrivate . . . 12
    - 4.1.2.8 mHasLastPrivate . . . 12
    - 4.1.2.9 mHasNoWait . . . 13
    - 4.1.2.10 mHasOrdered . . . 13
    - 4.1.2.11 mHasReduction . . . 13
    - 4.1.2.12 mNumSections . . . 13
    - 4.1.2.13 mRegionType . . . 13
    - 4.1.2.14 mScheduleType . . . 13
    - 4.1.2.15 mStartFileName . . . 13
    - 4.1.2.16 mStartLine1 . . . 13
    - 4.1.2.17 mStartLine2 . . . 13

4.1.2.18 mUserGroupName 13
4.1.2.19 mUserRegionName 14

5 File Documentation **15**
5.1 opari2.dox File Reference 15
5.2 pomp2_lib.h File Reference 15
5.2.1 Detailed Description 17
5.2.2 Typedef Documentation 17
5.2.2.1 POMP2_Region_handle 17
5.2.3 Function Documentation 17
5.2.3.1 POMP2_Assign_handle 17
5.2.3.2 POMP2_Atomic_enter 17
5.2.3.3 POMP2_Atomic_exit 17
5.2.3.4 POMP2_Barrier_enter 18
5.2.3.5 POMP2_Barrier_exit 18
5.2.3.6 POMP2_Begin 18
5.2.3.7 POMP2_Critical_begin 18
5.2.3.8 POMP2_Critical_end 18
5.2.3.9 POMP2_Critical_enter 19
5.2.3.10 POMP2_Critical_exit 19
5.2.3.11 POMP2_Destroy_lock 19
5.2.3.12 POMP2_Destroy_nest_lock 19
5.2.3.13 POMP2_End 19
5.2.3.14 POMP2_Finalize 20
5.2.3.15 POMP2_Flush_enter 20
5.2.3.16 POMP2_Flush_exit 20
5.2.3.17 POMP2_For_enter 20
5.2.3.18 POMP2_For_exit 20
5.2.3.19 POMP2_Get_num_regions 21
5.2.3.20 POMP2_Get_opari2_version 21
5.2.3.21 POMP2_Implicit_barrier_enter 21
5.2.3.22 POMP2_Implicit_barrier_exit 21
5.2.3.23 POMP2_Init 21
5.2.3.24 POMP2_Init_lock 21
5.2.3.25 POMP2_Init_nest_lock 21
5.2.3.26 POMP2_Init_regions 22
5.2.3.27 POMP2_Master_begin 22
5.2.3.28 POMP2_Master_end 22
5.2.3.29 POMP2_Off 22
5.2.3.30 POMP2_On 22
5.2.3.31 POMP2_Parallel_begin 22
5.2.3.32 POMP2_Parallel_end 23
5.2.3.33 POMP2_Parallel_fork 23
5.2.3.34 POMP2_Parallel_join 23
5.2.3.35 POMP2_Section_begin 23
5.2.3.36 POMP2_Section_end 24
5.2.3.37 POMP2_Sections_enter 24
5.2.3.38 POMP2_Sections_exit 24
5.2.3.39 POMP2_Set_lock 24
5.2.3.40 POMP2_Set_nest_lock 24

5.2.3.41 POMP2_Single_begin . . . 25
5.2.3.42 POMP2_Single_end . . . 25
5.2.3.43 POMP2_Single_enter . . . 25
5.2.3.44 POMP2_Single_exit . . . 25
5.2.3.45 POMP2_Test_lock . . . 25
5.2.3.46 POMP2_Test_nest_lock . . . 26
5.2.3.47 POMP2_Unset_lock . . . 26
5.2.3.48 POMP2_Unset_nest_lock . . . 26
5.2.3.49 POMP2_Workshare_enter . . . 26
5.2.3.50 POMP2_Workshare_exit . . . 26
5.3 pomp2_region_info.h File Reference . . . 27
5.3.1 Detailed Description . . . 27
5.3.2 Enumeration Type Documentation . . . 28
5.3.2.1 POMP2_Region_type . . . 28
5.3.2.2 POMP2_Schedule_type . . . 28
5.3.3 Function Documentation . . . 29
5.3.3.1 ctcString2RegionInfo . . . 29
5.3.3.2 freePOMP2RegionInfoMembers . . . 29
5.3.3.3 pomp2RegionType2String . . . 30
5.3.3.4 pomp2ScheduleType2String . . . 30

Chapter 1

opari2

opari2 is a tool to automatically instrument C, C++ and Fortran source code files in which OpenMP is used. Around OpenMP directives function calls to a POMP2 API are inserted. By implementing these API detailed measurements regarding the runtime behaviour of an OpenMP application can be made. A conforming POMP2 implementation needs to implement all POMP2 functions, see pomp2_lib.h for a list of those.

A detailed description of the first opari version has been published by Mohr et al. in "Design and prototype of a performance tool interface for OpenMP" (Journal of supercomputing, 23, 2002).

1.1 USAGE

To create an instrumented version of an OpenMP application, each file of interest needs to be transformed by the OPARI2 tool. The application is then linked against a POMP2 runtime measurement library and optionally to a special initialization file (see section LINKING (startup initialization only) and SUMMARY for further details).

A call to opari2 has the following syntax:

```
Usage: opari2 [OPTION] ... infile [outfile]

with following options and parameters:
[--f77|--f90|--c|--c++] [OPTIONAL] Specifies the programming language of the
                        input source file. This option is only necessary if
                        the automatic language detection based on the input
                        file suffix fails.

[--nosrc]               [OPTIONAL] If specified, OPARI2 does not generate
                        #line constructs in the transformation process which
                        allow to preserve the original source file and line
                        number information. This option might be necessary if
                        the OpenMP compiler does not understand #line
                        constructs.  The default is to generate #line
                        constructs.

[--tpd]                 [OPTIONAL] Adds the clause 'copyin(<pomp_tpd>)' to any
                        parallel construct. This allowes to pass data from the
```

```
                        creating thread to its children. The variable is
                        declared externally in all files, so it needs to be
                        defined by the pomp library.

[--disable constructs]  [OPTIONAL] Disable the instrumentation of the more
                        fine-grained OpenMP constructs such as !$OMP ATOMIC.
                        constructs is a comma separated list of the constructs
                        for which the instrumentation should be disabled.
                        Accepted tokens are atomic, critical, master, flush,
                        single or locks as well as sync to disable all of
                        them.

[--tpd-mangling         [OPTIONAL] If programming languages are mixed (C and
 gnu|intel|sun|pgi|ibm] Fortran), the <pomp_tpd> needs to use the Fortran
                        mangled name also in C files. This option specifies to
                        use the mangling scheme of the gnu, intel, sun, pgi or
                        ibm compiler. The default is to use the mangling
                        scheme of the compiler used to build opari2.

[--version]             [OPTIONAL] Prints version information.

[--help]                [OPTIONAL] Prints this help text.

infile                  Input file name.

[outfile]               [OPTIONAL] Output file name. If not specified, opari2
                        uses the name infile.mod.suffix if the input file is
                        called infile.suffix.

Report bugs to <zih-silc-dev@groups.tu-dresden.de>.
```

If you run opari2 on the input file `example.c` it will create two files:

- `example.mod.c` is the instrumented version of `example.c`, i.e. the original code plus calls to the POMP2 API referencing handles to the OpenMP regions identified by opari2.

- `example.c.opari.inc` contains the OpenMP region handle definitions accompanied with all relevant data needed by the handles. This compile time context (CTC) information is encoded into a string for maximum portability. For each region, the tuple (region_handle, ctc_string) is passed to an initializing function (POMP2_Assign_handle()). All calls to these initializing functions are gathered in a function named POMP2_Init_regions_XXX_YY, where XXX_YY is a unique for each compilation unit.

At some point during runtime of the instrumented application, the region handles need to be initialized using the information stored in the CTC string. This can be done in one of of two ways:

- during *startup* of the measurement/POMP2 system, or
- during *runtime* when a region handles is accessed for the first time.

We *highly* recommend using the first option as it incurs much less runtime overhead than the second one (no locking, no lookup needed). If we want to go with startup-time

initialization, we need to call all POMP2_Init_regions_XXX_YY functions introduced by opari2. How this can be done is described in section LINKING (startup initialization only). For runtime initialization we provide the ctc string as argument to the relevant POMP2 function calls.

1.2 CTC string decoding

As mentioned above, we pass ctc strings to different POMP2 functions. These functions need to parse the string in order to process the encoded information. With POMP2_Region_info and ctcString2RegionInfo() the opari2 package provides means of doing this, see pomp2_region_info.h.

1.3 LINKING (startup initialization only)

As explained above, we need to call all POMP2_Init_regions_XXX_YY functions that can be found in the object files and libraries the application consists of. We do this by creating an additional compilation unit that will contain calls to following POMP2 functions:

- POMP2_Init_regions(),
- POMP2_Get_num_regions(), and
- POMP2_Get_opari2_version().

The resulting object file additionally needs to be linked to the application. During startup of the measurement system the only thing we need to do is to call POMP2_Init_regions() which in turn will call all POMP2_Init_regions_XXX_YY functions.

In order to create the additional compilation unit (let's name it `pomp2_init_file.c`) we can use the following command sequence:

```
% `opari2_config --nm` <objs_and_libs> |            \
  `opari2_config --egrep` -i "pomp2_init_regions" | \
  `opari2_config --egrep` " T " |                   \
  `opari2_config --awk_cmd` -f                      \
  `opari2_config --awk_script` > pomp2_init_file.c
```

Here, <objs_and_libs> denotes the entire set of object files and libraries that were instrumented by opari2.

For portability reasons we don't call `nm`, `egrep` and `awk` directly but via the provided `opari2_config` tool.

Please see section EXAMPLE for a basic example that demonstrates the entire workflow.

1.4 EXAMPLE

The directory <prefix>/share/opari/doc/example contains the following files:

```
example.c
example.f
Makefile
```

The Makefile contains all required information for building the instrumented and uninstrumented binaries. It demonstrates the compilation and linking steps as described above.

1.5 NEWS

1.5.1 LINK STEP

Opari2 uses a new mechanism to link files together. The advantage is, that no opari.rc file is needed anymore. Because of that, libraries can be preinstrumented and parallel builds are now possible as well. To achieve this, the handles for parallel regions are instrumented using a ctc_string. Here you can find further information on LINKING (startup initialization only) and on the CTC string decoding.

1.5.2 POMP2

All prefixes for functions have been changed from POMP to POMP2 because the new API is incompatible to the old one. The entire API, that might be utilized by opari2, can be found in pomp2_lib.h.

1.5.3 POMP2_Parallel_fork

The POMP2_Parallel_fork() call has an additional argument to pass the requested number of threads to the POMP2 library. This allowes the library to prepare datastructures and allocate memory for the threads before they are created. The value passed to the library is determined as follows:

- If a `num_threads` clause is present, the expression inside this clause is evaluated into a local variable `pomp_num_threads`. This variable is afterwards used in the call to POMP2_Parallel_fork() and in the num_threads clause itself.
- If no num_threads clause is present, omp_get_max_threads() is used to determine the requested value for the next parallel region. This value is stored in `pomp_num_threads` and passed to the POMP2_Parallel_fork() call.

In Fortran, instead of omp_get_max_threads(), a wrapper function pomp_get_max_threads_XXX_X is used. This function is needed to avoid multiple definitions of omp_get_max_threads() since we do not know whether it is defined in the user code or not.

Removing all definitions in the user code would require much more Fortran parsing than is done with opari2, since function definitions cannot easily be distinguished from variable definitions.

1.5.4 pomp_tpd

If it is nessesary for the POMP2 library to pass information from the master thread to the children, the option `--tpd` can be used. opari2 uses the copyin clause to pass a threadprivate variable `pomp_tpd` to the newly spawned threads at the beginning of a parallel region. The variable pomp_tpd is a 64 bit integer variable, since fortran does not allow pointers. Of course a pointer can be stored in this variable, passed to child threads with the copyin clause (in C/C++ or Fortran) and later on casted back to a pointer in the pomp library. To support mixed programming (C/Fortran) the variable name depends on the name mangling of the Fortran compiler. This means, for GNU, Sun, Intel and PGI C compilers the variable is called pomp_tpd_ and for IBM it is called pomp_tpd in C. In Fortran it is of course always called pomp_tpd. The --tpd-mangling option can be used to change this. The variable is declared extern in all program units, so the pomp library needs to contain the actual variable definition of pomp_tpd as an 64 bit integer.

1.6 SUMMARY

The typical usage of OPARI2 consists of the following steps:

1. Call OPARI2 for each input source file

```
% opari2 file1.f90
...
% opari2 fileN.f90
```

2. Compile all modified output files ∗.mod.∗ using the OpenMP compiler
3. Generate the initialization file

```
% `opari2_config --nm` file1.mod.o ... fileN.mod.o | \
  `opari2_config --egrep` -i "pomp2_init_regions" |  \
  `opari2_config --egrep` " T " |                    \
  `opari2_config --awk_cmd` -f                       \
  `opari2_config --awk_script` > pomp2_init_file.c
```

4. Link the resulting object files against the pomp2 runtime measurement library.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

POMP2_Region_info (This struct stores all information on an OpenMP region, like the region type or corresponding source lines. The function ctcString2RegionInfo() can be used to fill this struct with data from a ctcString) 11

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

pomp2_lib.h (This file contains the declarations of all POMP2 functions) . . 15
pomp2_region_info.h (This file contains function declarations and structs which handle informations on OpenMP regions. POMP2_Region_info is used to store these informations. It can be filled with a ctcString by ctcString2RegionInfo()) 27

Chapter 4

Data Structure Documentation

4.1 POMP2_Region_info Struct Reference

This struct stores all information on an OpenMP region, like the region type or corresponding source lines. The function ctcString2RegionInfo() can be used to fill this struct with data from a ctcString.

`#include <pomp2_region_info.h>`

Data Fields

Required attributes

- POMP2_Region_type mRegionType
- char * mStartFileName
- unsigned mStartLine1
- unsigned mStartLine2
- char * mEndFileName
- unsigned mEndLine1
- unsigned mEndLine2

Currently not provided by opari

- bool mHasCopyIn
- bool mHasCopyPrivate
- bool mHasFirstPrivate
- bool mHasLastPrivate
- bool mHasNoWait
- bool mHasOrdered
- bool mHasReduction
- POMP2_Schedule_type mScheduleType
- char * mUserGroupName

Attributes for specific region types

- unsigned mNumSections
- char * mCriticalName
- char * mUserRegionName

4.1.1 Detailed Description

This struct stores all information on an OpenMP region, like the region type or corresponding source lines. The function ctcString2RegionInfo() can be used to fill this struct with data from a ctcString.

4.1.2 Field Documentation

4.1.2.1 char* POMP2_Region_info::mCriticalName

name of a named critical region

4.1.2.2 char* POMP2_Region_info::mEndFileName

name of the corresponding source file from the closing pragma

4.1.2.3 unsigned POMP2_Region_info::mEndLine1

line number of the first line from the closing pragma

4.1.2.4 unsigned POMP2_Region_info::mEndLine2

line number of the last line from the closing pragma

4.1.2.5 bool POMP2_Region_info::mHasCopyIn

true if a copyin clause is present

4.1.2.6 bool POMP2_Region_info::mHasCopyPrivate

true if a copyprivate clause is present

4.1.2.7 bool POMP2_Region_info::mHasFirstPrivate

true if a firstprivate clause is present

4.1.2.8 bool POMP2_Region_info::mHasLastPrivate

true if a lastprivate clause is present

4.1.2.9 bool POMP2_Region_info::mHasNoWait

true if a nowait clause is present

4.1.2.10 bool POMP2_Region_info::mHasOrdered

true if an ordered clause is present

4.1.2.11 bool POMP2_Region_info::mHasReduction

true if a reduction clause is present

4.1.2.12 unsigned POMP2_Region_info::mNumSections

number of sections

4.1.2.13 POMP2_Region_type POMP2_Region_info::mRegionType

type of the OpenMP region

4.1.2.14 POMP2_Schedule_type POMP2_Region_info::mScheduleType

schedule type in the schedule clause

4.1.2.15 char* POMP2_Region_info::mStartFileName

name of the corresponding source file from the opening pragma

4.1.2.16 unsigned POMP2_Region_info::mStartLine1

line number of the first line from the opening pragma

4.1.2.17 unsigned POMP2_Region_info::mStartLine2

line number of the last line from the opening pragma

4.1.2.18 char* POMP2_Region_info::mUserGroupName

user group name

4.1.2.19 char* POMP2_Region_info::mUserRegionName

name of a user defined region

The documentation for this struct was generated from the following file:

- pomp2_region_info.h

Chapter 5

File Documentation

5.1 opari2.dox File Reference

5.2 pomp2_lib.h File Reference

This file contains the declarations of all POMP2 functions.

Typedefs

- typedef void * POMP2_Region_handle

Functions

- void POMP2_Assign_handle (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Atomic_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Atomic_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Barrier_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Barrier_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Begin (POMP2_Region_handle *pomp2_handle)
- void POMP2_Critical_begin (POMP2_Region_handle *pomp2_handle)
- void POMP2_Critical_end (POMP2_Region_handle *pomp2_handle)
- void POMP2_Critical_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Critical_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Destroy_lock (omp_lock_t *s)
- void POMP2_Destroy_nest_lock (omp_nest_lock_t *s)
- void POMP2_End (POMP2_Region_handle *pomp2_handle)

- void POMP2_Finalize ()
- void POMP2_Flush_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Flush_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_For_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_For_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Implicit_barrier_enter (POMP2_Region_handle *pomp2_handle)
- void POMP2_Implicit_barrier_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Init ()
- void POMP2_Init_lock (omp_lock_t *s)
- void POMP2_Init_nest_lock (omp_nest_lock_t *s)
- void POMP2_Master_begin (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Master_end (POMP2_Region_handle *pomp2_handle)
- void POMP2_Off ()
- void POMP2_On ()
- void POMP2_Parallel_begin (POMP2_Region_handle *pomp2_handle)
- void POMP2_Parallel_end (POMP2_Region_handle *pomp2_handle)
- void POMP2_Parallel_fork (POMP2_Region_handle *pomp2_handle, int num_threads, const char ctc_string[])
- void POMP2_Parallel_join (POMP2_Region_handle *pomp2_handle)
- void POMP2_Section_begin (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Section_end (POMP2_Region_handle *pomp2_handle)
- void POMP2_Sections_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Sections_exit (POMP2_Region_handle *pomp2_handle)
- void POMP2_Set_lock (omp_lock_t *s)
- void POMP2_Set_nest_lock (omp_nest_lock_t *s)
- void POMP2_Single_begin (POMP2_Region_handle *pomp2_handle)
- void POMP2_Single_end (POMP2_Region_handle *pomp2_handle)
- void POMP2_Single_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Single_exit (POMP2_Region_handle *pomp2_handle)
- int POMP2_Test_lock (omp_lock_t *s)
- int POMP2_Test_nest_lock (omp_nest_lock_t *s)
- void POMP2_Unset_lock (omp_lock_t *s)
- void POMP2_Unset_nest_lock (omp_nest_lock_t *s)
- void POMP2_Workshare_enter (POMP2_Region_handle *pomp2_handle, const char ctc_string[])
- void POMP2_Workshare_exit (POMP2_Region_handle *pomp2_handle)

Functions generated by the instrumenter

- size_t POMP2_Get_num_regions ()
- void POMP2_Init_regions ()
- const char * POMP2_Get_opari2_version ()

5.2.1 Detailed Description

This file contains the declarations of all POMP2 functions. alpha

Authors

Daniel Lorenz <d.lorenz@fz-juelich.de> Dirk Schmidl <schmidl@rz.rwth-aachen.de>

5.2.2 Typedef Documentation

5.2.2.1 typedef void* POMP2_Region_handle

Handles to identify OpenMP regions.

5.2.3 Function Documentation

5.2.3.1 void POMP2_Assign_handle (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Registers a POMP2 region and returns a region handle.

Parameters

pomp2_handle	Returns the handle for the newly registered region.
ctc_string	A string containing the region data.

5.2.3.2 void POMP2_Atomic_enter (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called before an atomic statement.

Parameters

pomp2_handle	The handle of the started region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.3 void POMP2_Atomic_exit (POMP2_Region_handle * *pomp2_handle*)

Called after an atomic statement.

Parameters

pomp2_handle	The handle of the ended region.

5.2.3.4 void POMP2_Barrier_enter (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called before a barrier.

Parameters

pomp2_handle	The handle of the started region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.5 void POMP2_Barrier_exit (POMP2_Region_handle * *pomp2_handle*)

Called after a barrier.

Parameters

pomp2_handle	The handle of the ended region.

5.2.3.6 void POMP2_Begin (POMP2_Region_handle * *pomp2_handle*)

Called at the begin of a user defined POMP2 region.

Parameters

pomp2_handle	The handle of the started region.

5.2.3.7 void POMP2_Critical_begin (POMP2_Region_handle * *pomp2_handle*)

Called at the start of a critical region.

Parameters

pomp2_handle	The handle of the started region.

5.2.3.8 void POMP2_Critical_end (POMP2_Region_handle * *pomp2_handle*)

Called at the end of a critical region.

Parameters

pomp2_handle	The handle of the ended region.

5.2.3.9 void POMP2_Critical_enter (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called before a critical region.

Parameters

pomp2_handle	The handle of the started region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.10 void POMP2_Critical_exit (POMP2_Region_handle * *pomp2_handle*)

Called after a critical region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.11 void POMP2_Destroy_lock (omp_lock_t * *s*)

Wraps the omp_destroy_lock function.

Parameters

s	The OpenMP lock to destroy.

5.2.3.12 void POMP2_Destroy_nest_lock (omp_nest_lock_t * *s*)

Wraps the omp_destroy_nest_lock function.

Parameters

s	The nested OpenMP lock to destroy.

5.2.3.13 void POMP2_End (POMP2_Region_handle * *pomp2_handle*)

Called at the begin of a user defined POMP2 region.

Parameters

pomp2_handle	The handle of the started region.

5.2.3.14 void POMP2_Finalize ( )

Finalizes the POMP2 adapter. It is inserted at the #pragma pomp inst end.

5.2.3.15 void POMP2_Flush_enter (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called before an flush.

Parameters

pomp2_handle	The handle of the started region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.16 void POMP2_Flush_exit (POMP2_Region_handle * *pomp2_handle*)

Called after an flush.

Parameters

pomp2_handle	The handle of the ended region.

5.2.3.17 void POMP2_For_enter (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called before a for loop.

Parameters

pomp2_handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.18 void POMP2_For_exit (POMP2_Region_handle * *pomp2_handle*)

Called after a for loop.

Parameters

pomp2_handle	The handle of the region.

5.2.3.19 size_t POMP2_Get_num_regions ( )

Returns the number of instrumented regions.

The instrumenter scans all opari-created include files with nm and greps the POMP2_-INIT_uuid_numRegions() function calls. Here we return the sum of all numRegions.

5.2.3.20 const char* POMP2_Get_opari2_version ( )

Returns the opari version.

5.2.3.21 void POMP2_Implicit_barrier_enter (POMP2_Region_handle * *pomp2_handle*)

Called before an implicit barrier.

Parameters

pomp2_-handle	The handle of the started region.

5.2.3.22 void POMP2_Implicit_barrier_exit (POMP2_Region_handle * *pomp2_handle*)

Called after an implicit barrier.

Parameters

pomp2_-handle	The handle of the started region.

5.2.3.23 void POMP2_Init ( )

Initializes the POMP2 adapter. It is inserted at the #pragma pomp inst begin.

5.2.3.24 void POMP2_Init_lock (omp_lock_t * *s*)

Wraps the omp_init_lock function.

Parameters

s	The OpenMP lock to initialize.

5.2.3.25 void POMP2_Init_nest_lock (omp_nest_lock_t * *s*)

Wraps the omp_init_nest_lock function.

Parameters

s	The nested OpenMP lock to initialize.

5.2.3.26 void POMP2_Init_regions ( )

Init all opari-created regions.

The instrumentor scans all opari-created include files with nm and greps the POMP2_-INIT_uuid_numRegions() function calls. The instrumentor then defines this functions by calling all grepped functions.

5.2.3.27 void POMP2_Master_begin (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called at the start of a master region.

Parameters

pomp2_-handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.28 void POMP2_Master_end (POMP2_Region_handle * *pomp2_handle*)

Called at the end of a master region.

Parameters

pomp2_-handle	The handle of the ended region.

5.2.3.29 void POMP2_Off ( )

Disables the POMP2 adapter.

5.2.3.30 void POMP2_On ( )

Enables the POMP2 adapter.

5.2.3.31 void POMP2_Parallel_begin (POMP2_Region_handle * *pomp2_handle*)

Called at the start of a parallel region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.32 void POMP2_Parallel_end (POMP2_Region_handle * *pomp2_handle*)

Called at the end of a parallel region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.33 void POMP2_Parallel_fork (POMP2_Region_handle * *pomp2_handle*, int *num_threads*, const char *ctc_string[]*)

Called before a parallel region.

Parameters

pomp2_handle	The handle of the region.
num_threads	Upper bound for number of child threads
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.34 void POMP2_Parallel_join (POMP2_Region_handle * *pomp2_handle*)

Called after a parallel region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.35 void POMP2_Section_begin (POMP2_Region_handle * *pomp2_handle*, const char *ctc_string[]*)

Called at the start of a section.

Parameters

pomp2_handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.36 void POMP2_Section_end (POMP2_Region_handle * pomp2_handle)

Called at the end of a section.

Parameters

pomp2_handle	The handle of the region.

5.2.3.37 void POMP2_Sections_enter (POMP2_Region_handle * pomp2_handle, const char ctc_string[])

Called before a set of sections.

Parameters

pomp2_handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.38 void POMP2_Sections_exit (POMP2_Region_handle * pomp2_handle)

Called after a set of sections.

Parameters

pomp2_handle	The handle of the region.

5.2.3.39 void POMP2_Set_lock (omp_lock_t * s)

Wraps the omp_set_lock function.

Parameters

s	The OpenMP lock to set.

5.2.3.40 void POMP2_Set_nest_lock (omp_nest_lock_t * s)

Wraps the omp_set_nest_lock function

Parameters

s	The nested OpenMP lock to set.

5.2.3.41 void POMP2_Single_begin (POMP2_Region_handle * pomp2_handle)

Called at the start of a single region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.42 void POMP2_Single_end (POMP2_Region_handle * pomp2_handle)

Called at the end of a single region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.43 void POMP2_Single_enter (POMP2_Region_handle * pomp2_handle, const char ctc_string[])

Called before a single region.

Parameters

pomp2_handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.44 void POMP2_Single_exit (POMP2_Region_handle * pomp2_handle)

Called after a single region.

Parameters

pomp2_handle	The handle of the region.

5.2.3.45 int POMP2_Test_lock (omp_lock_t * s)

Wraps the omp_test_lock function

Parameters

s	the OpenMP lock to test for.

5.2.3.46 int POMP2_Test_nest_lock (omp_nest_lock_t * s)

Wraps the omp_test_nest_lock function

Parameters

s	The nested OpenMP lock to test for.

5.2.3.47 void POMP2_Unset_lock (omp_lock_t * s)

Wraps the omp_unset_lock function.

Parameters

s	the OpenMP lock to unset.

5.2.3.48 void POMP2_Unset_nest_lock (omp_nest_lock_t * s)

Wraps the omp_unset_nest_lock function

Parameters

s	The nested OpenMP lock to unset.

5.2.3.49 void POMP2_Workshare_enter (POMP2_Region_handle * pomp2_handle, const char ctc_string[])

Called before a workshare region.

Parameters

pomp2_handle	The handle of the region.
ctc_string	Initialization string. May be ignored if <pomp2_handle> is already initialized.

5.2.3.50 void POMP2_Workshare_exit (POMP2_Region_handle * pomp2_handle)

Called after a workshare region.

Parameters

pomp2_handle	The handle of the region.

5.3 pomp2_region_info.h File Reference

This file contains function declarations and structs which handle informations on OpenMP regions. POMP2_Region_info is used to store these informations. It can be filled with a ctcString by ctcString2RegionInfo().

Data Structures

- struct POMP2_Region_info

  This struct stores all information on an OpenMP region, like the region type or corresponding source lines. The function ctcString2RegionInfo() can be used to fill this struct with data from a ctcString.

Enumerations

- enum POMP2_Region_type {
  POMP2_No_type, POMP2_Atomic, POMP2_Barrier, POMP2_Critical,
  POMP2_Do, POMP2_Flush, POMP2_For, POMP2_Master,
  POMP2_Parallel, POMP2_Parallel_do, POMP2_Parallel_for, POMP2_Parallel_sections,
  POMP2_Parallel_workshare, POMP2_Sections, POMP2_Single, POMP2_User_region,
  POMP2_Workshare }
- enum POMP2_Schedule_type {
  POMP2_No_schedule, POMP2_Static, POMP2_Dynamic, POMP2_Guided,
  POMP2_Runtime, POMP2_Auto }

Functions

- void ctcString2RegionInfo (const char ctcString[ ], POMP2_Region_info *regionInfo)
- void freePOMP2RegionInfoMembers (POMP2_Region_info *regionInfo)
- const char * pomp2RegionType2String (POMP2_Region_type regionType)
- const char * pomp2ScheduleType2String (POMP2_Schedule_type scheduleType)

5.3.1 Detailed Description

This file contains function declarations and structs which handle informations on OpenMP regions. POMP2_Region_info is used to store these informations. It can be filled with a ctcString by ctcString2RegionInfo().

Author

Christian Rössel <c.roessel@fz-juelich.de> alpha

Date

Started Fri Mar 20 16:30:45 2009

5.3.2 Enumeration Type Documentation

5.3.2.1 enum POMP2_Region_type

POMP2_Region_type

Enumerator:

POMP2_No_type
POMP2_Atomic
POMP2_Barrier
POMP2_Critical
POMP2_Do
POMP2_Flush
POMP2_For
POMP2_Master
POMP2_Parallel
POMP2_Parallel_do
POMP2_Parallel_for
POMP2_Parallel_sections
POMP2_Parallel_workshare
POMP2_Sections
POMP2_Single
POMP2_User_region
POMP2_Workshare

5.3.2.2 enum POMP2_Schedule_type

type to store the scheduling type of a for worksharing constuct

Enumerator:

POMP2_No_schedule
POMP2_Static
POMP2_Dynamic
POMP2_Guided
POMP2_Runtime
POMP2_Auto

5.3.3 Function Documentation

5.3.3.1 void ctcString2RegionInfo (const char *ctcString[]*, POMP2_Region_info * *regionInfo*)

ctcString2RegionInfo() fills the POMP2_Region_info object with data read from the ctcString. If the ctcString does not comply with the specification, the program aborts with exit code 1.

Rationale: ctcString2RegionInfo() is used during initialization of the measurement system. If an error occurs, it is better to abort than to struggle with undefined behaviour or *guessing* the meaning of the broken string.

Note

Can be called from multiple threads concurrently, assuming malloc is thread-safe. ctcString2RegionInfo() will assign memory to the members of *regionInfo*. You are supposed to to release this memory by calling freePOMP2RegionInfoMembers().

Parameters

ctcString	A string in the format "length∗key=value∗[key=value]∗". The length field is parsed but not used by this implementation. Possible values for key are listed in ctcTokenMap. The string must at least contain values for the keys `regionType`, `sscl` and `escl`. Possible values for the key `regionType` are listed in regionTypesMap. The format for `sscl` resp. `escl` values is "`filename:lineNo1:lineNo2`".
regionInfo	must be a valid object

Postcondition

At least the required attributes (see POMP2_Region_info) are set.
All other members of *regionInfo* are set to 0 resp. false resp. POMP2_No_schedule.
If `regionType=sections` than POMP2_Region_info::mNumSections has a value > 0.
If `regionType=region` than POMP2_Region_info::mUserRegionName has a value != 0.
If `regionType=critical` than POMP2_Region_info::mCriticalName may have a value != 0.

5.3.3.2 void freePOMP2RegionInfoMembers (POMP2_Region_info * *regionInfo*)

Free the memory of the regionInfo members.

Parameters

regionInfo	The regioninfo to be freed.

5.3.3.3 const char* pomp2RegionType2String (POMP2_Region_type *regionType*)

converts regionType into a string

Parameters

regionType	The regionType to be converted.

5.3.3.4 const char* pomp2ScheduleType2String (POMP2_Schedule_type *scheduleType*)

converts scheduleType into a string

Parameters

schedule-Type	The scheduleType to be converted.

Index

ctcString2RegionInfo
    pomp2_region_info.h, 29

freePOMP2RegionInfoMembers
    pomp2_region_info.h, 29

mCriticalName
    POMP2_Region_info, 12
mEndFileName
    POMP2_Region_info, 12
mEndLine1
    POMP2_Region_info, 12
mEndLine2
    POMP2_Region_info, 12
mHasCopyIn
    POMP2_Region_info, 12
mHasCopyPrivate
    POMP2_Region_info, 12
mHasFirstPrivate
    POMP2_Region_info, 12
mHasLastPrivate
    POMP2_Region_info, 12
mHasNoWait
    POMP2_Region_info, 12
mHasOrdered
    POMP2_Region_info, 13
mHasReduction
    POMP2_Region_info, 13
mNumSections
    POMP2_Region_info, 13
mRegionType
    POMP2_Region_info, 13
mScheduleType
    POMP2_Region_info, 13
mStartFileName
    POMP2_Region_info, 13
mStartLine1
    POMP2_Region_info, 13
mStartLine2
    POMP2_Region_info, 13
mUserGroupName
    POMP2_Region_info, 13
mUserRegionName
    POMP2_Region_info, 13

opari2.dox, 15

POMP2_Atomic
    pomp2_region_info.h, 28
POMP2_Auto
    pomp2_region_info.h, 28
POMP2_Barrier
    pomp2_region_info.h, 28
POMP2_Critical
    pomp2_region_info.h, 28
POMP2_Do
    pomp2_region_info.h, 28
POMP2_Dynamic
    pomp2_region_info.h, 28
POMP2_Flush
    pomp2_region_info.h, 28
POMP2_For
    pomp2_region_info.h, 28
POMP2_Guided
    pomp2_region_info.h, 28
POMP2_Master
    pomp2_region_info.h, 28
POMP2_No_schedule
    pomp2_region_info.h, 28
POMP2_No_type
    pomp2_region_info.h, 28
POMP2_Parallel
    pomp2_region_info.h, 28
POMP2_Parallel_do
    pomp2_region_info.h, 28
POMP2_Parallel_for
    pomp2_region_info.h, 28
POMP2_Parallel_sections
    pomp2_region_info.h, 28
POMP2_Parallel_workshare
    pomp2_region_info.h, 28
pomp2_region_info.h

POMP2_Atomic, 28
POMP2_Auto, 28
POMP2_Barrier, 28
POMP2_Critical, 28
POMP2_Do, 28
POMP2_Dynamic, 28
POMP2_Flush, 28
POMP2_For, 28
POMP2_Guided, 28
POMP2_Master, 28
POMP2_No_schedule, 28
POMP2_No_type, 28
POMP2_Parallel, 28
POMP2_Parallel_do, 28
POMP2_Parallel_for, 28
POMP2_Parallel_sections, 28
POMP2_Parallel_workshare, 28
POMP2_Runtime, 28
POMP2_Sections, 28
POMP2_Single, 28
POMP2_Static, 28
POMP2_User_region, 28
POMP2_Workshare, 28
POMP2_Runtime
pomp2_region_info.h, 28
POMP2_Sections
pomp2_region_info.h, 28
POMP2_Single
pomp2_region_info.h, 28
POMP2_Static
pomp2_region_info.h, 28
POMP2_User_region
pomp2_region_info.h, 28
POMP2_Workshare
pomp2_region_info.h, 28
POMP2_Assign_handle
pomp2_lib.h, 17
POMP2_Atomic_enter
pomp2_lib.h, 17
POMP2_Atomic_exit
pomp2_lib.h, 17
POMP2_Barrier_enter
pomp2_lib.h, 17
POMP2_Barrier_exit
pomp2_lib.h, 18
POMP2_Begin
pomp2_lib.h, 18
POMP2_Critical_begin
pomp2_lib.h, 18
POMP2_Critical_end
pomp2_lib.h, 18
POMP2_Critical_enter
pomp2_lib.h, 19
POMP2_Critical_exit
pomp2_lib.h, 19
POMP2_Destroy_lock
pomp2_lib.h, 19
POMP2_Destroy_nest_lock
pomp2_lib.h, 19
POMP2_End
pomp2_lib.h, 19
POMP2_Finalize
pomp2_lib.h, 19
POMP2_Flush_enter
pomp2_lib.h, 20
POMP2_Flush_exit
pomp2_lib.h, 20
POMP2_For_enter
pomp2_lib.h, 20
POMP2_For_exit
pomp2_lib.h, 20
POMP2_Get_num_regions
pomp2_lib.h, 20
POMP2_Get_opari2_version
pomp2_lib.h, 21
POMP2_Implicit_barrier_enter
pomp2_lib.h, 21
POMP2_Implicit_barrier_exit
pomp2_lib.h, 21
POMP2_Init
pomp2_lib.h, 21
POMP2_Init_lock
pomp2_lib.h, 21
POMP2_Init_nest_lock
pomp2_lib.h, 21
POMP2_Init_regions
pomp2_lib.h, 22
pomp2_lib.h, 15
POMP2_Assign_handle, 17
POMP2_Atomic_enter, 17
POMP2_Atomic_exit, 17
POMP2_Barrier_enter, 17
POMP2_Barrier_exit, 18
POMP2_Begin, 18
POMP2_Critical_begin, 18
POMP2_Critical_end, 18
POMP2_Critical_enter, 19
POMP2_Critical_exit, 19
POMP2_Destroy_lock, 19
POMP2_Destroy_nest_lock, 19

POMP2_End, 19
POMP2_Finalize, 19
POMP2_Flush_enter, 20
POMP2_Flush_exit, 20
POMP2_For_enter, 20
POMP2_For_exit, 20
POMP2_Get_num_regions, 20
POMP2_Get_opari2_version, 21
POMP2_Implicit_barrier_enter, 21
POMP2_Implicit_barrier_exit, 21
POMP2_Init, 21
POMP2_Init_lock, 21
POMP2_Init_nest_lock, 21
POMP2_Init_regions, 22
POMP2_Master_begin, 22
POMP2_Master_end, 22
POMP2_Off, 22
POMP2_On, 22
POMP2_Parallel_begin, 22
POMP2_Parallel_end, 23
POMP2_Parallel_fork, 23
POMP2_Parallel_join, 23
POMP2_Region_handle, 17
POMP2_Section_begin, 23
POMP2_Section_end, 24
POMP2_Sections_enter, 24
POMP2_Sections_exit, 24
POMP2_Set_lock, 24
POMP2_Set_nest_lock, 24
POMP2_Single_begin, 24
POMP2_Single_end, 25
POMP2_Single_enter, 25
POMP2_Single_exit, 25
POMP2_Test_lock, 25
POMP2_Test_nest_lock, 25
POMP2_Unset_lock, 26
POMP2_Unset_nest_lock, 26
POMP2_Workshare_enter, 26
POMP2_Workshare_exit, 26
POMP2_Master_begin
pomp2_lib.h, 22
POMP2_Master_end
pomp2_lib.h, 22
POMP2_Off
pomp2_lib.h, 22
POMP2_On
pomp2_lib.h, 22
POMP2_Parallel_begin
pomp2_lib.h, 22
POMP2_Parallel_end
pomp2_lib.h, 23
POMP2_Parallel_fork
pomp2_lib.h, 23
POMP2_Parallel_join
pomp2_lib.h, 23
POMP2_Region_handle
pomp2_lib.h, 17
POMP2_Region_info, 11
mCriticalName, 12
mEndFileName, 12
mEndLine1, 12
mEndLine2, 12
mHasCopyIn, 12
mHasCopyPrivate, 12
mHasFirstPrivate, 12
mHasLastPrivate, 12
mHasNoWait, 12
mHasOrdered, 13
mHasReduction, 13
mNumSections, 13
mRegionType, 13
mScheduleType, 13
mStartFileName, 13
mStartLine1, 13
mStartLine2, 13
mUserGroupName, 13
mUserRegionName, 13
pomp2_region_info.h, 27
ctcString2RegionInfo, 29
freePOMP2RegionInfoMembers, 29
POMP2_Region_type, 28
POMP2_Schedule_type, 28
pomp2RegionType2String, 29
pomp2ScheduleType2String, 30
POMP2_Region_type
pomp2_region_info.h, 28
POMP2_Schedule_type
pomp2_region_info.h, 28
POMP2_Section_begin
pomp2_lib.h, 23
POMP2_Section_end
pomp2_lib.h, 24
POMP2_Sections_enter
pomp2_lib.h, 24
POMP2_Sections_exit
pomp2_lib.h, 24
POMP2_Set_lock
pomp2_lib.h, 24
POMP2_Set_nest_lock
pomp2_lib.h, 24

POMP2_Single_begin
pomp2_lib.h, 24
POMP2_Single_end
pomp2_lib.h, 25
POMP2_Single_enter
pomp2_lib.h, 25
POMP2_Single_exit
pomp2_lib.h, 25
POMP2_Test_lock
pomp2_lib.h, 25
POMP2_Test_nest_lock
pomp2_lib.h, 25
POMP2_Unset_lock
pomp2_lib.h, 26
POMP2_Unset_nest_lock
pomp2_lib.h, 26
POMP2_Workshare_enter
pomp2_lib.h, 26
POMP2_Workshare_exit
pomp2_lib.h, 26
pomp2RegionType2String
pomp2_region_info.h, 29
pomp2ScheduleType2String
pomp2_region_info.h, 30