www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Hardware Hetergeneous Task Scheduling for Task-based Programming Models

Xubin Tan

OpenMPCon 2018

Advisors: Carlos Álvarez, Daniel Jiménez-González

Agenda

> Background, Motivation

> Picos++ accelerated hardware runtime

> Hardware Heterogeneous task scheduling

> Results

Agenda

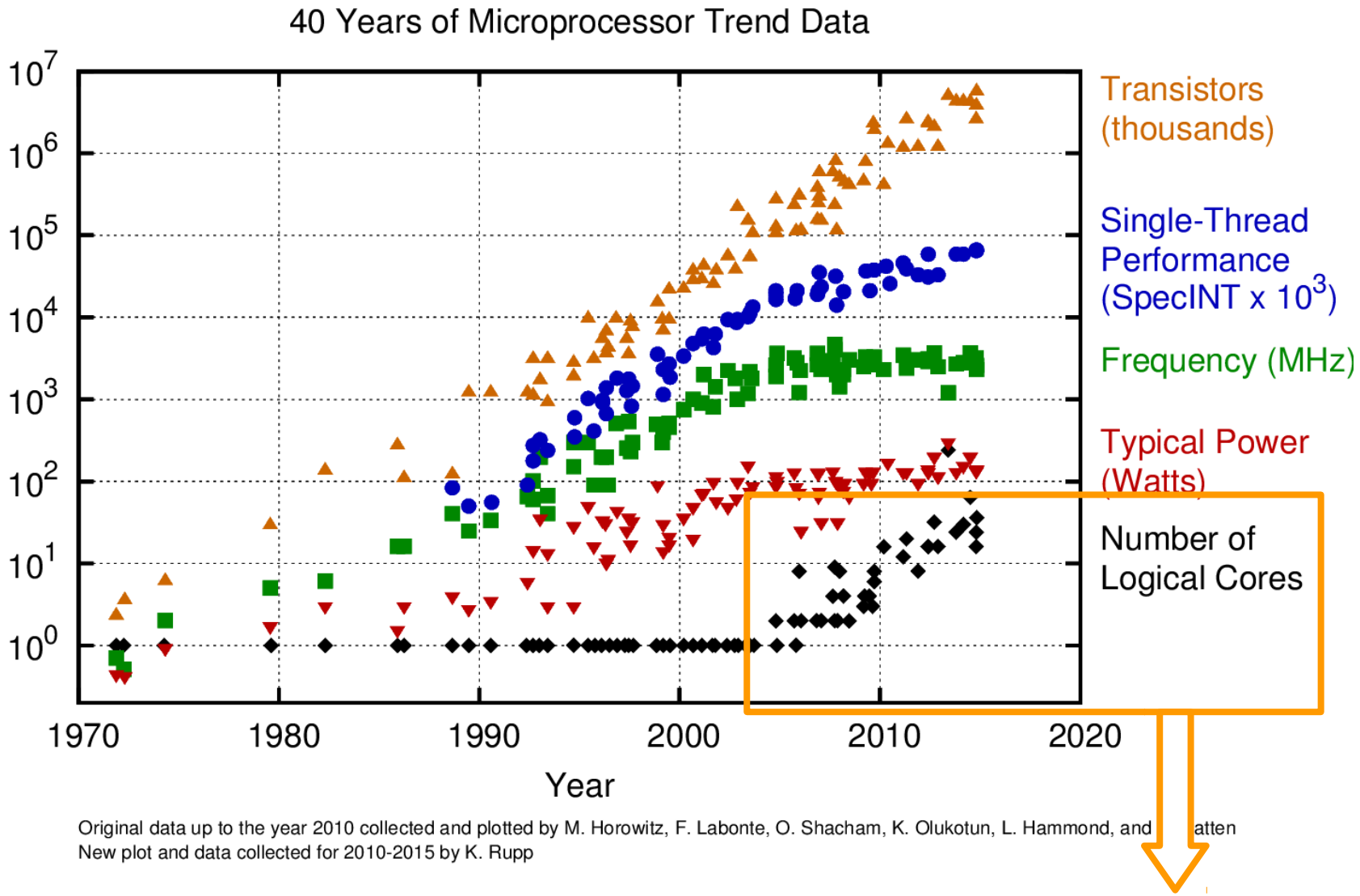
> Background, Motivation

> Picos++ accelerated hardware runtime

> Hardware Heterogeneous task scheduling

> Results

Moore's Law – Commodity Microprocessor

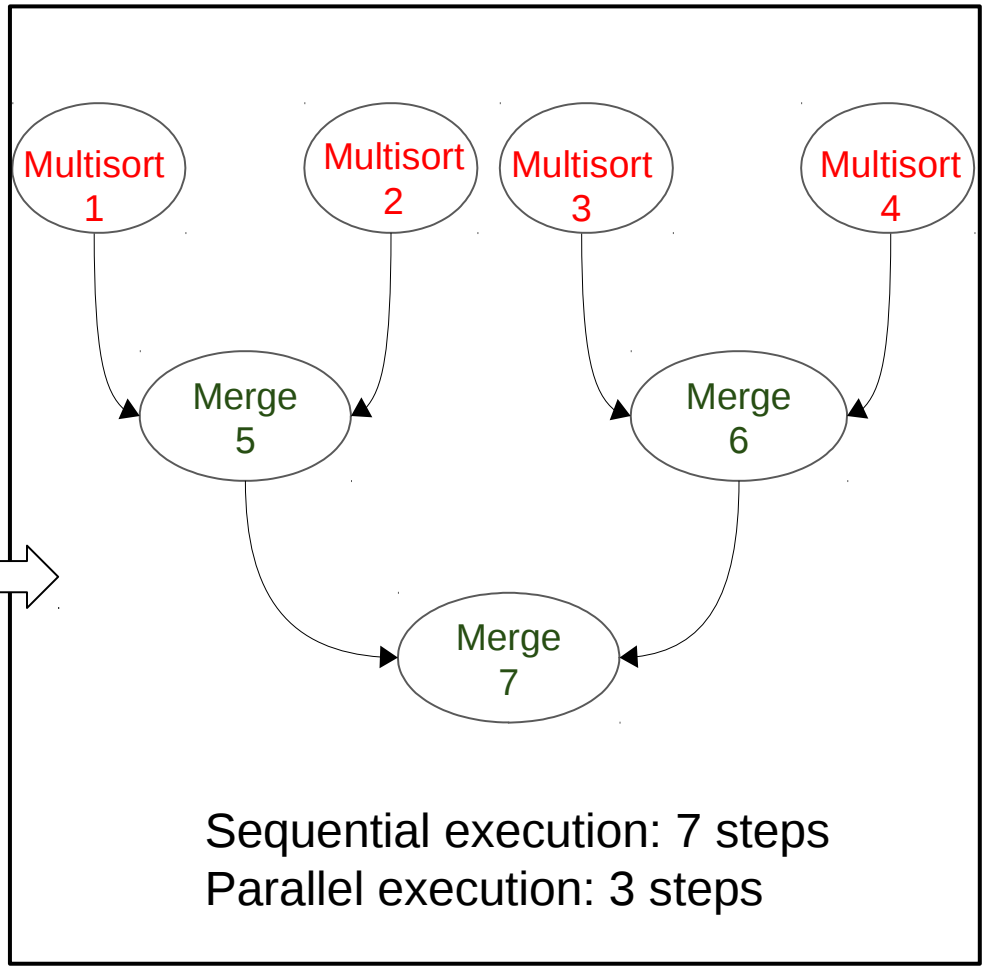


Multi-core, many-core, heterogeneous architectures

Task-based programming models

With task-based programming models (OpenMP, OmpSs, Codelet, StarPU, …, etc), an application can be expressed as a collection of tasks with dependences

```
void multisort (long n, T data[n], T tmp[n]) {

if(n<CUTOFF){sequential_sort();}

else{
   multisort(n/4, &data[0], &tmp[0]);
   multisort(n/4, &data[n/4], &tmp[n/4]);
   multisort(n/4, &data[n/2], &tmp[n/2]);
   multisort(n/4, &data[3n/4], tmp[3n/4]);

   merge(n/4, &data[0], &data[n/4], &tmp[0]);
   merge(n/4, &data[n/2], &data[3n/4], &tmp[n/2]);

   merge(n/2, &tmp[0], &tmp[n/2], &data[0]);
 }}
```



OmpSs programming model and its runtime

```
void multisort (size_t n, T data[n], T tmp[n]) {
```

Programmer →

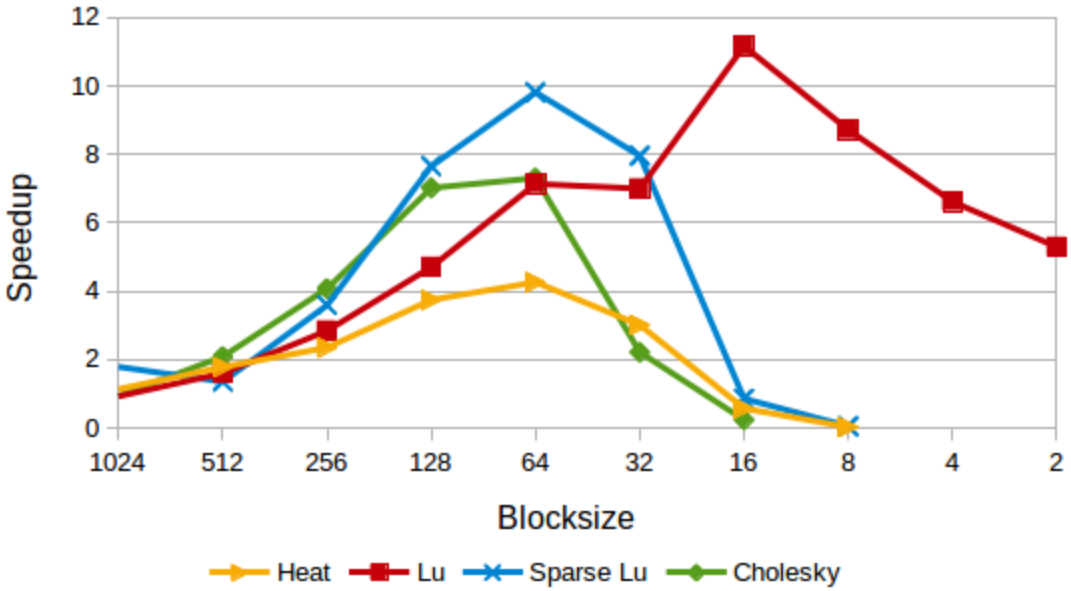
```
#pragma omp task inout(data[0], tmp[0])
    multisort(i, &data[0], &tmp[0]);
    ...
#pragma omp task in(data[0], data[i]) out(tmp[0])
    merge(i, &data[0], &data[i], &tmp[0]);
    ...
}
```

Compiler (Mercurium) → generates runtime calls to create tasks (task descriptor)

Runtime (Nanos++) → manages task creation, constructs task dependence graph, and schedules ready tasks dynamically

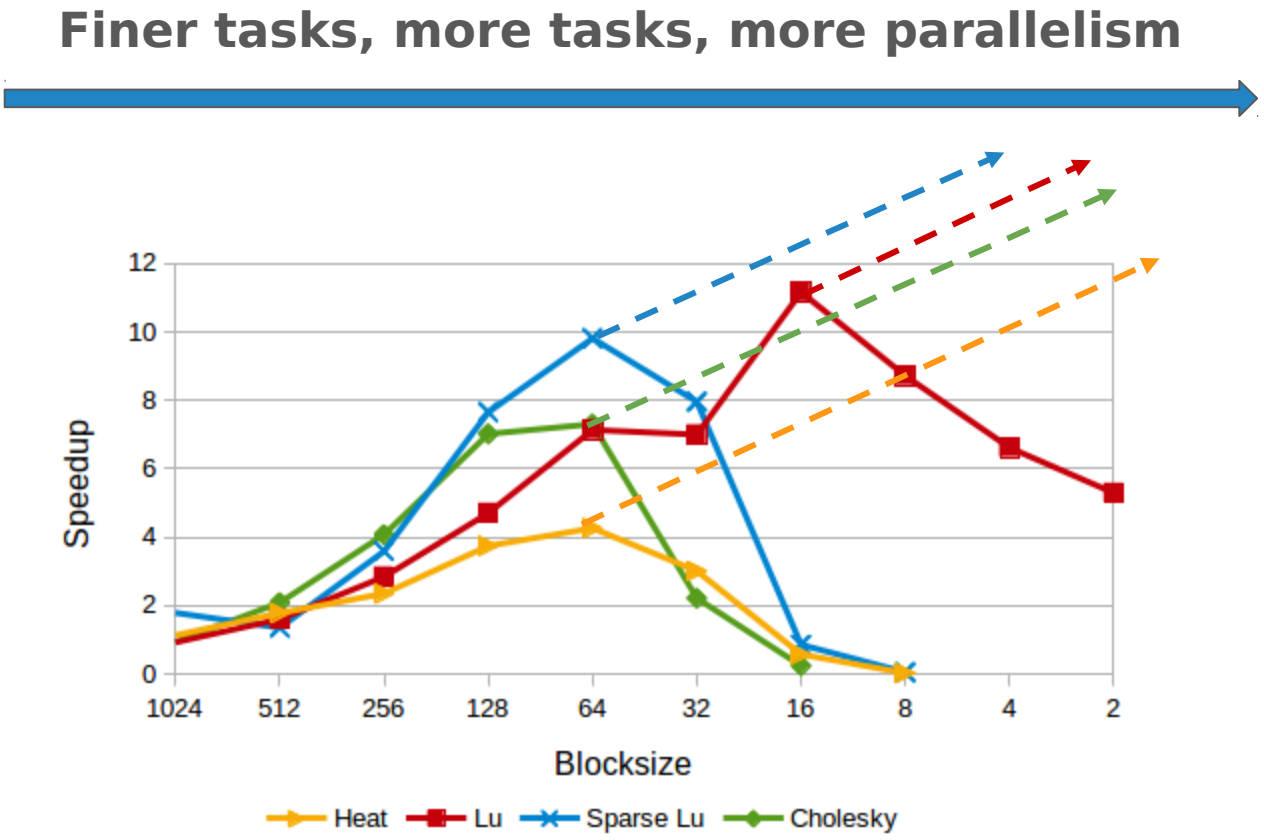
SW runtime for exploiting fine-grained task parallelism

Smaller tasks, more tasks, more parallelism



Four OmpSs applications with problem size 2K*2K, with 12 cores

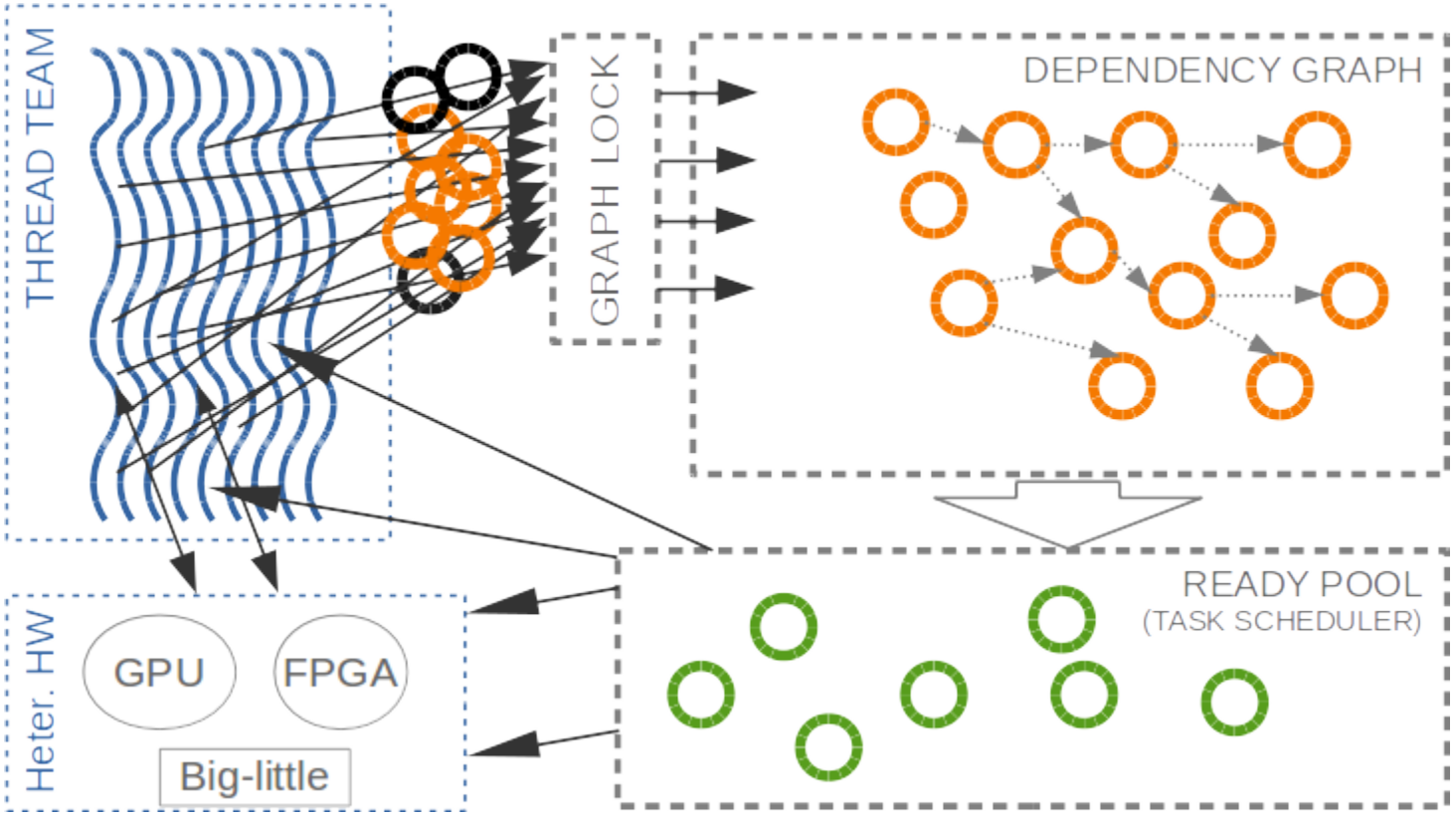
Use hardware to accelerate SW runtime



Four OmpSs applications with problem size 2K*2K, with 12 cores

SW-only runtime overheads

Costly task-dependence analysis and heterogeneous task scheduling



New task Ready task Finish task

Agenda

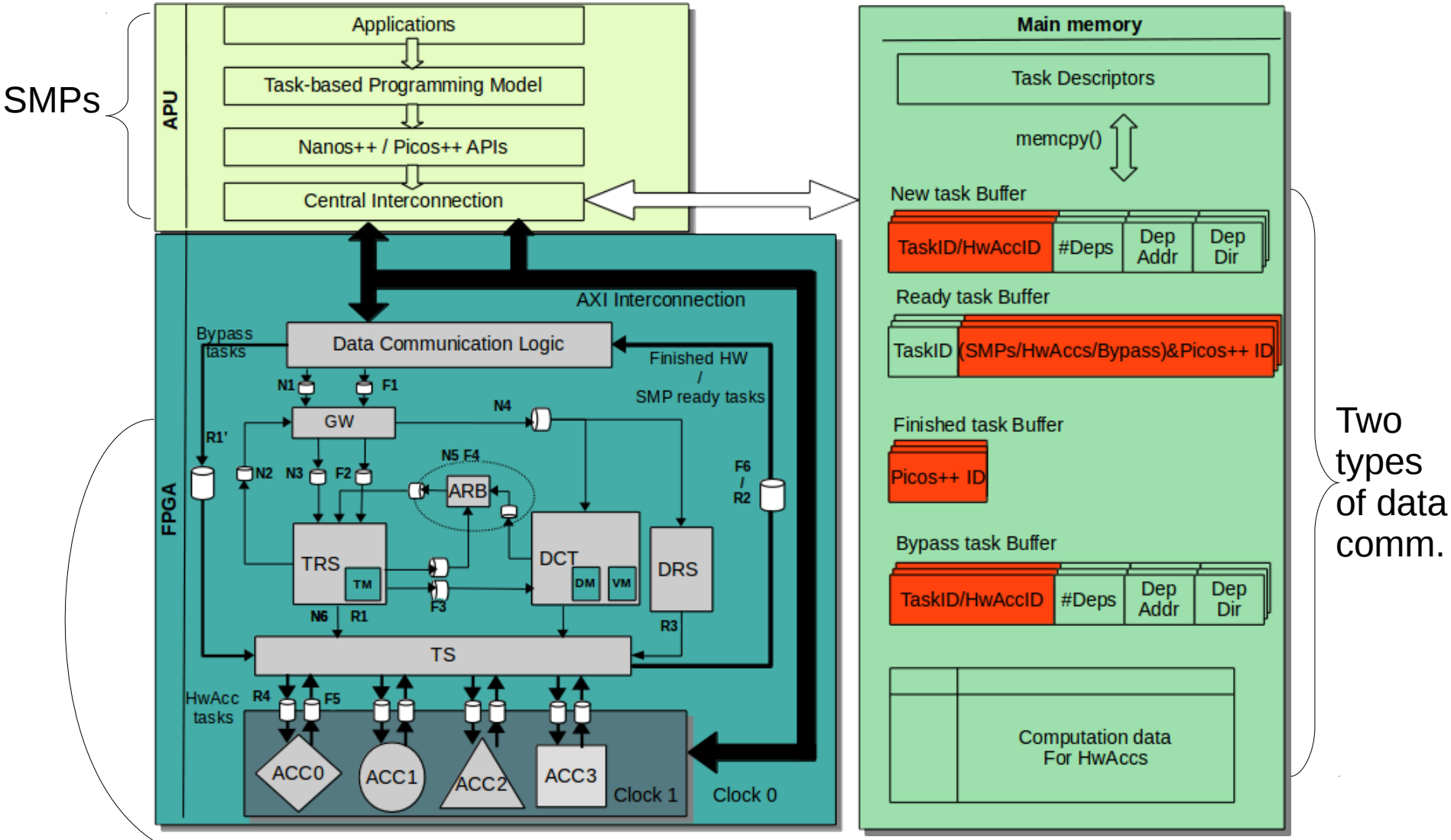
> Background, Motivation

> Picos++ accelerated hardware runtime

> Hardware Heterogeneous task scheduling

> Results

Picos++ System



Agenda

> Background, Motivation

> Picos++ accelerated hardware runtime

> Hardware Heterogeneous task scheduling

> Results

Heterogeneous Task Scheduling

Two main issues concerning scheduling:

A. Load imbalance

Scheduling task to a suitable hardware device with the least number of waiting tasks

B. Hardware to hardware communication

Picos++ directly manages task scheduling to different HW units

Related work:

A. Static algorithms

Heterogeneous Earlist Finish Time (HEFT), Crtitical-Path-on-a-Processor (CPOP), etc

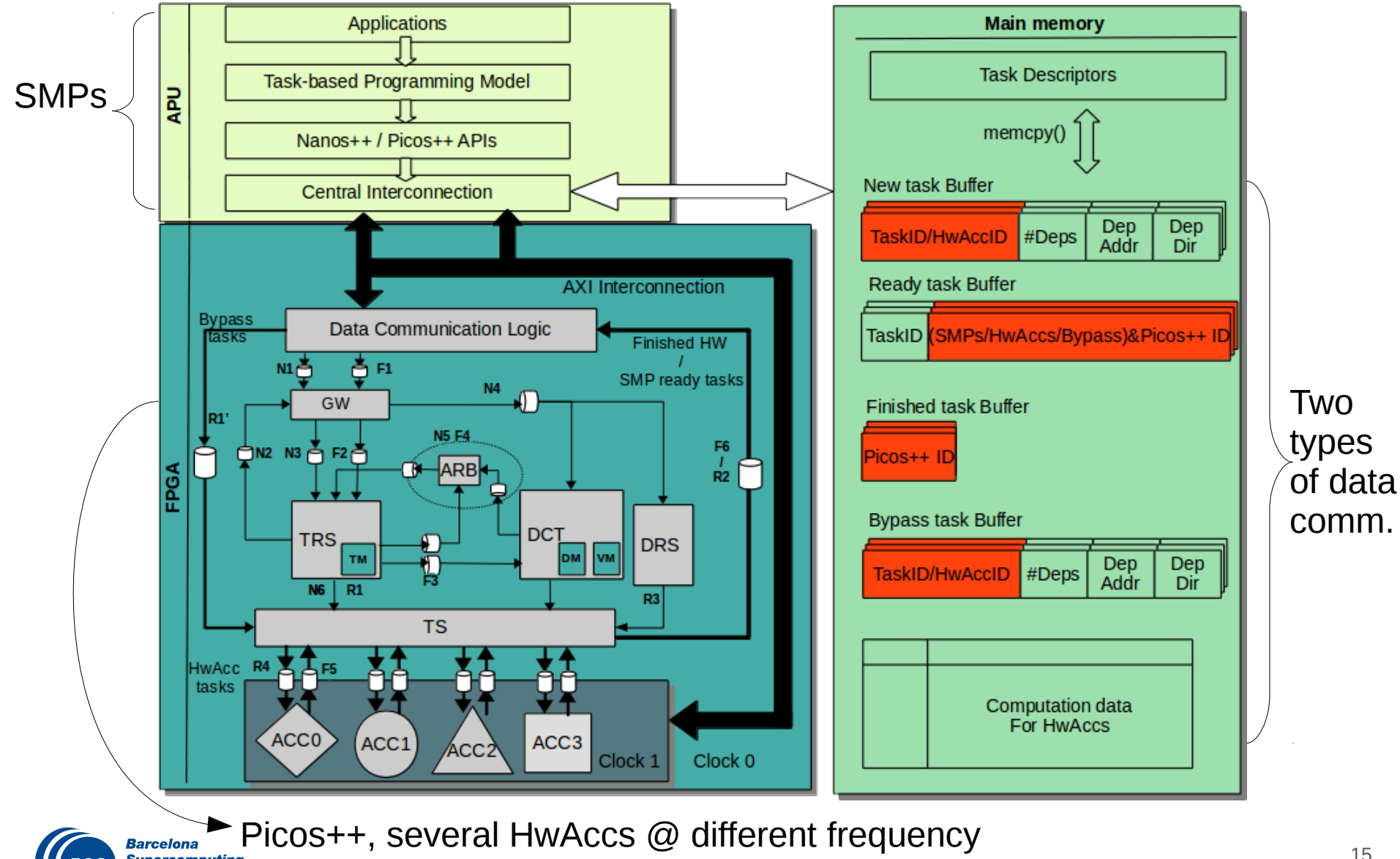
B. Dynamic algorithms

Cricality-aware task scheduler (CATS), Critical-path scheduler (CPATH), Version Scheduler from Judit Planas, etc

C. Scheduling hierarchies

Intel CARBON, Asychronous Direct Messages (ADM), Task Scheduling Unit, Programmable Task Management Unit (TMU), etc

Picos++ System



Heterogeneous Task Scheduling

> Ready tasks come from Picos or Bypass

> Hardware Registers for each device to keep information

> Device selection based on:
- Target device of the task
- Hardware Register information about the number of pending tasks for each device
- HW accelerator priority

Agenda

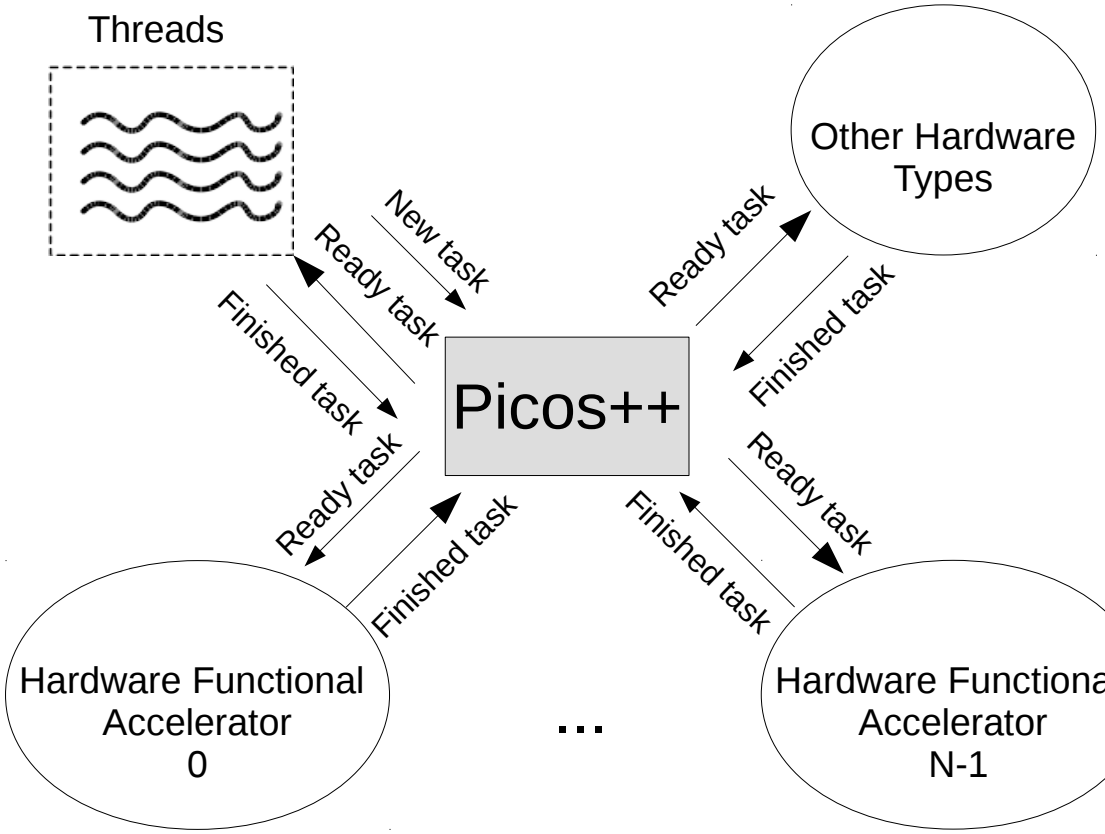
> Background, Motivation

> Picos++ accelerated hardware runtime

> Hardware Heterogeneous task scheduling

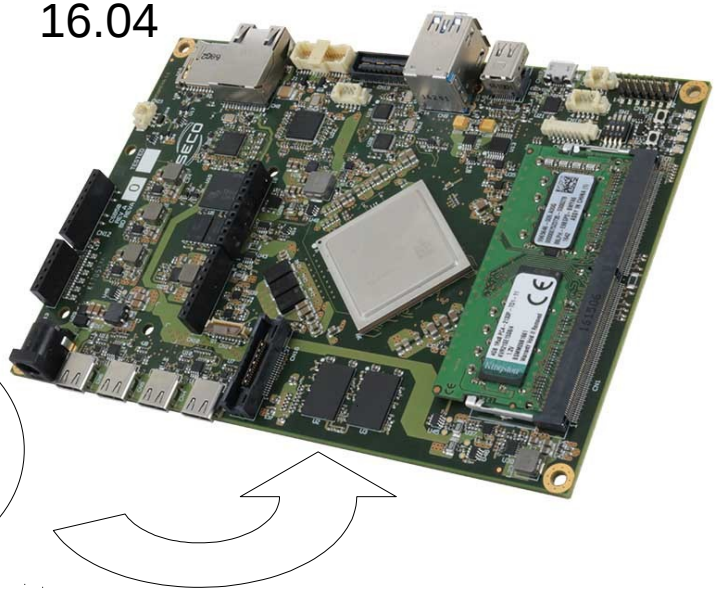
> Results

Experimental Setup



A Xilinx Ultrascale+ MPSoC:

- 4 ARM cores@ 1.1GHz, and a FPGA
- HW registers for power samples
- OmpSs up running in Linux 16.04

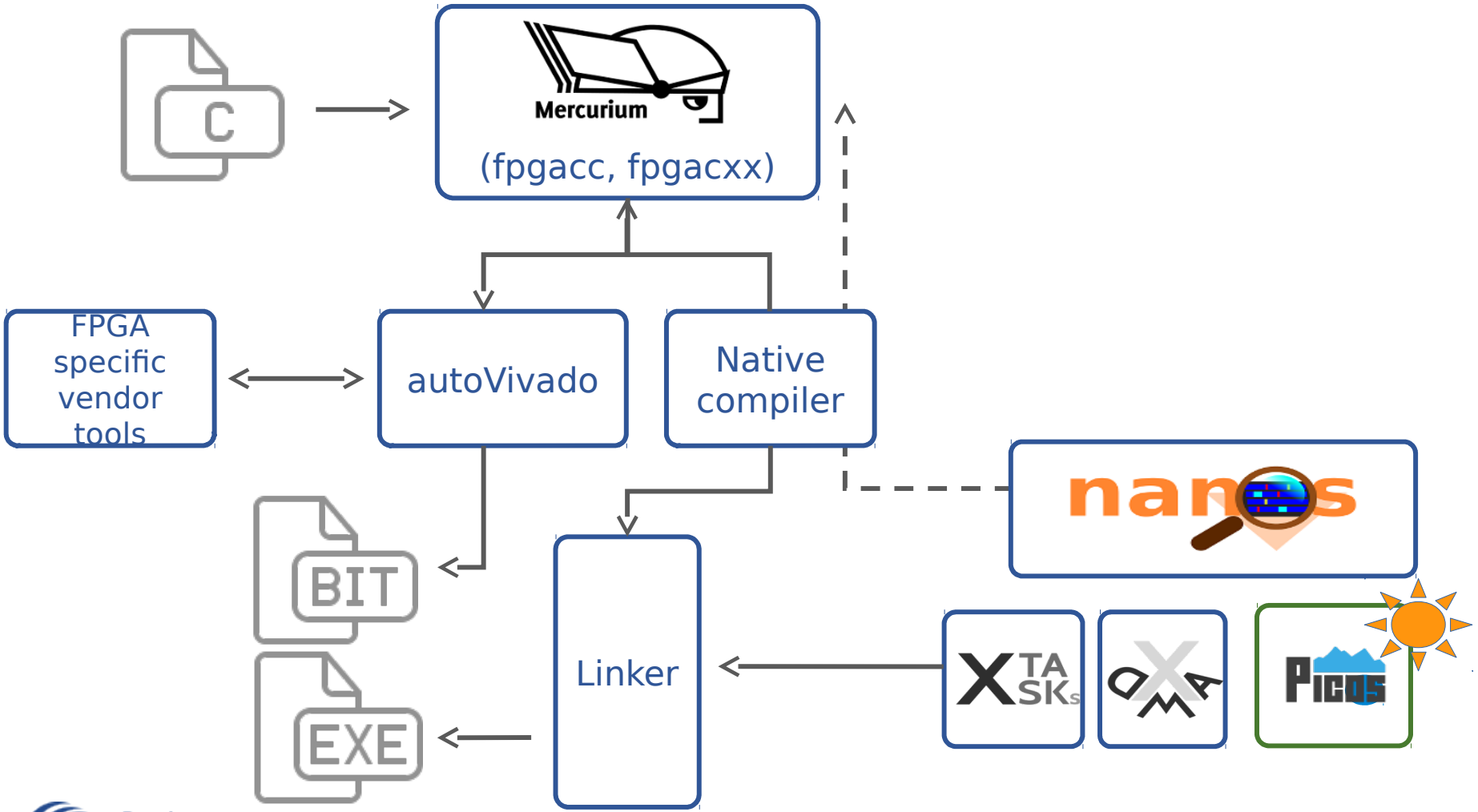


The SW-only runtime@1.1GHz is supported by OmpSs @ FPGA, Picos++ @ 100 or 200MHz

An example code of gemm function for Cholesky

```
#pragma omp target device(fpga) copy_deps onto(0) num_instances(4)
#pragma omp task inout([bs]C) in([bs]A, [bs]B)

void matmulBlock(T (*A)[bs], T (*B)[bs], T (*C)[bs]){
unsigned int i, j, k;

#pragma HLS array_partition variable=A block factor=bs/2 dim=2
#pragma HLS array_partition variable=B block factor=bs/2 dim=1
  for (i = 0; i < bs; i++) {
    for (j = 0; j < bs; j++) {
#pragma HLS pipeline II=1
      T sum = 0;
      for (k = 0; k < bs; k++) {
        sum += A[i][k] * B[k][j];
      }
      C[i][j] += sum;
    }}}

#pragma omp target device(smp) no_copy_deps  implements(matmulBlock)
#pragma omp task in([bs]A, [bs]B) inout([bs]C)

void matmulBlockSmp(T (*A)[bs], T (*B)[bs], T (*C)[bs]){
  T const alpha = 1.0; T const beta = 1.0;
  cblas_gemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
      bs, bsize, bs, alpha, a, bs, b, bs, beta, c, bs);
}
```

OmpSs@FPGA | Compilation process

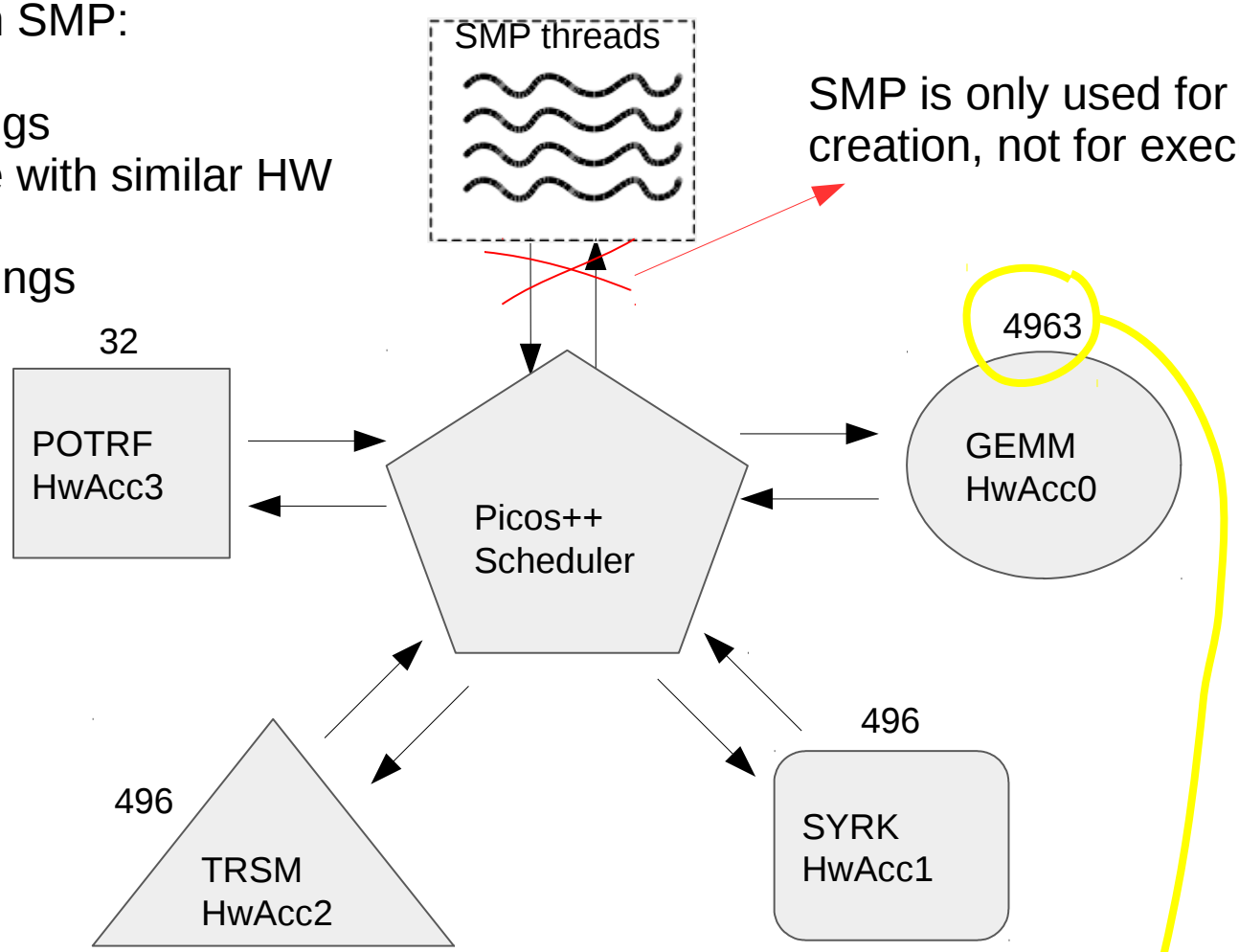


Task scheduling using 4 different HwAccs-only

Cholesky 2K, 64

Vs Seq execution in SMP:
- 1.7x speedup,
- 0% energy savings

Vs SW-only runtime with similar HW
- 1.1x speedup
- 18% energy savings

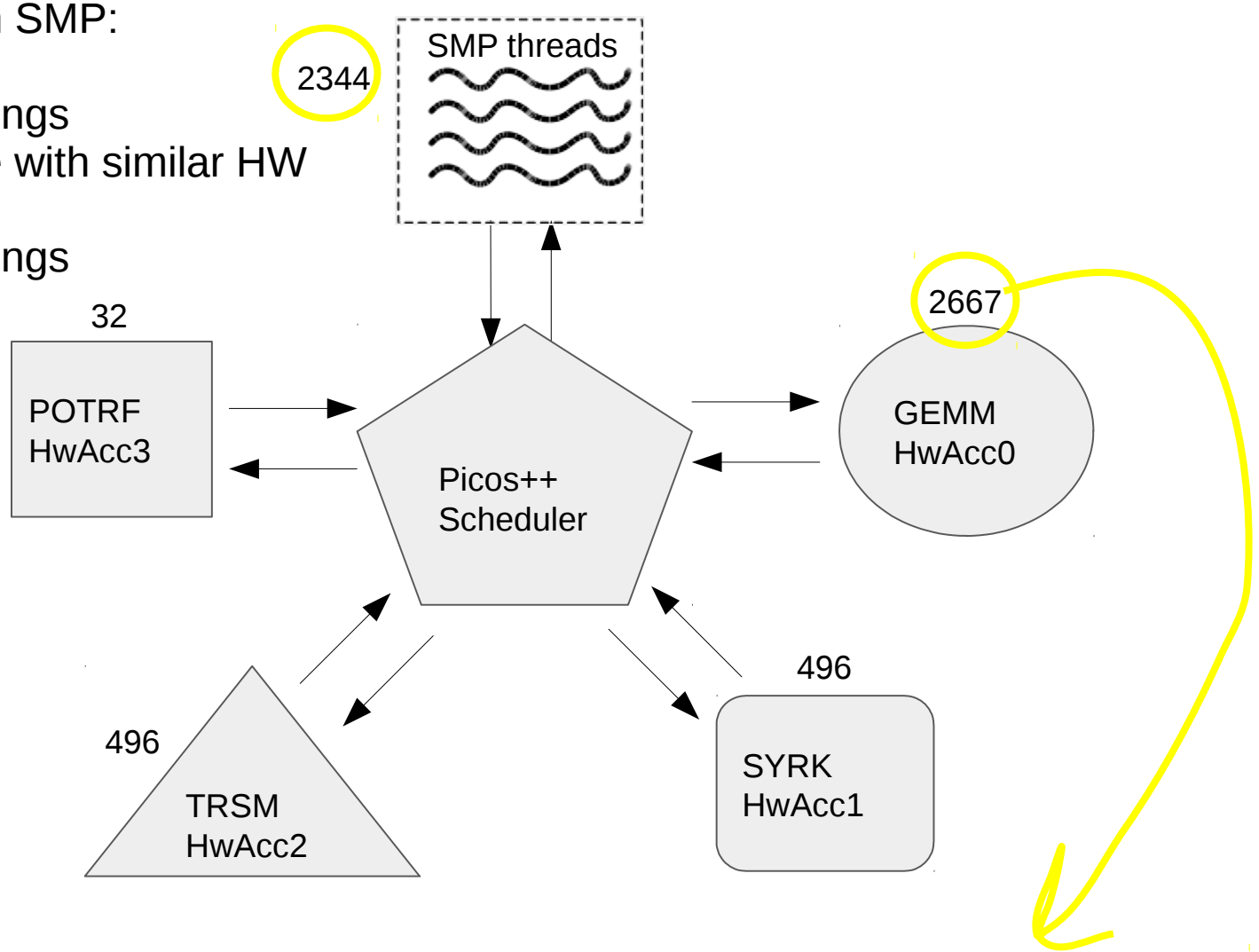


Task scheduling using 4 different HwAccs+SMP

Cholesky 2K, 64

Vs Seq execution in SMP:
- 3x speedup,
- 42% energy savings

Vs SW-only runtime with similar HW
- 1.95x speedup
- 53% energy savings



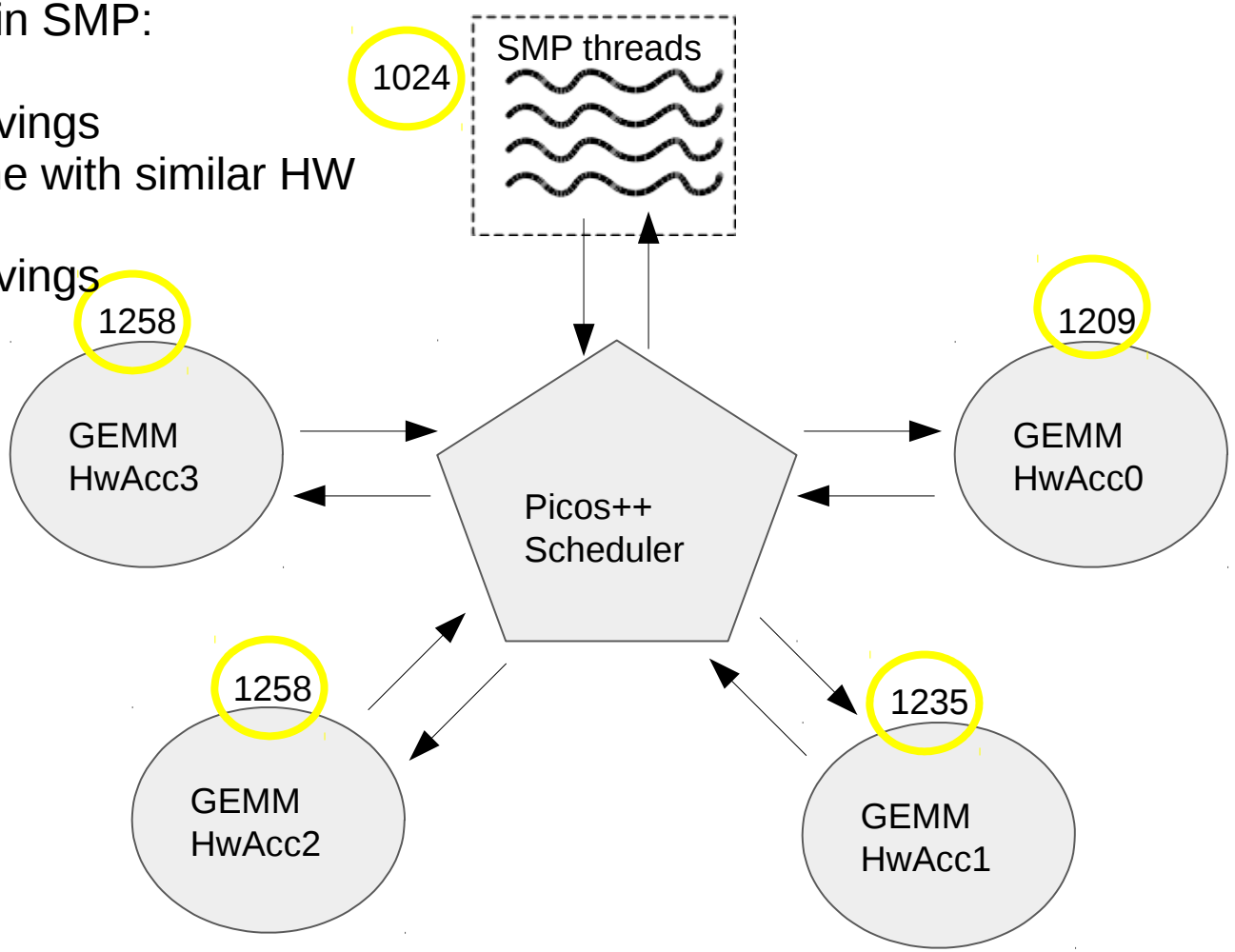
Picos++ schedules nearly half of GEMM tasks to SMP

Task scheduling using 4 same HwAccs+SMP

Cholesky 2K, 64

Vs Seq execution in SMP:
- 6.5x speedup,
- 73% energy savings

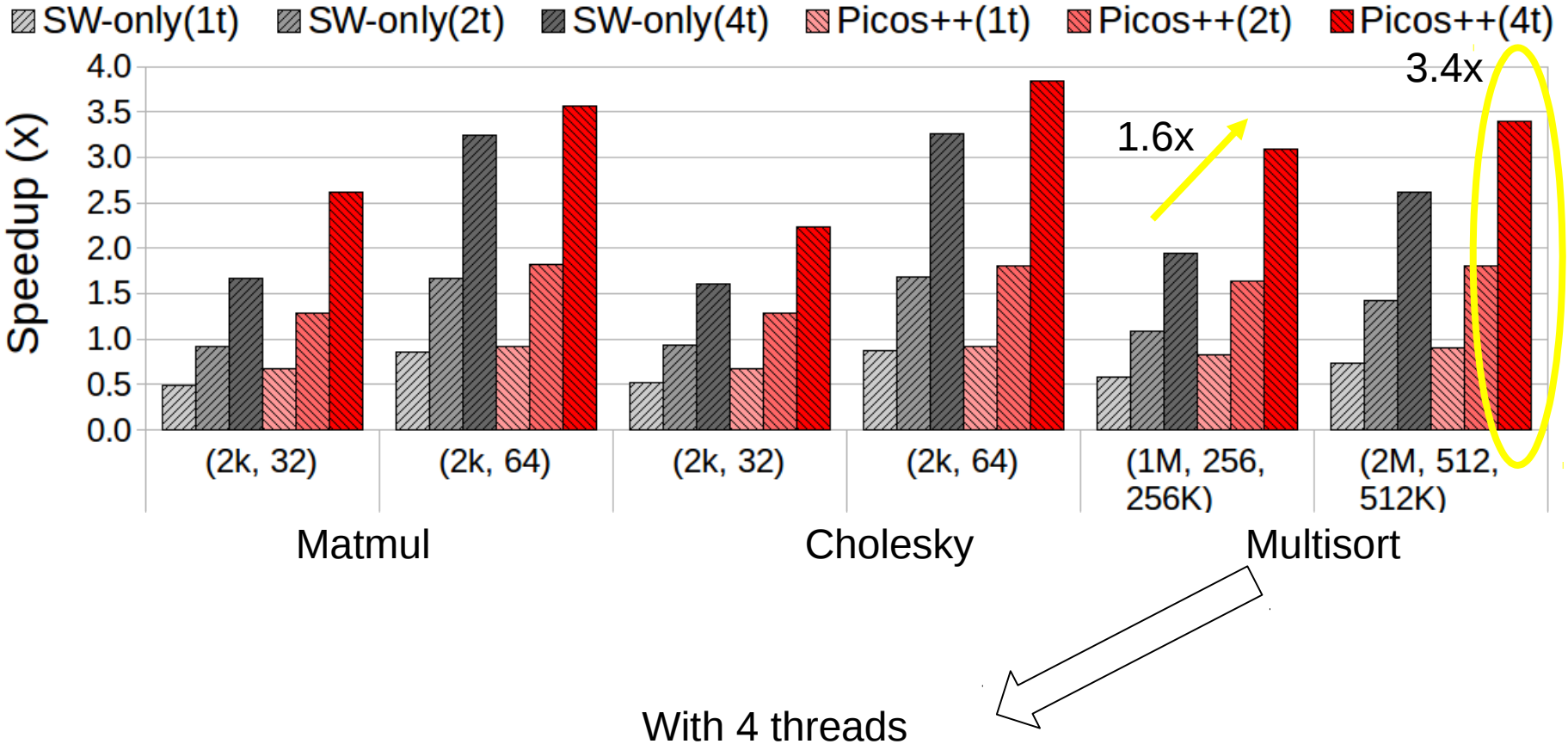
Vs SW-only runtime with similar HW
- 1.36x speedup
- 33% energy savings



Use HwAccs only for GEMM task execution, further balance the workloads

HwAcc3 has the highest priority, SMP has the lowest

Performance Impact of the Task Granularity

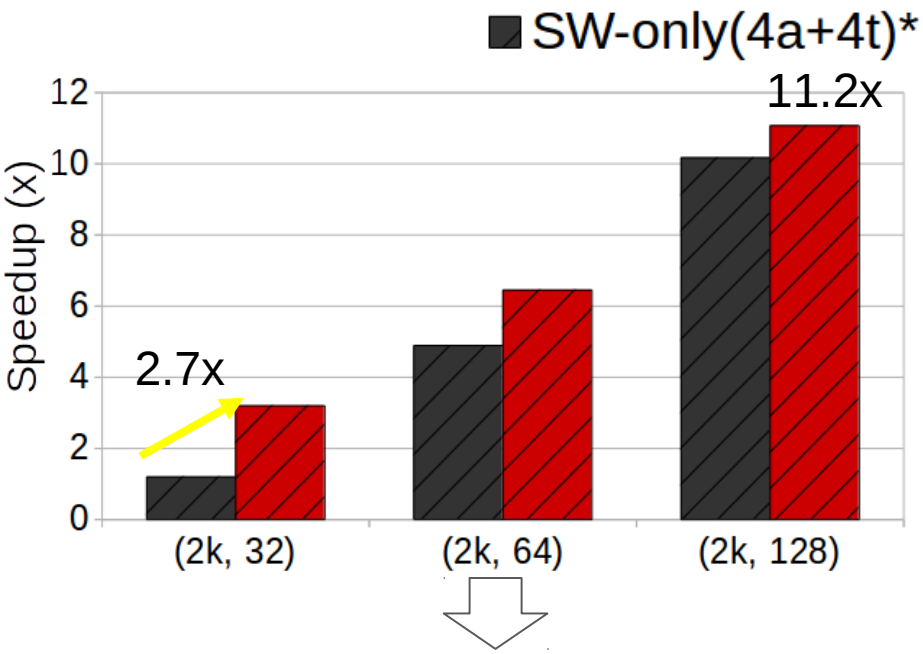


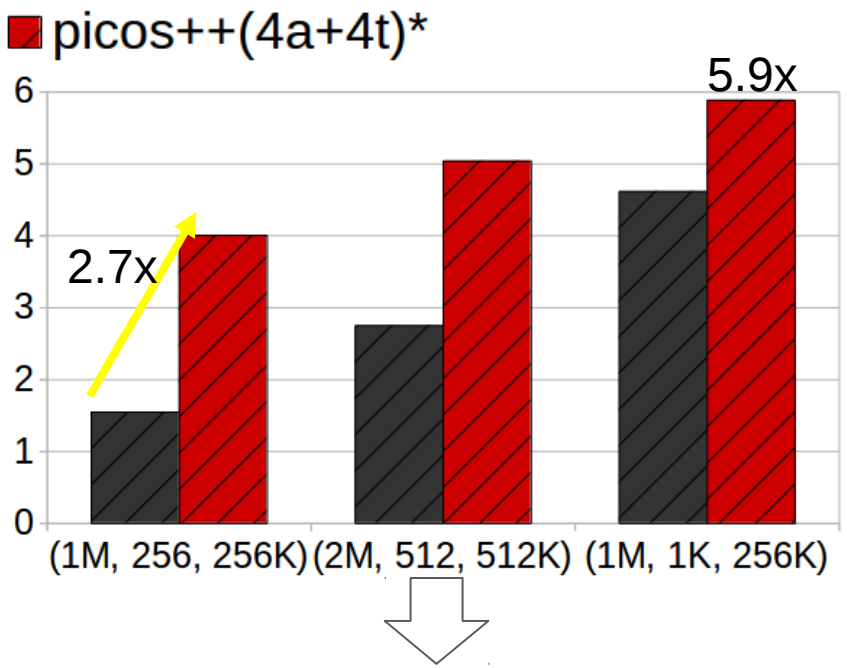
With 4 threads

Picos++ vs Seq: up to 3.4x speedup, 65% energy saving

vs SW-only: up to 1.6x speedup, 40% energy saving

Performance Impact of the Heterogeneous Task Management



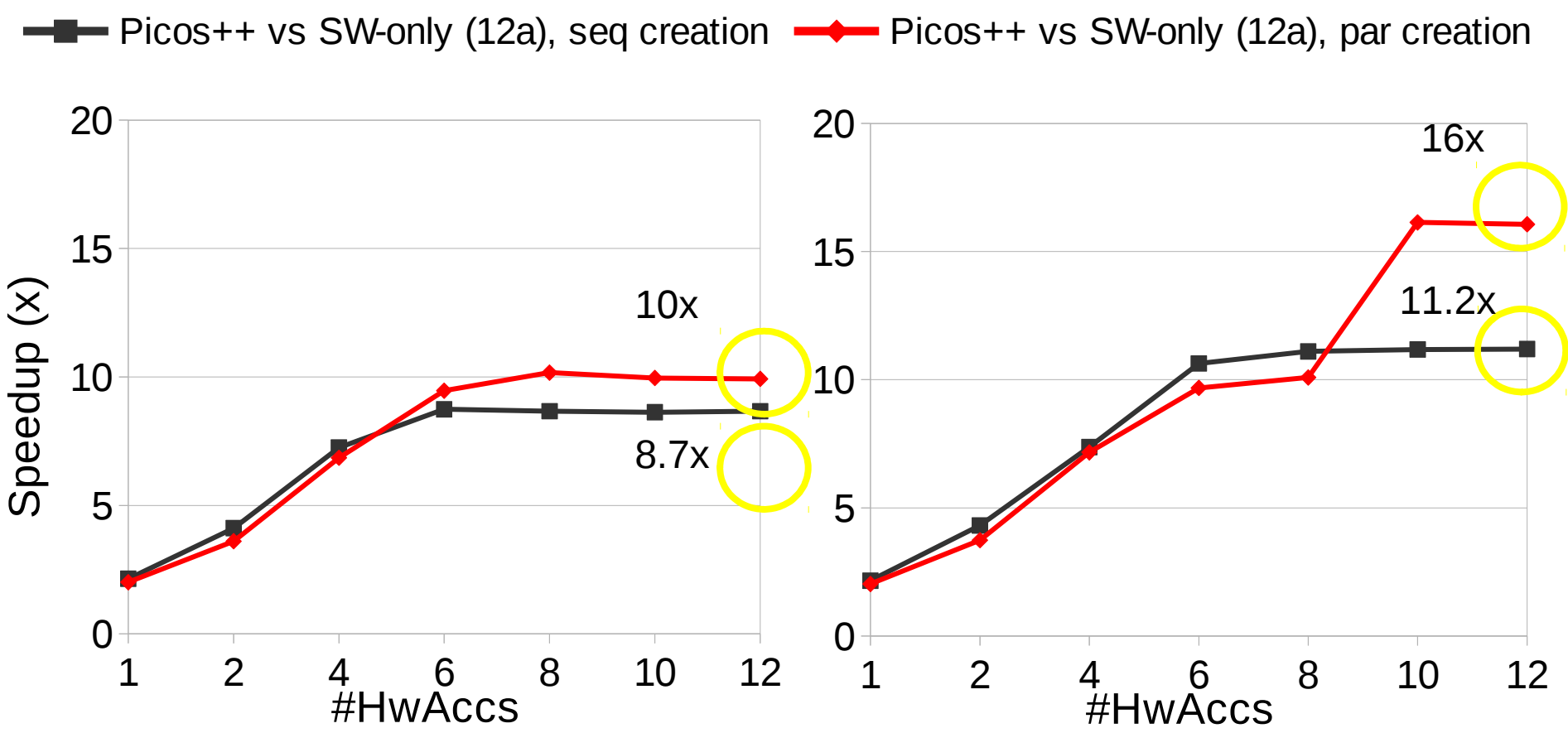


Matmul:
Vs seq: 11.2x, 70% of energy savings
Vs SW-only: 2.7x, 65% of energy savings

Multisort:
Vs seq: 5.9x, 85% of energy savings
Vs SW-only: 2.7x, 69% of energy savings

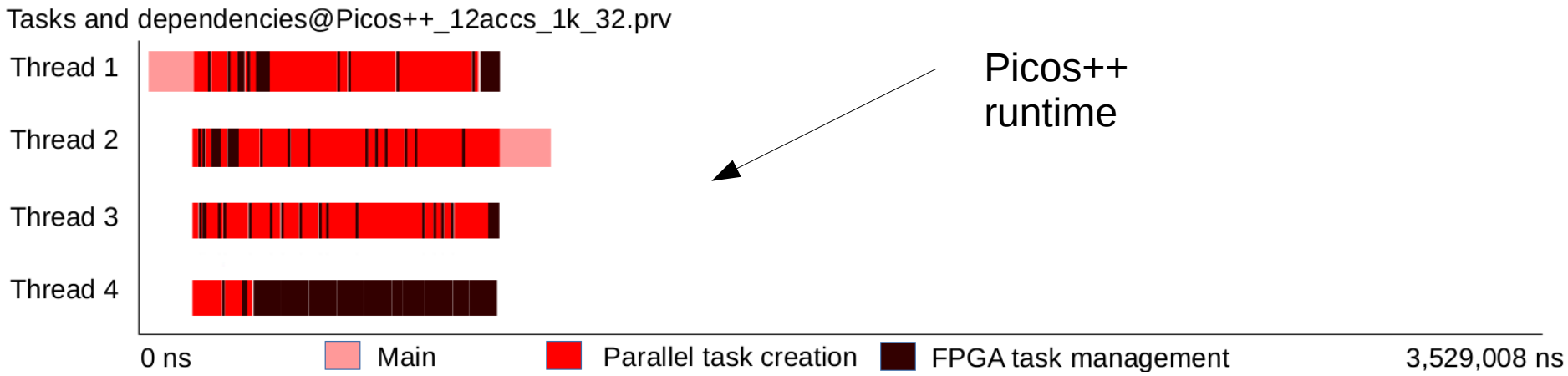
with SMP+4 HwAccs*

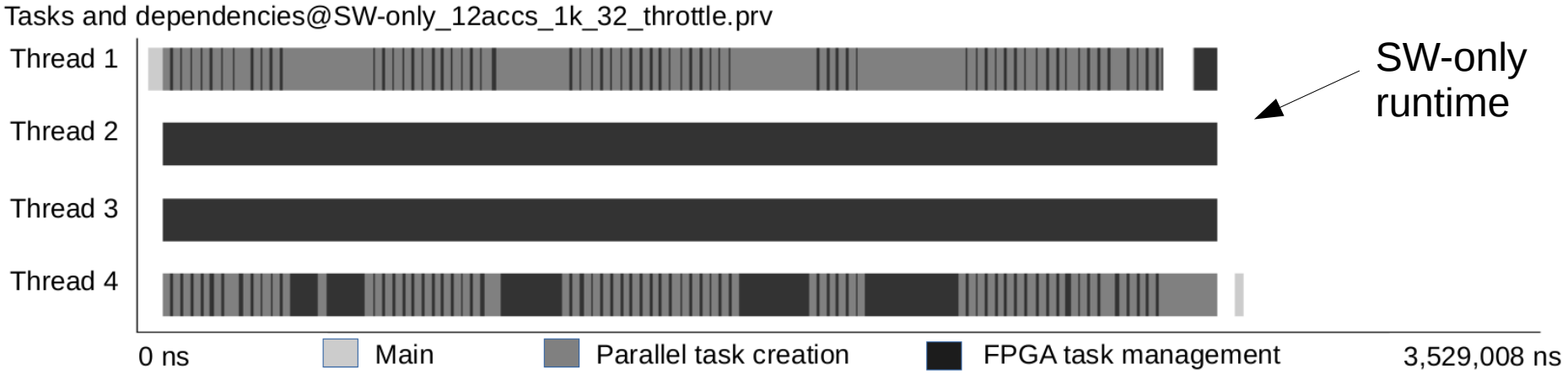
Scaling Up the Number of HwAccs



Matmul (2k, 32) with Picos++ at 100 or 200 MHz

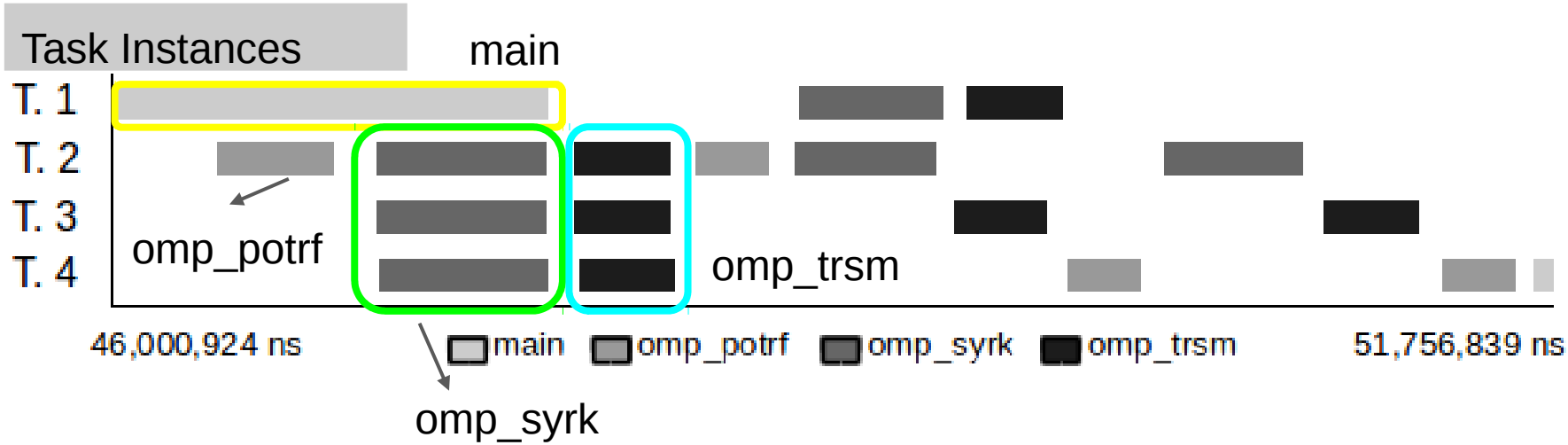
Paraver Trace of Matmul(2K, 32) with 12 HwAccs

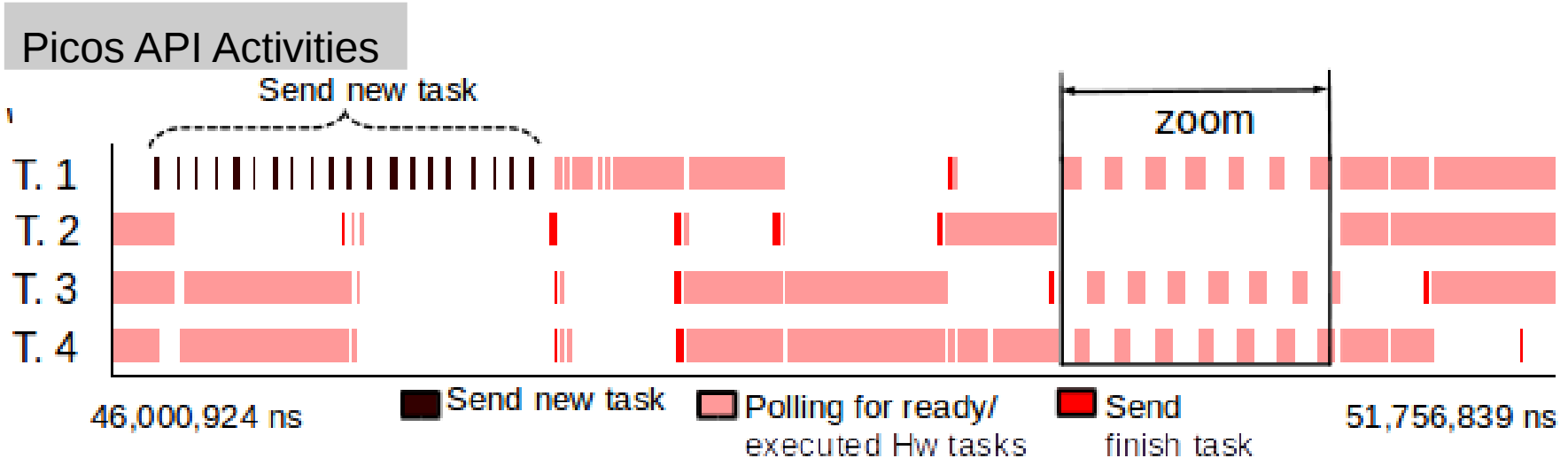




By using Picos++ runtime, the cost of FPGA task management is significantly smaller

Paraver Trace of Cholesky@SMP+4GEMM Accs





Potential Time Saving During Cholesky Execution

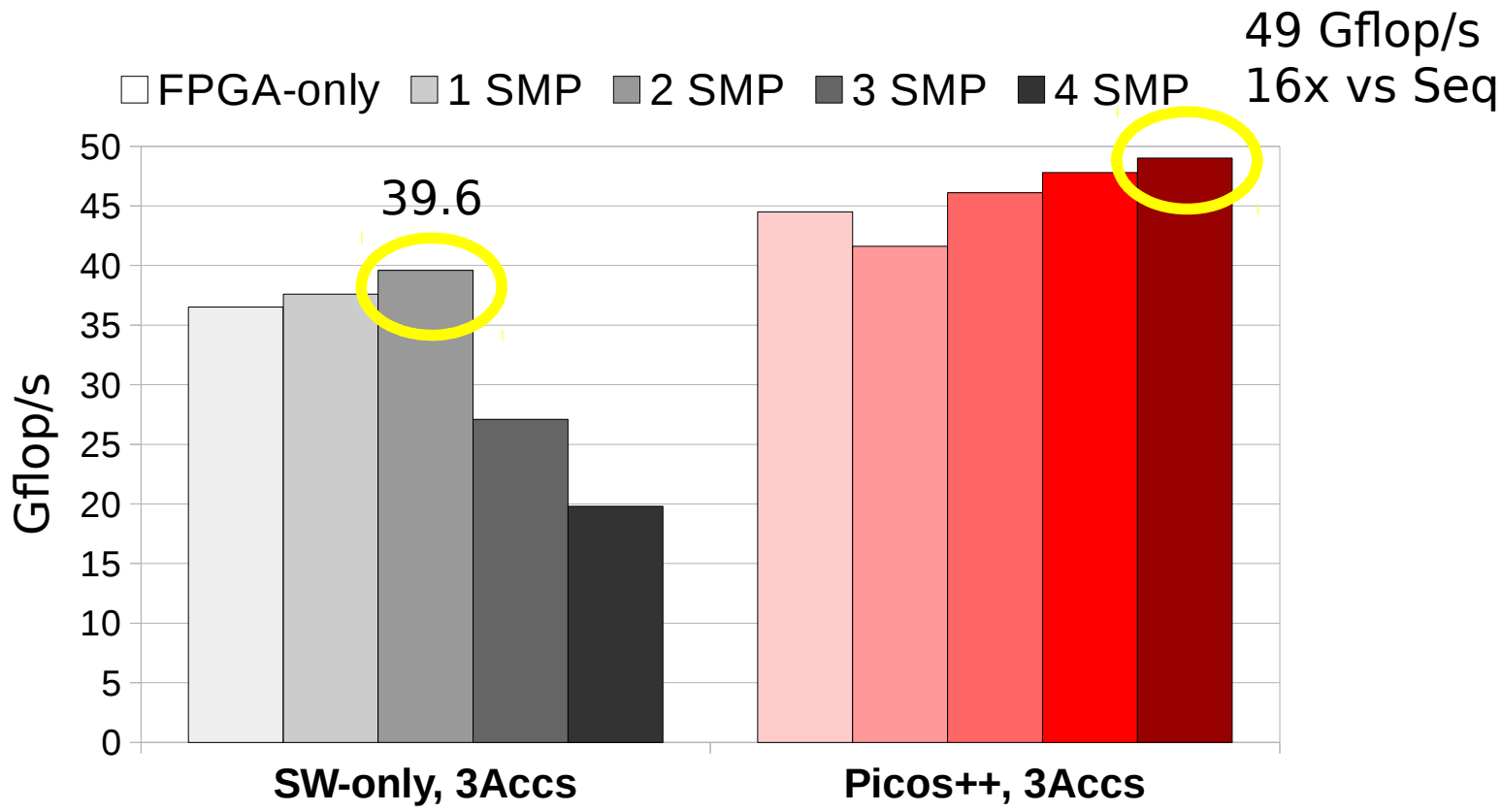
TABLE 7: The useful time consumption in a 2K*2K trace

Category	Name	T1/Hw0*	T2/Hw1*	T3/Hw2*	T4/Hw3*
Measured Useful Time in the trace					
Tasks	main	97%	1.24%	0	0
	omp_potrf	0.14%	0.78%	0.76%	0.75%
	omp_syrk	0.28%	15.32%	14.47%	15.00%
	omp_trsm	0.26%	20.79%	20.79%	22.64%
	omp_gemm*	13.34%	56.52%	56.52%	60.89%
Picos++APIs	New task	33.00%	0	0	0
	Polling ready	0.81%	19.19%	19.16%	18.49%
	Successful ready	0.05%	5.89%	5.52%	4.78%
	Finish task	0.04%	3.08%	3.07%	3.06%
Potential time candidate for reduction					
HwAccs	omp_gemm*	86.66%	43.48%	43.48%	39.11%
Threads	Upper bound	0	48.63%	54.75%	53.11%

Clock stopping, frequency scaling, sleep/wake-up

GFLOPS: Picos++ with HPC

Matmul (2K, 128), Picos++@100MHz, HwAccs @ 300MHz



With **more** resources, Picos++ gains **more** GFLOPS

Up to **49 GFLOPS**, **24% faster** than SW-only

5.74 watts average power consumption

GFLOPS: Picos++ with HPC

Matmul (2K, 128), Picos++ @ 100MHz, HwAccs @ 300MHz

Name	Gflops per watt	Num.Threads, Frequency
Intel(R) Core(TM) i5-3470	0.51	4t, 3GHz
Intel(R) Xeon(R) CPU E5-2020 V2	4.14	24t, 2.1GHz
Intel(R) Core(TM) i7-4600U CPU	4.75	4t, 2.1GHz
Picos++ with 3 blocksize 128 Accs	8.53	4t, 1.1GHz

Conclusion

- Fine-grained parallelism offers a lot of opportunities for desirable performance at a low energy cost
- HW task-dependence manager and heterogeneous task scheduler are **fast, energy efficient, and general purpose**
- The more hardware resources, the greater the impact of Picos hardware
- Picos++ is compatible for task-based programming model runtimes as gomp for OpenMP

Publication List

Jaume Bosch, **Xubin Tan**, Jaume Bosch, Antonio Filgueras, Miquel Vidal, Marc Mateu, Daniel Jiménez-González, Carlos Álvarez, Xavier Martorell, Eduard Ayguadé, Jesus Labarta. *"Application Acceleration on FPGAs with OmPss@FPGA"* **in FPT 2018**.

Xubin Tan, Jaume Bosch, Carlos Álvarez, Daniel Jiménez-González, Eduard Ayguadé, Mateo Valero. *"General Purpose Task-Dependence Management Hardware for Task-based Dataflow Programming Models"* **in IPDPS'17**.

Jaume Bosch, **Xubin Tan**, Carlos Álvarez, Daniel Jiménez-González, Eduard Ayguadé, Mateo Valero. *"Characterizing and Improving the Performance of Many-Core Task-Based Parallel Programming Runtimes"* **in IPDPS Workshop IPDRM'17**.

Xubin Tan, Jaume Bosch, Carlos Álvarez, Daniel Jiménez-González, Eduard Ayguadé, Mateo Valero. *"Performance Analysis of a Hardware Accelerator of Dependence Management for Task-based Dataflow Programming models"* **in IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'16**).

Acknowledgement

This work has been financially supported by the Spanish Government through Programa Severo Ochoa (SEV-2015-0493),
by the Spanish Ministry of Science and Technology through TIN2015-65316-P project,
by the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272), by the European Research
Council RoMoL Grant Agreement number 321253 and by the OmpSs on Android and Hardware support for runtime Project
Cooperation Agreement with LG Electronics.
We also thank the Xilinx University Program for its hardware and software donations

Hardware Hetergeneous Task Scheduling for Task-based Programming Models

Xubin Tan

OpenMPCon 2018

Advisors: Carlos Álvarez, Daniel Jiménez-González

Backup Slides

HW and Power cost in a Xilinx Ultrascale+ MPSoC

TABLE 3: Hardware Resource and Power Consumption

Name	FPGA resource				Power
	BRAM18Kb	DSP48E	FFs	LUTs	Watts
XCZU9EG	1824	2520	548160	274080	
Picos++	87/5.0%	0	5478/1.0%	9793/4.0%	0.07
Comm. Logic	2/0%	0	3421/0.6%	4244/1.0%	0.03
APU					**Watts**
4 ARM Cortex-A53s	-	-	-	-	1.4

Backup Slides

Task size in Ultrascale+ executions

TABLE 1: The characteristics of real benchmarks

Name	Configs	#Tasks	Seq time(us)	Latency(us)
Matmul	(2K, 32)	262144	6820850	26
	(2K, 64)	32768	5463956	167
	(2K, 128)	4096	5435528	1327
Cholesky	(2K, 32)	45760	1232664	27
	(2K, 64)	5984	1087485	182
Multisort	(1M, 256, 256k)	9565	395594	414
	(2M, 512, 512K)	9565	832882	871
	(1M, 1K, 512K)	2397	407648	1701

Backup Slides

Task size in Ultrascale+ executions

TABLE 2: Characteristics of HwAccs in XCZU9EG-FFVC900

Name	HWACCs				Latency
	B_18Kb	DSP48E	FFs	LUTs	us
fgemm32	68/3.7%	160/6.4%	19771/3.6%	15559/5.7%	27
fsyrk32	36/2.0%	160/6.4%	19822/3.6%	16149/5.9%	63
ftrsm32	36/2.0%	104/4.1%	11482/2.1%	10875/4.0%	67
fpotrf32	10/0.6%	22/0.9%	3487/0.6%	3302/1.2%	168
fgemm64	74/4.1%	160/6.4%	23887/4.4%	30032/11.0%	126
fsyrk64	42/2.3%	160/6.4%	23849/4.4%	30727/11.2%	270
ftrsm64	42/2.3%	250/9.9%	28734/5.2%	25753/9.4%	314
fpotrf64	28/1.5%	22/0.9%	3514/0.6%	3350/1.2%	981
fmatmul32	68/3.7%	162/6.4%	20106/3.7%	14671/5.4%	27
fmatmul64	138/7.6%	322/12.8%	38770/7.1%	27668/10.9%	105
fmatmul128	287/15.7%	642/25.5%	76147/13.9%	54462/19.9%	415
sort256	68/3.7%	0	20106/3.7%	14671/5.5%	27
sort512	138/7.6%	0	38770/7.1%	27668/10.1%	105
sort1024	159/8.7%	0	47034/8.6%	71124/26.0%	497