Implementing a minimal OpenMP runtime

Rafel Albert Bros Esqueu

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

rafel.albert.bros@est.fib.upc.edu

Funcionalitats implementades

S'ha seguit el primer itinerari:

- `Parallel`.
- `Critical`.
- `Barrier`.
- `Single`.
- `For` dinàmic amb clàusules `wait` i `nowait`.
- Opcional: `Barrier` utilitzant operacions atòmiques.
- Opcional: Intent (no funcional) d'implementar gestió de tasques.

Barrier (amb operacions atòmiques)

```
void GOMP_barrier ()
{
/* Barrera simple: */
  //pthread_barrier_wait (&miniomp_barrier);

/* Barrera casolana: */
  int numT = omp_get_num_threads ();

  while (__atomic_load_n (&sortirB, __ATOMIC_SEQ_CST));

  if (__atomic_add_fetch (&comptadorB, 1, __ATOMIC_SEQ_CST) == numT)
    __atomic_store_n (&sortirB, true, __ATOMIC_SEQ_CST);

  while (! __atomic_load_n (&sortirB, __ATOMIC_SEQ_CST));

  if (__atomic_add_fetch (&comptadorB, -1, __ATOMIC_SEQ_CST) == 0)
    __atomic_store_n (&sortirB, false, __ATOMIC_SEQ_CST);

  return;
}
```

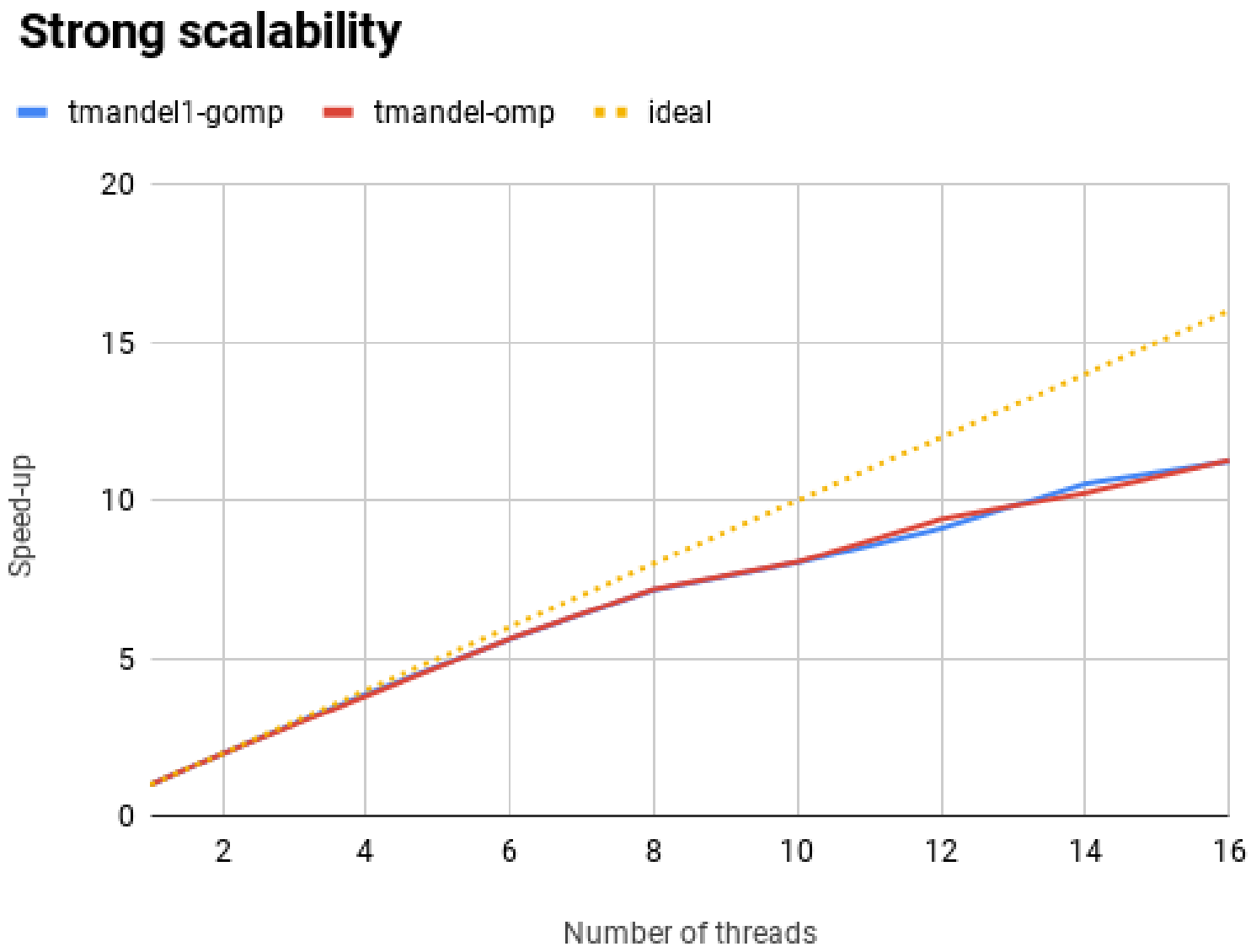
Single

```
/* This routine is called when first encountering
   a SINGLE construct. Returns true if this is the
   thread that should execute the clause. */

bool
GOMP_single_start (void)
{
  int id = omp_get_thread_num ();
  miniomp_parallel[id].comptadorS++;

  int val = miniomp_parallel[id].comptadorS;

  return (__sync_bool_compare_and_swap (&comptadorS, val - 1, val));
}
```

For

```
[...]

int id = miniomp_parallel[omp_get_thread_num ()].loop;

if (__sync_bool_compare_and_swap (&loopC, id - 1, loopC + 1))
{
  loops = (miniomp_loop_t *) realloc (loops,
      (loopC + 1) * sizeof (miniomp_loop_t));
  loops[id].ini = false;
}

if (__sync_bool_compare_and_swap (&loops[id].ini, false, true))
{

[...]
```

Estudi de rendiment



Implementing a minimal OpenMP runtime

Rafel Albert Bros Esqueu

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

rafel.albert.bros@est.fib.upc.edu