Asynchronous Task Creation for Task-Based Parallel Programming Runtimes

Jaume Bosch (jbosch@bsc.es),
Xubin Tan, Carlos Álvarez, Daniel Jiménez,
Xavier Martorell and Eduard Ayguadé

Barcelona, Sept. 24, 2018

1) Introduction

2) OmpSs@FPGA
3) Implementation and design
4) Conclusion

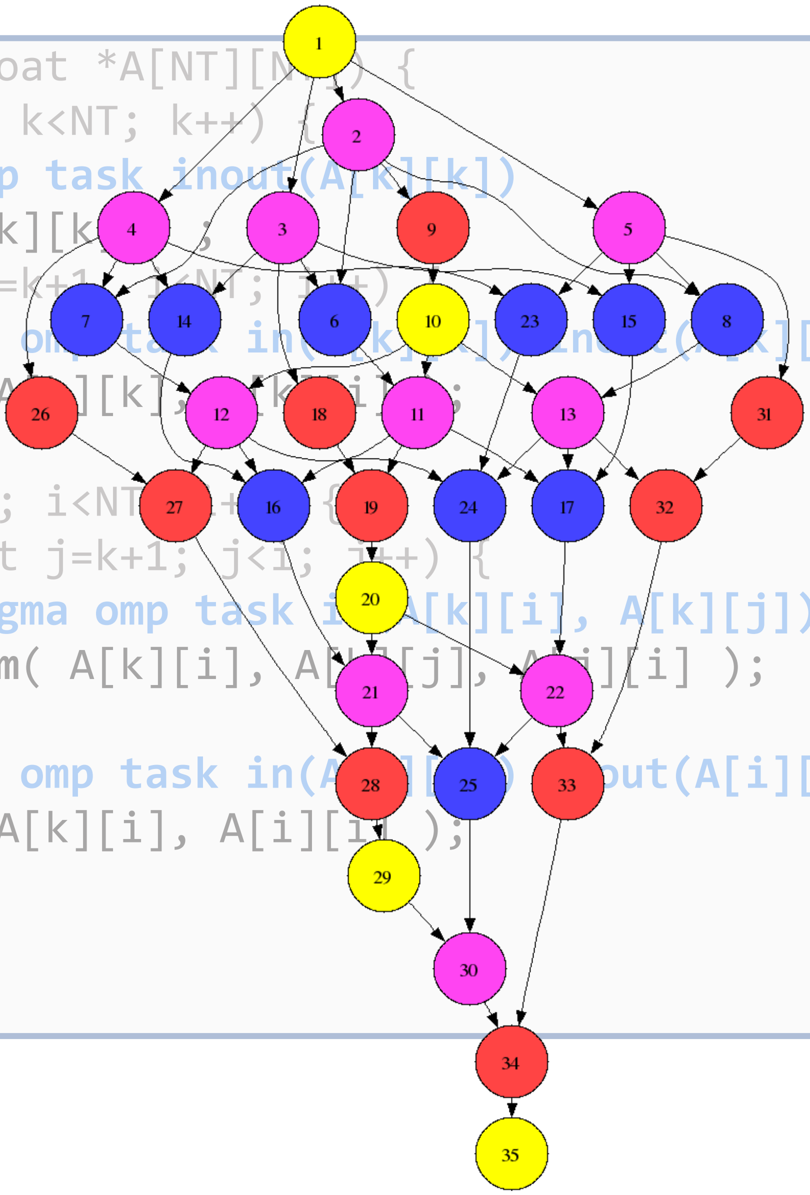
Task-Based Parallel Programming Models

```
void cholesky(float *A[NT][NT]) {
   for (int k=0; k<NT; k++) {
    ● #pragma omp task inout(A[k][k])
      spotrf( A[k][k] ) ;
      for (int i=k+1; i<NT; i++) {
       ● #pragma omp task in(A[k][k]) inout(A[k][i])
         strsm( A[k][k], A[k][i] );
      }
      for (i=k+1; i<NT; i++) {
         for (int j=k+1; j<i; j++) {
          ● #pragma omp task in(A[k][i], A[k][j]) inout(A[j][i])
            sgemm( A[k][i], A[k][j], A[j][i] );
         }
       ● #pragma omp task in(A[k][i]) inout(A[i][i])
         ssyrk( A[k][i], A[i][i] );
      }
   }
}
```

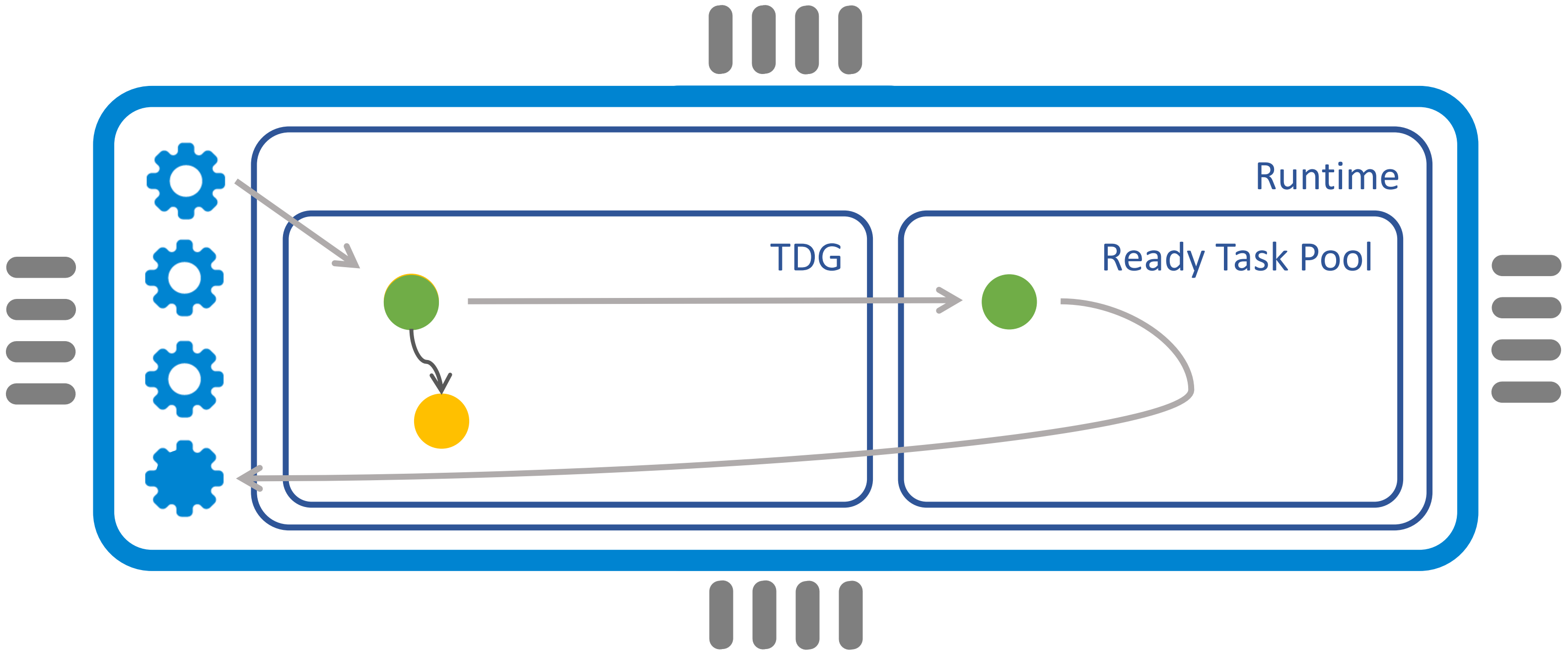
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

Task-Based Parallel Programming Models

```
void cholesky(float *A[NT][NT]) {
   for (int k=0; k<NT; k++) {
      #pragma omp task inout(A[k][k])
      spotrf( A[k][k] );
      for (int i=k+1; i<NT; i++)
         #pragma omp task in(A[k][k]) inout(A[k][i])
         strsm( A[k][k], A[k][i] );
      }
      for (i=k+1; i<NT; i++) {
         for (int j=k+1; j<i; j++) {
            #pragma omp task in(A[k][i], A[k][j]) inout(A[j][i])
            sgemm( A[k][i], A[k][j], A[j][i] );
         }
         #pragma omp task in(A[k][i]) inout(A[i][i])
         ssyrk( A[k][i], A[i][i] );
      }
   }
}
```



Runtime operational flow of a task

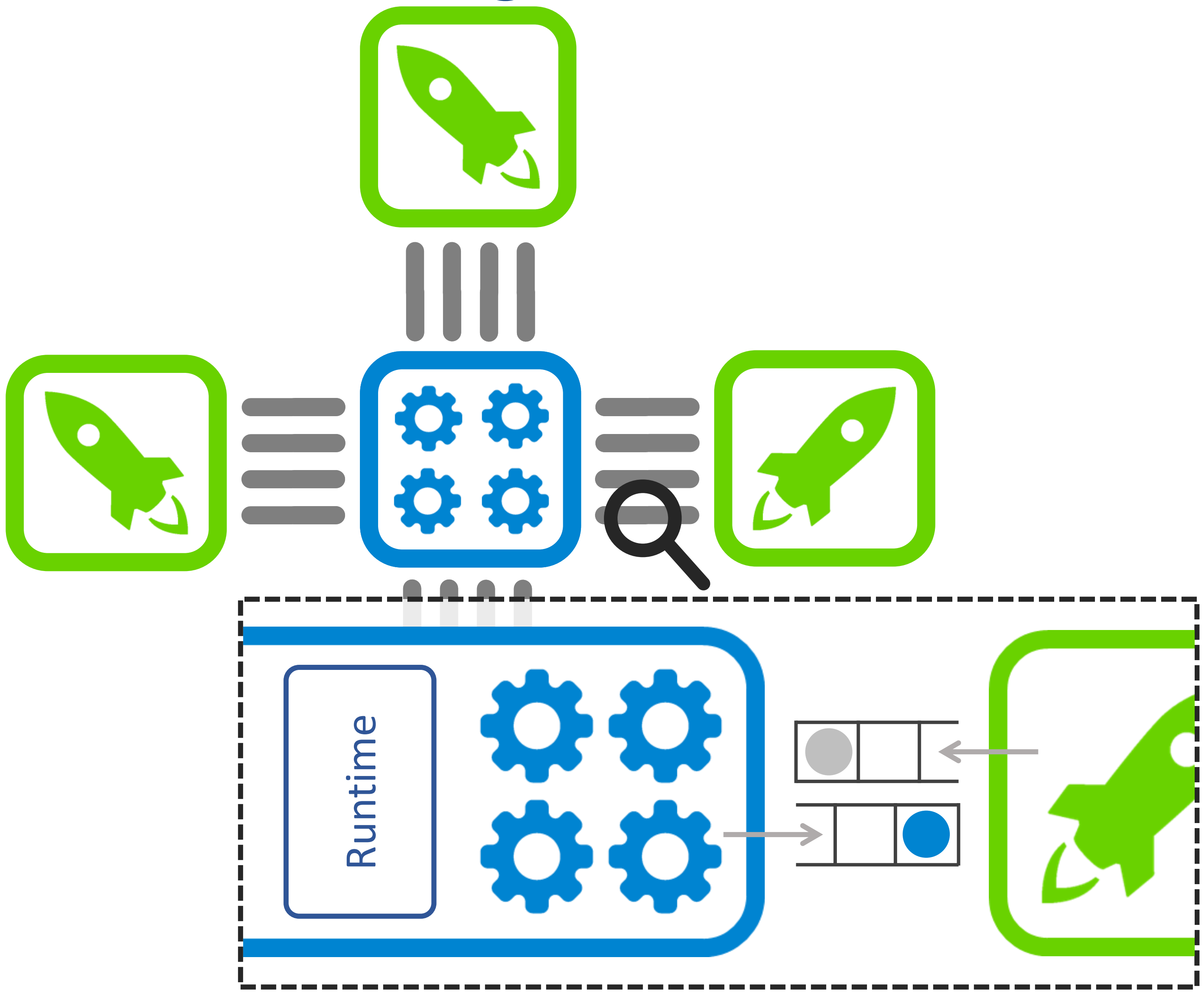


Task states:

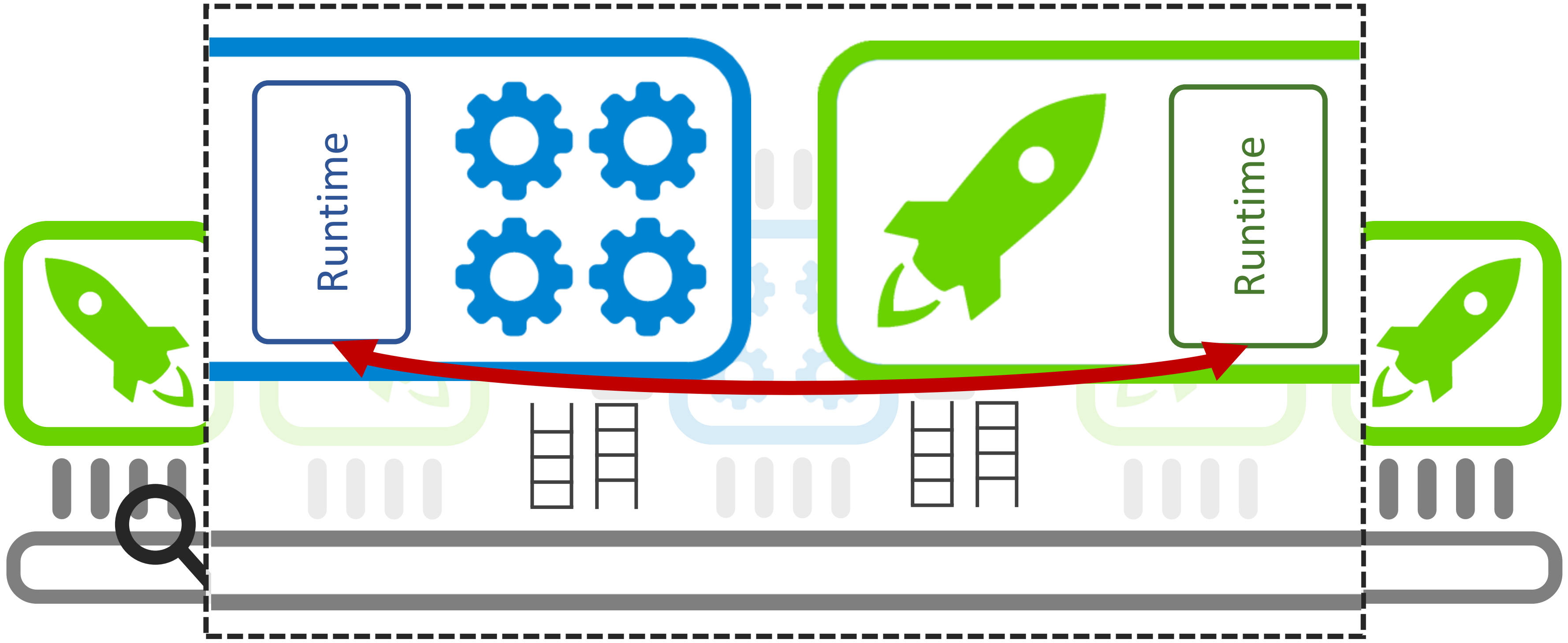
Instantiated → Ready → Active → Completed

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

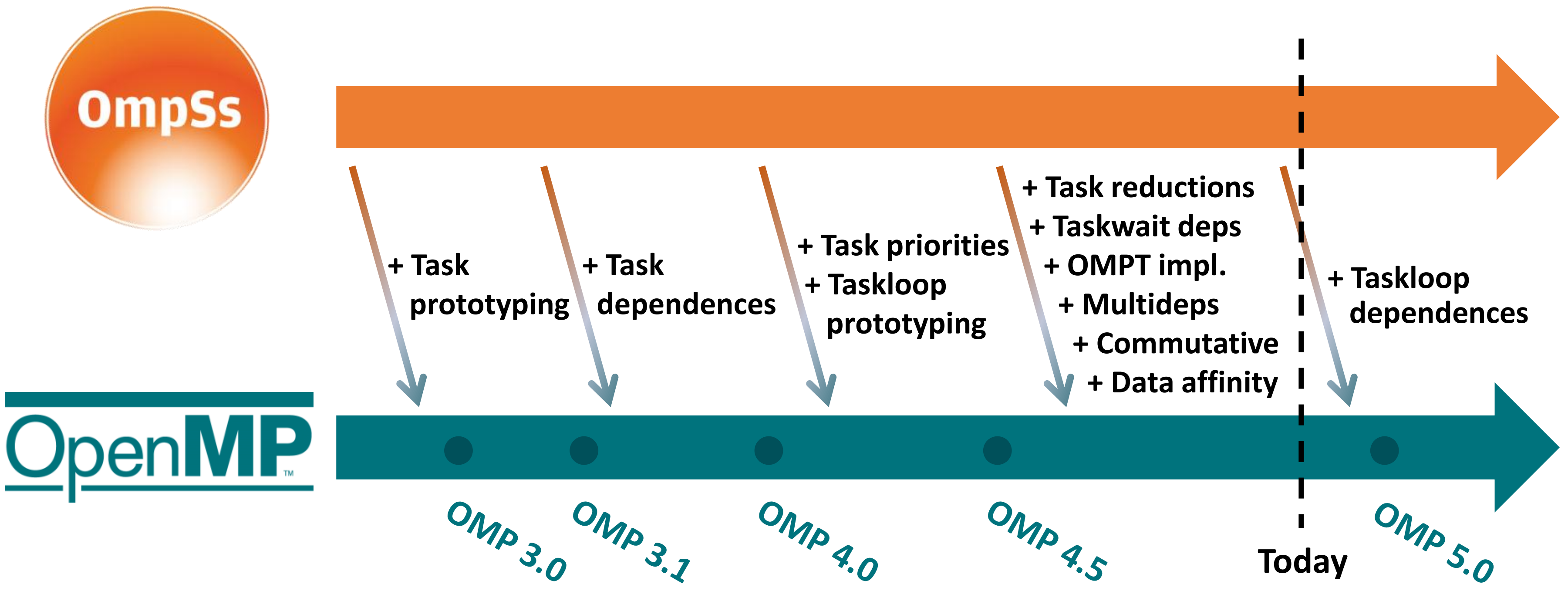
Current target model



Next? target model



OmpSs: Forerunner of the OpenMP tasking



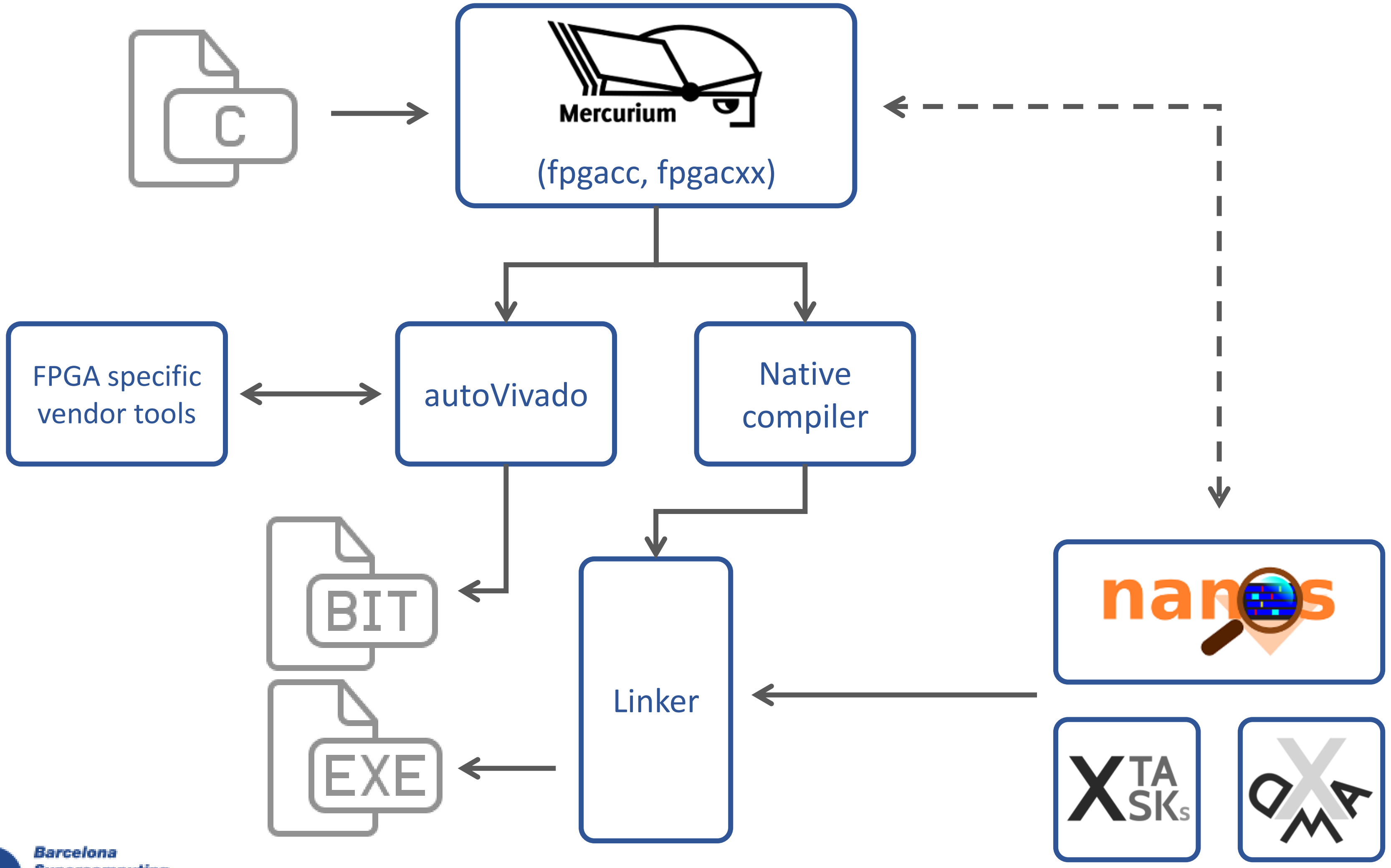
OmpSs@FPGA

- Easily offloading tasks to FPGA devices
  - Automatic generation of FPGA bitstream from C/C++ tasks
  - Automatic data movements and task synchronization between the host and the FPGA
- Support for HW instrumentation integrated in Extrae

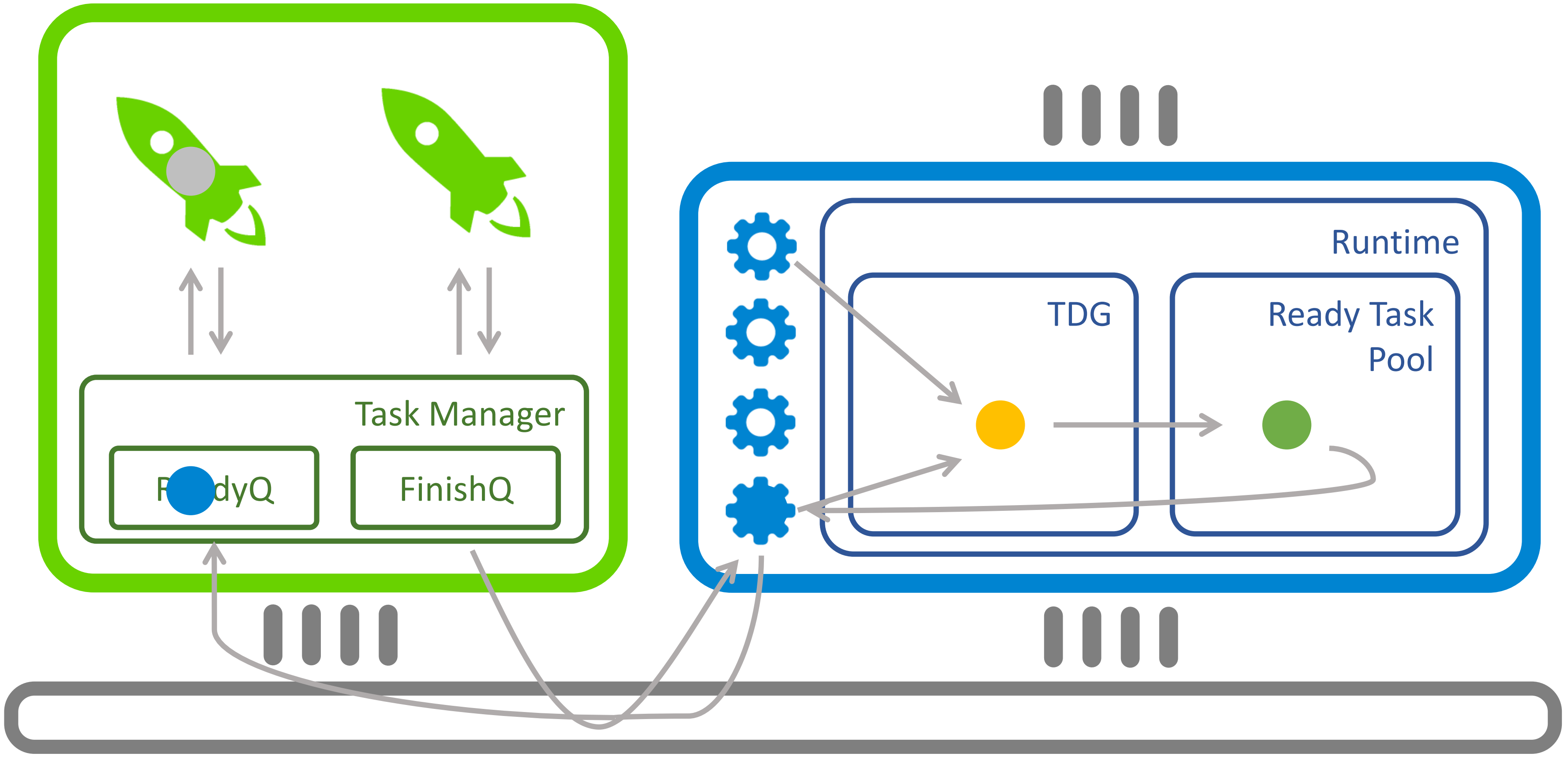
OmpSs@FPGA | Source code example

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task in([BSIZE]v1, [BSIZE]v2) inout([1]result)
void dotProductBlock(float *v1, float *v2, float *result) {
  int resultLocal = result[0];
  for (size_t i = 0; i < BSIZE; ++i) {
    resultLocal += v1[i]*v2[i];
  }
  result[0] = resultLocal;
}

int main() {
  ...
  for (size_t i = 0; i < VSIZE; i += BSIZE) {
    dotProductBlock(v1+i, v2+i, results+i/BSIZE);
  }
  #pragma omp taskwait
  ...
}
```

OmpSs@FPGA | Compilation process

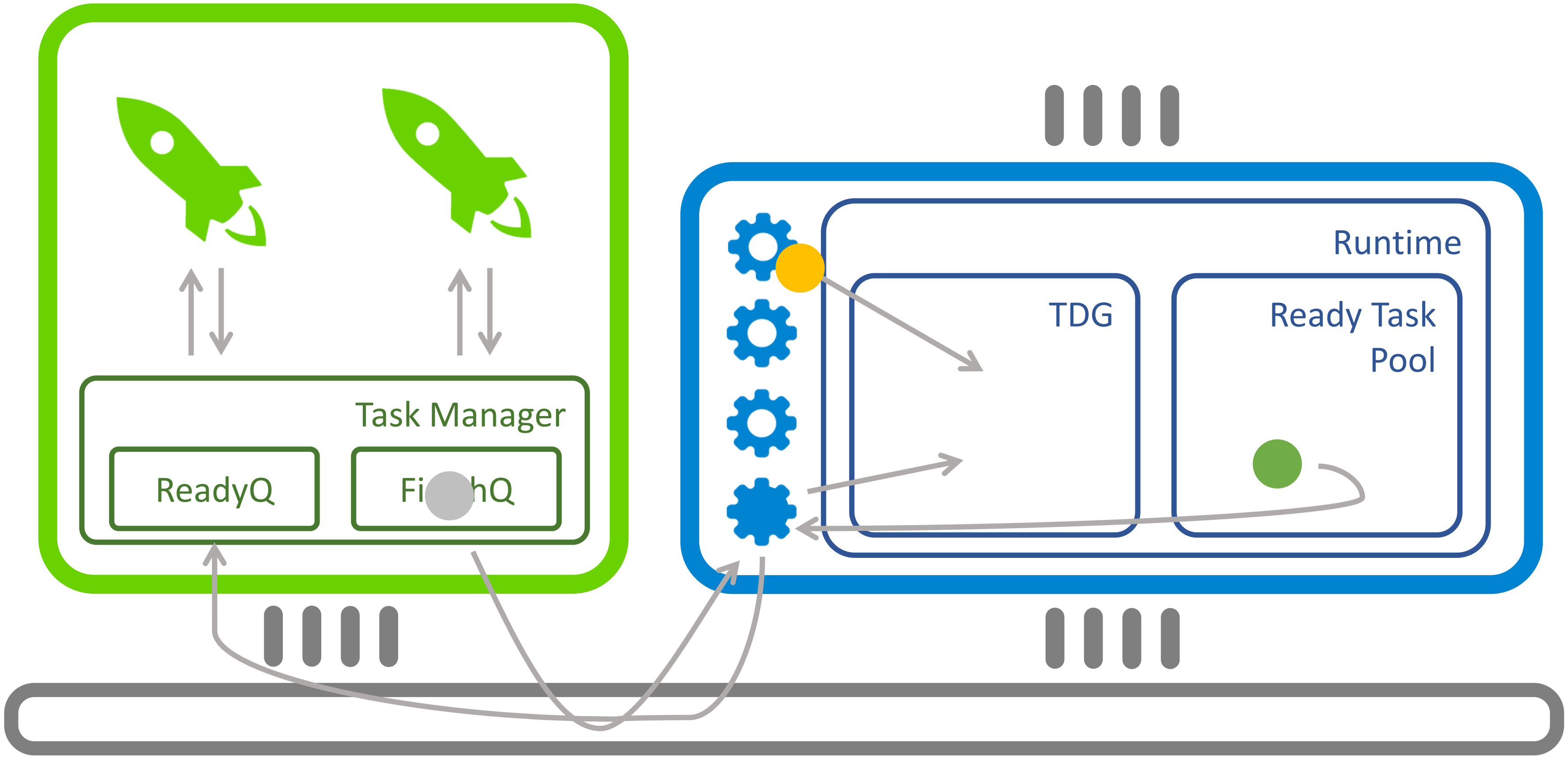


OmpSs@FPGA | Execution



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

OmpSs@FPGA | Execution



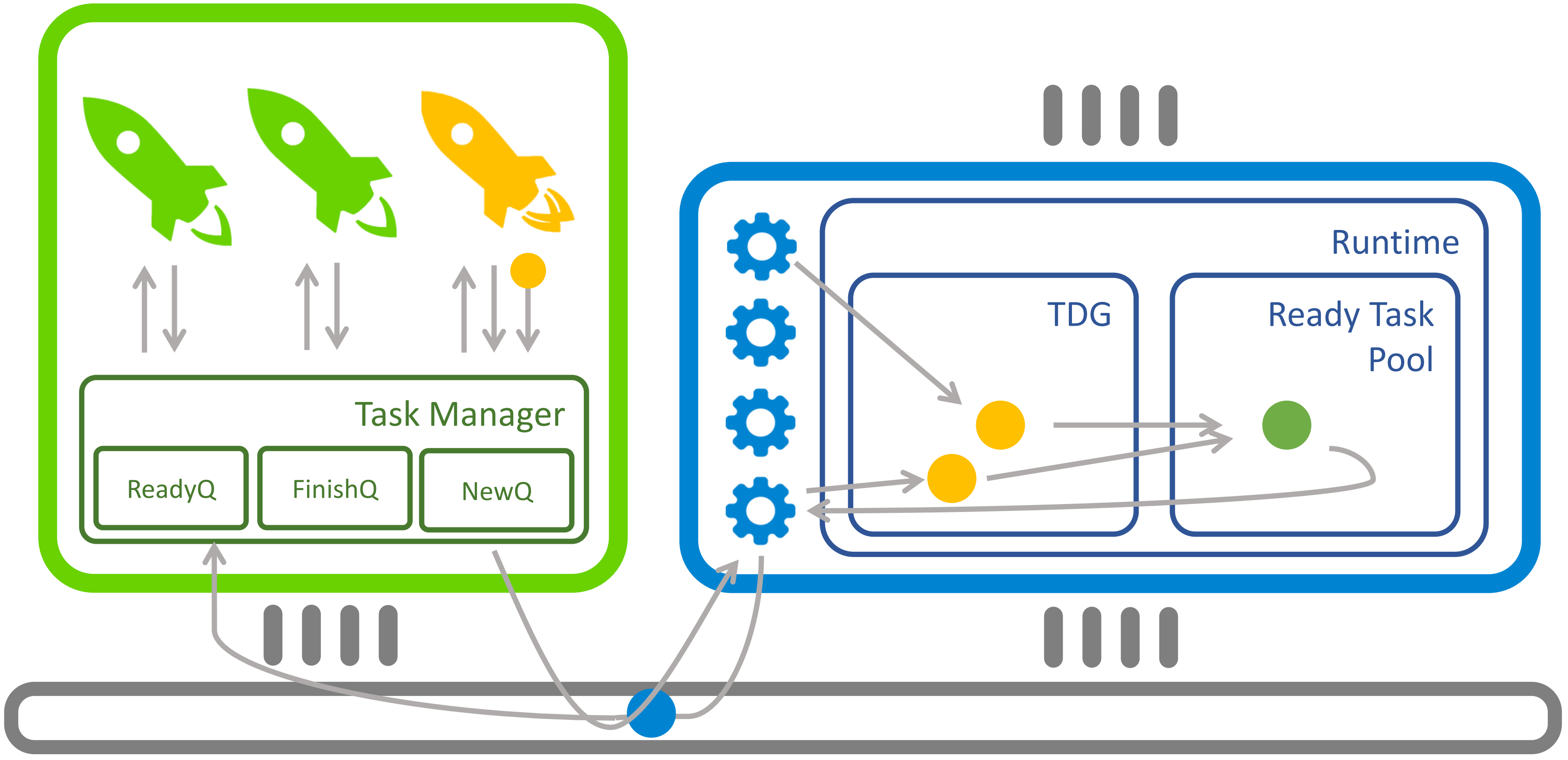
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

Task creation inside a fpga task

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task in([BSIZE]v1, [BSIZE]v2) inout([1]result)
void dotProductBlock(float *v1, float *v2, float *result) {
  ...
}

#pragma omp target device(fpga)
#pragma omp task in([VSIZE]v1, [VSIZE]v2) inout([1]result)
void dotProduct(float *v1, float *v2, float *result) {

  ...
  for (size_t i = 0; i < VSIZE; i += BSIZE) {
    dotProductBlock(v1+i, v2+i, results+i/BSIZE);
  }

  ...
}

int main() {
  ...
  dotProduct(v1, v2, &result);
  ...
}
```

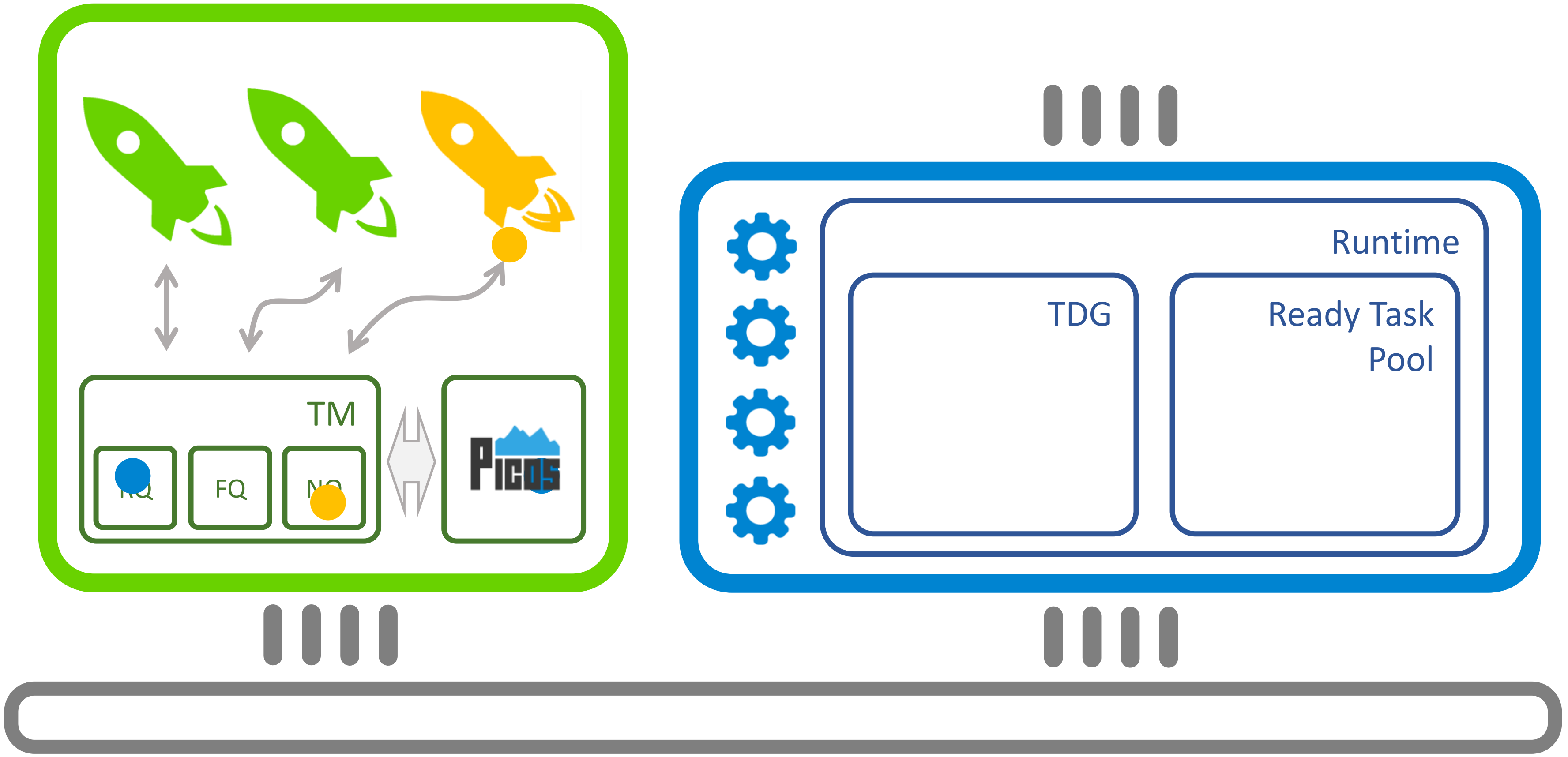
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Task creation from FPGA



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Task creation from FPGA with Picos



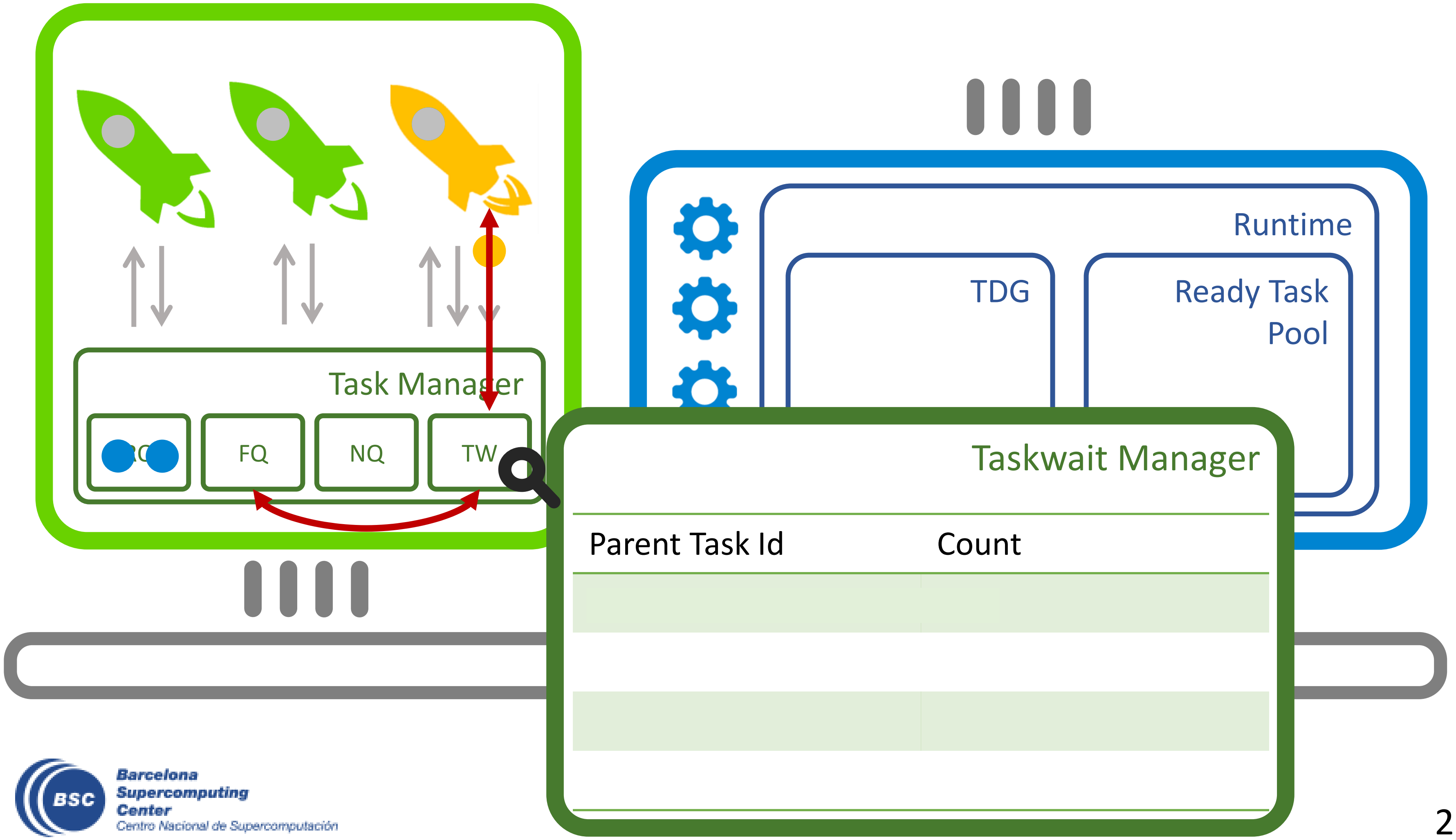
Asynchronous task creation

- Shared memory region between the accelerator and the handler
  - Coherent and consistent
- Tasks must be added and handled in the same order
  - Ensure sequential order execution
  - Handling must cannot be parallelized for the same parent task
- Shared memory can be splited into sub-regions, one for each accelerator

Taskwait inside a fpga task

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task in([BSIZE]v1, [BSIZE]v2) inout([1]result)
void dotProductBlock(float *v1, float *v2, float *result) {
  ...
}

#pragma omp target device(fpga)
#pragma omp task in([VSIZE]v1, [VSIZE]v2) inout([1]result)
void dotProduct(float *v1, float *v2, float *result) {
  for (size_t i = 0; i < VSIZE; i += BSIZE) {
    dotProductBlock(v1+i, v2+i, results+i/BSIZE);
  }
  #pragma omp taskwait
}

int main() {
  ...
  dotProduct(v1, v2, &result);
  ...
}
```

Taskwait inside a fpga task



Parent Task Id	Count

Asynchronous taskwait

- Counter of tasks executed in each context (parent task id)
  - Children of SMP tasks not considered, do not have a parent fpga task id
- Accelerator blocks until receives a signal from the Taskwait Manager
  - The signal is sent when the count becomes 0
- The entry for a context is deleted when the task finalizes
  - Garbage collector

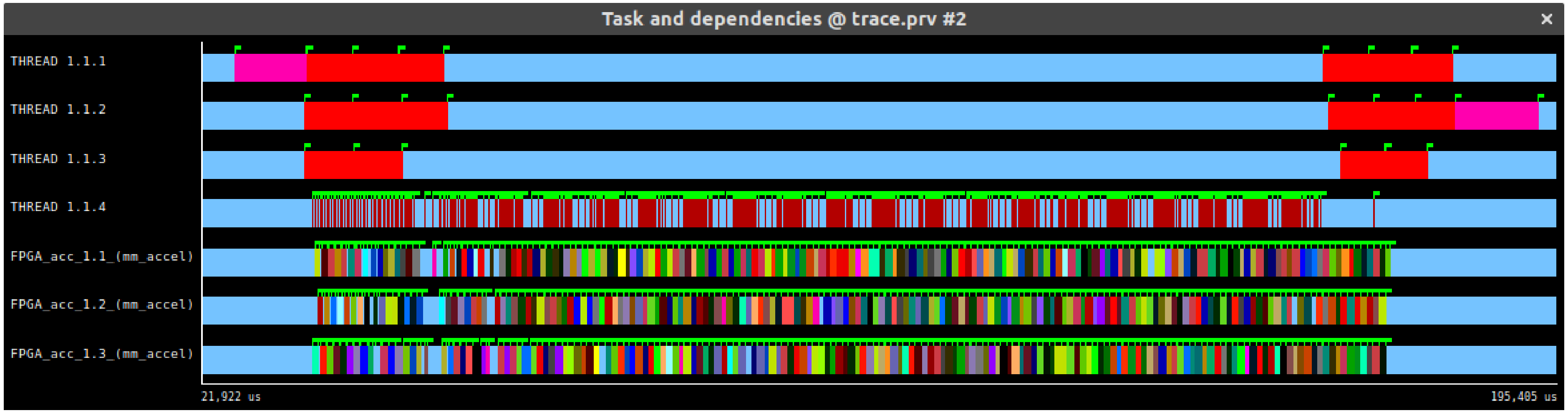
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Execution trace

Matrix Multiply 1024*1024, 3 accelerators at 333Mhz 128*128



- With current implementation, we can save around 50% the time of 1 thread
- With the Picos implementation, we can save an additional thread (fpga helper thread)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Conclusions

- Design of a general mechanism for asynchronous task creation
  - Suitable for accelerators
- Design successfully implemented over OmpSs@FPGA on an Axiom Board
  - Similar performance to SMP task creation version despite the round trip
- Next steps
  - Integration with Picos HW manager
  - Performance evaluation for iterative solvers

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

THANK YOU!

FOR MORE INFO:

pm.bsc.es/ompss-at-fpga

ompss-fpga-support@bsc.es

www.bsc.es

OpenMPCon