

Parallel Real-Time Tasks, as viewed by WCET Analysis and Task Scheduling Approaches

Christine Rochange

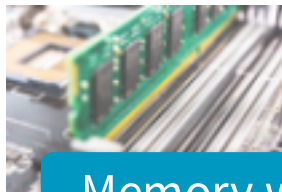


16th Workshop on Worst-Case Execution Time Analysis
5th July, 2016



Motivation

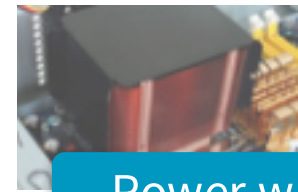
The performance of single-core processors does not grow anymore



Memory wall

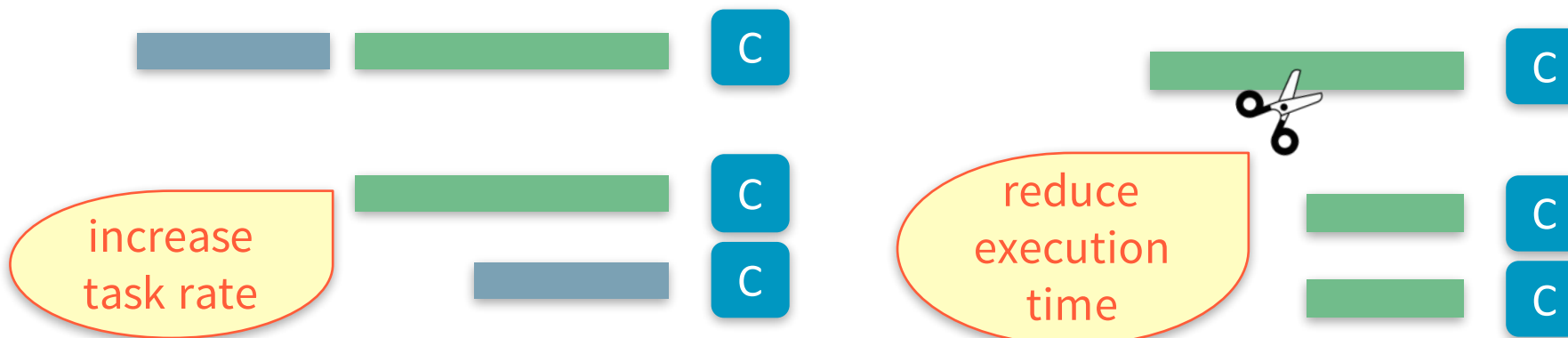


ILP wall



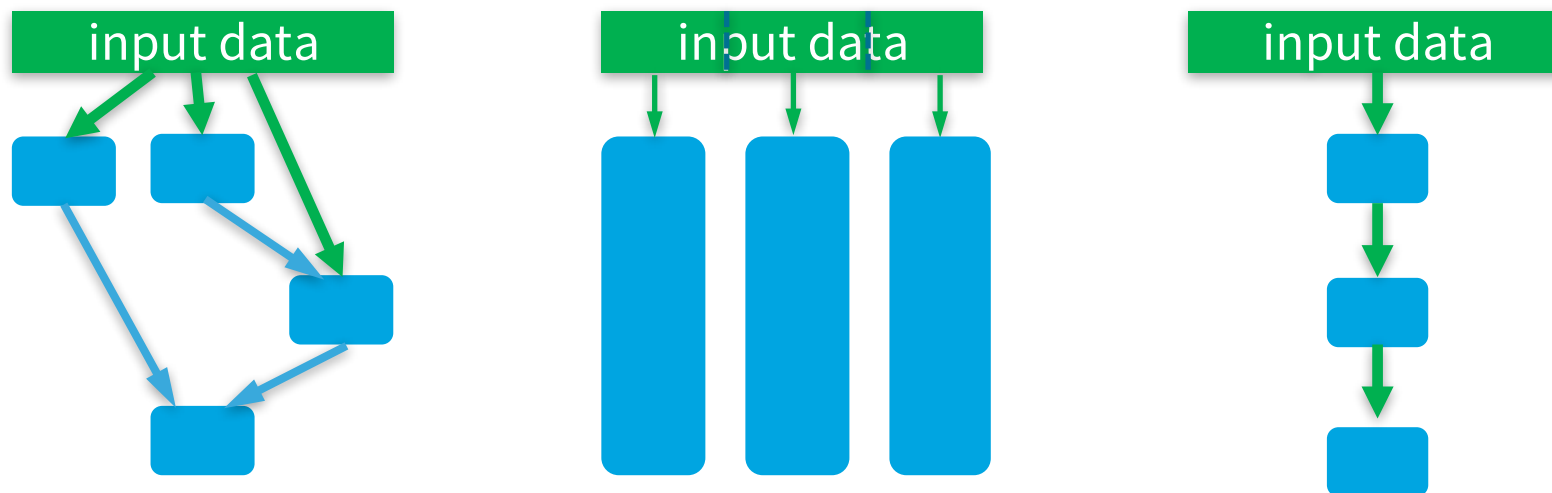
Power wall

Multi-core processors have become the norm



Different flavours of parallel programming

Task vs. data parallelism



Interactions between threads

- exchange of data (message passing, mutual exclusion, etc.)
- progress synchronisation

Disclaimer

Hardware-level interferences are out of the scope

- they may have a strong impact on WCETs
 - bandwidth-sharing (bus, memory controller, etc.)
 - space-sharing (e.g. shared L2 cache)
- research on how this impact can be accounted for is ongoing
 - still some open questions

BUT

 - let's assume the problem solved (analysable/predictable platform)
- an orthogonal issue wrt. analysing software-level interferences

Literature

Timing analysis of parallel real-time tasks



Do they all speak of the same parallel tasks?

How do they fit together?

Schedulin

Kar

Introdu

I. Int

In the ubiquitous exploit th

real-time systems community, there has been significant research on scheduling task sets with *inter-task parallelism*. In this case, increasing the number of cores allows us to increase the number of tasks in the task set. However, since each task can only use one core at a time, the computational requirement of a single task is still limited by the capacity of a single core. Recently, there has been some interest in the design and analysis of scheduling strategies

1068-3070/14 \$31.00 © 2014 IEEE
DOI 10.1109/ECRTS.2014.23

1052-8725/10
DOI 10.1109/RTSS.2010.42

1052-8725
DOI 10.1109

DOI 10.1109

Analysis of Federat

Jing Li¹, Jian-Jia Chen², K

li.jing@wustl.edu, jian-jia

Towards WCET An
Architectures Usin

Andreas Gustavsson¹,
Pettersson¹

1 School of Innovation,
Box 883, S-721 23 Vä
(andreas.g.gustavss

Integrated Wo
Estimation of

Dumitru Potop-B

1 INRIA, Paris-R
2 University of R

Abstract

Worst-case execution
sequential programs
analysis technique t

Automatic WC
Applications*

Haluk Ozaktas, Ch

IRIT – Université de
France
firstname.lastname@irit

Abstract

Tomorrow's real-time e
two challenges. First,
independent throu-

Toward Static Timing Analysis of Parallel Software*

Andreas Gustavsson, Jan Gustafsson, and Björn Lisper

School of Innovation Design and Engineering, Mälardalen U
(andreas.g.gustavsson, jan.gustafsson, bjorn.lisper)

Abstract

The current trend within computer, and e
parallel systems, e.g., regarding
executed on a gi
lyses the tim

This paper time tasks with characterized a We analyze th gies: two well a deadline-first an algorithm, name scheduling algor. alization of part this strategy, each high-utilization task (utilization 2 is assigned a set of dedicated cores and the remain low-utilization tasks share the remaining cores. We p particular, we show that if on uni-speed cores, a task set has total utilization of at most m and the critical path length of each task is smaller than its deadline, then federated scheduling can schedule that task set on m cores of speed 2; G-EDF can schedule it with speed $\frac{3+\sqrt{3}}{2} \approx 2.618$; and G-RM can schedule it with speed $2 + \sqrt{3} \approx 3.732$. We also provide lower bound scheduling and show that the best

Keywords and UPPAAL

Digital Obj

1 Intro

The execu system b real-tim

Multi-cor Their ad the price involving on the software of ever its ter (i) Wo (ii) W Seven decompy is to

time-predict T-CHEST¹ have to be a high ta cores. Other require sh paralleliz program them pro Seven decompy is to

For reasons of energy consumption and performance, development in hardware today strives toward massively parallel architectures, like many-core, GPU made in providing WCET analyses for such systems. This paper focuses on analysing the provided programming model is arbitrary underlying model is derived the

Worst-Case Execution Times (WCET) of the tasks in heterogeneous platforms. Thus, it is very likely that software made in providing WCET analyses for such systems. This paper focuses on analysing the provided programming model is arbitrary underlying model is derived the

Abstract interpretation of Pro

Abstract interpretation of Pro

Abstract interpretation of Pro

Abstract interpretation of Pro

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

At tasks utilizat task set cores. a resource antes a u

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord

A utilizat notation in ord



WCET analysis of parallel tasks

Background on WCET analysis for *sequential* tasks

TASK UNDER ANALYSIS

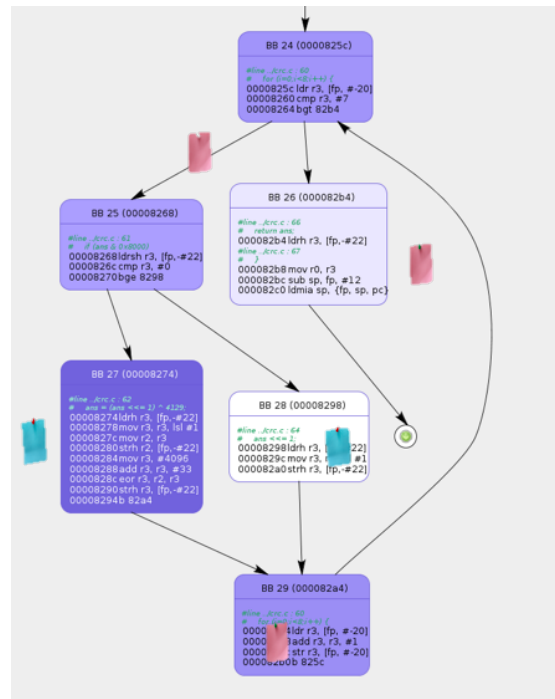
source
code

binary
code

STATIC ANALYSES

FLOW ANALYSES

LOW-LEVEL ANALYSES



WCET COMPUTATION

Integer Linear Programming



WCET upper-bound

WCET analysis of parallel tasks

Example 1

```
void core1() {
    int tqmf[24]; long xa, xb, el;
    int xin1, xin2, decis_level;
    for(;;) { //Infinite loop
        // Computation phase 1
        xa = 0; xb = 0;
        for (i=0;i<12;i++) { // 12 iterations
            xa += (long) tqmf[2*i]*h[2*i];
            xb += (long) tqmf[2*i+1]*h[2*i+1];
        }
        // Send the results to core 2
        send(channel1, (int)((xa+xb)>>15)); -----
        // Read inputs
        xin1=read_input(); xin2=read_input();
        // Computation phase 2
        for(i=23;i>=2;i--) { // 22 iterations
            tqmf[i]=tqmf[i-2];
        }
        tqmf[1] = xin1; tqmf[0] = xin2;
        // Receive data from core2 and output it
        decis_level = receive(channel2) ; <-----
        write_output(decis_level) ;
    }
}
```

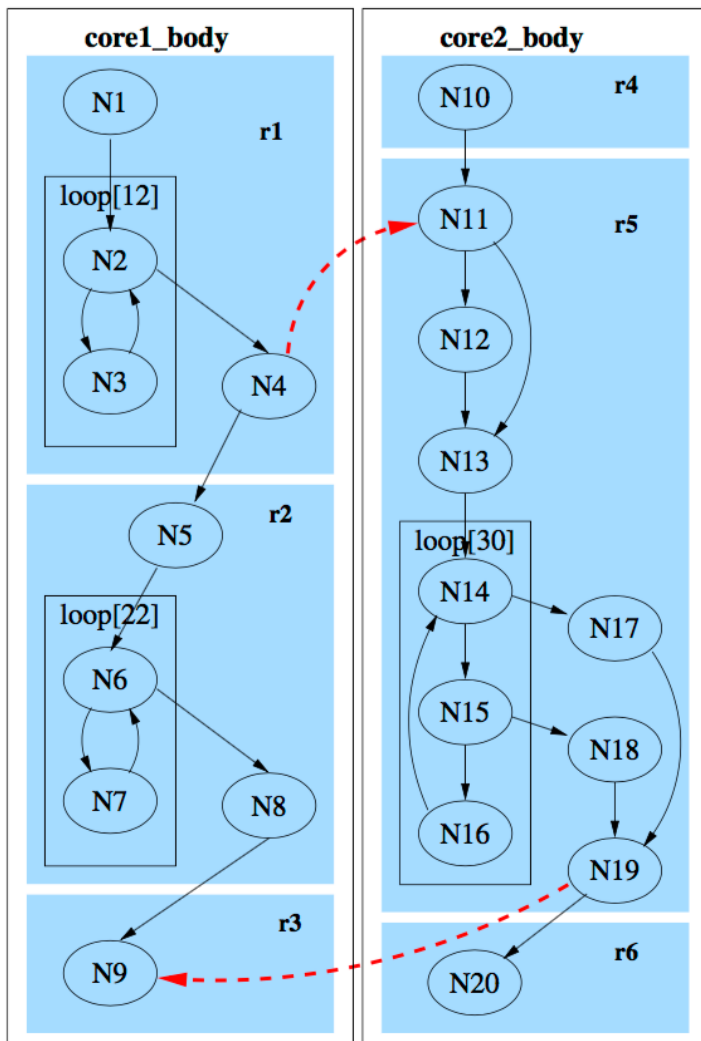
```
const int decis_level [30];
int core2() {
    int q, el;

    for(;;) { //Infinite loop

        // Receive data from core1
        el = receive(channel1);
        // Computation phase 1
        el = (el>=0)?el:(-el);
        for (q = 0; q < 30; q++) {
            // 30 iterations
            if (el <= decis_level[q])
                break;
        }
        // Send result to core1
        send(channel2, decis_level) ;
    }
}
```

WCET analysis of parallel tasks

Example 1



Joined CFGs

- additional edges to express precedence

```
void core1() {  
  int tqmf[24]; long xa, xb, el;  
  int xin1, xin2, decis_level;  
  for(;;) { //Infinite loop  
    // Computation phase 1  
    xa = 0; xb = 0;  
    for (i=0;i<12;i++) { // 12 iterations  
      xa += (long) tqmf[2*i]*h[2*i];  
      xb += (long) tqmf[2*i+1]*h[2*i+1];  
    }  
    // Send the results to core 2  
    send(channel1, (int)((xa+xb)>>15));  
    // Read inputs  
    xin1=read_input(); xin2=read_input();  
    // Computation phase 2  
    for(i=23;i>=2;i--) { // 22 iterations  
      tqmf[i]=tqmf[i-2];  
    }  
    tqmf[1] = xin1; tqmf[0] = xin2;  
    // Receive data from core2 and output it  
    decis_level = receive(channel2);  
    write_output(decis_level);  
  }  
}
```

```
const int decis_level [30];  
int core2() {  
  int q, el;  
  
  for(;;) { //Infinite loop  
  
    // Receive data from core1  
    el = receive(channel1);  
    // Computation phase 1  
    el = (el>=0)?el:(-el);  
    for (q = 0; q < 30; q++) {  
      // 30 iterations  
      if (el <= decis_level[q])  
        break;  
    }  
    // Send result to core1  
    send(channel2, decis_level);  
  }  
}
```


WCET analysis of parallel tasks

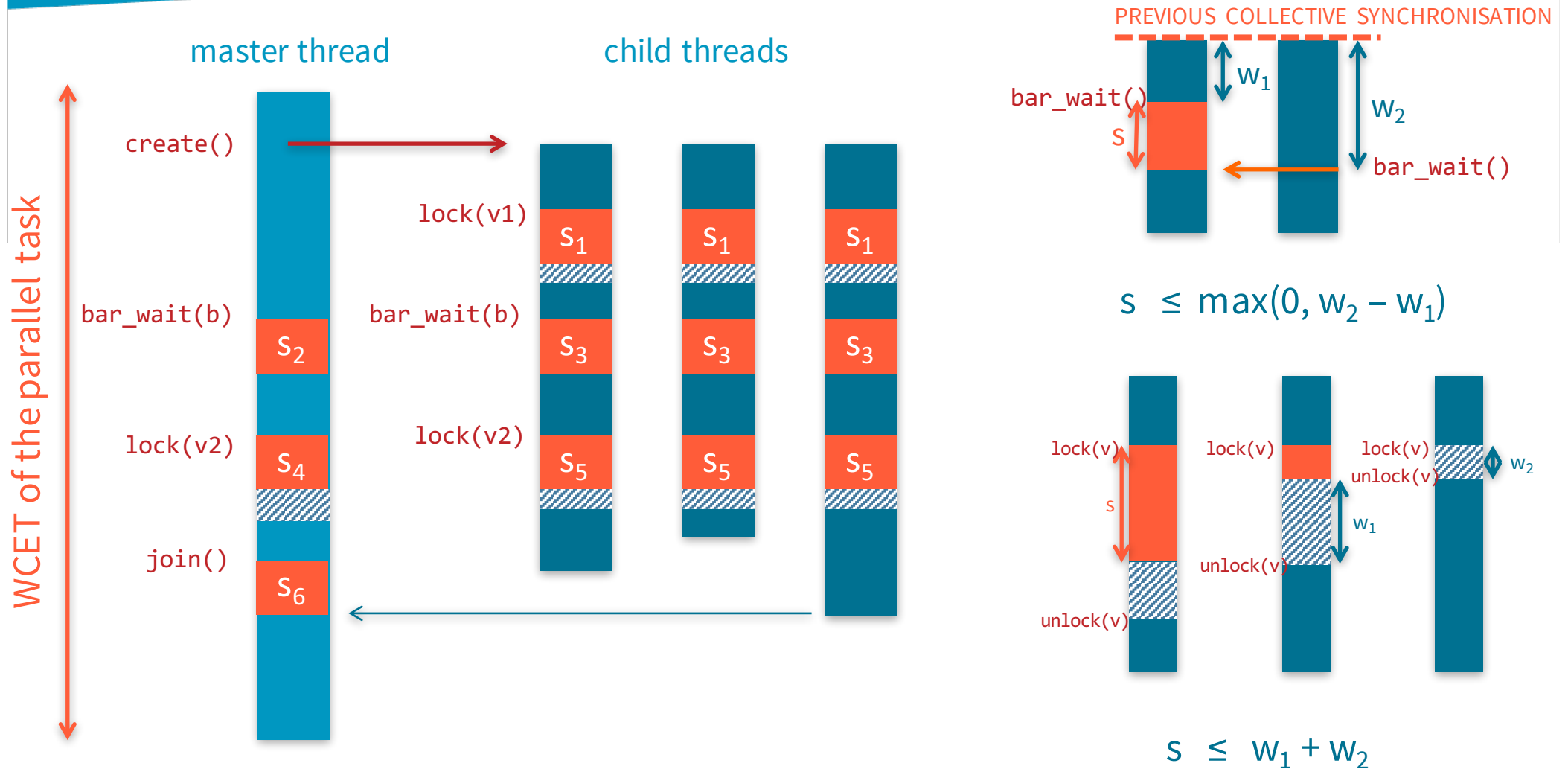
Example 2

```
int main() {  
    for (int i=0; i<2; i++)  
        CREATE_THREAD(&work);  
    ...  
    BARRIER(&bar,3); // ID=bar  
    ...  
    for (int i=0; i<2; i++)  
        JOIN(i+1); // ID=join  
}
```

```
void work() {  
    ...  
    BARRIER(&bar,3); // ID=bar  
    ...  
    MUTEX_LOCK(&lock); // ID=cs  
    ... // critical section  
    MUTEX_UNLOCK(&lock); // ID=cs  
    ...  
}
```

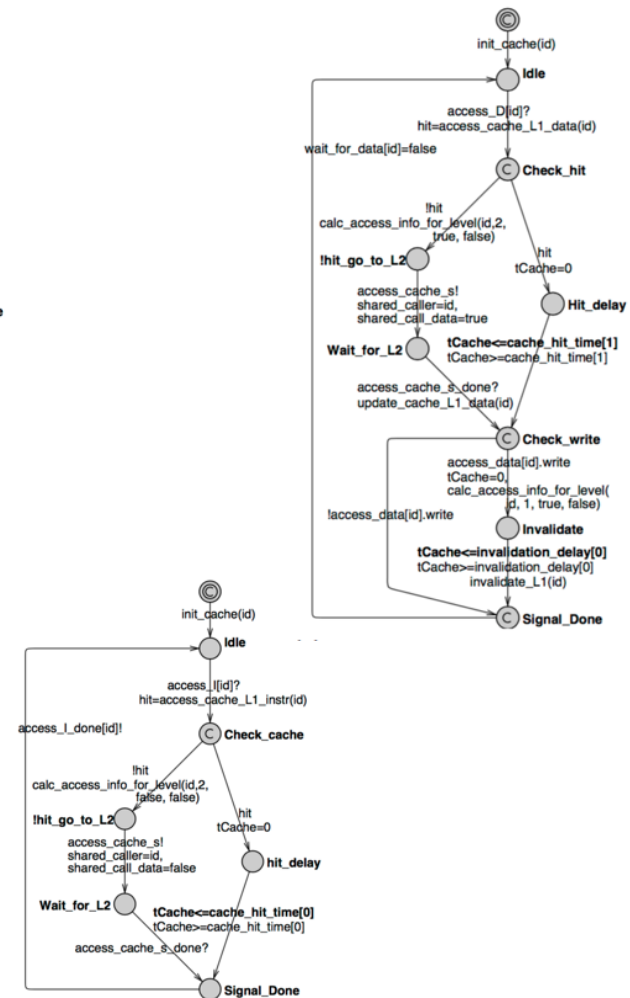
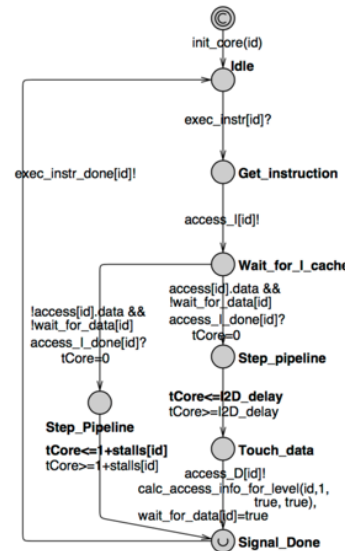
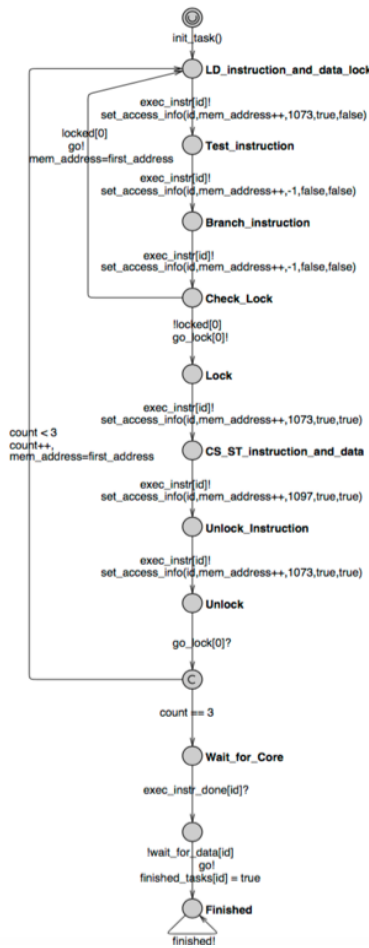
WCET analysis of parallel tasks

Example 2



WCET Analysis of Parallel Tasks

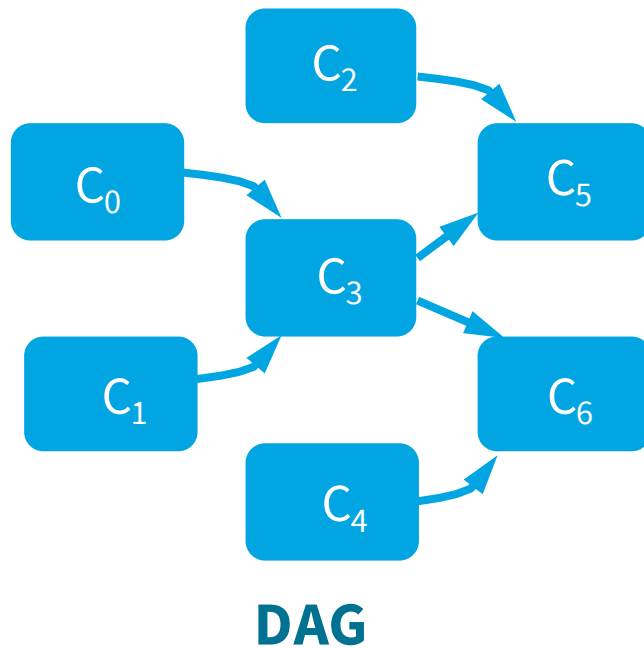
Example 3



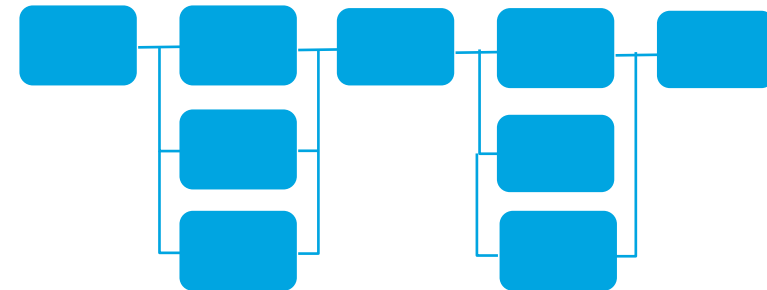
Scheduling parallel tasks

Parallel task models

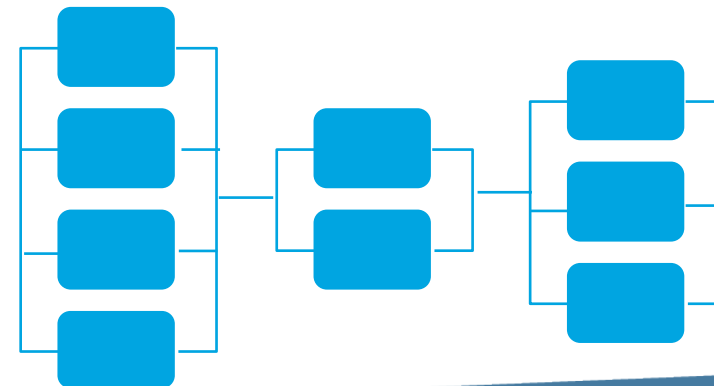
P, D, ~~X~~



fork-join



multi-frame segment



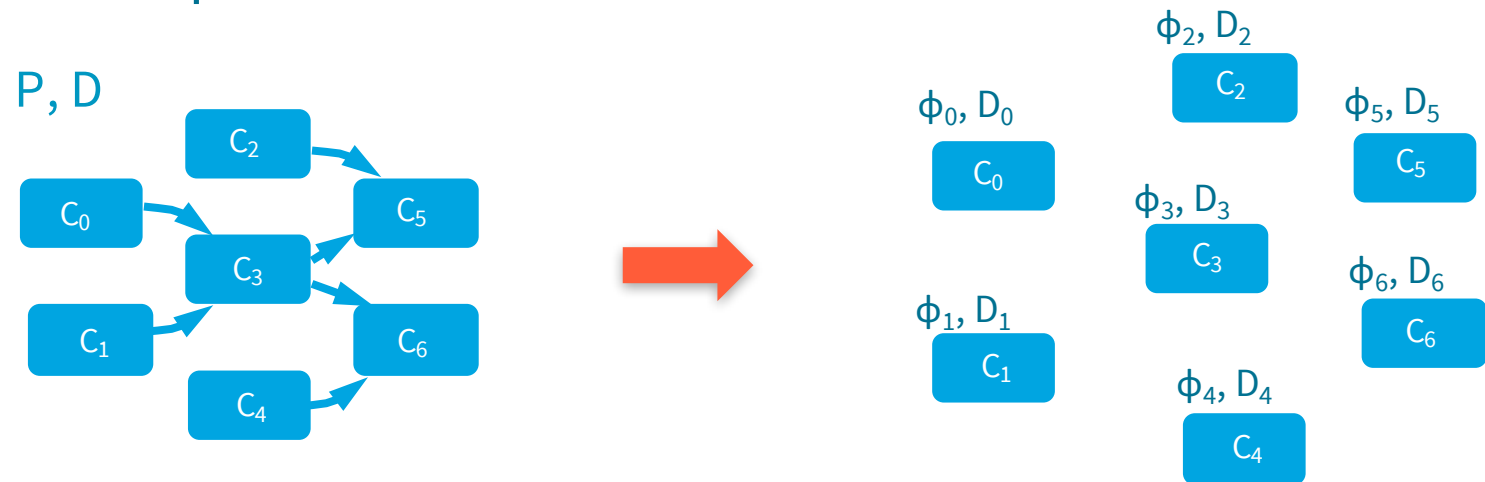
Scheduling parallel tasks

Two kinds of approaches

- DAG transformation
- Direct scheduling

DAG transformation

- transforms a parallel task (DAG of subtasks) into a set of independent sequential tasks



- schedules independent tasks using techniques for scheduling sequential tasks on multicores

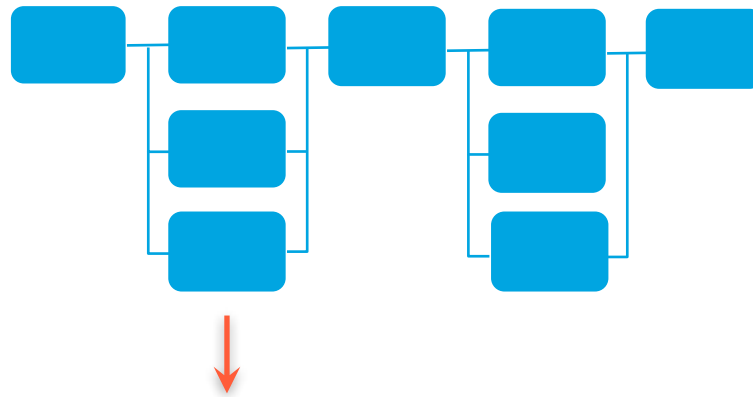
Scheduling parallel tasks

Two kinds of approaches

- DAG transformation
- Direct scheduling

Direct scheduling

- Gang scheduling



*one thread per core
threads run simultaneously*

Scheduling parallel tasks

Two kinds of approaches

- DAG transformation
- Direct scheduling

Direct scheduling

- Gang scheduling
- Federated scheduling
 - allocates a dedicated cluster of cores
 - schedules subtasks with a greedy strategy

$$n_i = \left\lceil \frac{C_i - L_i}{D_i - L_i} \right\rceil$$

Scheduling parallel tasks

Two kinds of approaches

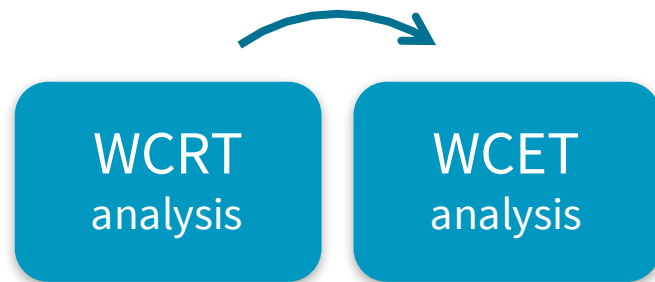
- DAG transformation
- Direct scheduling

Direct scheduling

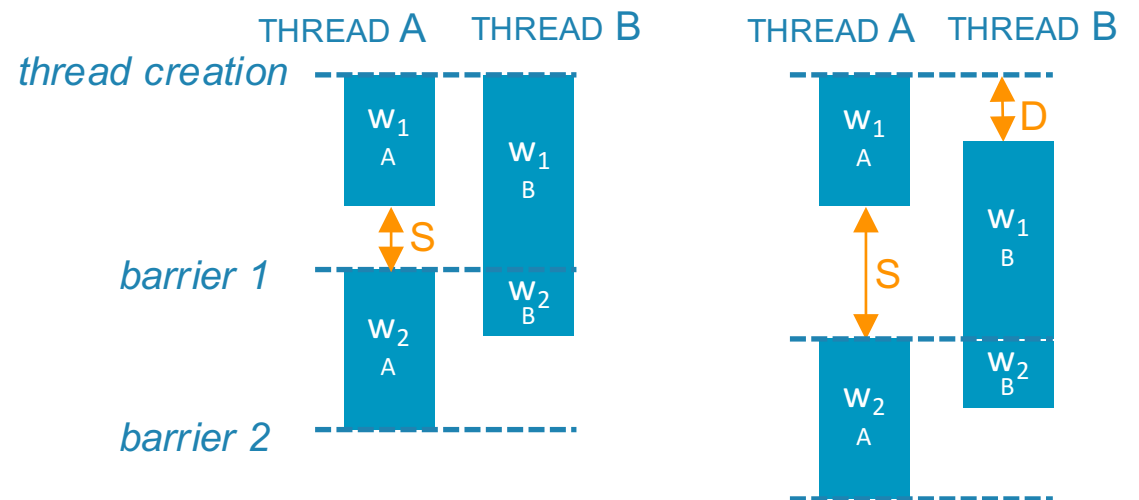
- Gang scheduling
- Federated scheduling
- Global EDF scheduling
 - schedules ready nodes with the earliest deadline

Key assumptions

gang-style schedule



safe WCETs
(for any schedule)



Summary

Different focus

- independent subtasks with precedence constraints
vs.
communicating/synchronising subtasks

sched./WCRT

WCET

Idealized assumptions

- schedule-independent subtask WCETs
vs.
gang-like schedule

sched./WCRT

WCET

Need for tighter cooperation

Preliminary ideas

- Schedule-dependent WCET analysis
 - release offsets for parallel threads
 - safer/tighter estimations of worst-case stall times
- Enriched task model
 - scheduling directives/constraints
 - schedule-dependent WCETs
- Scheduling/WCET-aware parallel programming

