

OpenMP in GCC

Ben Elliston
bje@au.ibm.com

Australia Development Laboratory
IBM Australia

19/05/09

1

Legal stuff

This presentation represents the work of the author and does not necessarily represent the views of IBM.

OpenMP is a trademark of the OpenMP Architecture Review Board in the United States and other countries.

Company, product, and service names may be trademarks or service marks of others.

19/05/09

2

Introduction

What is OpenMP?

Why is it important?

History

Implementation in GCC

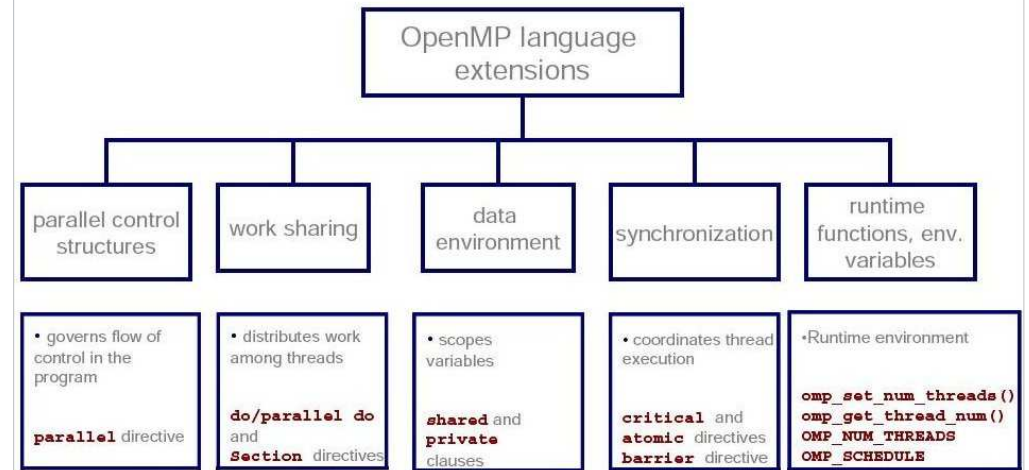
Tour of the transformations

A peek at the runtime library code

19/05/09

3

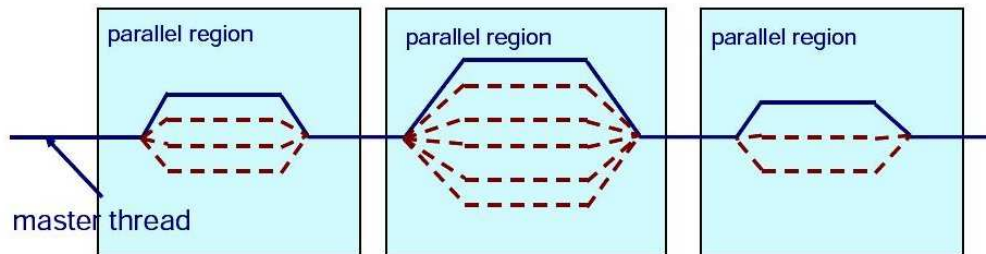
OpenMP constructs



19/05/09
Source: http://en.wikipedia.org/wiki/Image:Omp_lang_ext.jpg

4

OpenMP parallelism



19/05/09
Source: http://en.wikipedia.org/wiki/Image:Fork_join.jpeg

5

History of OpenMP ARB

FORTRAN spec 1.0, Oct '97

C/C++ spec 1.0, Oct '98

FORTRAN spec 2.0, Nov '00

C/C++ spec 2.0, Mar '02

Combined spec 2.5, May '05

Spec version 3.0, Nov '08

19/05/09

6

History of GNU OpenMP

Developed mostly by Red Hat engineers

Branch formed ~2004

Merged into mainline, Dec 2005

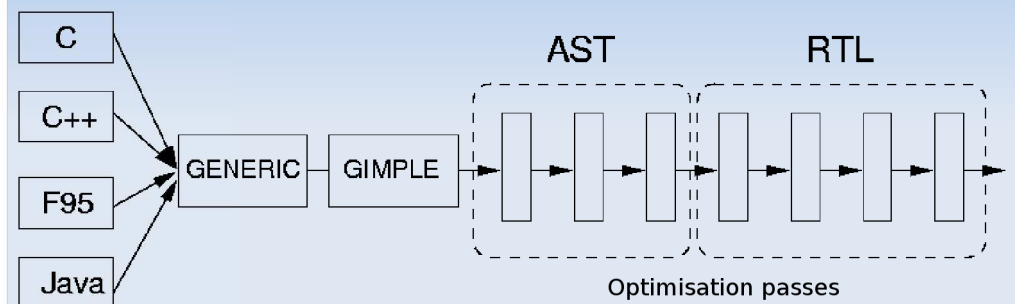
In gcc 4.2 and above

19/05/09

7

Implementation (directives)

source file → intermediate representations (gimple, rtl) → assembly source



19/05/09

8

Implementation (runtime)

Library of support routines: libgomp

Provides portable interface for GCC

On UNIX systems, call down to pthreads

19/05/09

9

C example

```
$ gcc foo.c -S -fdump-tree-all -fopenmp

#include <omp.h>
#include <stdio.h>

int main (int argc, char *argv[]) {
    int id, nthreads;
    #pragma omp parallel private (id)
    {
        id = omp_get_thread_num ();
        printf ("Hello World from thread %d\n", id);
        #pragma omp barrier
        if (id == 0) {
            nthreads = omp_get_num_threads ();
            printf ("There are %d threads\n", nthreads);
        }
    }
    return 0;
}
```

19/05/09

10

GIMPLE representation

foo.c.004t.gimple

```
main (argc, argv)
{
  int D.2221;
  int id;
  int nthreads;

  #pragma omp parallel private(id) shared(nthreads)
  {
    {
      int id.0;
      int nthreads.1;

      id.0 = omp_get_thread_num ();
      id = id.0;
      printf (&"Hello World from thread %d\n"[0], id);
      builtin_GOMP_barrier ();
      if (id == 0)
        {
          nthreads.1 = omp_get_num_threads ();
          nthreads = nthreads.1;
          printf (&"There are %d threads\n"[0], nthreads);
        }
      else
        {
        }
    }
  }
  D.2221 = 0;
  return D.2221;
}
```

19/05/09

11

GIMPLE after lowering

foo.c.010t.omplower

```
main (argc, argv)
{
  int D.2229;
  int D.2230;
  struct .omp_data_s.2 .omp_data_o.3;

  {
    .omp_data_o.3.nthreads = nthreads;
    #pragma omp parallel private(id) shared(nthreads) [child fn: main.omp_fn.0 (.omp_data_o.3)]
    {
      .omp_data_i = &.omp_data_o.3;
      id.0 = omp_get_thread_num ();
      id = id.0;
      printf (&"Hello World from thread %d\n"[0], id);
      builtin_GOMP_barrier ();
      if (id == 0)
        {
          nthreads.1 = omp_get_num_threads ();
          .omp_data_i->nthreads = nthreads.1;
          D.2229 = .omp_data_i->nthreads;
          D.2230 = D.2229;
          printf (&"There are %d threads\n"[0], D.2230);
        }
      else
        {
        }
    }
  }
  OMP_RETURN
  nthreads = .omp_data_o.3.nthreads;
}
D.2221 = 0;
return D.2221;
}
```

19/05/09

12

GIMPLE expansion (1/2)

foo.c.022t.ompexp

```
main (argc, argv)
{
  int nthreads;
  int id;
  int D.2221;
  struct .omp_data_s.2 .omp_data_o.3;

  <bb 2>:
  .omp_data_o.3.nthreads = nthreads;
  __builtin_GOMP_parallel_start (main.omp_fn.0, &.omp_data_o.3, 0);
  main.omp_fn.0 (&.omp_data_o.3);
  __builtin_GOMP_parallel_end ();
  nthreads = .omp_data_o.3.nthreads;
  D.2221 = 0;
  return D.2221;
}
```

19/05/09

13

GIMPLE expansion (2/2)

foo.c.022t.ompexp

```
main.omp_fn.0 (.omp_data_i)
{
  int D.2230;
  int D.2229;
  int nthreads.1;
  int id.0;
  int id;
  int nthreads [value-expr: .omp_data_i->nthreads];

  <bb 2>:
  id.0 = omp_get_thread_num ();
  id = id.0;
  printf (&"Hello World from thread %d\n"[0], id);
  __builtin_GOMP_barrier ();
  if (id == 0) goto <L0>; else goto <L1>;

  <L1>;
  return;

  <L0>;
  nthreads.1 = omp_get_num_threads ();
  .omp_data_i->nthreads = nthreads.1;
  D.2229 = .omp_data_i->nthreads;
  D.2230 = D.2229;
  printf (&"There are %d threads\n"[0], D.2230);
  goto <bb 3> (<L1>);
}
```

19/05/09

14

Finally, generated code!

foo.s

```
main:
    leal    4(%esp), %ecx
    andl   $-16, %esp
    pushl  -4(%ecx)
    pushl  %ebp
    movl   %esp, %ebp
    pushl  %ecx
    subl   $36, %esp
    movl   -8(%ebp), %eax
    movl   %eax, -16(%ebp)
    movl   $0, 8(%esp)
    leal   -16(%ebp), %eax
    movl   %eax, 4(%esp)
    movl   $main.omp_fn.0, (%esp)
    call   GOMP_parallel_start
    leal   -16(%ebp), %eax
    movl   %eax, (%esp)
    call   main.omp_fn.0
    call   GOMP_parallel_end
```

19/05/09

15

Peek into libgomp: parallel.c

```
void
GOMP_parallel_start (void (*fn) (void *), void *data, unsigned num_threads)
{
    num_threads = gomp_resolve_num_threads (num_threads);
    gomp_team_start (fn, data, num_threads, NULL);
}

void
GOMP_parallel_end (void)
{
    gomp_team_end ();
}
```

19/05/09

16

Peek into libgomp: team.c

```
/* Launch a team. */
void
gomp_team_start (void (*fn) (void *), void *data, unsigned nthreads,
                 struct gomp_work_share *work_share)
{
  /* ... */

  /* Launch new threads. */
  for (; i < nthreads; ++i, ++start_data)
    {
      /* ... */
      err = pthread_create (&pt, attr, gomp_thread_start, start_data);
      if (err != 0)
        gomp_fatal ("Thread creation failed: %s", strerror (err));
    }
}
```

19/05/09

17

Summary

OpenMP support now established in GCC

Implemented at various levels through compiler

Implementation is conceptually simple

Resources:

GCC source code

<http://www.openmp.org/>

<http://gcc.gnu.org/projects/gomp/>

19/05/09

18