Timing analysis of parallel tasks running on parallel architectures
Christine Rochange
Static Worst-Case Execution Time Analysis

**TASK UNDER ANALYSIS**
- source code
- binary code

**STATIC ANALYSES**

**FLOW ANALYSES**
- indirect branch targets
- Control Flow Graph
- loop bounds
- infeasible paths

**LOW-LEVEL ANALYSES**
- global analyses
  - memory hierarchy
  - branch prediction
- local analyses
  - pipeline

**WCET COMPUTATION**
- Integer Linear Programming

*WCET upper-bound*
WCET analysis of parallel tasks

Why is it different from the analysis of sequential tasks?

- hardware-level interferences: tasks compete for shared resources
  - time-sharing (bus, memory controller, etc.)
    - transactions might be delayed due to conflicts ⇒ longer latencies
    - can we bound the local/global overhead?
  - space-sharing (e.g. shared L2 cache)
    - the contents of shared storage might be corrupted by competing tasks
    - can we predict the impact of "random" cache line replacement?

out of the scope of this talk
WCET analysis of parallel tasks

Why is it different from the analysis of sequential tasks?

- software-level interferences: tasks communicate and synchronize
  - data-sharing:
    - message passing for distributed memory
    - mutual exclusion for shared memory
  - progress synchronisation:
    - conditions, barriers

STALL/WAITING TIME
- suspension
- spinning
WCET analysis of parallel tasks

State of the art

- distributed-memory blocking message passing
  - D. Potop-Butucaru, I. Puaut
    Integrated WCET Estimation of Multicore Applications
    Workshop on WCET Analysis, 2013
    [PoPu13]

- shared-memory mutual exclusion with spin locks
  - A. Gustavsson, A. Ermedahl, B. Lisper, P. Pettersson
    Towards WCET Analysis of Multicore Architectures Using UPPAAL
    Workshop on WCET Analysis, 2010
  - H. Ozaktas, C. Rochange, P. Sainrat
    Automatic WCET Analysis of Real-Time Parallel Applications
    [OzRS13]
    Workshop on WCET Analysis, 2013
WCET analysis of parallel tasks

Common assumptions

- cooperating tasks/threads run simultaneously on different cores
  - one task/thread per core
  - no hardware-level interferences

\[\text{PoPu13}] \text{ task = control loop}\]

\[
\text{core 0:}
\text{for (;;)}{
    \text{... // sampling}
    \text{... // processing}
    \text{... // actuation}
}\]

\text{similar on other cores}

\[\text{OzRS13}] \text{ fork-join model}\]

\[
\text{int main()}{
    \text{for (int i=1; i<N; i++)}
    \text{create_thread(&work);}\text{work();}
    \text{for (int i=1; i<N; i++)}
    \text{join(i);}\}
\text{void work(){}
    \text{...;}
}\]
WCET analysis of parallel tasks

Common point of view

- the analysis of stall/waiting times is *integrated* in the WCET analysis

[PoPu13] joins the CFGs of tasks

[OzRS13] estimates worst-case waiting times
WCET analysis of tasks synchronizing through message passing [PoPu13]

Joined CFGs
- additional edges to express precedence
WCET analysis of tasks synchronizing through message passing [PoPu13]

Analysis phases

- flow analyses, low-level analyses on each task
- modeling of communications
  - edges to reflect blocking primitives
- ILP-based WCET computation
  - longest path in the compound CFG
WCET analysis of tasks synchronizing through shared memory [OzRS13]

POSIX threads with time-analysable sync. routines
WCET analysis of tasks synchronizing through shared memory [OzRS13]

POSIX threads with time-analysable sync. routines
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Stall/waiting times for progress synchronizations

\[ s \leq WCST \]
\[ WCST = \max(0, w_2 - w_1) \]

Stall time depends on relative ‘speed’ of threads
reference = previous common synchr.
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Stall/waiting times for mutual exclusion
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Stall/waiting times for mutual exclusion

mutual exclusion \Rightarrow \text{serialisation of threads}

Worst-case stall time:
\[ s \leq WCST \]
\[ WCST = w_1 + w_2 \]
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

- main
- create()
- join()
- lock(v1)
- bar_wait(b)
- lock(v2)
- bar_wait(b)
- lock(v2)
- join()

estimate the WCET of the main thread
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

main
create()
bar_wait(b)
lock(v2)
join()

lock(v1)
bar_wait(b)
lock(v2)

S2
S3
S4
S5
S6

estimate the WCET of the main thread including stall/waiting times
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

- main
  - create()
  - bar_wait(b)
  - lock(v2)
  - join()

Stages:
- lock(v1) → S1
- bar_wait(b) → S2
- lock(v2) → S3
- lock(v2) → S4
- lock(v2) → S5

Analysis of stall/waiting times

WCET(main) → S2
WCET analysis of tasks synchronizing through shared memory [OzRS13]

**Computation process**

- **main**: create(), bar_wait(b), lock(v1), bar_wait(b), lock(v2), join()
- **analysis of stall/waiting times**: WCET(main)

Time steps:
- $s_1$
- $s_2$
- $s_3$
- $s_4$
- $s_5$
- $s_6$
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

- main
  - create()
  - bar_wait(b)
  - lock(v2)
  - join()

- lock(v1)
- bar_wait(b)
- lock(v2)

- s1
- s2
- s3
- s4
- s5
- s6

analysis of stall/waiting times

WCET(main)
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

main:
  create()
  bar_wait(b)
  lock(v2)
  join()

lock(v1)
  S1
bar_wait(b)
  S2
  S3
lock(v2)
  S4
  S5

analysis of stall/waiting times

WCET(main)

s1 → s2 → s4
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

main
create()
bar_wait(b)
lock(v2)
join()

lock(v1) bar_wait(b) lock(v2)

s1 s1 s1
s2 s3 s3
s4 s3 s3
s5 s5 s5
s6

s1 s2 s4 s6

analysis of stall/waiting times

WCET(main)
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

main
create()

bar_wait(b)

lock(v2)

join()

lock(v1)

bar_wait(b)

analysis of stall/waiting times

WCET(main)

s1
s2
s3
s4
s5
s6
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

Analysis of stall/waiting times

WCET(main)

s_1

s_2

s_3

s_4

s_5

s_6
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

Determine flow facts

Compute local WCETs

Compute global WCET

ILP formulation

\[
\begin{align*}
\max \ T &= \sum x_i w_i \\
x_0 &= 1; \quad \text{//structure} \\
x_0 &= x_1 + x_2; \\
\ldots \\
x_4 &\leq 10; \quad \text{// flow} \\
\ldots 
\end{align*}
\]

Implicit Path Enumeration Technique
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Computation process

- Determine flow facts
- Compute local WCETs
- Identify synchronisations
- Compute WCSTs
- Append WCSTs to local WCETs
- Compute global WCET (main thread)

ILP formulation

\[
\text{max } T = \sum x_i \cdot w_i
\]

\[ x_0 = 1; \quad \text{//structure} \]
\[ x_0 = x_1 + x_2; \]
\[ ... \]
\[ x_4 \leq 10; \quad \text{// flow} \]
\[ ... \]

Implicit Path Enumeration Technique
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Required user annotations

```
<sync id="b_get">
  <thread id="7,2,8,5">
    <contender id="7,2,8,5">
      <sync ref="b_get"/>
    </contender>
    <contender id="7">
      <sync ref="b_set"/>
    </contender>
  </thread>
</sync>

<sync id="b_set">
  <thread id="7">
    <contender id="2,8,5">
      <sync ref="b_get"/>
    </contender>
  </thread>
</sync>
```
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Required user annotations

```xml
<thread id="0-3">
  <routine id="compartmentThd"/>
</thread>

<barrier id="bar_step">
  <thread id="0-3">
    <last_sync ref="bar_ready" />
  </thread>
</barrier>
```
WCET analysis of tasks synchronizing through shared memory [OzRS13]

Required user annotations

```xml
<barrier id="bar_first">
  <in_loop id="loop_step"/>
  <thread id="0-3">
    <last_sync ref="bar_step_start"/>
    <last_sync_in_loop ref="bar_next"/>
  </thread>
</barrier>
```
Summary

Static WCET analysis of parallel tasks

- assumptions:
  - all threads/tasks run in parallel on separate cores
- accounting for communication/synchronisation overhead
  - integrated to WCET analysis
    - CFGs joined with edges that denote precedence (message passing)
    - estimation of worst-case stall times (shared memory)
  - benefits from contextual hardware-state information
    - better for accuracy