ENCORE

ENabling technologies for a programmable many-CORE

Alex Ramirez, BSC

ENCORE consortium

ROYAL INSTITUTE
OF TECHNOLOGY

ISRAEL INSTITUTE
OF TECHNOLOGY

- Funded under FP7 Objective ICT 2009.3.6 - Computing Systems
  - 3-year STREP project (March 2010 - February 2013)

encore

Key issues – Programmability

Until now, we had **faster** processors

1x ⇨ 2x

Now, we have **more** processors

1x 1x
1x 1x

*But **more** doesn't necessarily mean **better**!*

If we want to run with more processors,

Original Code

```
{
   int tid=0;
   int tids[nThreads];
   for(i=0; i<nThreads; i++) {
      tids[i]=i;
      bs_thread(&tids[i]);
   }
}

void bs_thread(void *tid_ptr)
{
...
}
```

we need to re-write the application

Re-written Code

```
pthread_t _M4_threadsTable[MAX_THREADS];
pthread_mutexattr_t _M4_normalMutexAttr;
int _M4_numThreads = MAX_THREADS;

int tids[nThreads];
for(i=0; i<nThreads; i++) {
   tids[i]=i;
   {
      int _M4_i;
      for ( _M4_i = 0; _M4_i < MAX
         if ( _M4_threadsTable[
      }
      pthread_create(&
                  NU                                   (void *)&tids[i]);
   };
}

   {

               _i < MAX_THREADS;_M4_i++) {
               _M4_i] == -1)    break;
               n( _M4_threadsTable[_M4_i], &_M4_ret);

  bs_thread(void *tid_ptr)
{
...
}
```

more code = more time =
more €€ and slower time-to-market!

… or find an alternative solution: a **better** programming model

Re-written Code using ENCORE (OmpSs)

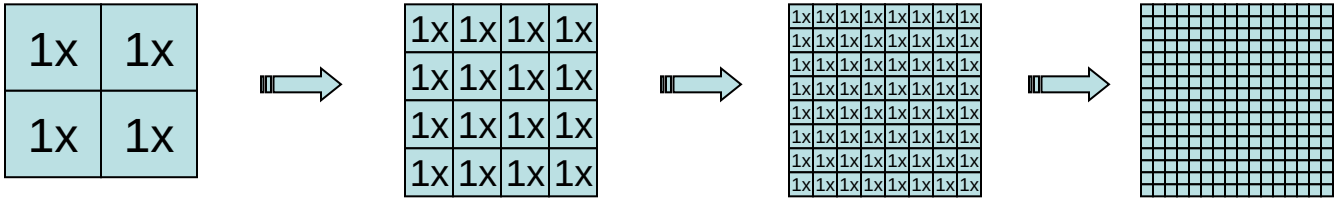
```
{
   int tid=0;
   int tids[nThreads];
   for(i=0; i<nThreads; i++) {
      tids[i]=i;
      bs_thread(&tids[i]);
   }
}

#pragma omp task input(tid_ptr)
void bs_thread(void *tid_ptr)
{
...
}
```

ENCORE Programming Model is:

- Easy to use
- Fewer lines of code
- Same (or better performance)

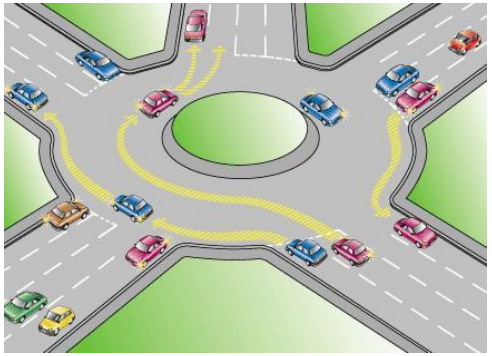
Key issues – Multicore Architecture

- Will the application run faster on a multi-core processor?



We also need to organize those 256+ cores in the right way

vs.



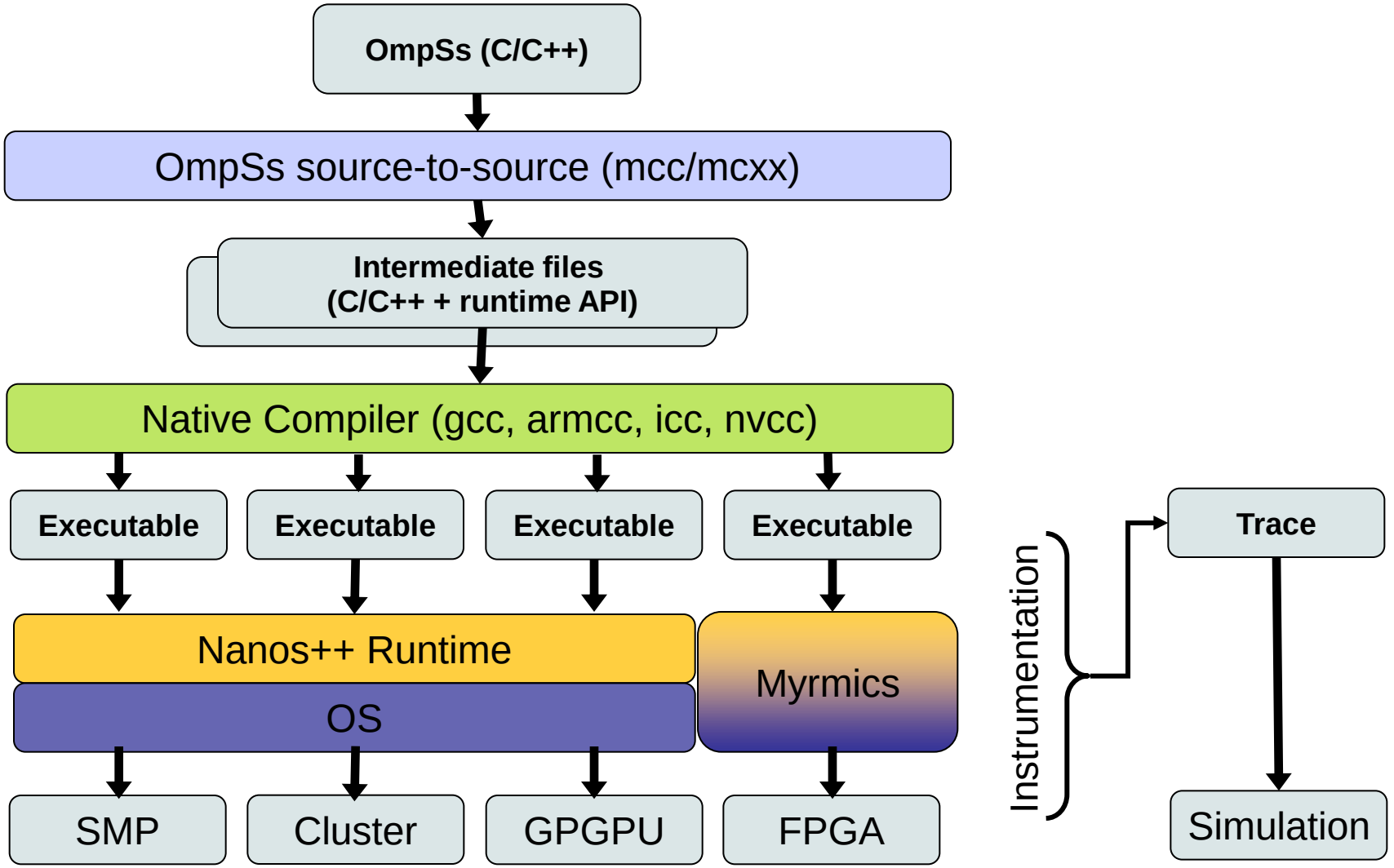
Project Objectives

- **Objective 1**: To define an easy to use parallel programming model that offers code portability across several parallel architectures
  - Extended OpenMP 3.0 specification

- **Objective 2**: To develop a runtime management system to dynamically detect, manage, and exploit parallelism, data locality, and resources across several parallel architectures
  - Fine-tuned runtime libraries implementing the runtime calls described in Objective 1

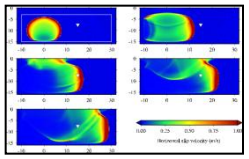
- **Objective 3**: To provide the right kind of hardware support for the parallel programming and runtime environment that ensures scalability, performance, and cost-efficiency
  - Multicore architecture, fine-tuned to support scalable parallel programming using an efficient runtime environment

The ENCORE Toolchain



ENCORE Applications

- Goal: porting applications to OmpSs
  - HPC kernels and applications
    - Fixedgrid, GROMACS…
  - Embedded and Consumer Applications
    - Raytracing, md5…
  - Real Time
    - Micro- and evaluation- benchmarks (e.g. H264)
  - MapReduce
    - MapReduce runtime



- Expected improvements
  - Better scalability of the applications
  - Improving the portability of applications
  - Demonstrating the ‘ease-of-use’ of the OmpSs

OmpSs Programming Model

- Integration of StarSs features to OpenMP 3.0 standard

- From OpenMP 3.0
  - Loop parallelism, iteration scheduling and tasking

- From StarSs
  - Expressiveness
    - Data direction (input/output hints)
    - Dependency checking at runtime and automatic data movement
  - Possibility to run on various platforms
    - SMPSs, CellSs, ClusterSs, GPUSs

OmpSs: An example

Imperative code

```
for (i=0; i<height; i+=16)
  for (j=0; j<width; j+=16)
    mb_decode(&frame[i][j]);
```

OmpSs

Annotated code

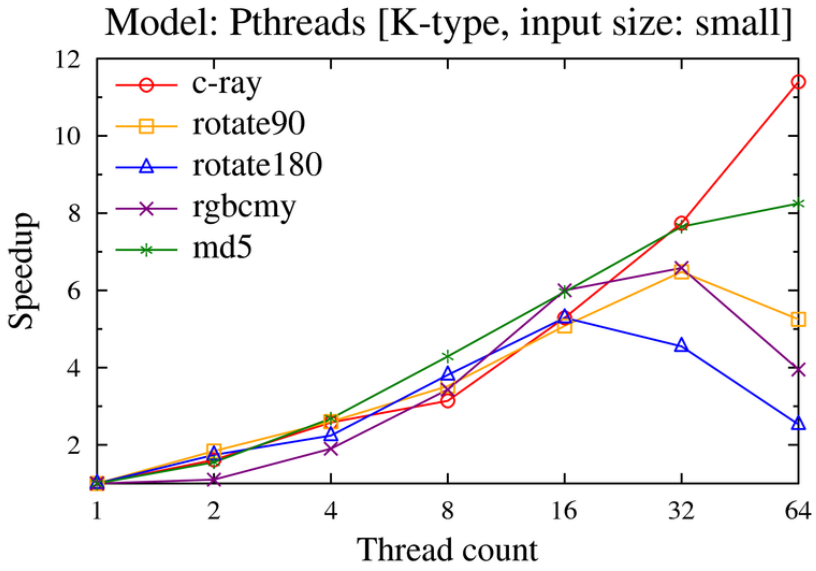
```
for (i=0; i<height; i+=16)
  for (j=0; j<width; j+=16)
#pragma omp target device(encore)\
        copy_deps \
        require (…)
#pragma omp task inout(frame[i][j]) \
  deadline(deadline_frame_1 + frame_no * 0.02) \
  onerror(OMP_ERR_DEADLINE_EXPIRED:OMP_SKIP)
    mb_decode(&frame[i][j]);
```

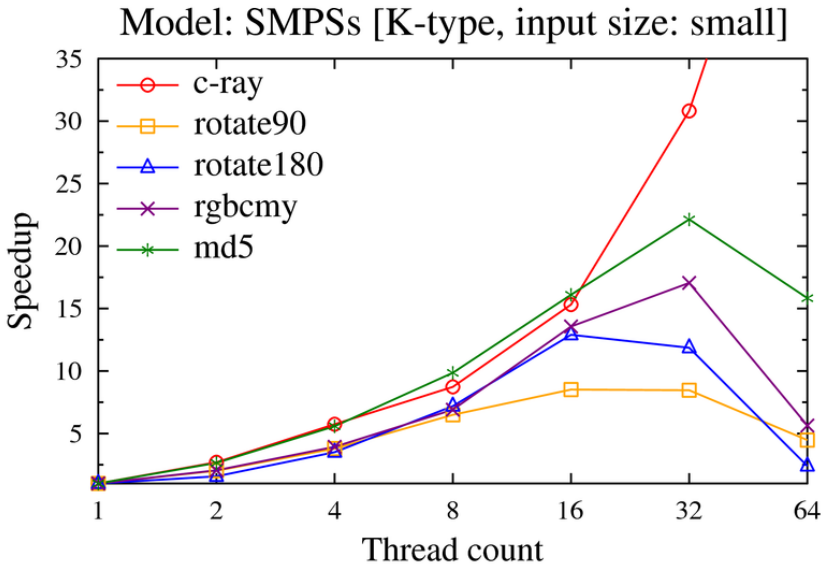
Mercurium

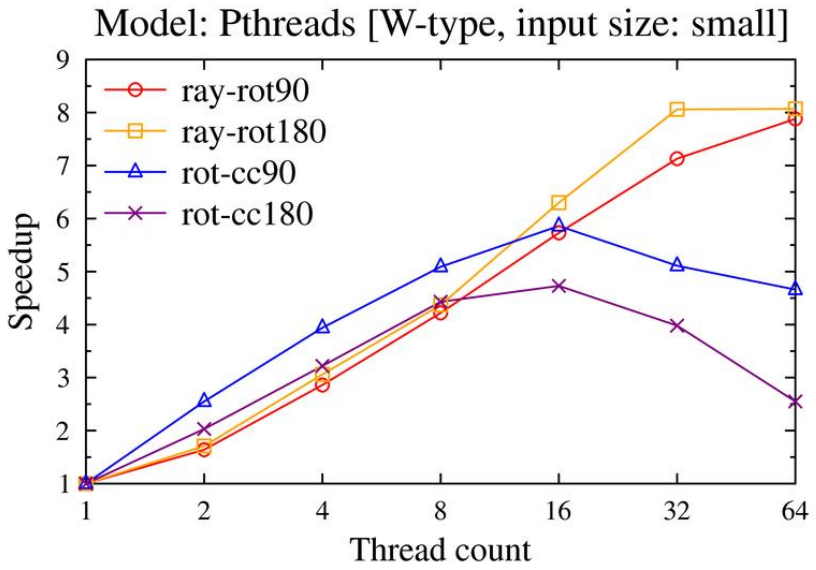
ENCORE application

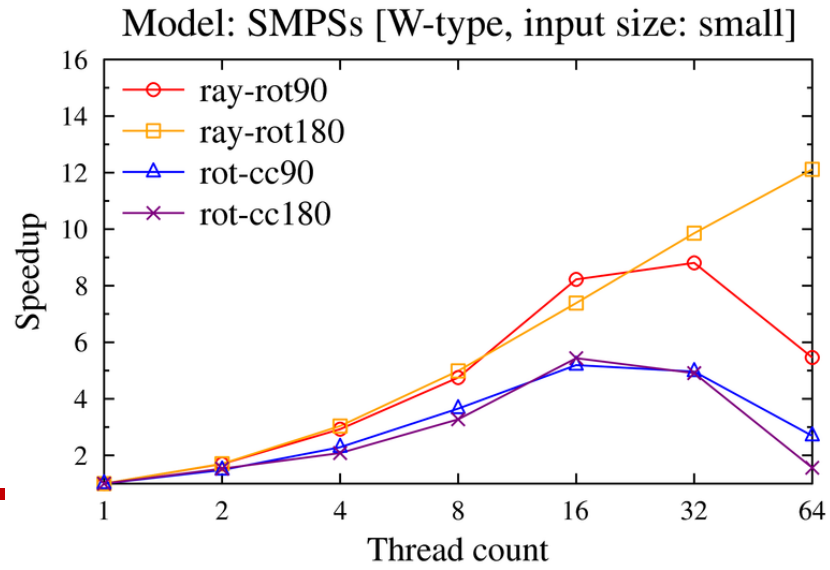
```
for (i=0; i<n; i+=16)
  for (j=0; j<n; j+=16) {
    wd = nanos_create_wd(...,
         dependences_info,
         copies_info );

    nanos_submit(wd);
 }
```

OmpSs: Embedded and Consumer applications
Initial results

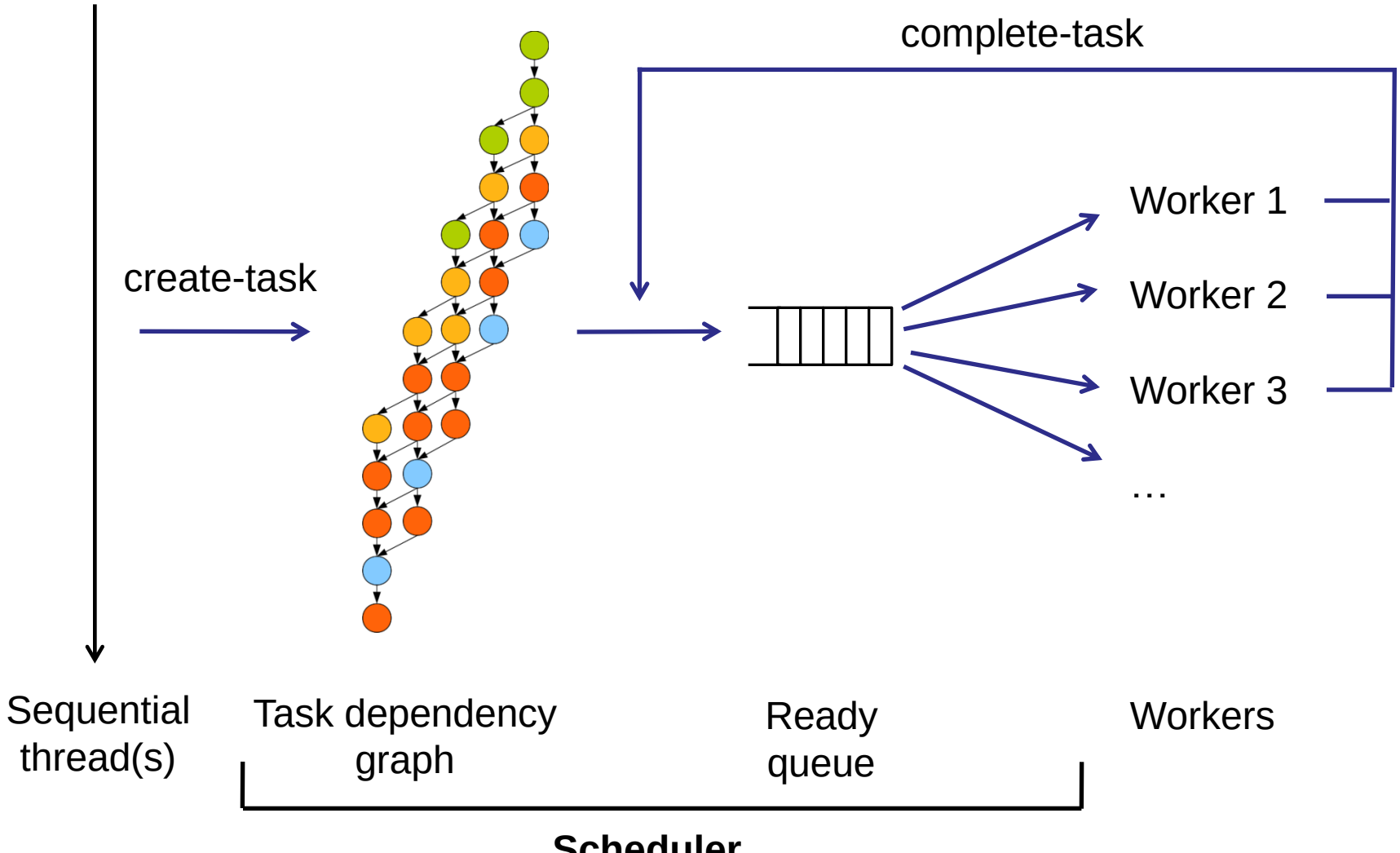








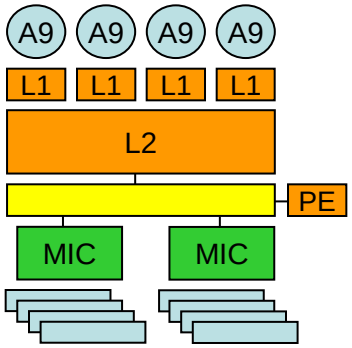
Runtime: Scheduling



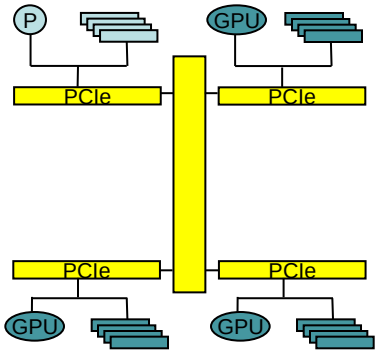
Architecture

- ENCORE targets several architectures

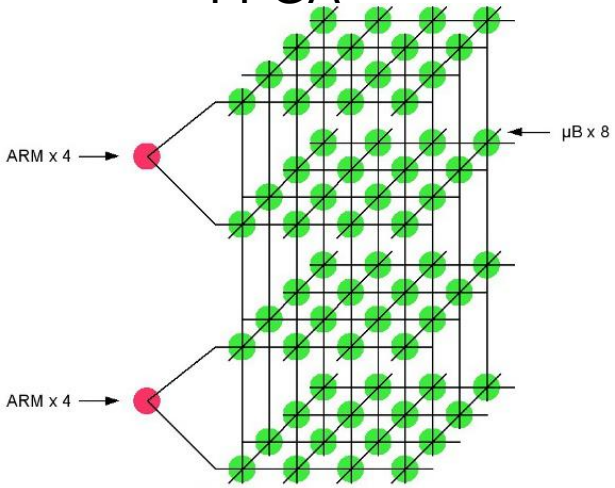
SMP



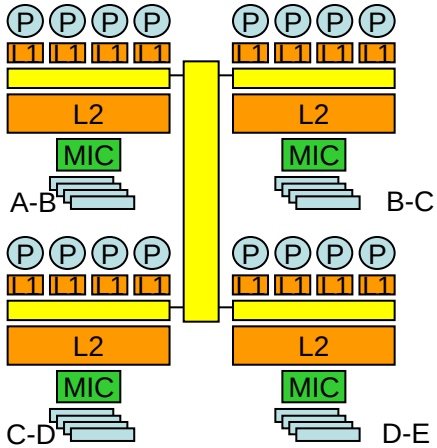
GPGPU



FPGA



cc-NUMA/Cluster



Expected Improvements

- Programming model
  - Support for C++
  - More expresiveness to the programmer
    - e.g. Multi-dimensional regions, QoS, dynamic memory
- Runtime
  - Scheduling improvements
    - e.g. resource-aware scheduling, real-time, optimizations for SIMT
  - Support for more platforms
    - FPGA, DSM
- Architecture
  - HW support
    - e.g. task management, explicit memory management
  - Prototyping – FPGA, virtual many-core platform

Long Term Impact

- Enable exploitation of many-core heterogeneous systems
  - Increased performance and power-efficiency
  - Improved scalability
- Code portability across platforms
  - Accelerated system development and production
  - Shorter time to market
- Extended OpenMP specification

Beyond ENCORE

- Parallel programming
  - Debug and performance analysis of parallel systems
  - Runtime management
- Locality management
  - Work at object granularity vs. fixed-size blocks
  - Dealing with indirect accesses
    - Possibly with architecture support
- Scalable system architecture
  - Beyond the one-chip many-core (10K-core systems)
  - Design of heterogeneous processors
    - Focus on performance / power-efficiency for a particular application domain
  - Coping with unreliable hardware + out-of-order delivery
- Simulation of large-scale systems
  - Scalability issues only detected on the full-scale system