Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Tracing Heterogeneous Executions Through OMPT

M. Wagner[1], G. Llort[1,2], A. Filgueras[1,2], D. Jiménez-González[1,2], H.Servat[3], X.Teruel[1,2], E. Mercadal[1,2], C. Álvarez[1,2], J.Giménez[1,2], X.Martorell[1,2], E. Ayguadé[1,2], J. Labarta[1,2]

michael.wagner@bsc.es

1) Department of Computer Sciences, Barcelona Supercomputing Center
2) Department of Computer Architecture, Polytechnic University of Catalonia - BarcelonaTech
3) Intel Corporation

Outline

- Introduction
  - Tools, OmpSs, OMPT
- How it all works together
- Examples

BSC Tools

- Since 1991
- Based on traces
- Open source
  - http://www.bsc.es/paraver
- Core tools
  - Paraver → Offline trace analysis
  - Dimemas → Message-passing simulator
  - Extrae → Instrumentation
- Focus
  - Detail, variability, flexibility
  - Behavioral structure vs. Syntactic structure
  - Intelligence: Performance Analytics

Extrae – Trace Generation

Parallel programming models

- MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python…

Performance Counters

- CPU – PAPI and PMAPI
- Network – Myrinet (GM and MX)
- OS – Memory allocation, resource usage

Links to source

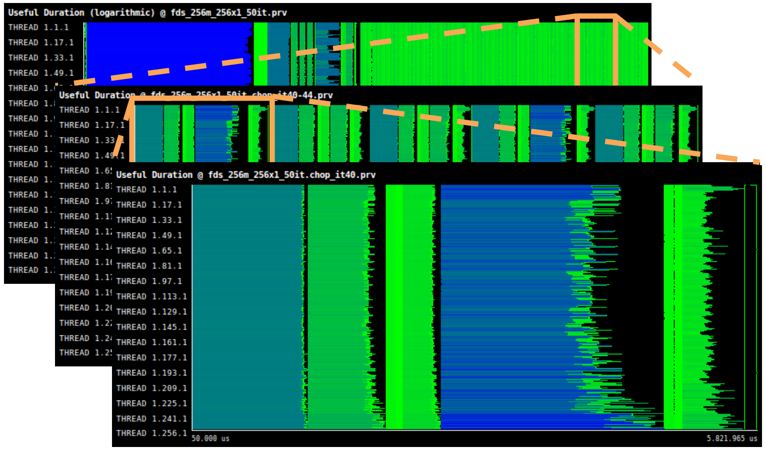
- Callstack at MPI calls
- OpenMP outlined routines
- Selected user functions

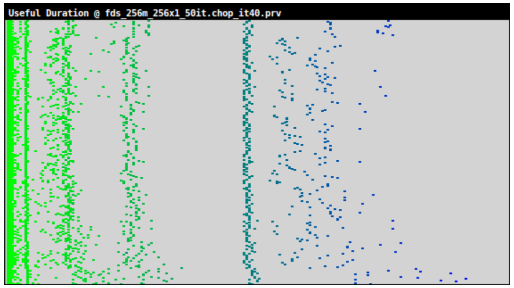
Periodic sampling

User events (Extrae API)

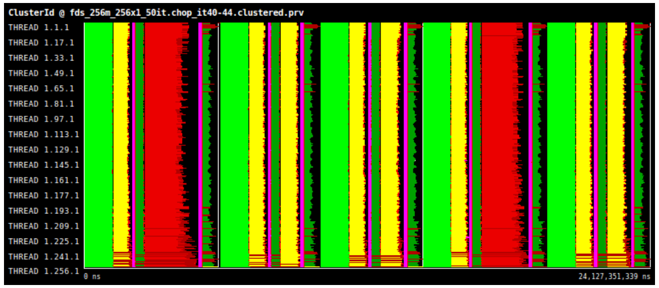
Paraver – Trace Analyzer

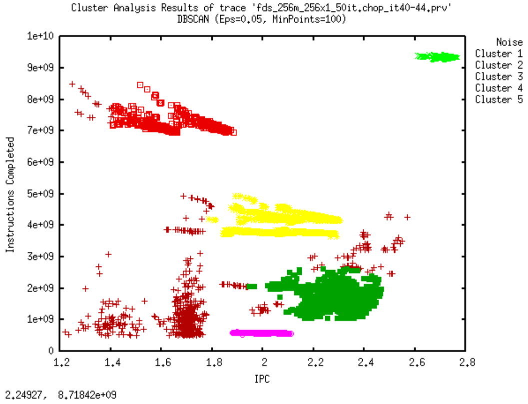
Visual and statistical analysis

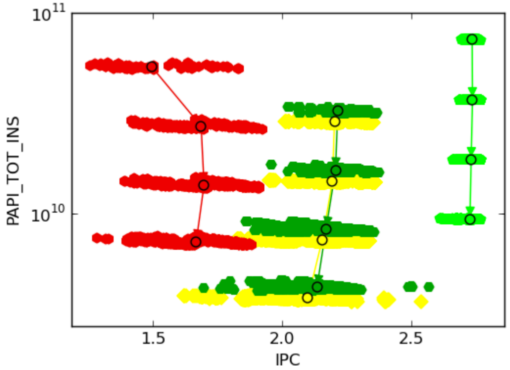




Clustering, Folding, Tracking







OpenMP Instrumentation Challenges

- Multiple implementations
  - GNU, IBM, Intel, Oracle Studio, LLVM…
- Proprietary runtimes with unknown API's
  - Non-disclosure agreements?
  - Reverse engineering?
- Changes in newer versions that break compatibility
  - E.g. Changes in tasking routines in GNU libgomp v4.9
  - Users find out these problems with sudden crashes
- High development burden
- OMPT was proposed to improve this scenario

OMPT: OpenMP Performance Tools API

Goal: a standardized tool interface for OpenMP

- Prerequisite for portable tools for debugging and performance analysis
- Missing piece of the OpenMP language standard

Design objectives

- Enable tools to measure and attribute costs to application source and runtime system
  - Support low-overhead tools based on asynchronous sampling
  - Attribute to user-level calling contexts
  - Associate a thread's activity at any point with a descriptive state
- Minimize overhead if OMPT interface is not in use
  - Features that may increase overhead are optional
- Define interface for trace-based performance tools
- Don't impose an unreasonable development burden

Includes specification to collect performance data from target devices

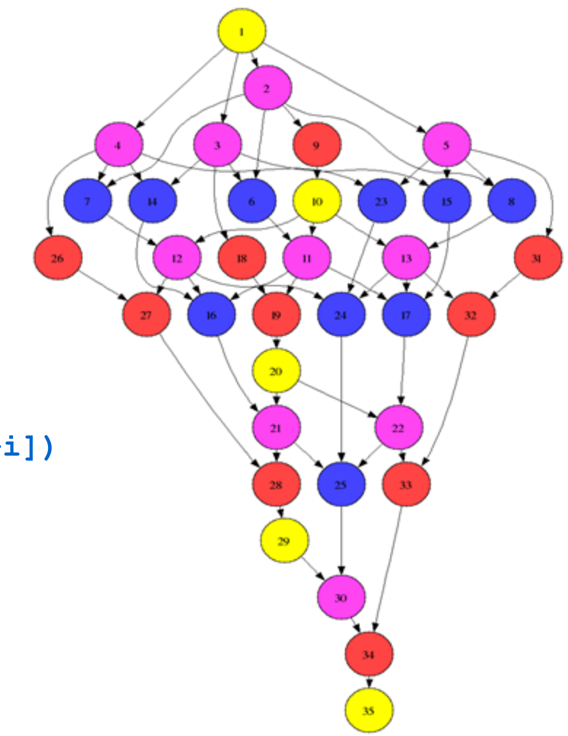
- GPU's, DSP's, FPGA's, Xeon Phi…

Source: OpenMP Tools API, John Mellor-Crummey, STW 2015

OmpSs (Cholesky Example)

- Forerunner of OpenMP that added task data dependencies

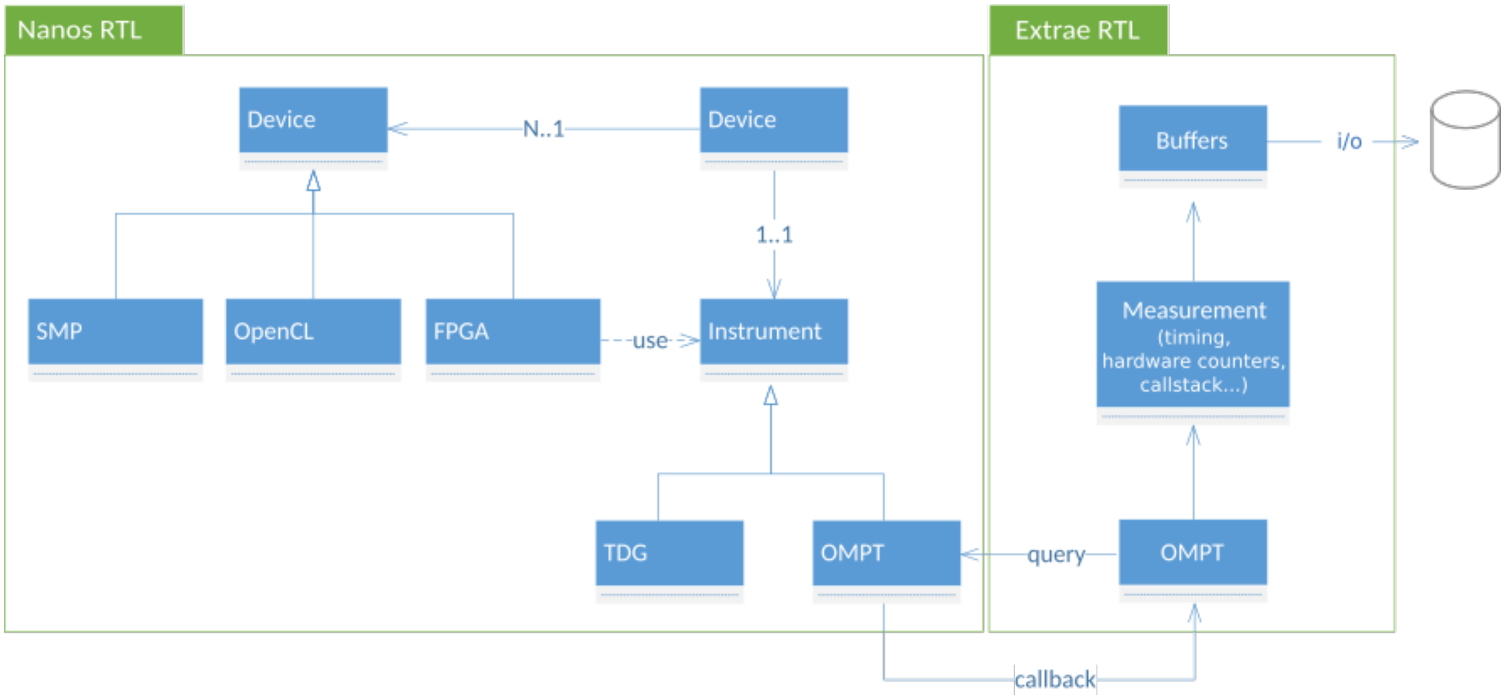
```
void Cholesky( float *A ) {
  int NB, i, j, k;
  for (k=0; k<NB; k++) {
#pragma omp task inout(A[k*NB+k])
      ● dpotrf (A[k*NB+k]);
    for (i=k+1; i<NB; i++)
#pragma omp task input(A[k*NB+k]) inout(A[k*NB+i])
        ● dtrsm (A[k*NB+k], A[k*NB+i]);
    for (i=k+1; i<NB; i++) {
      for (j=k+1; j<i; j++)
#pragma omp task input(A[k*NB+i], A[k*NB+j]) inout(A[j*NB+i])
          ● dgemm( A[k*NB+i], A[k*NB+j], A[j*NB+i]);
#pragma omp task input (A[k*NB+i]) inout(A[i*NB+i])
        ● dsyrk (A[k*NB+i], A[i*NB+i]);
    }
  }
}
```



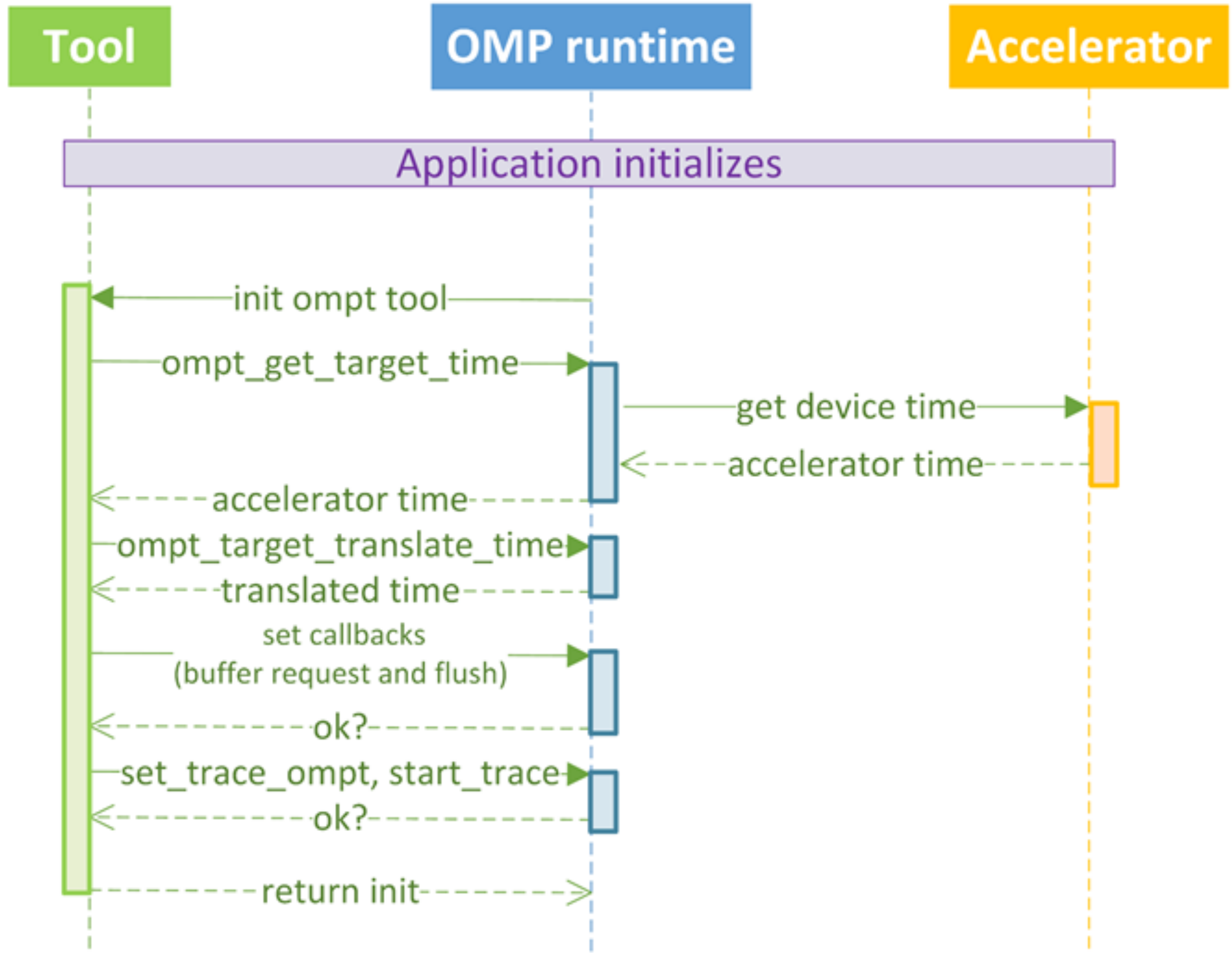
- Tasks can be submitted to accelerators
  - #pragma omp target device(fpga)

Synergy

- OmpSs + Extrae + OMPT = Traces from FPGA devices
- Why?
  - Improve the analysis retrieving data only known by the parallel runtime
    - Activity of accelerator devices (FPGA)
- How?
  - Implement the OMPT standard to connect runtime and performance tool

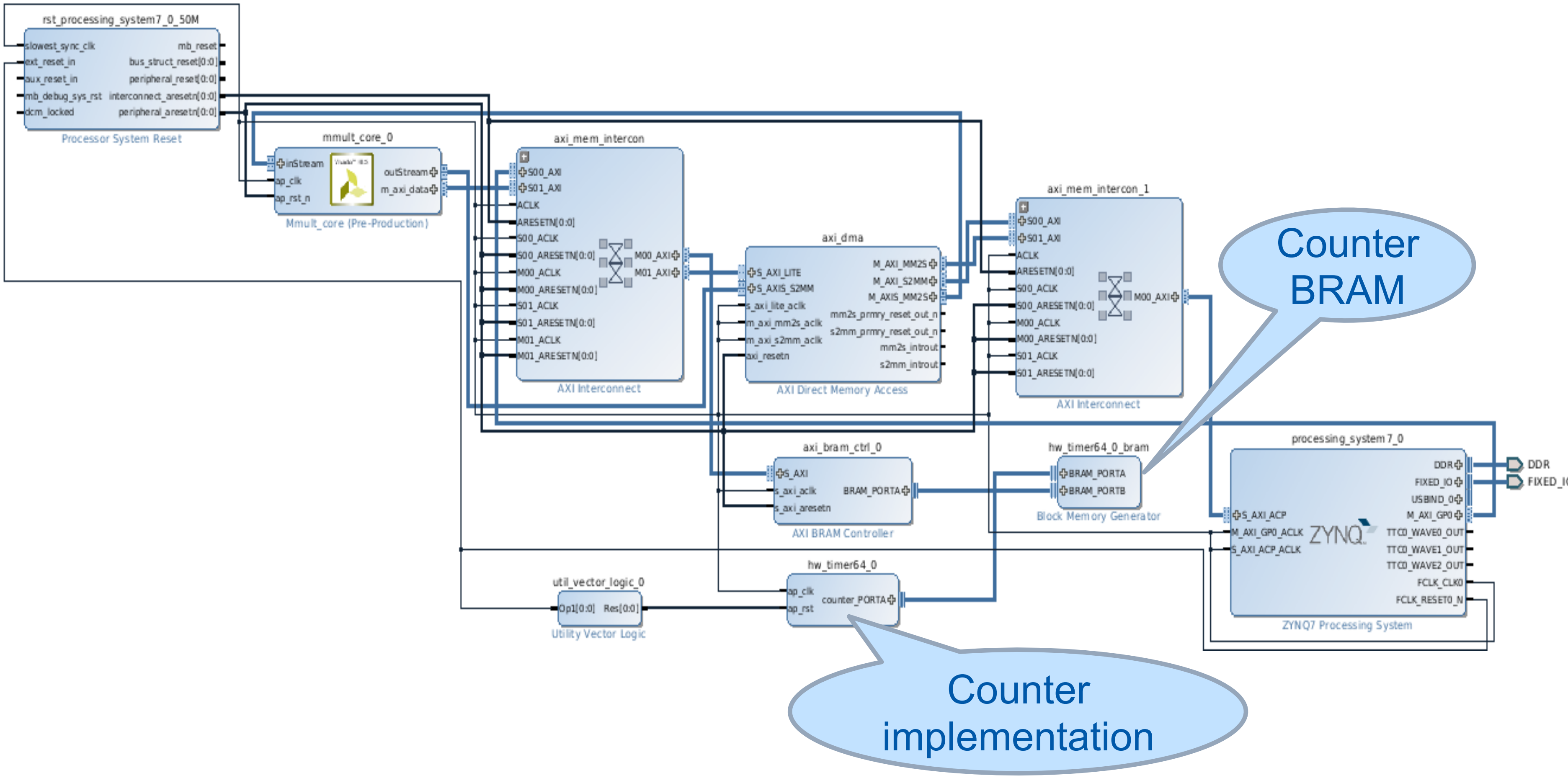


FPGA Tracing: Initialization and Time Correction



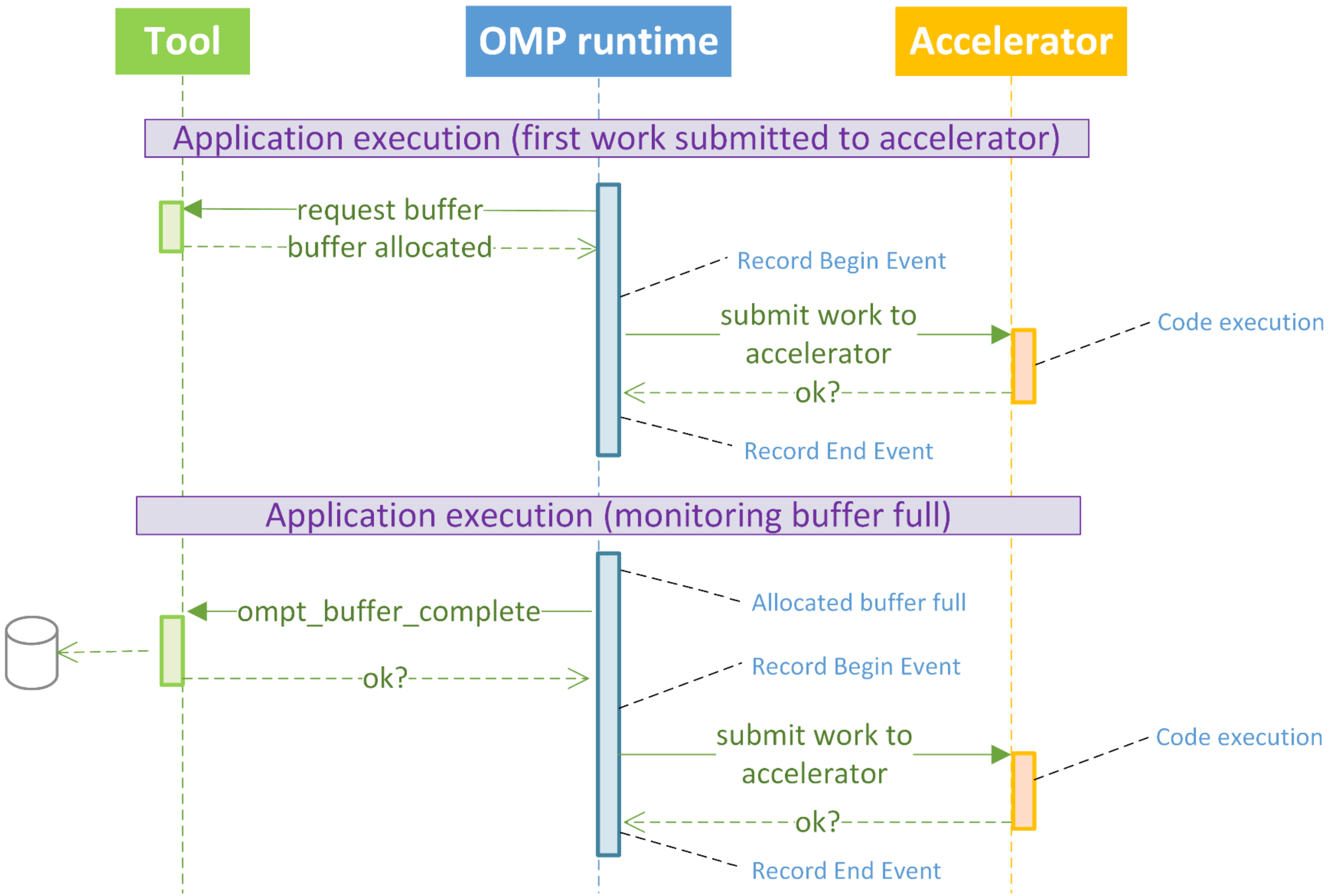
FPGA Tracing: Cycles Counter

- Hardware (access from PL and PS)



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

FPGA Tracing: Execution and Event Recording



Experimental Setup

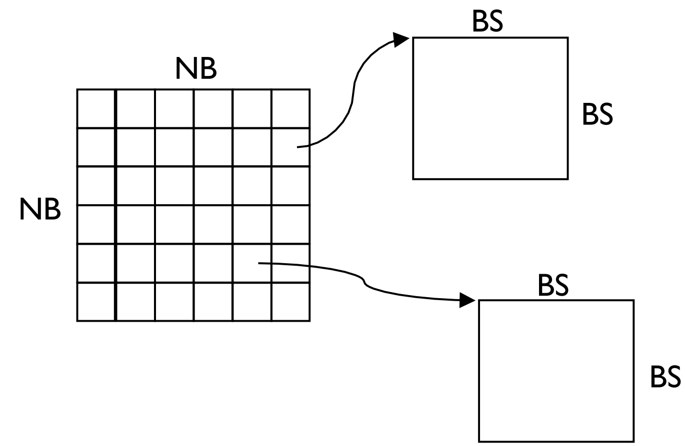
- Zynq SoC 702 board
  - SMP dual core ARM Cortex A9 at 666 MHz + FPGA Xilinx Artix 7
- OmpSs ecosystem for FPGA/SMP heterogeneous execution
  - Mercurium compiler v1.99.9
  - Nanos++ runtime 0.10a
- Instrumentation
  - Extrae tracing framework v3.3.0
- Bitstream generation to implement FPGA logic
  - Xilinx's Vivado and Vivado HLS v2015.4
    - For FP apps, HLS synthesizes code compliant with the IEEE-754 standard
- All applications and libraries cross-compiled
  - arm-linux-gnueabihf-gcc 4.8.4 (Ubuntu/Linaro 4.8.4-2ubuntu1 14.04.1)

Example: Matrix Multiply

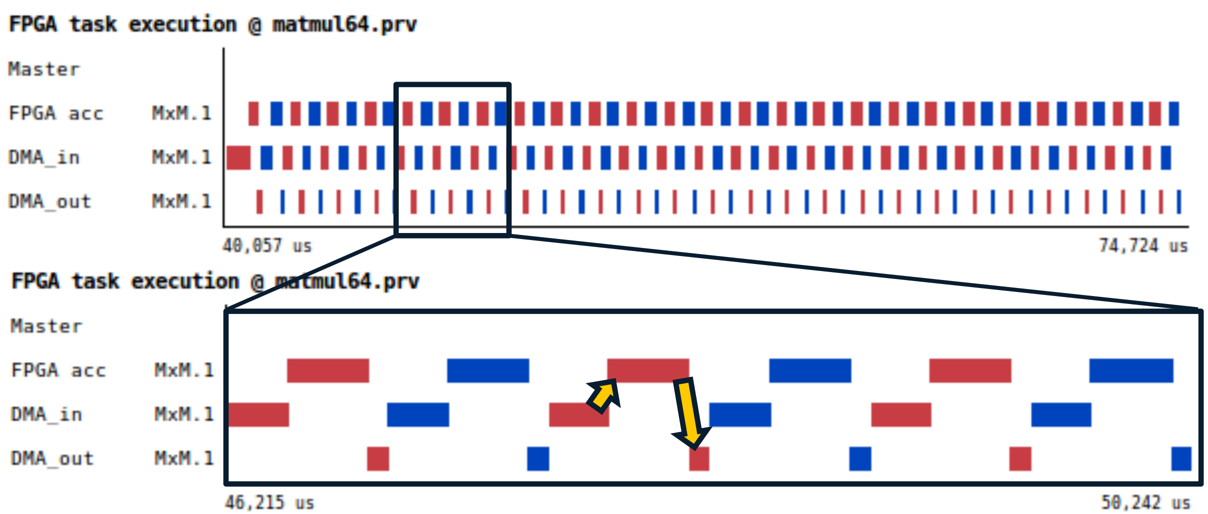
```
#pragma omp target device (fpga)
#pragma omp task in([BS*BS]A, [BS*BS]B) inout([BS*BS]C)
void MxM(REAL *A, REAL *B, REAL *C);

void matmul(REAL **AA, REAL **BB, REAL **CC, int NB) {
  for (int k=0; k<NB; k++)
    for (int i=0; i<NB; i++)
      for (int j=0; j<NB; j++) {
        MxM(AA[i*NB+k], BB[k*NB+j], CC[i*NB+j]);
        MxM( … );
      }
}
```

Unrolled by 2



- 2 MxM, 256 nblocks, 64 bsize, 1 FPGA accelerator



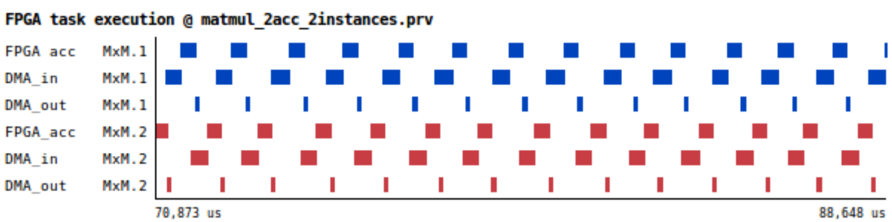
Example: Matrix Multiply

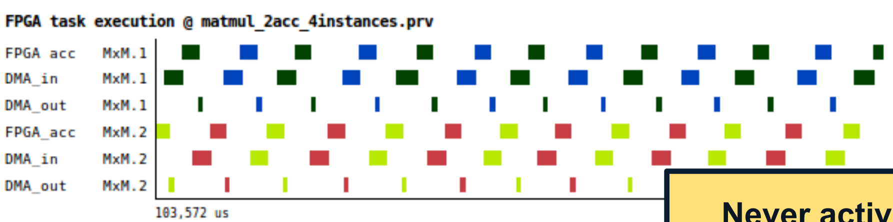
- Task avg. execution time per stage
  - Input/output DMA ratio close to 3x
  - DMA transfer performance may vary
    - Different DMA input/output bandwidth
    - Different waiting time for the corresponding DMA submit

64 x 64	Task #1	Task #2
FPGA acc	334.81 us	337.53 us
DMA_in	260.43 us	246.53 us
DMA_out	82.82 us	82.76 us

32 x 32	Task #1	Task #2
FPGA acc	46.51 us	44.51 us
DMA_in	229.02 us	256.37 us
DMA_out	80.49 us	82.33 us

- Task overlapping using 2 FPGA accelerators





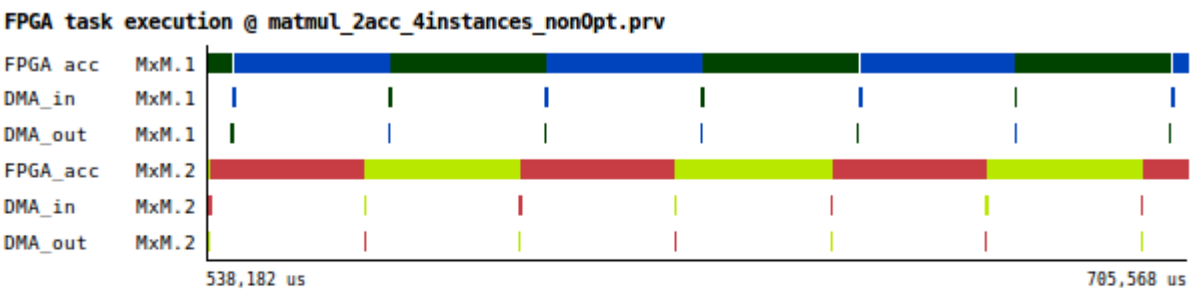
	DMA_in 1	DMA_out 1	FPGA acc 1
DMA_in 2	0%	21.4%	17.0%
DMA_out 2	20.1%	0%	0%
FPGA_acc 2	11.2%	0%	0%

Never active simultaneously due to scheduling pattern

Computations do not overlap due to the small execution times

Example: Matrix Multiply

- >90% computation overlap using a non-optimized accelerator



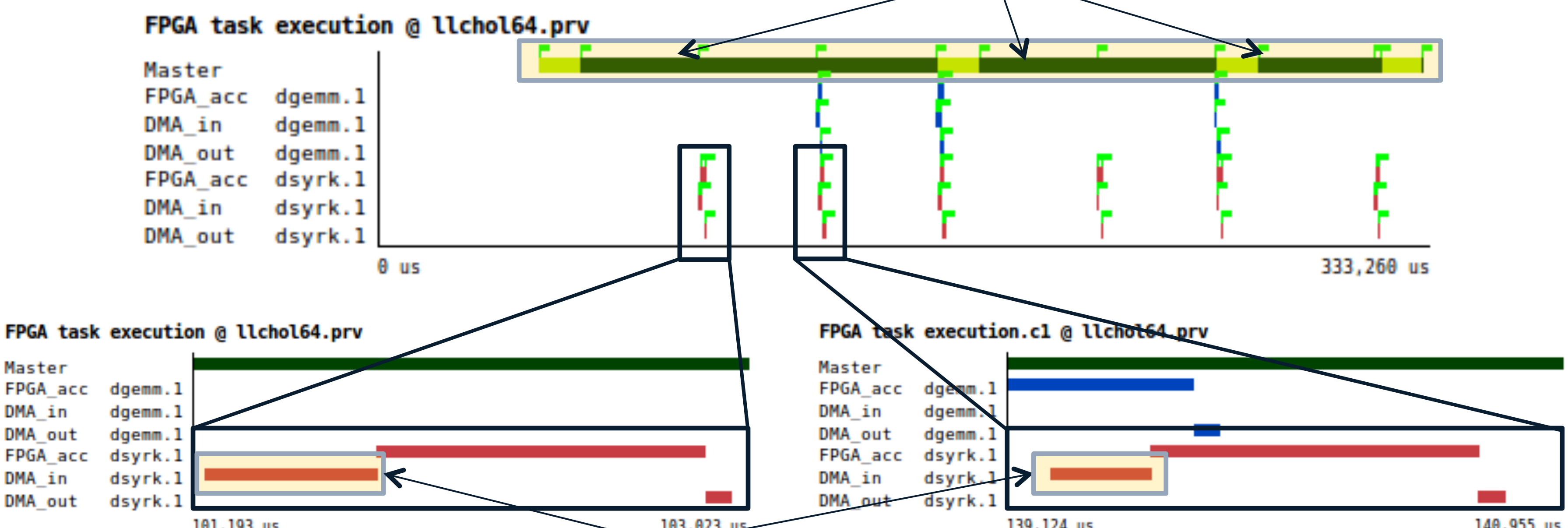
- We can gain insight about…
  - DMA memory transfers and computation overlap
  - Latency information
  - Runtime memory management
  - Scheduling policy

Example: Cholesky Decomposition (SMP and FPGA Issues)

Evaluated 1 possible target device partition

- dgemm/dsyrk running in FPGA; dpotrf/dtrsm running in SMP
- Possible improvements
  - Accelerate dtrsm or use mo…
  - Use another accelerator for c…

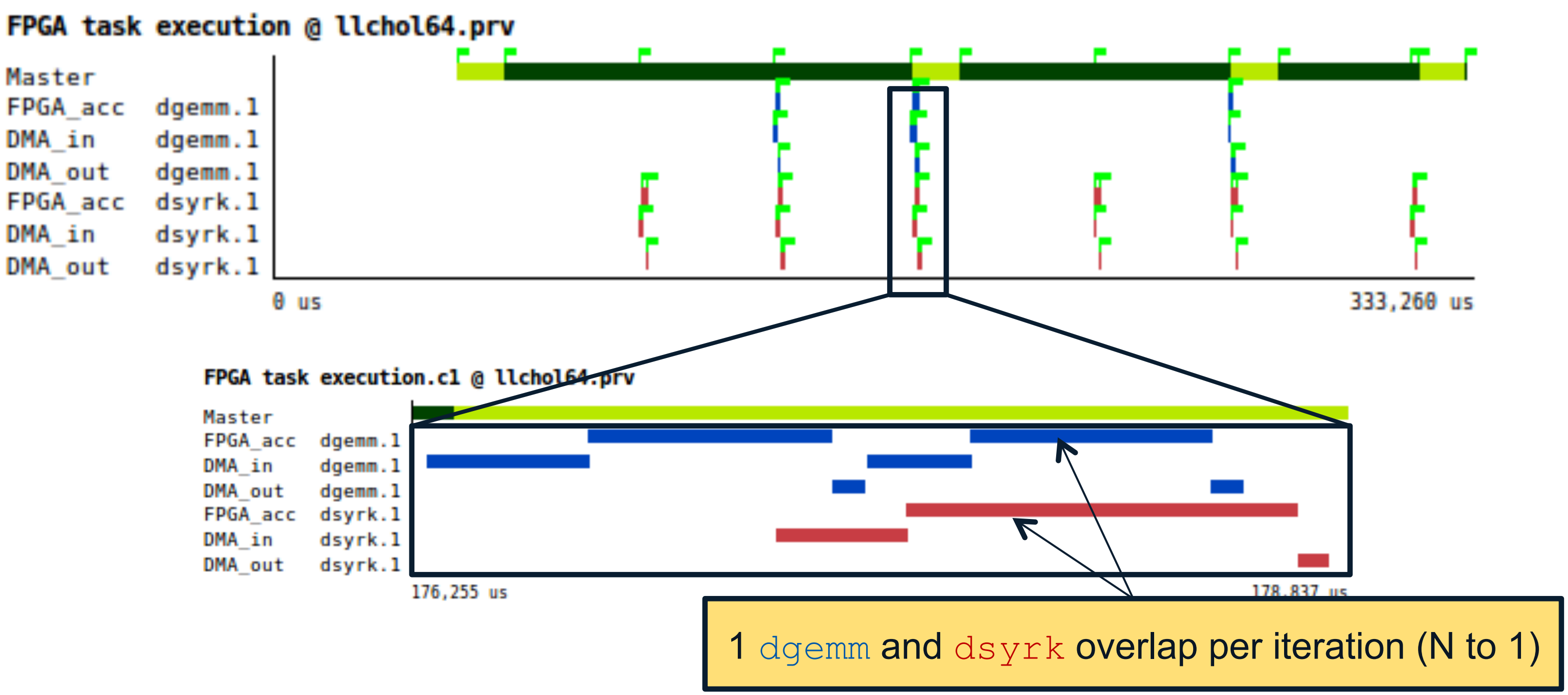
dtrsm is the main critical execution path resulting in long idles in the accelerators



First DMA transfer costs 2x more than any other due to driver initialization

Example: Cholesky Decomposition (SMP and FPGA Issues)

- Evaluated 1 possible target device partition
  - dgemm/dsyrk running in FPGA; dpotrf/dtrsm running in SMP
  - Possible improvements
    - Accelerate dtrsm or use more SMP cores
    - Use another accelerator for dgemm tasks



Conclusions and Future Work

- OMPT enabled cooperation between OmpSs and Extrae
- Seamless software interoperability
- About 1 y.o. spec already supported different types of accelerators
- Current spec is practically in final version and under consideration to be integrated in OpenMP 5
- Analysis provided insight to make SW/HW improvements of 3x
- Extend runtime to support CUDA, OpenCL and Xeon Phi
- Extend instrumentation to include further information in the trace
  - Algorithm phases, logical states, task dependences, runtime internals, power efficiency…