Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Breaking the master-slave model in heterogeneous computing

OmpSs@FPGA team

Barcelona, 2020/09/10

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

1) OmpSs@FPGA

2) Distributed runtime
3) Evaluation
4) Conclusion

Task-Based Parallel Programming Models

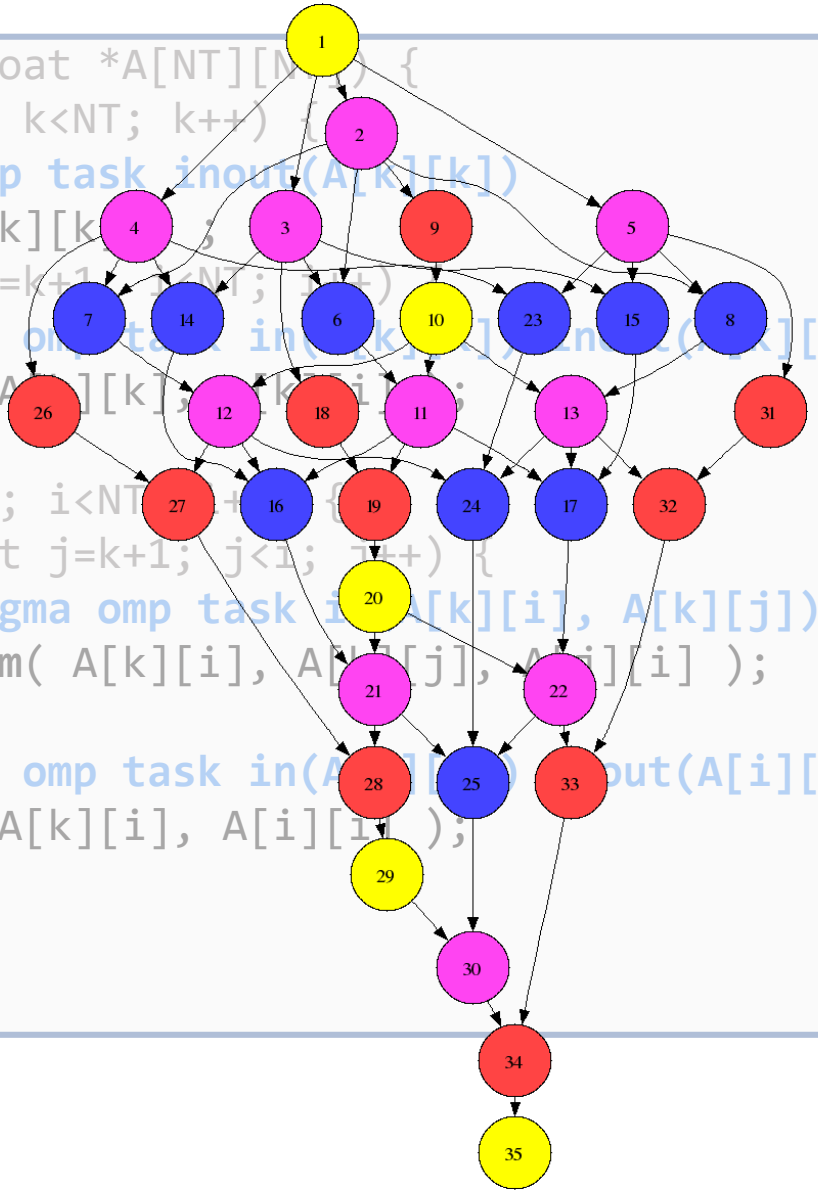
```
void cholesky(float *A[NT][NT]) {
   for (int k=0; k<NT; k++) {

      spotrf( A[k][k] ) ;
      for (int i=k+1; i<NT; i++) {

         strsm( A[k][k], A[k][i] );
      }
      for (i=k+1; i<NT; i++) {
         for (int j=k+1; j<i; j++) {

            sgemm( A[k][i], A[k][j], A[j][i] );
         }

         ssyrk( A[k][i], A[i][i] );
      }
   }
}
```

Task-Based Parallel Programming Models

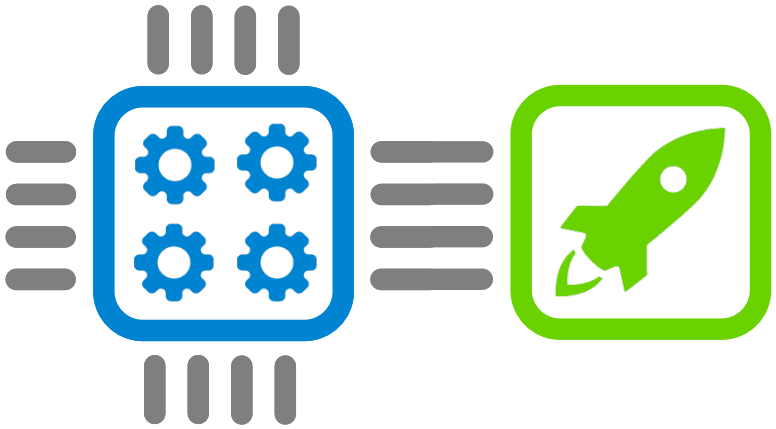
```
void cholesky(float *A[NT][NT]) {
   for (int k=0; k<NT; k++) {
      #pragma omp task inout(A[k][k])
      spotrf( A[k][k] ) ;
      for (int i=k+1; i<NT; i++) {
         #pragma omp task in(A[k][k]) inout(A[k][i])
         strsm( A[k][k], A[k][i] );
      }
      for (i=k+1; i<NT; i++) {
     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

Task-Based Parallel Programming Models

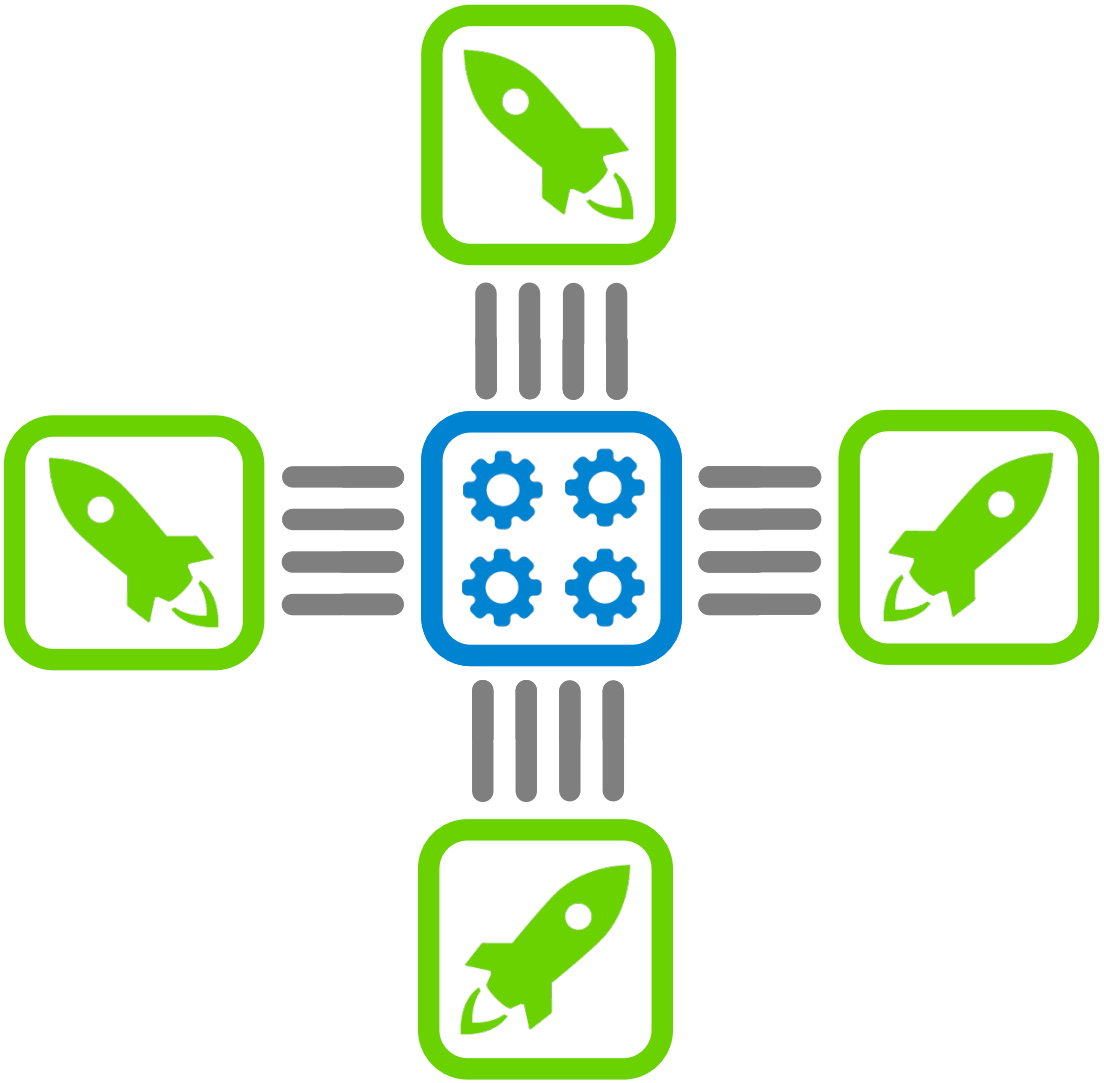
```
void cholesky(float *A[NT][NT]) {
   for (int k=0; k<NT; k++) {
      #pragma omp task inout(A[k][k])
      spotrf( A[k][k] );
      for (int i=k+1; i<NT; i++)
         #pragma omp task in(A[k][k]) inout(A[k][i])
         strsm( A[k][k], A[k][i] );
      }
      for (i=k+1; i<NT; i++) {
        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



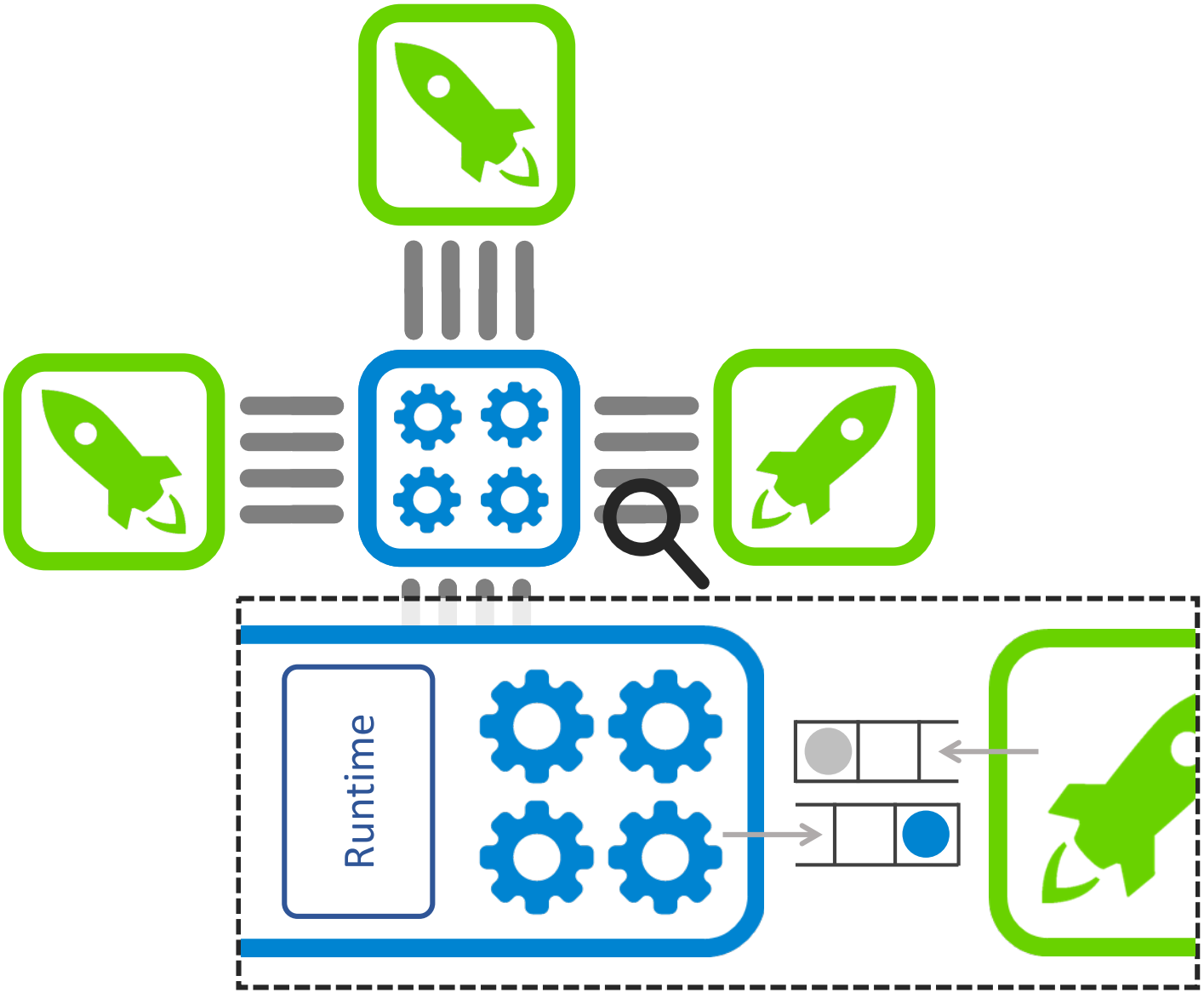
Master-slave target model



Master-slave target model



Master-slave target model



“Classic” 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

```
void update_fpga(int *a, int val, size_t BS) {
  for (size_t i=0; i<BS; ++i) a[i] += val;
}


void update_blocked(int *a, int val, size_t LEN, size_t BS) {
  for (size_t i=0; i<LEN; i+=BS)
    update_fpga(a+i, val, BS);

}
int main(...) {
  int *a = (int *)malloc(NUM_ELEMENTS*sizeof(int));
  update_blocked(a, 2019, NUM_ELEMENTS, NUM_ELEMENTS_BLOCK);
  #pragma omp taskwait
}
```

OmpSs@FPGA | Source code example

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task copy_inout([BS]a)
void update_fpga(int *a, int val, size_t BS) {
  for (size_t i=0; i<BS; ++i) a[i] += val;
}


void update_blocked(int *a, int val, size_t LEN, size_t BS) {
  for (size_t i=0; i<LEN; i+=BS)
    update_fpga(a+i, val, BS);

}
int main(...) {
  int *a = (int *)malloc(NUM_ELEMENTS*sizeof(int));
  update_blocked(a, 2019, NUM_ELEMENTS, NUM_ELEMENTS_BLOCK);
  #pragma omp taskwait
}
```

OmpSs@FPGA | Source code example

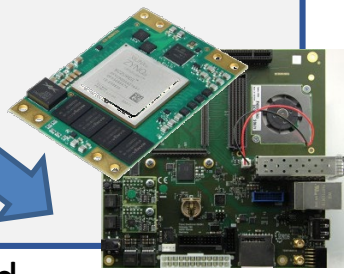
```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task copy_inout([BS]a)
void update_fpga(int *a, int val, size_t BS) {
  for (size_t i=0; i<BS; ++i) a[i] += val;
}


void update_blocked(int *a, int val, size_t LEN, size_t BS) {
  for (size_t i=0; i<LEN; i+=BS)
    update_fpga(a+i, val, BS);

}
int main(...) {
  int *a = (int *)malloc(NUM_ELEMENTS*sizeof(int));
  update_blocked(a, 2019, NUM_ELEMENTS, NUM_ELEMENTS_BLOCK);
  #pragma omp taskwait
}
```

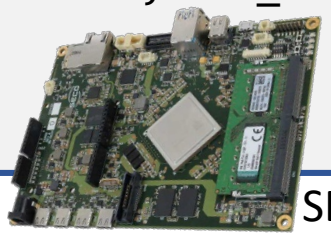
Alveo & Cloud WiP

AlphaData 7v3

Xilinx ZCU 102
XCZU9EG-2FFVB1156

Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1





SECO AXIOM Board
Zynq U+ XCZU9EG-ES2
4x Cortex-A53 cores + FPGA (64-bit platforms)

Zynq-7000 Family
2x Cortex-A9 cores + FPGA (32-bit platforms)

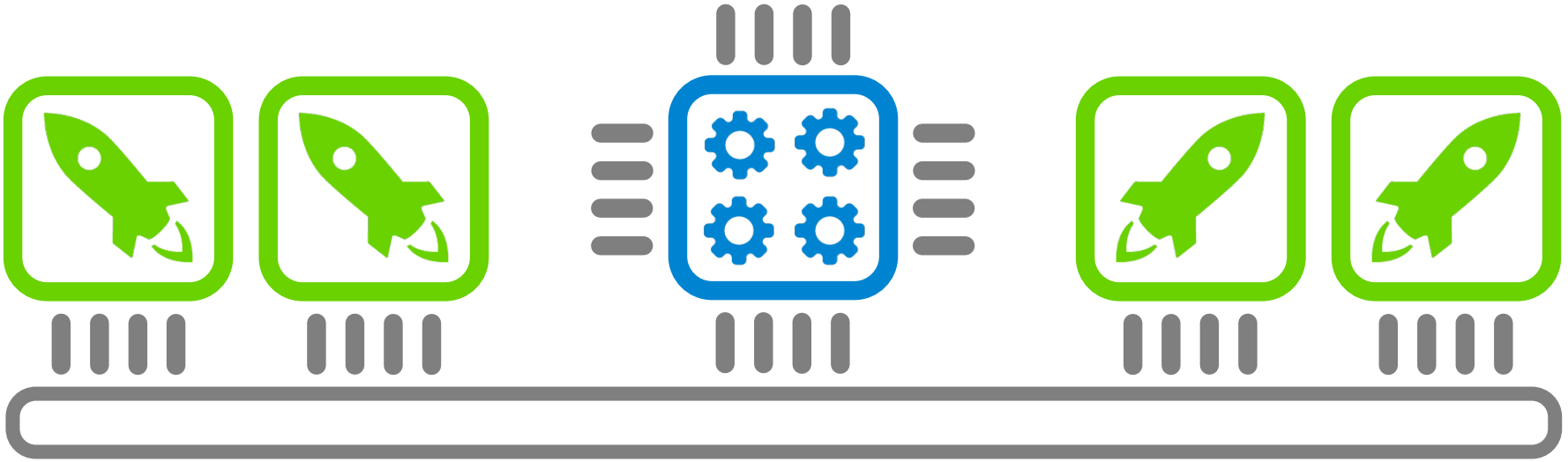
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

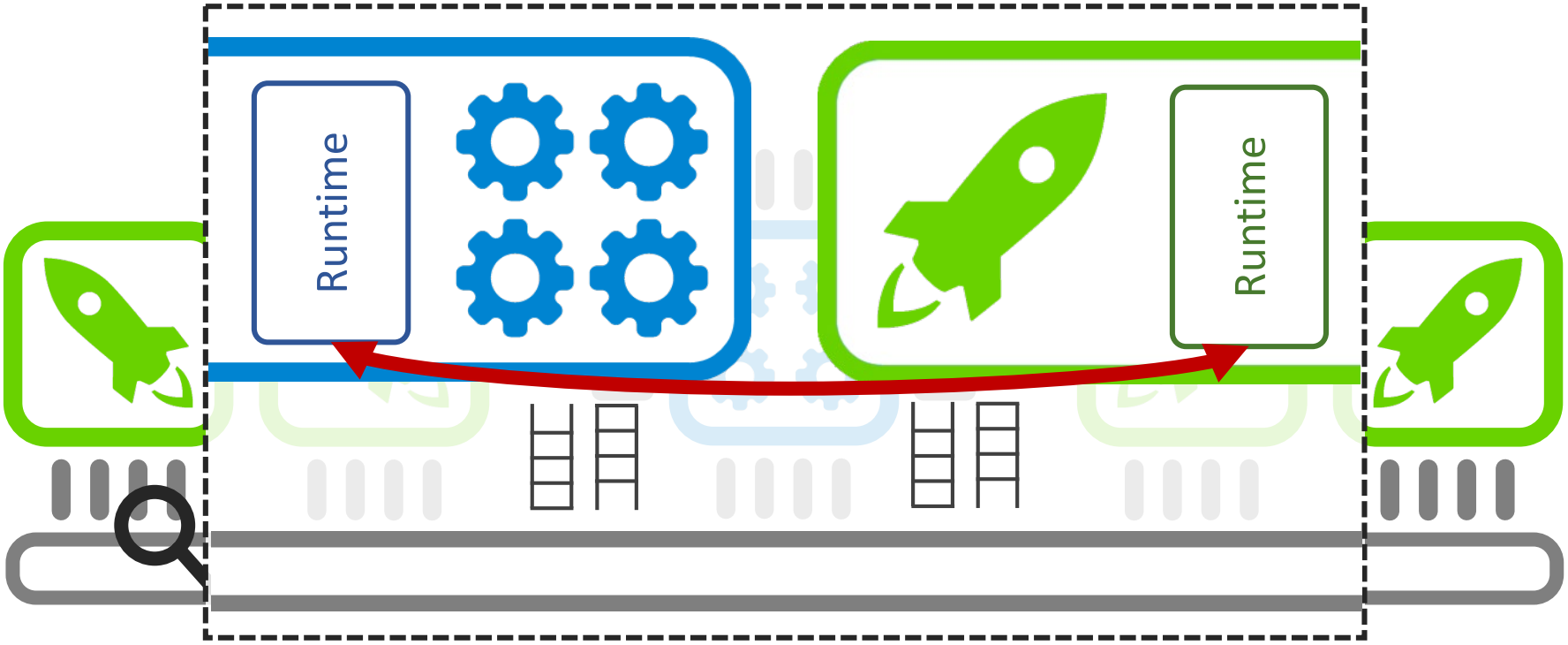
EXCELENCIA SEVERO OCHOA

1) OmpSs@FPGA
2) **Distributed runtime**
3) Evaluation
4) Conclusion

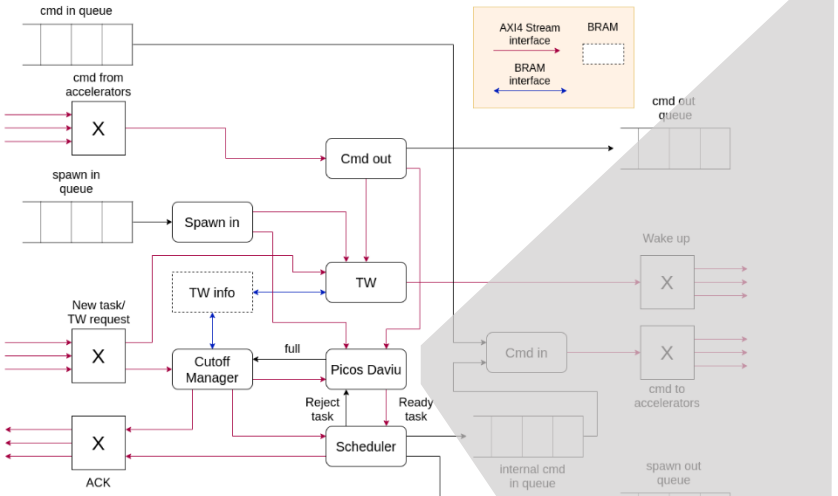
Actual target model

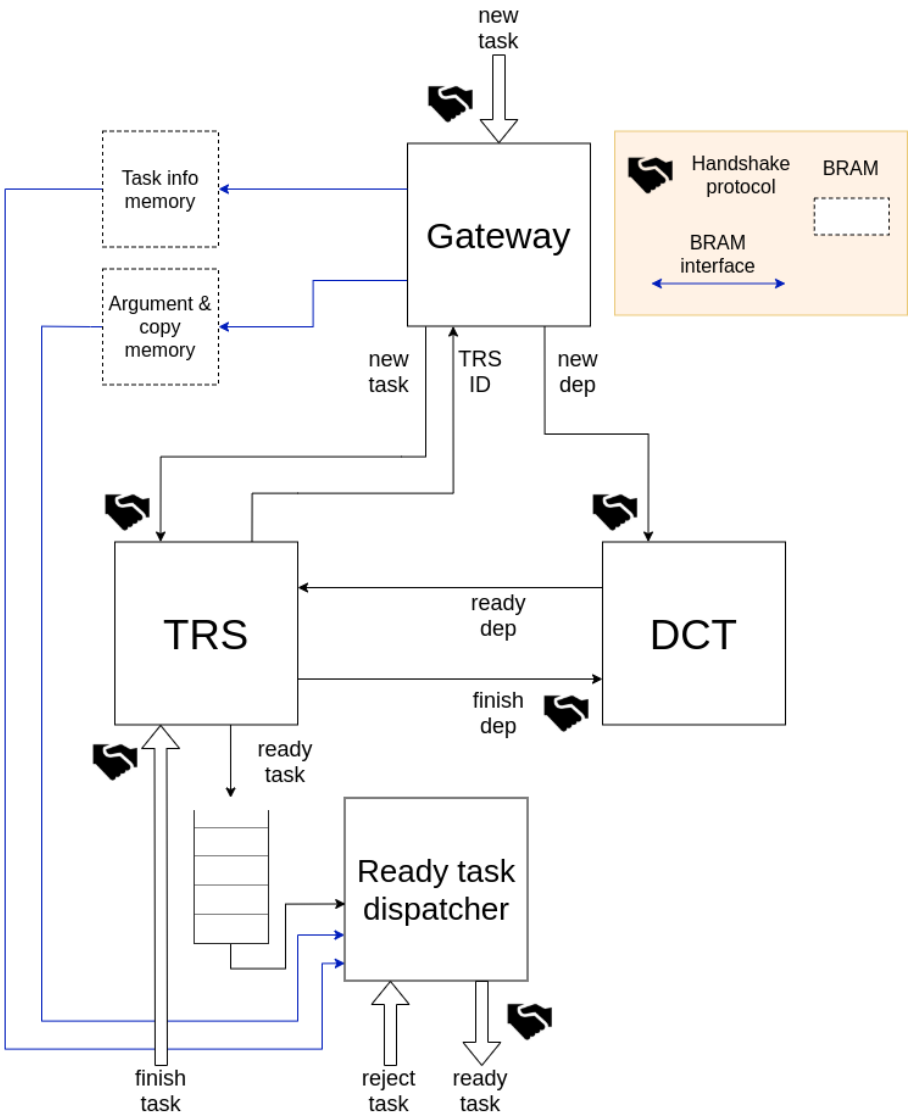


Actual target model



Task creation inside a fpga task: design





In-FPGA task dependence management

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task inout([BS]a)
void update_fpga(int *a, int val, size_t BS) {
  for (size_t i=0; i<BS; ++i) a[i] += val;
}
#pragma omp target device(fpga)
#pragma omp task inout([LEN]a) inout([LEN/BS]index)
void update_blocked(int *a, int *index, int val, size_t LEN, size_t BS) {
  for (size_t i=0; i<(LEN/BS); i++])
    update_fpga(a+BS*index[i], val, BS);
  #pragma omp taskwait
}
int main(...) {
  int *a = (int *)malloc(NUM_ELEMENTS*sizeof(int));
  int *index = (int *)malloc(NUM_ELEMENTS/NUM_ELEMENTS_BLOCK*sizeof(int));
  update_blocked(a, index, 2020, NUM_ELEMENTS, NUM_ELEMENTS_BLOCK);
  #pragma omp taskwait
}
```

In-FPGA task dependence management

```
#pragma omp target device(fpga) num_instances(2)
#pragma omp task inout([BS]a)
void update_fpga(int *a, int val, size_t BS) {
  for (size_t i=0; i<BS; ++i) a[i] += val;
}
#pragma omp target device(fpga)
#pragma omp task inout([LEN]a) inout([LEN/BS]index)
void update_blocked(int *a, int *index, int val, size_t LEN, size_t BS) {
  for (size_t i=0; i<(LEN/BS); i++])
    update_fpga(a+BS*index[i], val, BS);
  #pragma omp taskwait
}
int main(...) {
  int *a = (int *)malloc(NUM_ELEMENTS*sizeof(int));
  int *index = (int *)malloc(NUM_ELEMENTS/NUM_ELEMENTS_BLOCK*sizeof(int));
  update_blocked(a, index, 2020, NUM_ELEMENTS, NUM_ELEMENTS_BLOCK);
  #pragma omp taskwait
}
```

Fast low-latency task & dependence In-FPGA management

New model key features

- In-FPGA task creation
  - Reduce synchronization overheads between host and FPGA devices
  - Free dedicated host resources to offload tasks (power efficient)
  - Better programmability of heterogeneous systems

- In-FPGA task management
  - Support for dependences
  - All the task model advantages

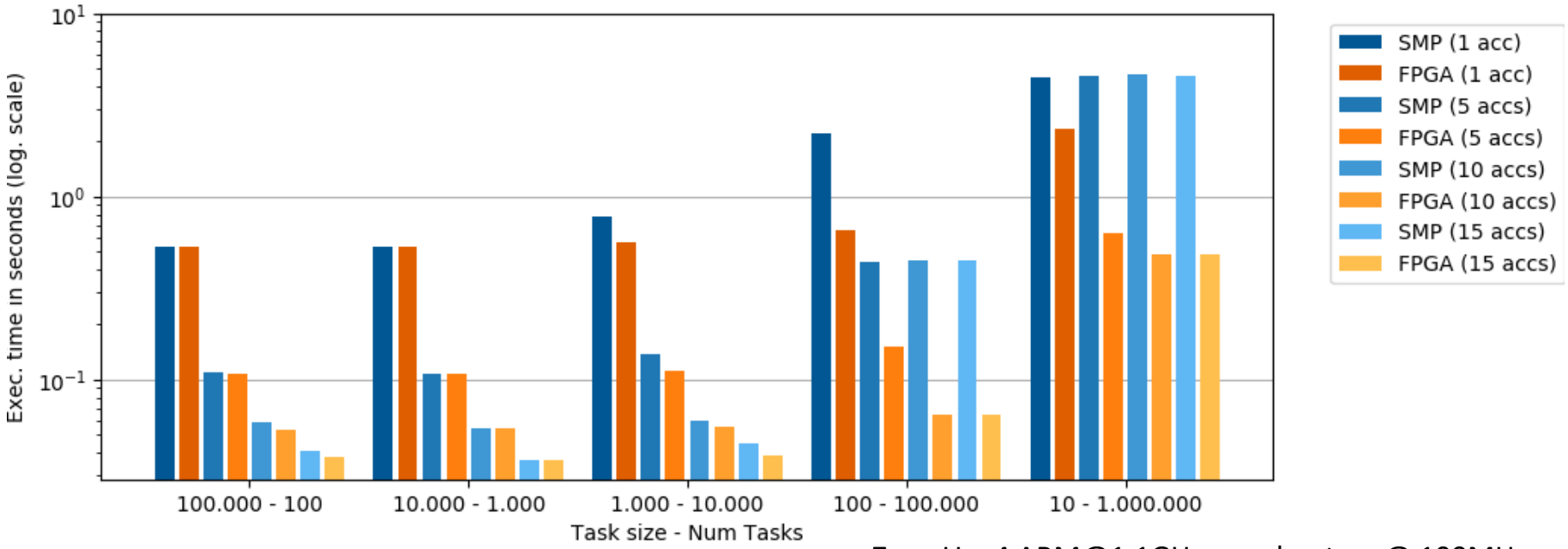
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

1) OmpSs@FPGA
2) Distributed runtime
3) **Evaluation**
4) Conclusion

Allowing larger parallelism

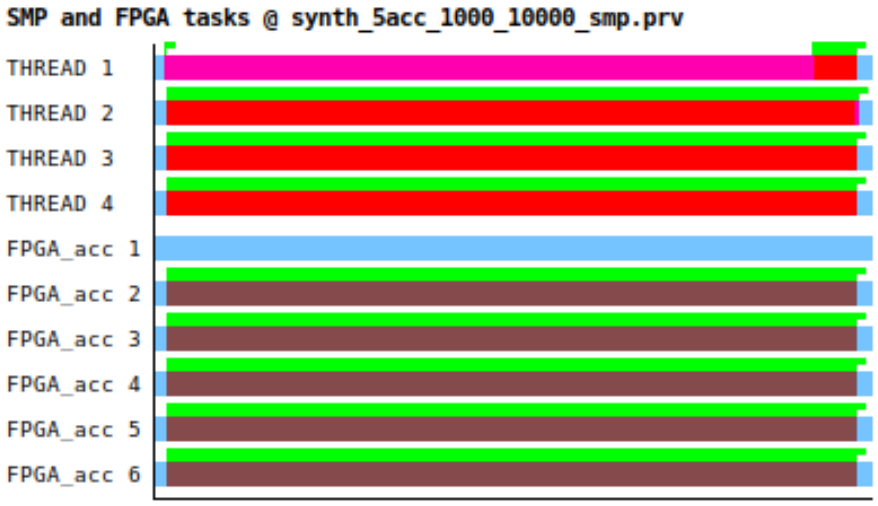
- FPGA management is always better than SMP management
- The smaller the tasks the larger the performance gap
- FPGA management is the only option for parallelism in smallest task sizes

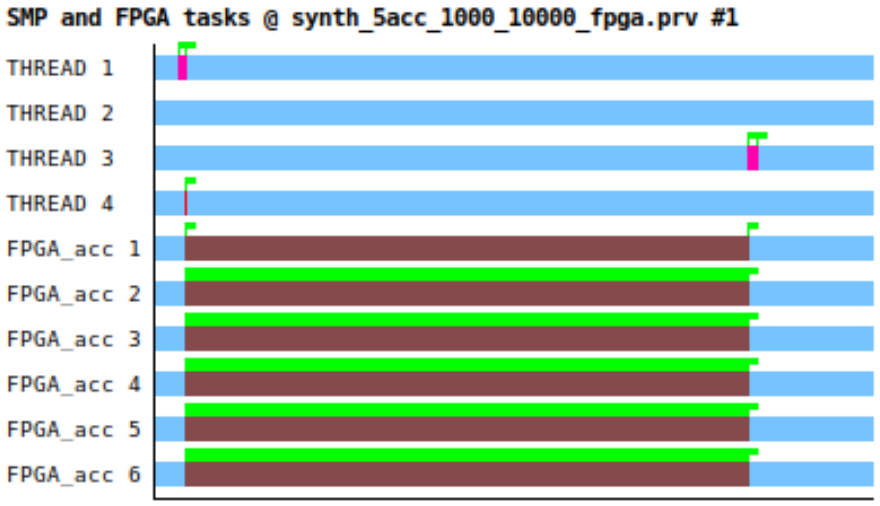


Zynq U+, 4 ARM@1.1GHz, accelerators @ 100MHz

Autonomous FPGA task management

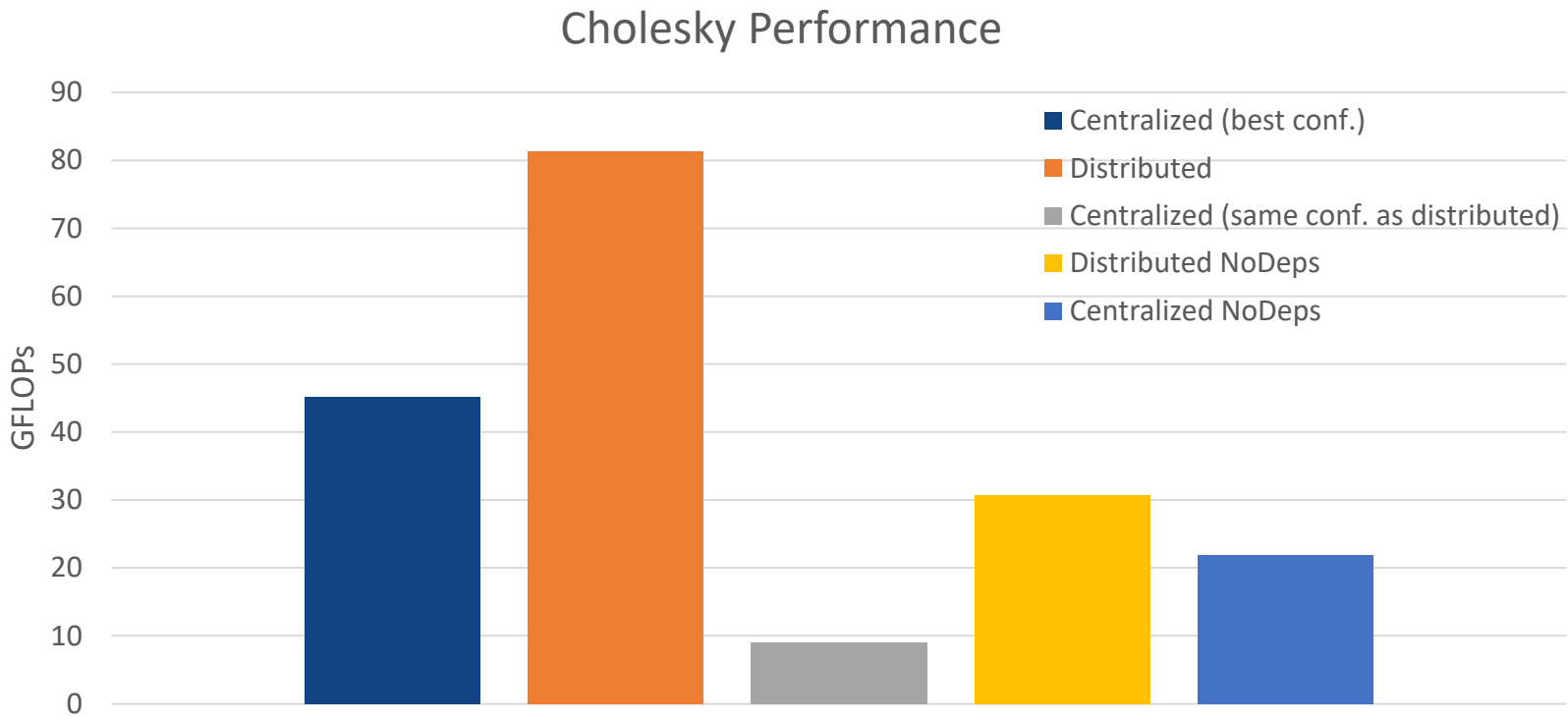
- SMP threads are not utilized when the FPGA management is used
  - This allows to use them to do useful computation
  - FPGA_acc 1 & FPGA Task Manager replace SMP threads using only 0,5% + 1% of FPGA area
- FPGA Task management has smaller execution time





Cholesky computation

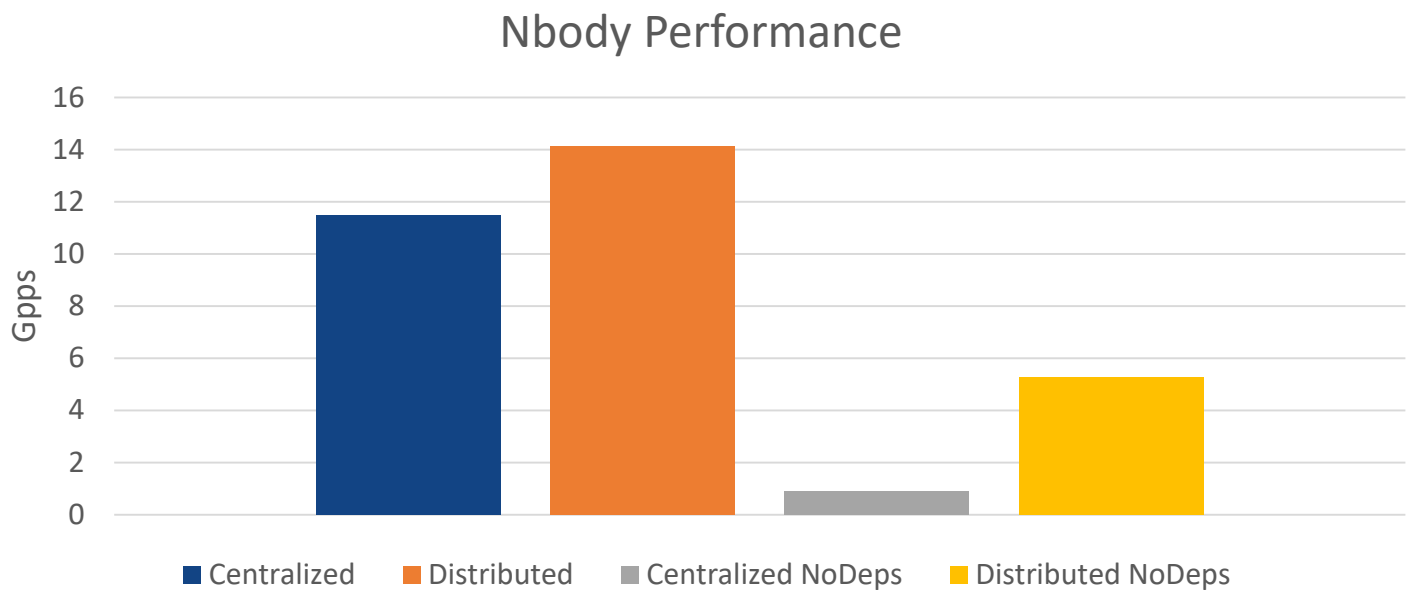
- FPGA management beats Host (Centralized) management
- potrf function is performed in SMP in any case (with a SMP task spawn from the FPGA in Distributed)



Zynq U+, 4 ARM@1.1GHz, accelerators @ 350MHz

NBody computation

- Centralized can not fully use the accelerators due to limited SMP performance



Zynq U+, 4 ARM@1.1GHz, accelerators @ 300MHz

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

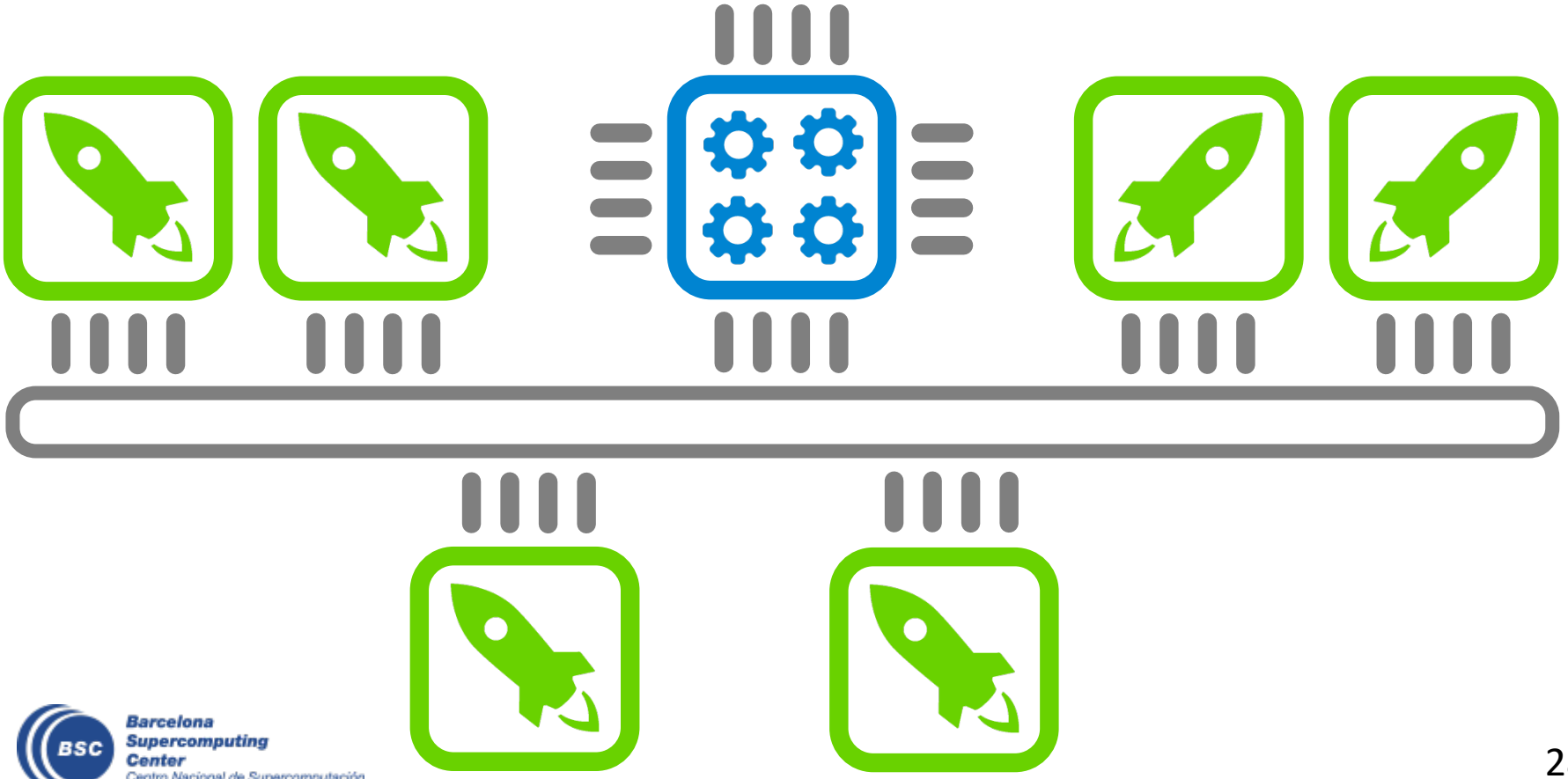
1) OmpSs@FPGA
2) Distributed runtime
3) Evaluation
4) **Conclusion**

Conclusions

- In order to scale we need to move control away from the host
- Modern FPGAs are big enough to support higher level features
- The bigger and more complex the environment, the greater the benefits

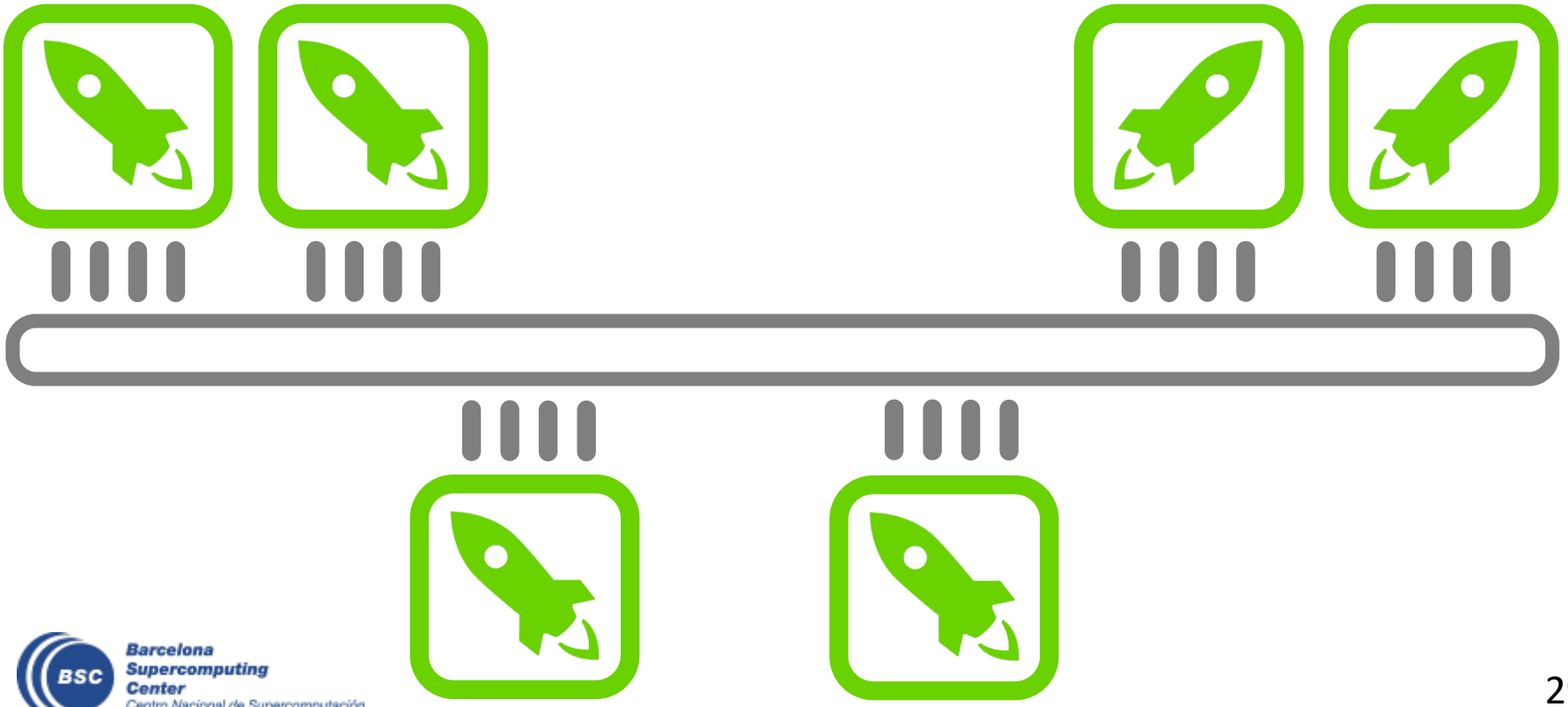
Future

- Make FPGA independent of SMP



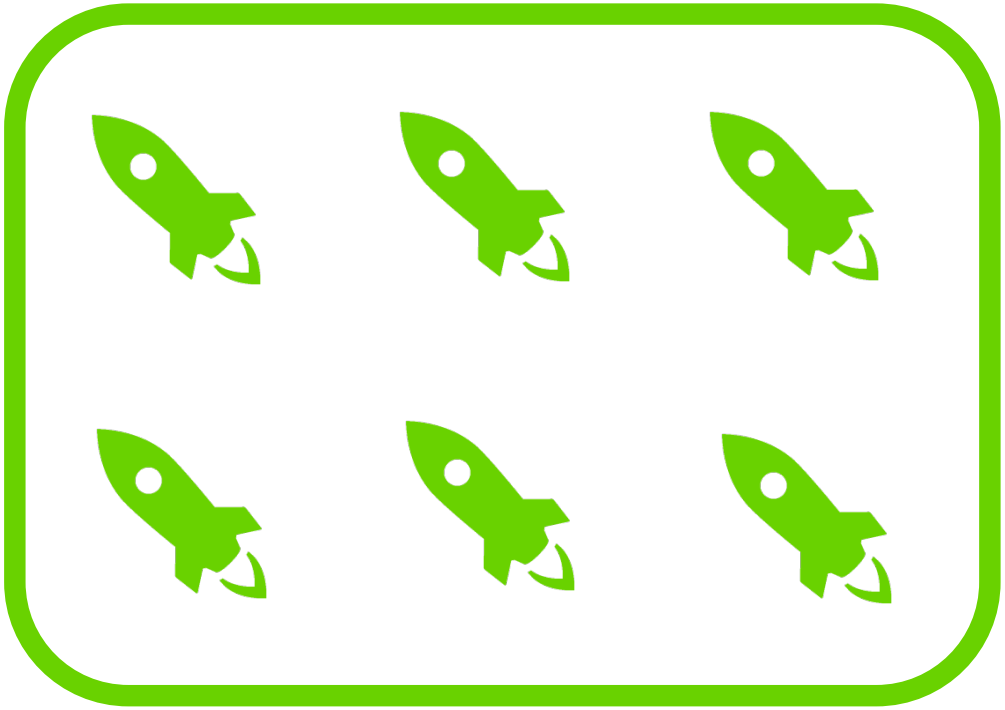
Future

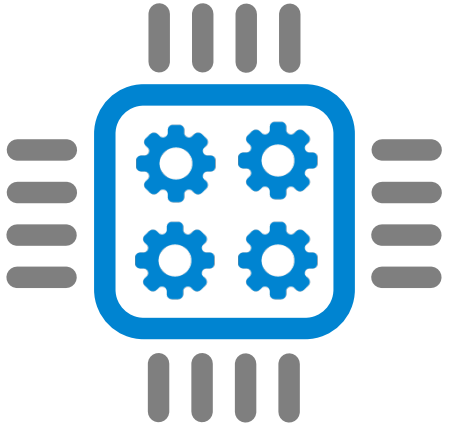
- Make FPGA independent of SMP



Future

- Make FPGA independent of SMP
- Hardware runtime orchestrates multiple FPGAs into a BIG ONE





THANK YOU!

FOR MORE INFO:

pm.bsc.es/ompss-at-fpga

ompss-fpga-support@bsc.es

www.bsc.es