Module 6: Parallel Execution

Technical University of Vienna
Vienna, Austria, May 2022
Parallelism

Multicore execution preserves deterministic semantics.
Event and Reaction Queues

Event queue, sorted by tag

Reaction queue, sorted by deadline and level.

Worker 1  Worker 2  Worker 3  Worker 4

ForkJoin

Source

TakeTime

Destination

(level: 0) (0, 200 msec)

(level: 1)

(level: 2)
To get parallelism, the pipeline pattern requires careful attention to tags.

```c
/* Basic pipeline pattern where a periodic source feeds a chain of reactors that can all execute in parallel at each logical time step. */

/* The workers argument specifies the number of worker workers, which enables the reactors in the chain to execute on multiple cores simultaneously. */

/* This uses the TakeTime reactor to perform computation. If you reduce the number of worker workers to 1, the execution time will be approximately four times as long. */

/* @author Edward A. Lee */
/* @author Marten Lohstroh */

```
Sporadic events are assigned a time stamp based on the local physical-time clock.

Computations have logically zero delay.

Every reactor handles events in time-stamp order. If time-stamps are equal, events are “simultaneous”.

Actuators can have a deadline \( D \). An input with time stamp \( t \) is required to be delivered to the actuator before the local clock hits \( t + D \).
Determinism

Whether the two triggers are present simultaneously depends only on their timestamps, not on when they are received nor on where in the network they are sent from.
Simple, Sequential Execution

- Sort reactions topologically based on precedences.
- Global notion of “current tag” $g$.
- Event queue containing future events.
- Choose earliest tag $g'$ on the event queue.
- Wait for the real-time clock to match the timestamp of $g$.
- Execute reactions sequentially in topological sort order.

When a sporadic sensor triggers (or an asynchronous event like a network message arrives), assign a time stamp based on the local physical-time clock.
Sort reactions topologically based on precedences.
- Global notion of “current tag” $g$.
- Event queue containing future events.
- Choose earliest tag $g'$ on the event queue.
- Wait for the real-time clock to match the timestamp of $g$.
- **Execute reactions in parallel where possible.**
Reactions with the same level can always execute in parallel.
Parallel Execution Using Levels

Reactions with the same level can always execute in parallel.

Reactions within a reactor never execute in parallel.
More Deterministic Timing

Actuators will now execute between 9 and 10 msec after sensors, unless a deadline violation occurs.

What could cause a deadline violation?
This strategy is closely related to the notion of **Logical Execution Time (LET)** but generalizes that concept to permit zero execution time and to allow deadline violation handlers.
More Deterministic Timing

Classical real-time systems scheduling and execution-time analysis determines whether the specification can be met.

[Buttazzo, 2005]  [Wilhelm et al., 2008]
Determinism does not imply a cost in performance.

Parallel execution (multicore) does not imply nondeterminism.

Christian Menard  
(TU Dresden)
Scalability

Ping Pong

Execution time (ms)

Threads

All-Pairs Shortest Path (APSP)

Execution time (ms)

Threads

Framework
- Akka
- CAF
- LF C++
Active Research

- More aggressive parallel execution.
- Reducing contention for reaction queue.
- Supporting parallel execution at multiple tags.
- Direct support for Logical Execution Time (LET)
- Leveraging lock-free concurrency.