Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

OpenMP Tasking: *Past, Present and Future*

Sergi Mateo Bellido
Compiler engineer

11/2018

SC18 Dallas, TX | hpc inspires.

Tasking history: past

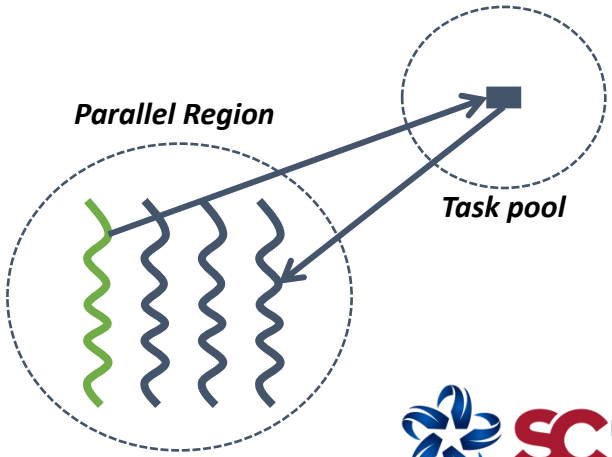
Tasking history: OpenMP 3.0

- A task is a piece of code, its data environment and ICVs

```
int foo(int x) {
  int res = 0;
  #pragma omp task shared(res) firstprivate(x)
  {
    res += x;
  }
 #pragma omp taskwait
  return res;
}
```

```
int main() {
  int var = 0;
  #pragma omp parallel
  #pragma omp master
  var = foo(4);
}
```

- Task's execution may be deferred
  - Liveness of variables
  - Guarantee task completeness



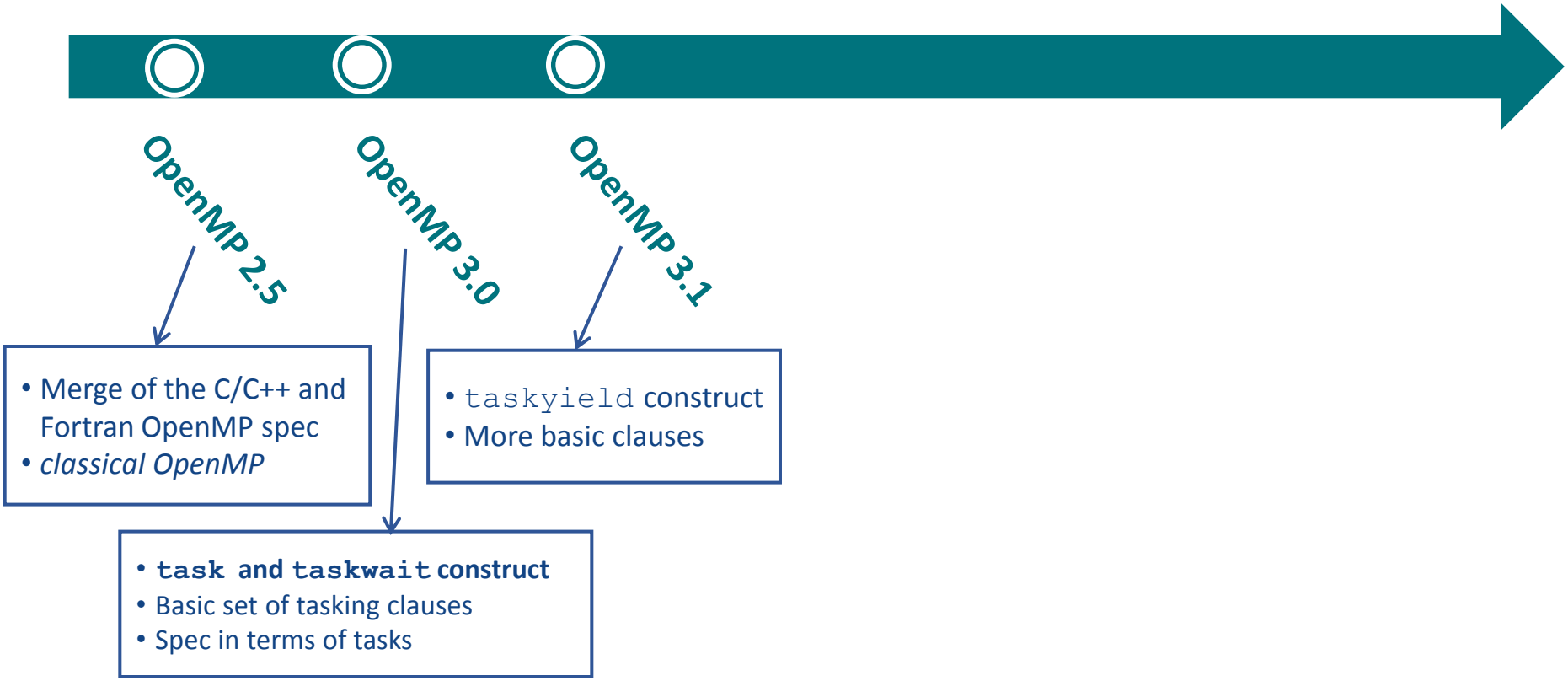
Tasking history: OpenMP 3.0

- `if(expr)` clause
  - "Switch off" mechanism
  - The new task is *undeferred* and executed immediately
  - The encountering task is suspended until the new task is completed
  - Data-sharing clauses are honored!

```
int foo(int x) {
  printf("entering foo function\n");
  int res = 0;
  #pragma omp task shared(res) if(false)
  {
     res += x;
  }
  printf("leaving foo function\n");
}
```

Really useful to debug tasking applications :D

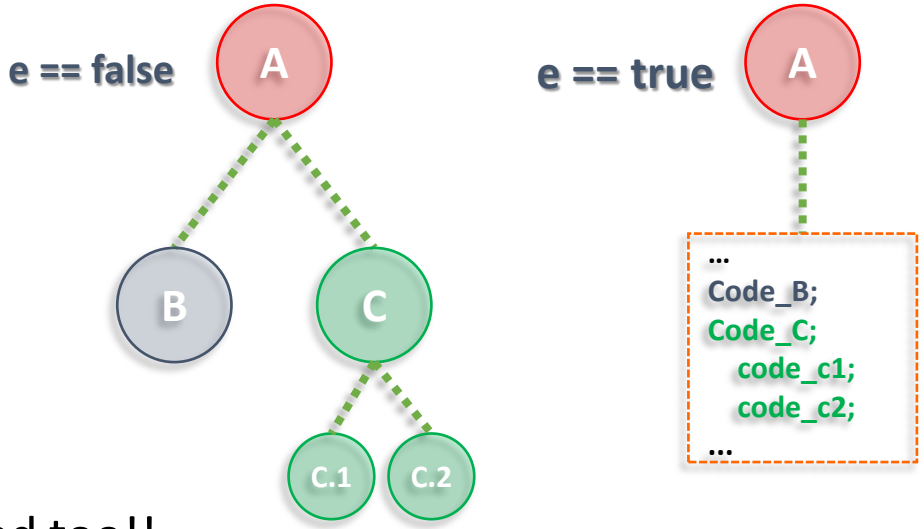
Tasking history: past



Tasking history: OpenMP 3.1

- `final(expr)` clause
  - For recursive & nested applications
  - The new task is created and executed normally but in its context all tasks will be executed immediately by the same thread (*included tasks*)

```
#pragma omp task final(e)
{
  #pragma omp task
  { … }
  #pragma omp task
  { … #C.1; #C.2 … }
  #pragma omp taskwait
}
```

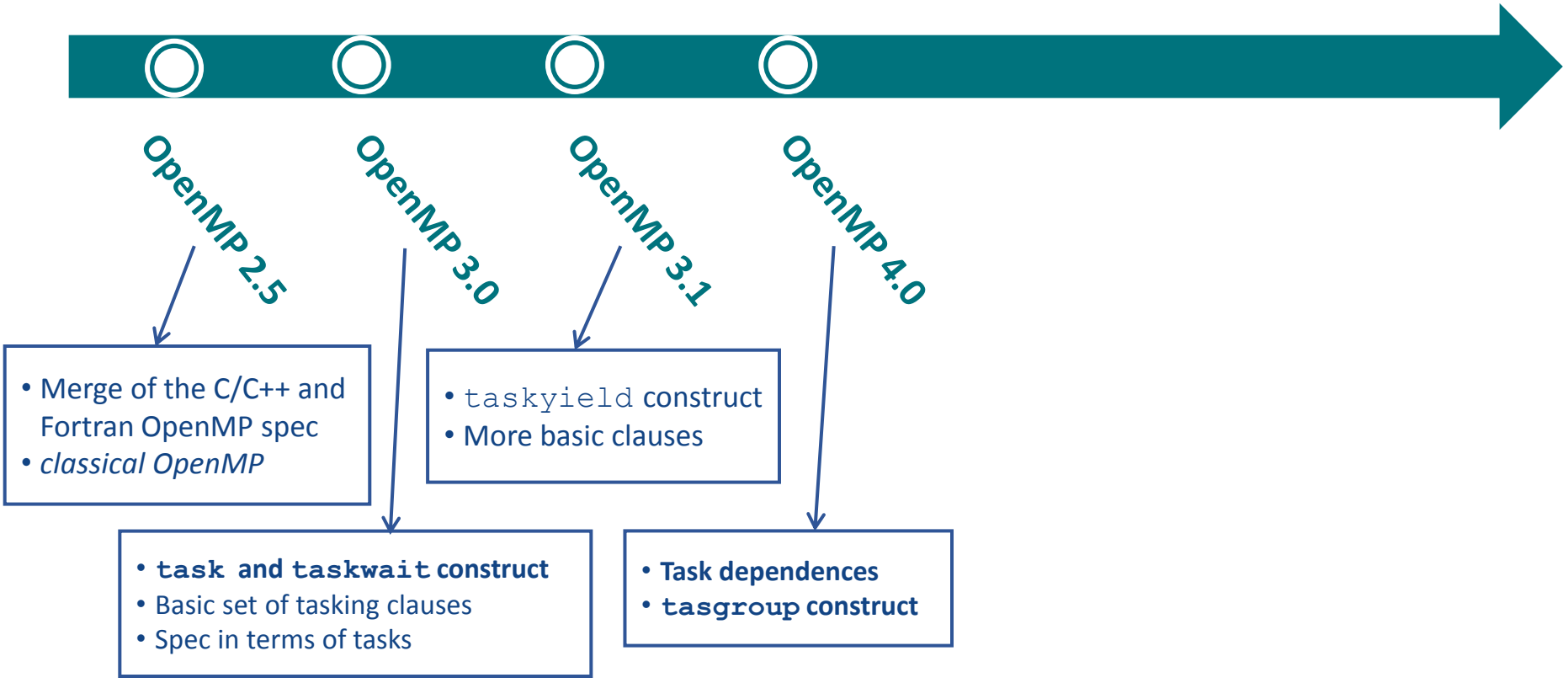


  - Data-sharing clauses are honored too!!

- The `mer` clause
  - Optimiz

Unfortunately, there are no OpenMP commercial implementations taking advantatge of `final` neither `mergeable` =(

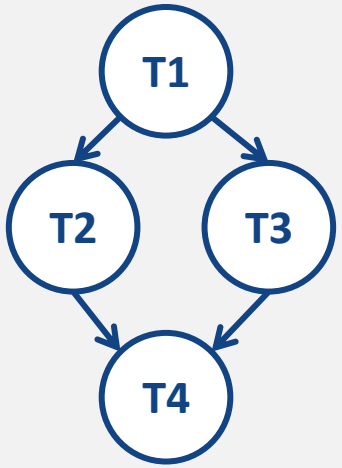
Tasking history: past



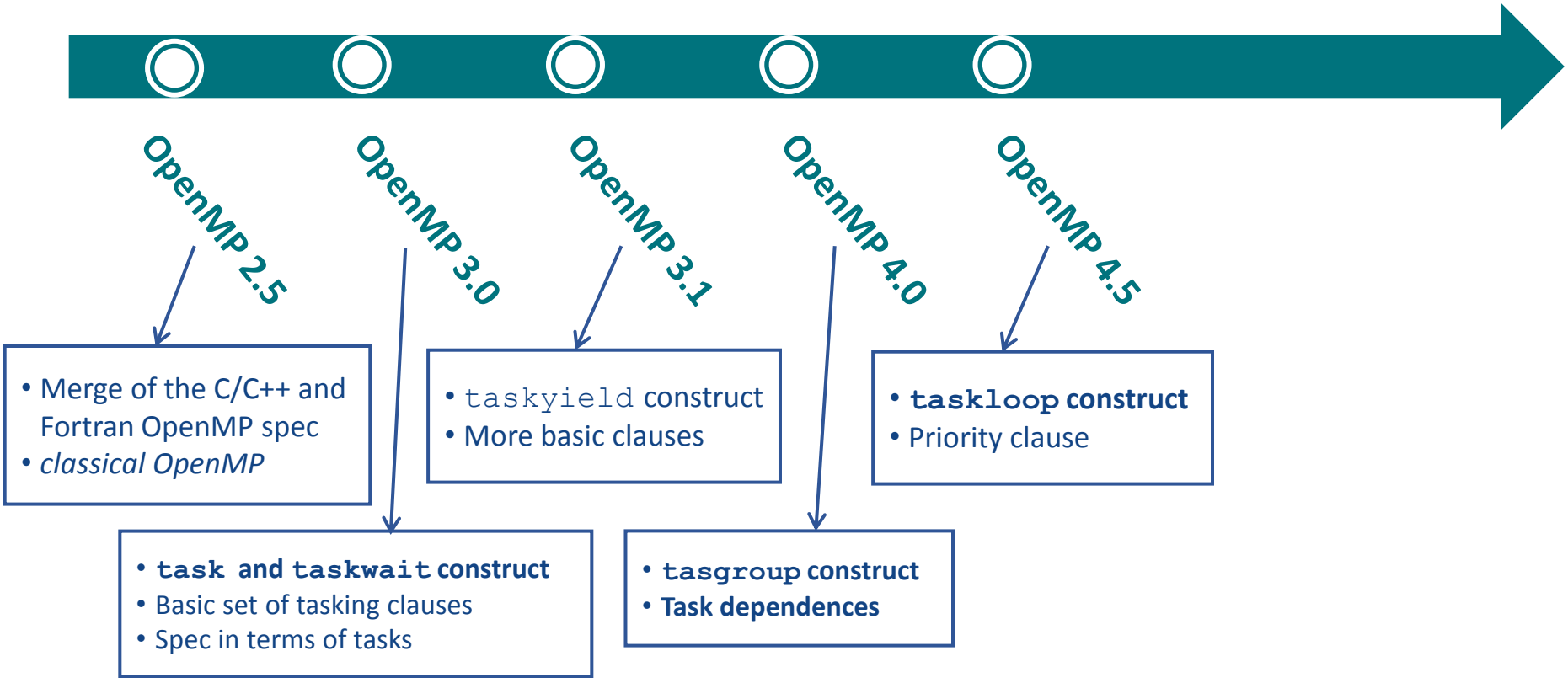
Tasking history: OpenMP 4.0

- `depend(dep-type: list-items)`, where:
  - `dep-type` may be `in`, `out` or `inout`
  - `item` may be:
- A t
cor
- If a
  -
- If a
  -

```
int x = 0;
#pragma omp parallel
#pragma omp single
{
  #pragma omp task depend(inout: x) //T1
  { ... }

  #pragma omp task depend(in: x)    //T2
  { ... }

  #pragma omp task depend(in: x)    //T3
  { ... }

  #pragma omp task depend(inout: x) //T4
  { ... }
}
```



Tasking history: recent past



Tasking history: OpenMP 4.5

- The `taskloop` construct
  - specifies that the iterations of a loop will be executed in parallel using tasks

```
int foo(int n, int *v) {
  #pragma omp parallel for
  for(int i = 0; i < n; ++i)
    compute(v[i]);
}
```

```
int foo(int n, int *v) {
  #pragma omp parallel
  #pragma omp single
  #pragma omp taskloop
  for(int i = 0; i < n; ++i)
    compute(v[i]);
}
```

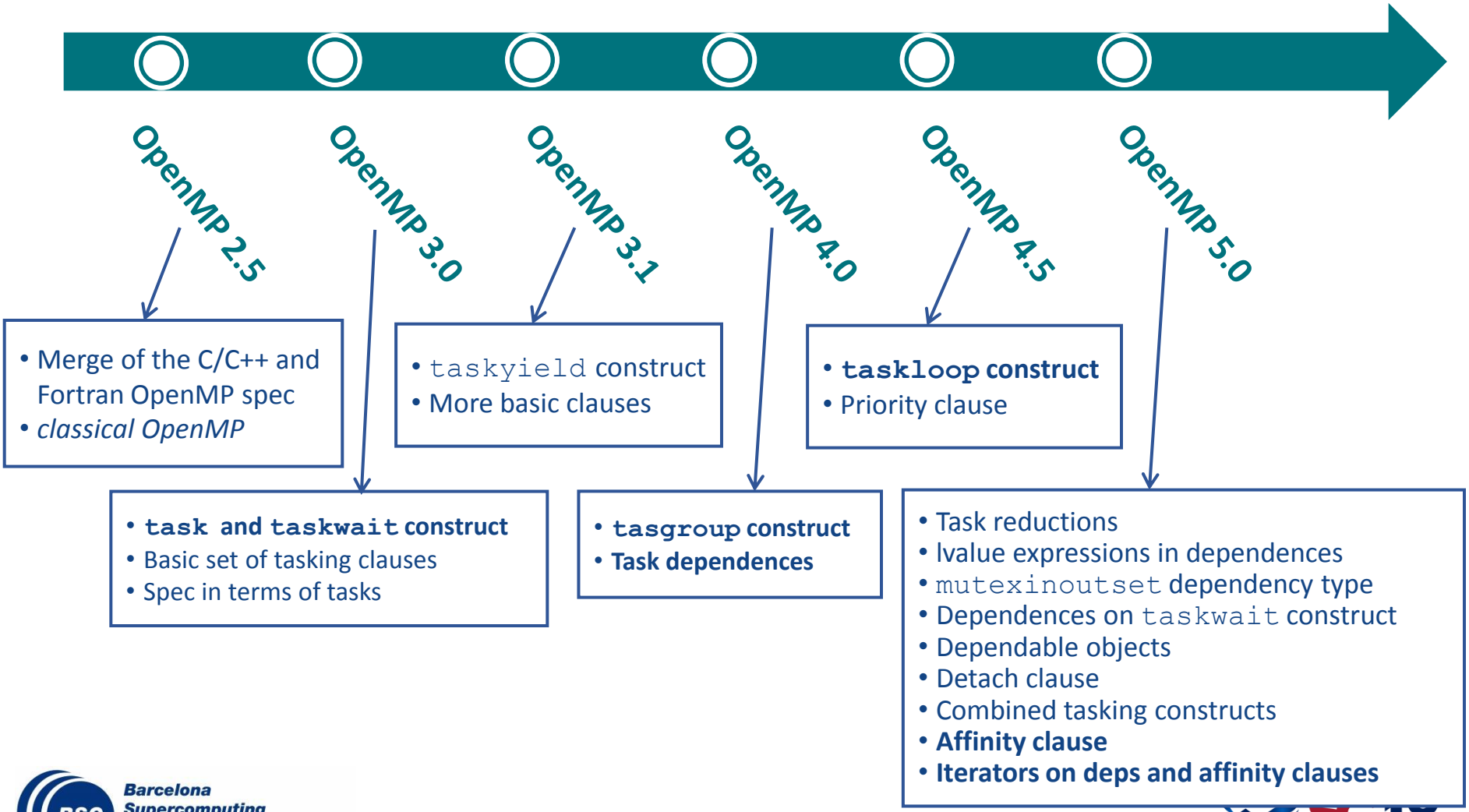
OpenMP 4.5

```
      compute(v[i]);
  }
}
```

OpenMP 3.0

  - Use `num_tasks` or `gra`

Tasking history: present

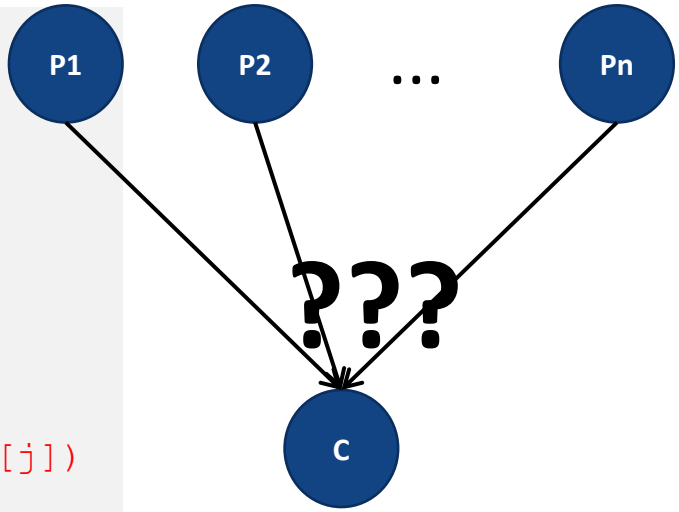


Tasking history: OpenMP 5.0

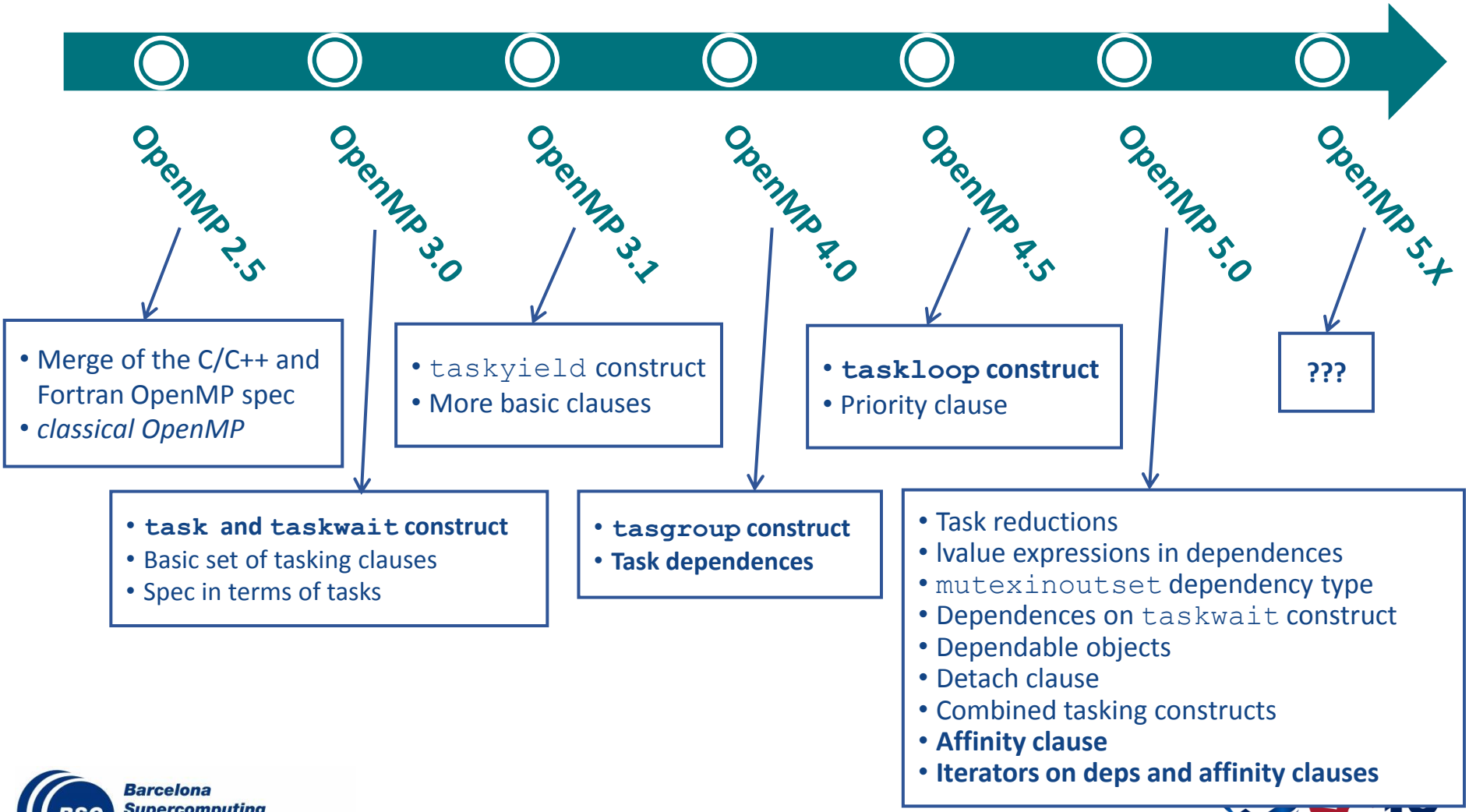
- Affinity clause: hint to the runtime
  - *Try to execute this task as close as possible to some data*

- Iterators on deps (also on the affinity clause):
  - define a dynamic number of dependences

```
std::list<int> l = ...;
int n = l.size();
#pragma omp parallel
#pragma omp single
{
  for (int i = 0; i < n; ++i)
    #pragma omp task depend(out: l[i])
     compute_elem(l[i]);  //Px

  #pragma omp task depend(iterator(j=0:n),in: l[j])
  print_elems(l);          // C
}
```



Tasking history: future?



Tasking history: OpenMP 5.X

- What I would like to see in OpenMP 5.X:
  - A way to express task dependences in a taskloop
  - Idem for task affinity
  - Task-only threads
  - `inoutset` dependency type
  - Better taskloop scheduling policies
  - …
- What would you like to see?

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you!

sergi.mateo@bsc.es

SC18 Dallas, TX | hpc inspires.