WCET analysis of a parallel 3D multigrid solver executed on the MERASA multi-core

Christine Rochange, Armelle Bonenfant, Pascal Sainrat (U. Toulouse)
Mike Gerdes, Julian Wolf, Theo Ungerer (U. Augsburg)
Zlatko Petrov, Frantisek Mikulu (Honeywell)

Workshop on WCET Analysis, July 2010

Motivation

- **Hardware trends**
  - common **requirements**
    - low energy consumption
    - high integration / low cost
    - high performance
  - architectural solution: **multicore processors**

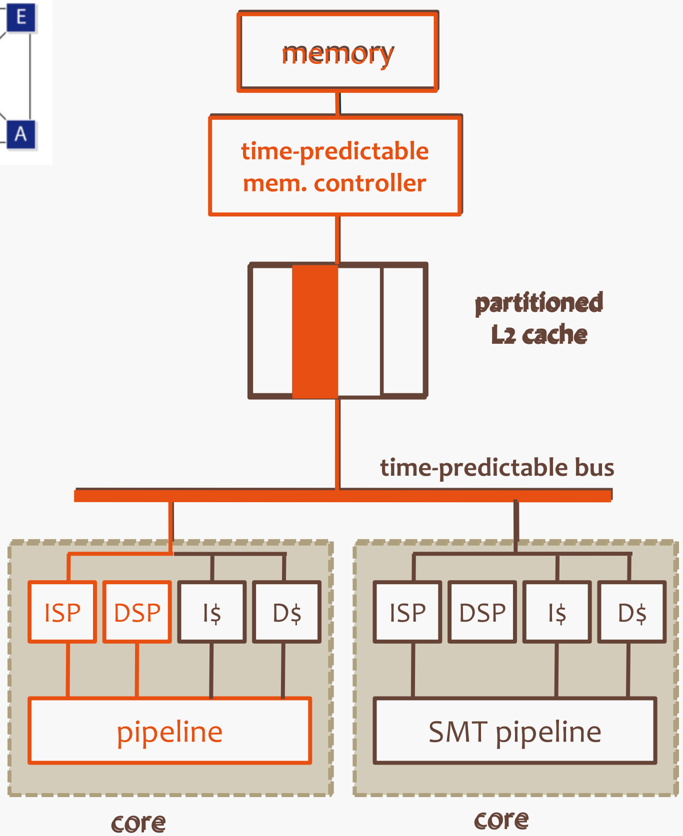
- **New issues**
  - time-predictable multicore architectures
    - projects: **MERASA**, PREDATOR
  - parallel programming of embedded applications
  - **WCET analysis of parallel programs**
    - reporting experiments on a data parallel industrial application

The MERASA project

- **Objectives:**
  - develop **multi-core processor designs** (from 2 to 16 cores) and **system-level software** support for mixed-critical workloads
  - provide **WCET analysis tools**
    - static analysis: OTAWA
    - hybrid measurement-based: RapiTime
  - evaluate with **pilot studies** submitted by industrial partners

- **Partners:**
  - **Participants:** Univ. Augsburg, Barcelona Supercomputing Center, Univ. Toulouse, Rapita Systems Ltd., Honeywell
  - **Industrial Advisory Board:** Airbus, BAUER Maschinen GmbH, European Space Agency, Infineon , NXP

The MERASA multicore and system software



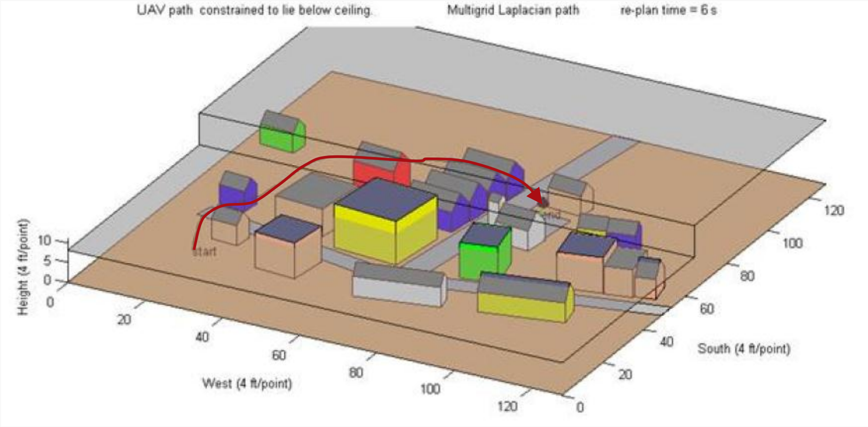
- **Isolation of critical threads**
  - specific scheduling in pipeline
  - time-predictable bus
  - partitioned L2 cache
  - time-predictable memory controller

- **System software**
  - time-predictable memory management
  - POSIX-compliant time-predictable synchronizations
  - two-level thread scheduling

The 3D multigrid solver

Université de Toulouse

- **Part of application for airbone collision avoidance (Honeywell)**
  - path planning using Laplace's equation
  - numerical solution:
    - domain partitioning → voxels
    - set potential of each point to average of neighbors
    - several iterations
  - multigrid technique:
    - several phases, varying grid size



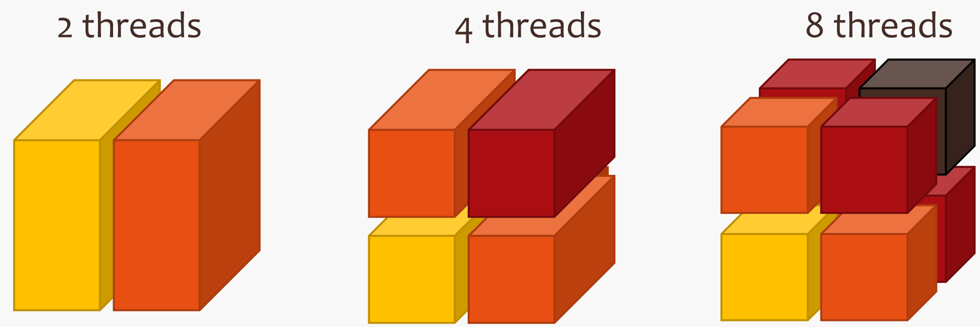
- **Version considered in this study**
  - three phases -- each phase = **interpolation** step + **iteration** step

```
for (x=0; x<NX; x++)
   for (y=0; y<NY; y++)
      for (z=0; z<NZ; z++)
         v[x][y][z] = compInterpolate(old_v);
```

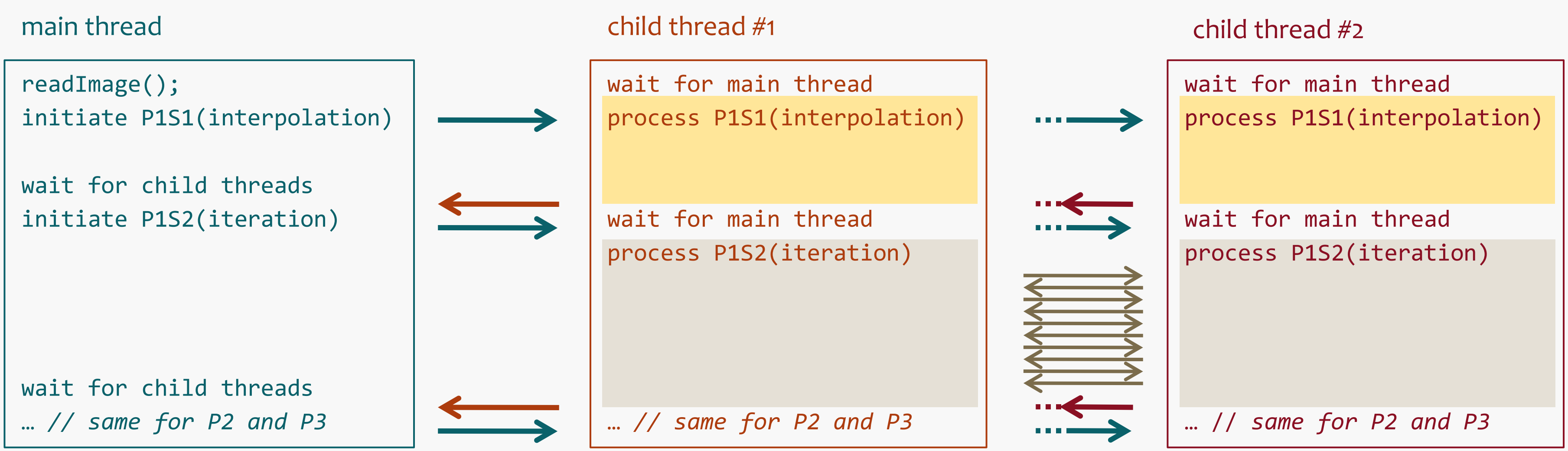
```
for (i=0; i<NUM_ITE; i++)
   for (x=0; x<NX; x++)
      for (y=0; y<NY; y++)
         for (z=0; z<NZ; z++)
            v[x][y][z] = compIterate(v);
```

The parallelized 3D multigrid solver

- Domain splitting



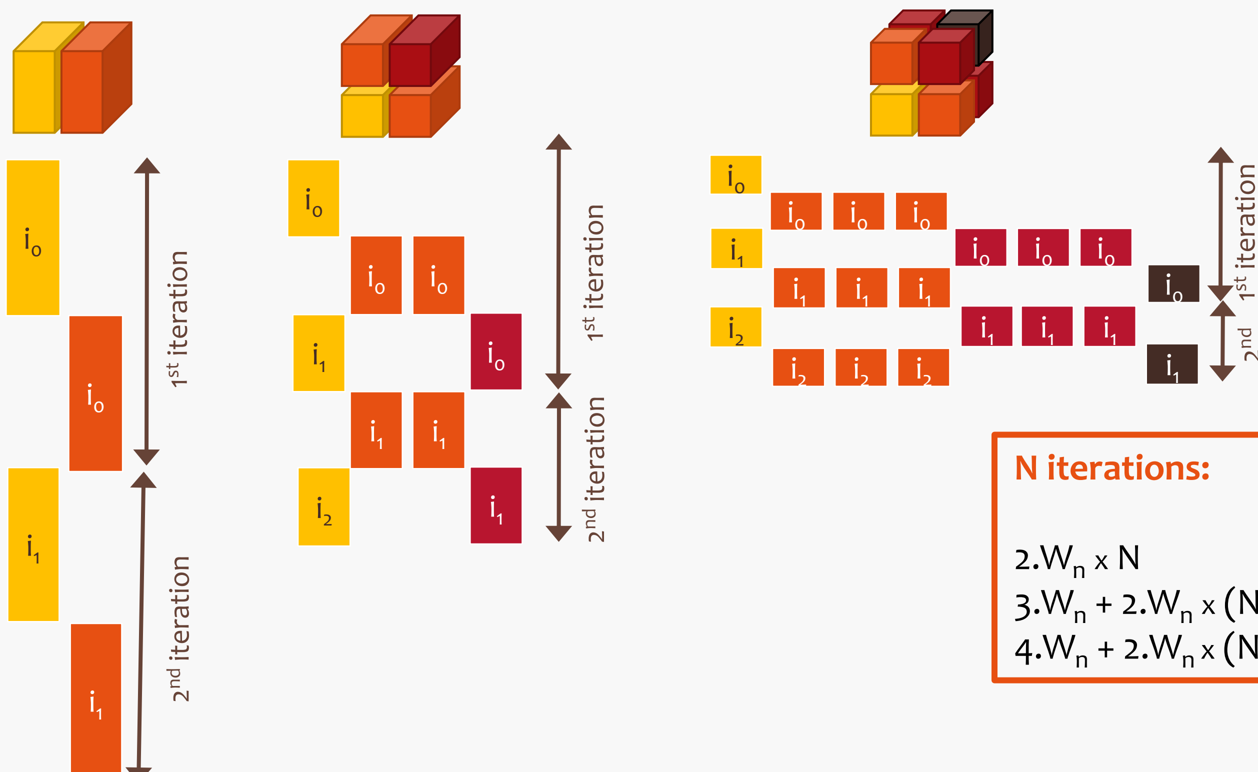
- Orchestration



Breakdown of the WCET

- First level

- Second level



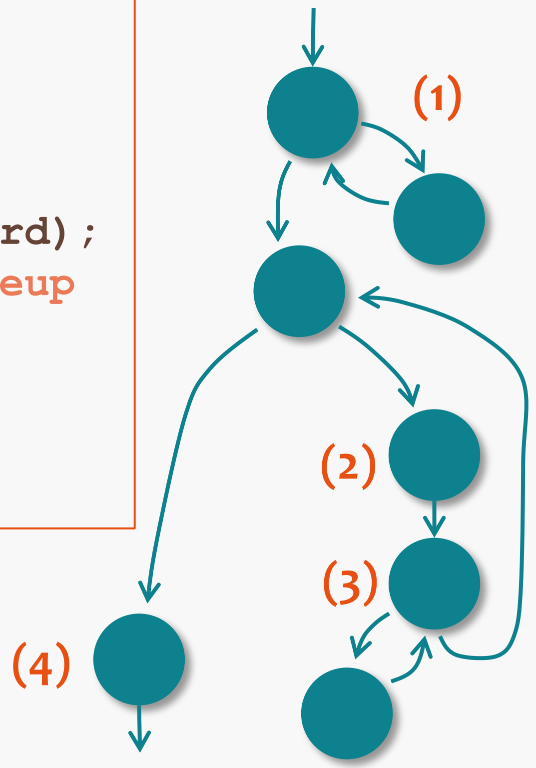
N iterations:

$2.W_n \times N$	(2 threads)
$3.W_n + 2.W_n \times (N-1)$	(4 threads)
$4.W_n + 2.W_n \times (N-1)$	(8 threads)

Analysis of synchronizations

- Example: mutex lock

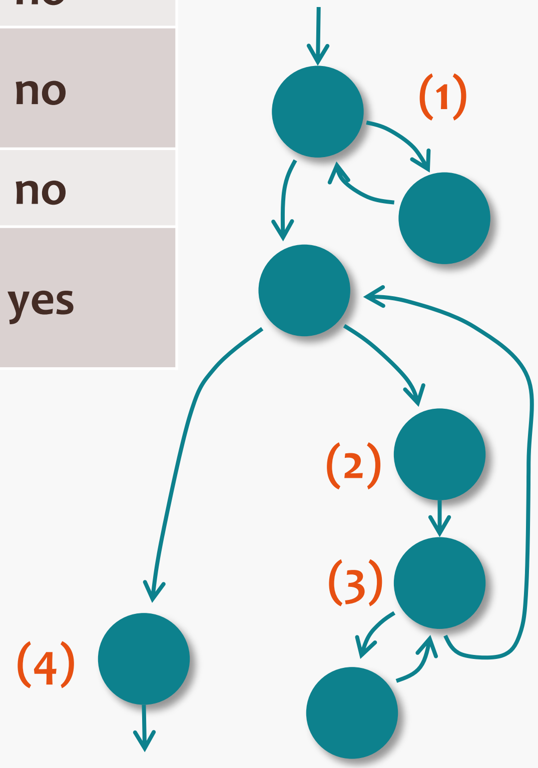
```
    int mutex_lock(mutex_t mutex){
(1)     sl_lock(&mutex->guard);
        …
        while (trylock(&mutex->thelock){
            … // insert thread into queue
(2)         sl_unlock_and_set_suspended(&mutex->guard);
(3)         spinlock_lock(&mutex->guard); // on wakeup
        }
        …
(4)     sl_unlock(&mutex->guard);
    }
```



Analysis of synchonizations (cont'd)

Université de Toulouse

Term		depends on …	
		# threads	application
T_e	Execution time when the variable is free	no	no
T_{w1}	Overhead execution time when the thread has to wait for the variable	no	no
T_{w2}	Waiting time related to system-level variables	yes	no
T_{w3}	Waiting time related to application-level variables	yes	yes



Experimental results

- **Estimated WCETs** (OTAWA – perfect ISP – no L2 cache)

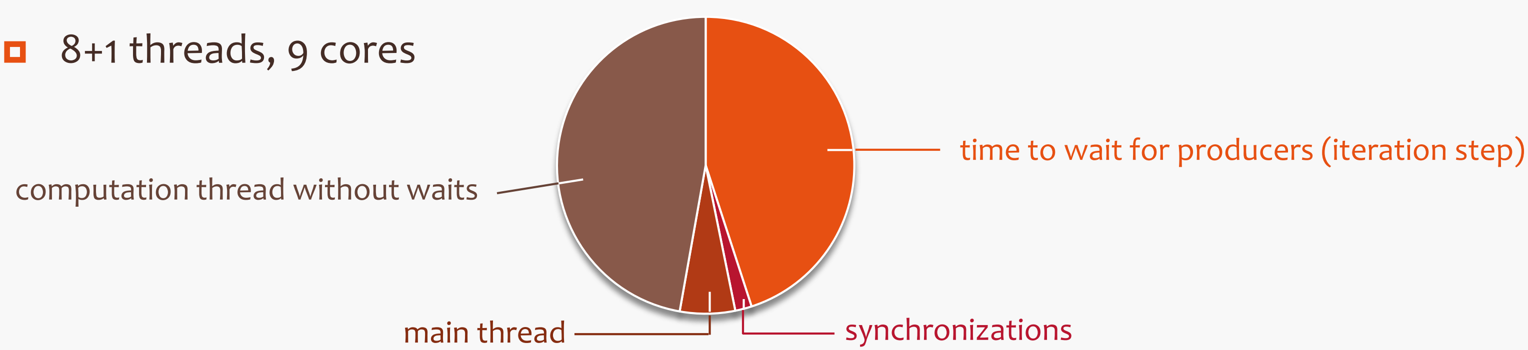
	# threads			
	1	2+1	4+1	8+1
1 core	53,990,765	-	-	-
3 cores	58,297,375	66,849,369	-	-
5 cores	62,603,985	73,372,109	40,389,428	-
9 cores	71,217,205	86,417,589	47,678,968	28,105,761

× 1.3

÷ 1.9

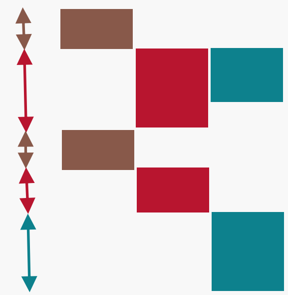
÷ 2.5

- **WCET breakdown**
  - 8+1 threads, 9 cores



Lessons learnt and future work

- **WCET analysis of a parallel application requires:**
  - understanding the structure of the application
    - dependency analysis
  - analyzing the synchronizations
    - where?
    - number of concurrent threads?
    - paths on which locks are hold



- **Automatic WCET analysis of parallel code:**
  - retrieve required information (structure + synchronizations)
    - source/object code analysis
    - user annotations
  - orchestrate the WCET computation
- **Parallelisation guidelines**
  - to facilitate WCET analysis

Université de Toulouse

Questions?