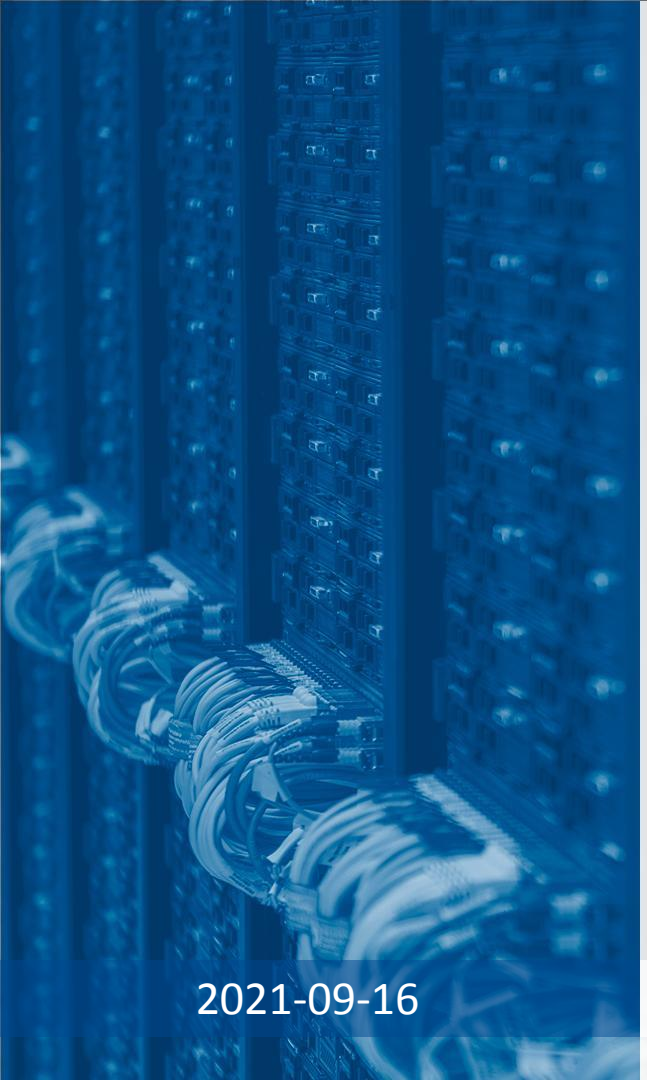


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# An OpenMP Free Agent Threads Implementation

**Victor Lopez**, Joel Criado, Raúl Peñacoba, Roger Ferrer, Xavier Teruel and Marta Garcia-Gasulla

2021-09-16

IWOMP 2021

# Free agent threads background

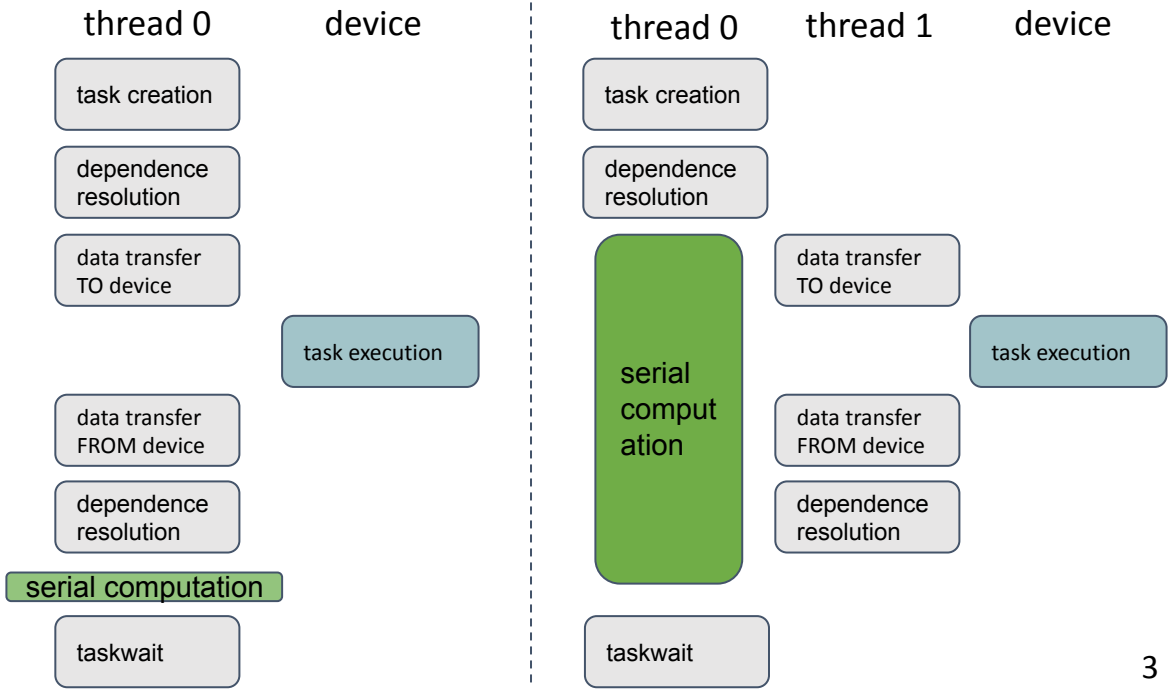
- Hopefully, a new feature in OpenMP 6.0
- Objective: run OpenMP explicit tasks outside of an explicit parallel region
  - Prevent CPU starvation outside of the outermost parallel region
  - Execute tasks / target tasks without a fork-join model

- Bibliography:
  - Sunderland, D., Olivier, S.L., Hollman, D.S., Evans, N., de Supinski, B.R.: Making OpenMP Ready for C++ Executors (2019). https://www.osti.gov/biblio/1559921
  - de Supinski, B.R.: Recent, Current and Future OpenMP Directions: OpenMP 5.1 and More!. https://www.openmp.org/wp-content/uploads/OpenMP_SC20-deSupinski.pdf
  - Tian, S., Doerfert, J., Chapman, B.: Concurrent Execution of Deferred OpenMP Target Tasks with Hidden Helper Threads. https://tianshilei.me/wp-content/uploads/concurrent-lcpc2020.pdf

# Hidden Helper Threads in LLVM OpenMP runtime

- Target tasks are deferred, even tasks encountered in the implicit parallel region
- The deferred target task is executed by a thread that is not visible to the user

```
int main() {
    int val = 0;
    #pragma omp target map(val) nowait
    {
        update_value(&val);
    }

    serial_computation();
    #pragma omp taskwait

    printf("value: %d\n", val);
    return 0;
}
```



# Task based parallelism

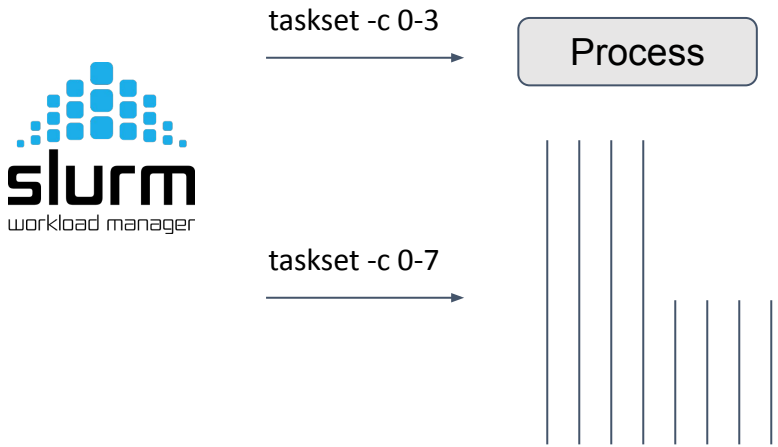
- Free agent threads simplify syntax for task based parallelism

```
void foo() {
    #pragma omp parallel
    {
        #pragma omp single
        {
            #pragma omp taskgroup
            {
                #pragma omp task
                compute_tree(tree);
            }
        } /* implicit barrier */
    }
}
```

```
void foo() {
    #pragma omp taskgroup
    {
        #pragma omp task
        compute_tree(tree);
    }
}
```

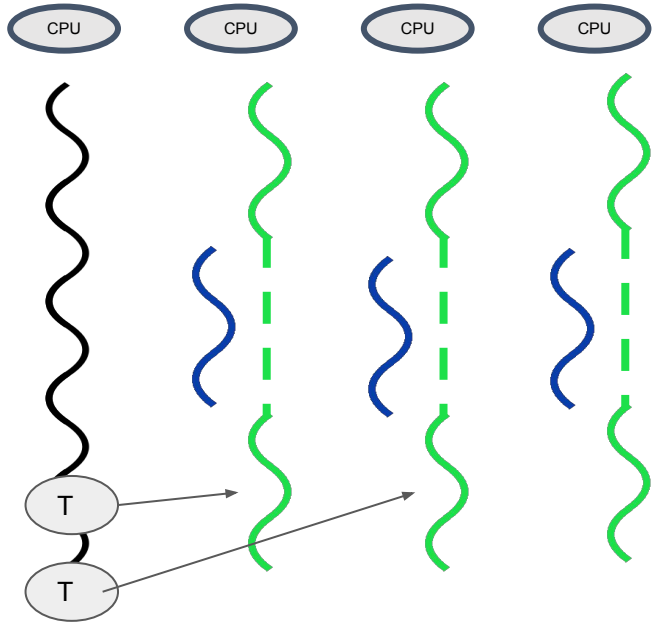
# Extra motivation: Free agents as a malleability tool

- Great opportunity to add malleability to OpenMP programs
  - Free agent threads can be increased/reduced where team size cannot
  - Use free agent threads to perform load balance operations
  - Give OMPT tools the power to manage free agent threads



# Implementation

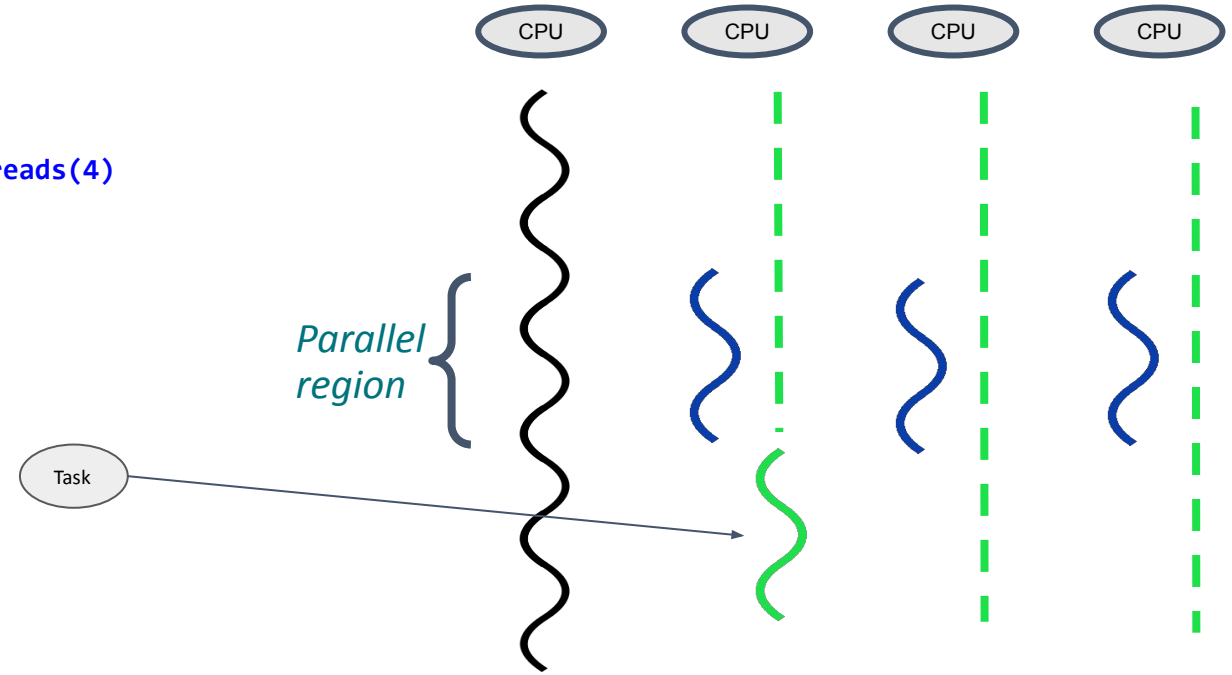
- Not organized in teams
  - Can execute/create explicit tasks in any team
- Not the same OpenMP thread as regular workers
- Part of the initial thread contention group
- Flexible: add threads / enable / disable




**Initial thread**
**Worker thread**
**Free agent thread**

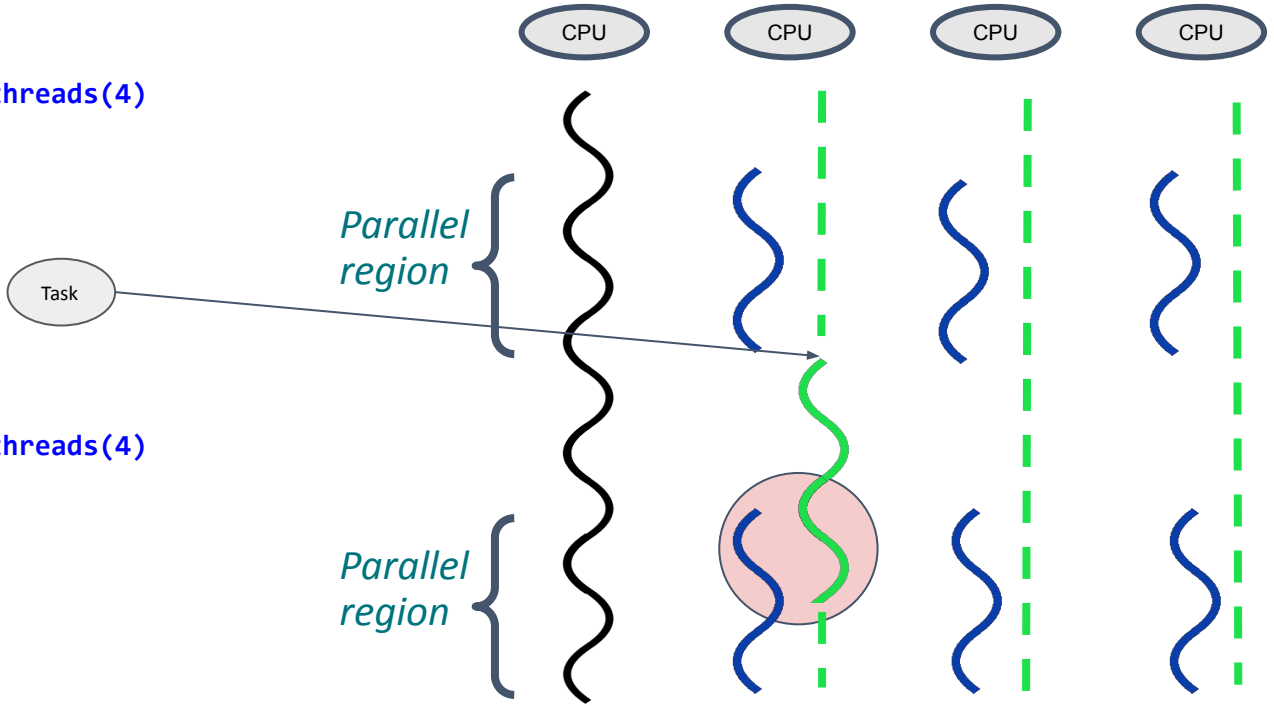
# Example 1: introducing free agent threads

```
void foo()
{
    /* sequential code */

    #pragma omp parallel num_threads(4)
    {
        …
    }

    /* sequential code */

    #pragma omp task
    {
        …
    }
}
```



# Example 2: free agent threads between phases

```
void foo()
{
    #pragma omp parallel num_threads(4)
    {
        …
    }

    #pragma omp task
    {
        …
    }

    #pragma omp parallel num_threads(4)
    {
        …
    }
}
```



# Free agent clause

- Free agent threads should not break existing codes
  - What if a task calls omp_get_thread_num?
  - What if a task uses threadprivate variables?
  - What if a task participates in a reduction?

```
<T> my_buffer[omp_get_max_threads()];
#pragma omp parallel
{
    #pragma omp taskloop
    for (...)
        my_buffer[omp_get_thread_num()] = …
}
```

- Let the programmer decide using a clause/env. var.
  - #pragma omp task **free_agent(bool-expr)**
  - #pragma omp taskloop **free_agent(bool-expr)**
  - export OMP_FREE_AGENT_TASKS={true,false} (if no clause)

# Configuring free agent threads

- Environment variables

| | |
|---|---|
| **OMP_FREE_AGENT_NUM_THREADS** | sets the max initial number of free agent threads to use<br>(num_free_agents = [0-N], they count towards thread-limit) |
| **OMP_FREE_AGENT_PROC_BIND** | equivalent to OMP_PROC_BIND for free agent threads |
| **OMP_FREE_AGENT_PLACES** | equivalent to OMP_PLACES for free agent threads |
| **OMP_FREE_AGENT_WAIT_POLICY** | equivalent to OMP_WAIT_POLICY for free agent threads |
| **OMP_FREE_AGENT_POLICY** | sets the initial policy/state for free agent threads<br>{ enable, disable, … } |
| **OMP_FREE_AGENT_TASKS** | whether all tasks are considered free agent tasks<br>{ true, false } |

# Configuring free agent threads

- Interfaces

| | |
|---|---|
| int **omp_get_num_free_agent_threads**(void) | returns the number of free agent threads |
| void **omp_set_num_free_agent_threads** (int num_threads) | sets the number of free agent threads (enabled or disabled) |
| void **omp_set_free_agent_policy** (omp_free_agent_policy_t policy) | sets the global policy for free agent threads |

| | |
|---|---|
| int **ompt_get_num_free_agent_threads**(void) | returns the number of free agent threads |
| int **ompt_get_free_agent_thread_id**(void) | returns the id of the free agent thread (id=0..n-1) |
| void **ompt_set_free_agent_thread_state**(int free_agent_id, int state) | sets the specific thread state {enabled, disabled} |

# Implementation detail

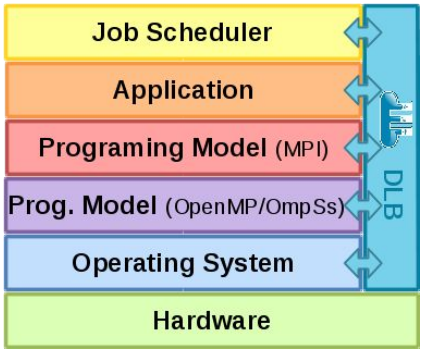
- Based on LLVM 11.0.0
- The runtime creates a second pool of threads at the start of the execution
- Free agent threads may steal tasks from any thread in a parallel region, or suspend their execution if there are no tasks to execute:

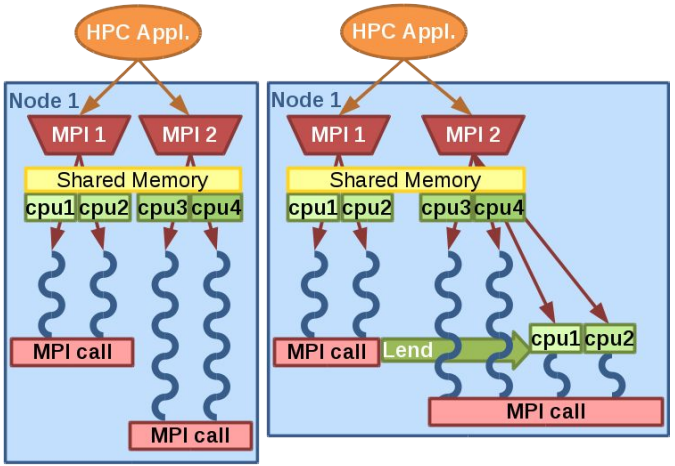
```
while(!done) {
    for (team : allowed_teams)
        for (thread : team)
            steal_task(thread);

    if (wait_policy)
        suspend();
}
```

- The thread that encounters a task must resume free agent threads (gradually)
- Free agent threads can create child tasks, which will be added to the victim thread's queue

# DLB

- It is a dynamic library transparent to the user
- Maximizes the utilization of computational resources
  - Improves the load balance of hybrid applications
  - Manages the number of threads in the shared memory level

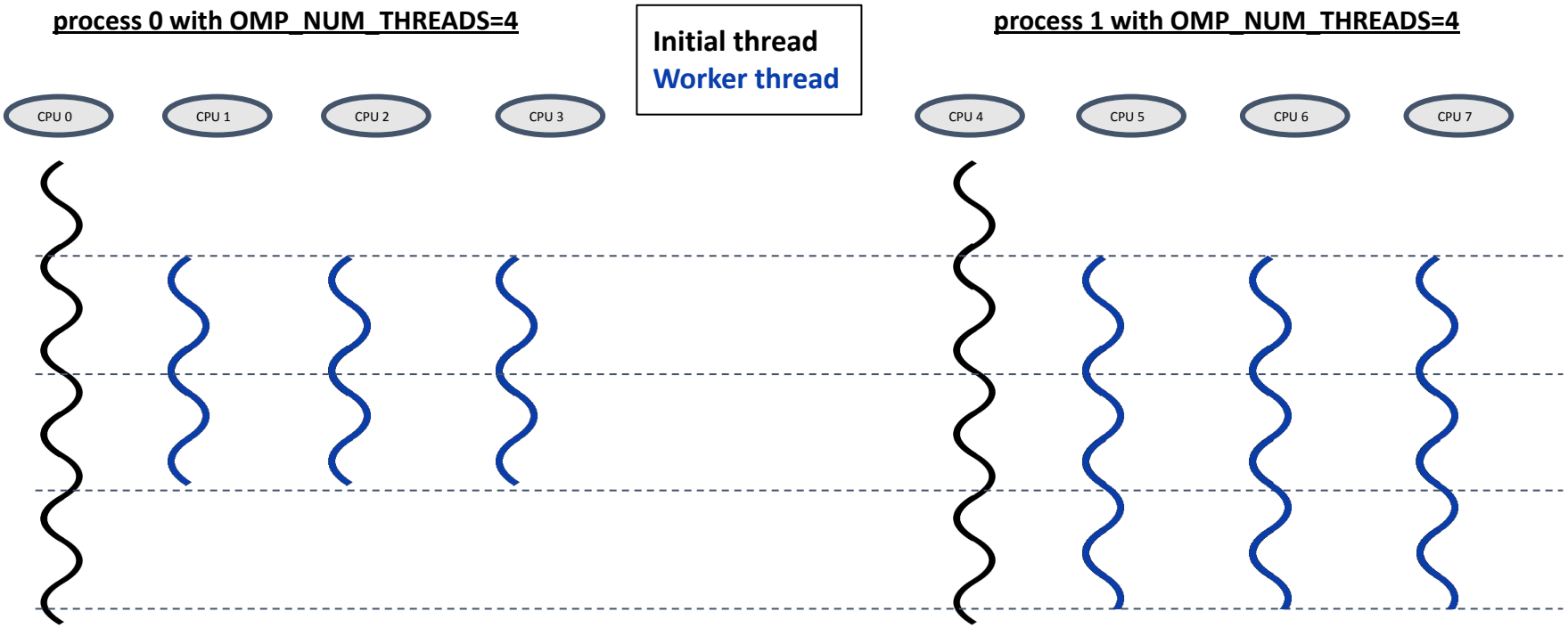


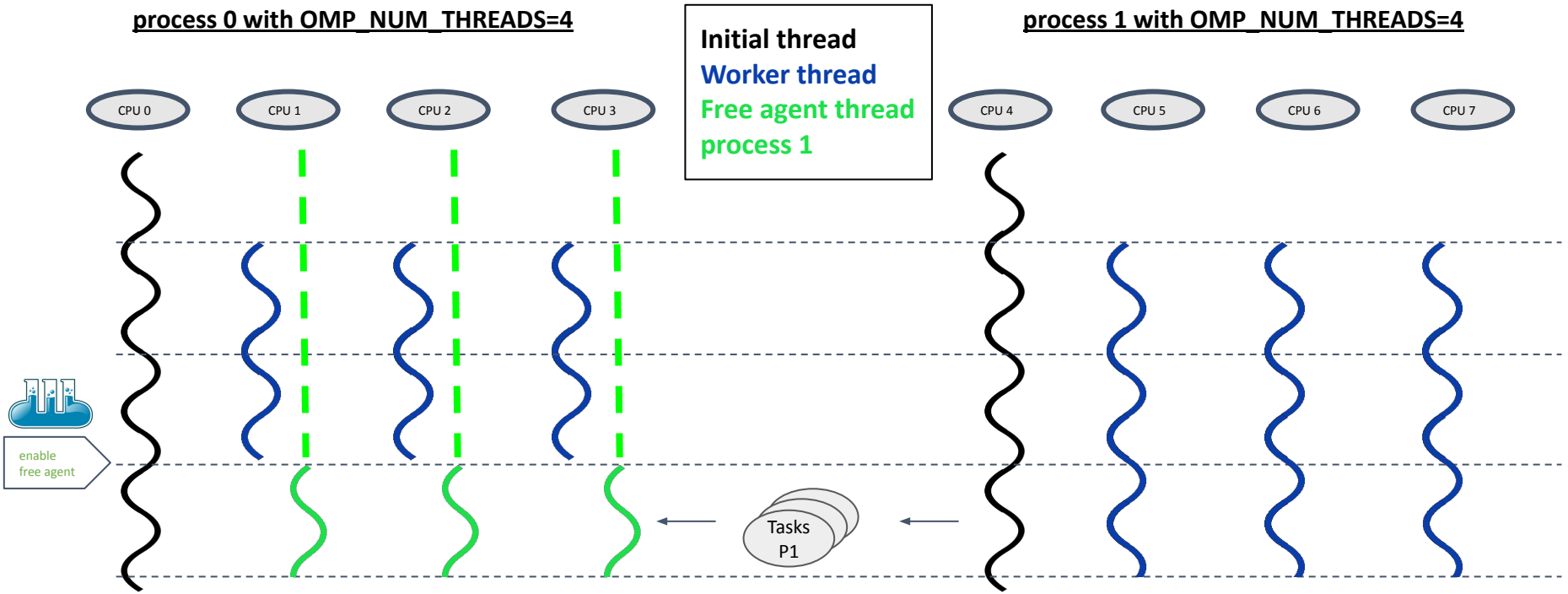

a) Unbalanced MPI application

b) Unbalanced MPI application balanced with DLB

# Example 4: DLB (as OMPT tool), MPI+OpenMP



# Example 4: DLB (as OMPT tool), MPI+OpenMP




An OpenMP Free Agent Threads Implementation

# Evaluation

- Performance evaluation performed on MareNostrum4 at BSC
  - two sockets Intel Xeon Platinum 8160 2.1GHz 24-core and 96GB of main memory
  - Intel compiler 2017.4
  - Modified LLVM 11.0.0
  - DLB 3.0
- DMRG++
  - Density Matrix Renormalization Group developed at ORNL
  - Pure OpenMP (self-balanced)
  - Fixing load imbalance between nested parallel regions
- Alya
  - Computational fluid dynamics (CFD) code optimized for HPC environments developed at BSC
  - Hybrid MPI + OpenMP + DLB
  - Fixing load imbalance between MPI processes

# Evaluation: DMRG++

- Using free agent threads to solve load imbalance between nested parallel regions

- DMRG++
  - Task granularity: 20 us - 20 ms
  - Speedup: 1.36x

```
#pragma omp parallel num_threads(4)
{
    #pragma omp parallel num_threads(4)
    {
        for (...)
            #pragma omp task
            { … }
    }
}
```


16 regular threads + 4 free agent threads on 16 CPUs (same time scale)

# Evaluation: DMRG++

- Using free agent threads to solve load imbalance between nested parallel regions

- DMRG++
  - Task granularity: 20 us - 20 ms
  - Speedup: 1.36x

# Evaluation: Alya

- Using free agent threads to solve load imbalance

- Alya
  - Task granularity: 200 us
  - Load Balance: 0.74
  - Speedup: 1.2x

# Evaluation: Alya

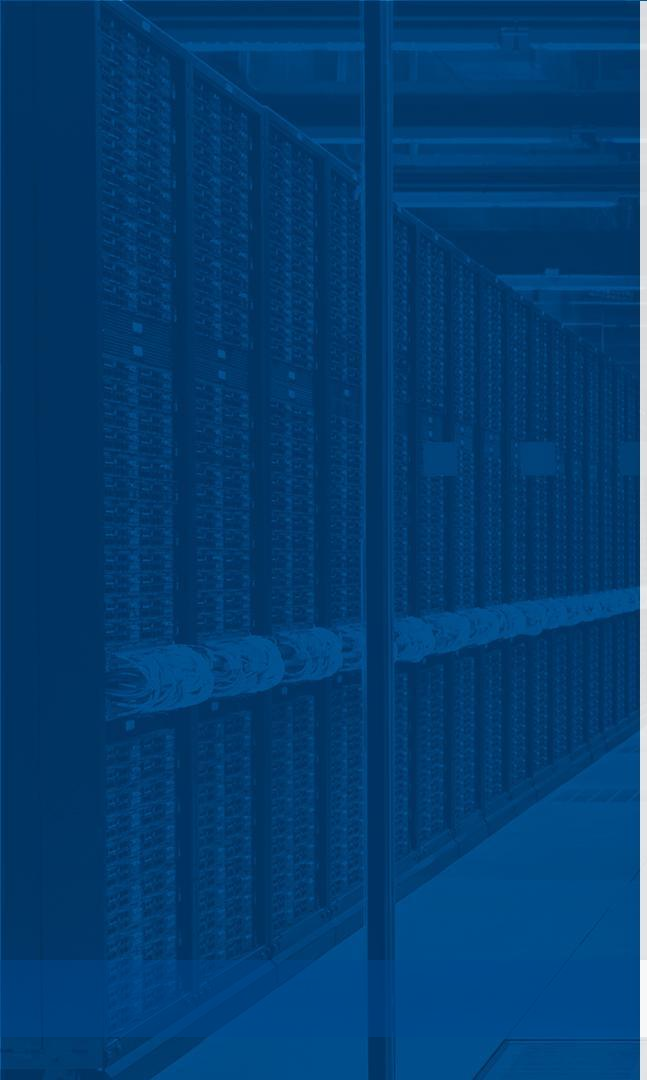
- Using free agent threads to solve load imbalance

- Alya
  - Task granularity: 200 us
  - Load Balance: 0.74
  - Speedup: 1.2x

# Conclusions

- Free agent threads improves the expressivity of the program
- Already demonstrated for target tasks in LLVM by hidden helper threads
- Free agent threads are an opportunity to increase the flexibility to the OpenMP model

- Future work
  - We are starting another design of a free agent threads implementation to overcome some of the difficulties found with this approach
    - Respect OMP_THREAD_LIMIT
    - Avoid thread OS context switch



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# Thank you

victor.lopez@bsc.es