A Measurement-based Model for Parallel Real-Time Tasks

KUNAL AGRAWAL & SANJOY BARUAH

Washington University in St. Louis

Current models for parallel real-time workloads

Graph based: < graph G = (V, E); relative deadline D; period T>

Vertices ≈ sequential code; edges ≈ dependencies

- Fork-join tasks – a layered graph
- Sporadic DAG task – any acyclic graph
- **Conditional** DAG task – add conditional constructs

Current models – shortcomings

Graph based: < graph G = (V, E); relative deadline D; period T>

1. Analysis requires knowledge of DAG structure

   - may not be available/ known

2. Conditional tasks: many possible behaviors

   - Exhaustive enumeration yields exponential-time algorithms

3. Conditional tasks: worst-case behavior may be very rare

Proposed model – background

Federated scheduling: each (high-util.) task gets exclusive access to some processors

work and ***span*** parameters of (regular) DAG tasks

- *work*: cumulative WCET of all vertices (on a single processor)
- *span*: maximum sum of WCET's of chain of vertices (on infinitely many processors)

Response time upon m dedicated processors

$$\textit{response time} \geq \max\left(\frac{\textit{work}}{m}, \textit{span}\right)$$

The **list scheduling** guarantee

$$\textit{response time} \leq \left[\frac{\textit{work}}{m} + \left(1 - \frac{1}{m}\right) \times \textit{span}\right]$$

- a 2-approximation

Proposed model

Federated scheduling. Run-time dispatching using **list scheduling**

IDEA I. Represent each task by its (***work***, ***span***) parameters

IDEA II. Measurement-based: *work, span* values obtained via measurement experiments – profiling run-time behavior (as in **probabilistic WCET**)

work – upon a single processor

span – upon a large number of processors

IDEA III. Two pairs of estimates

($work_H$, $span_H$) are very conservative estimates

($work_L$, $span_L$) are less conservative estimates

For efficient implementation, assume that ($work_L$, $span_L$) values hold

Guarantee correctness if ($work_H$, $span_H$) values hold

Proposed model – scheduling algorithms

PREPROCESS($\langle work_H, span_H, work_L, span_L, D \rangle, m$)

1 **if** $\left(\frac{work_H}{m} + \left(1 - \frac{1}{m}\right) \times span_H\right) > D$ **then return** FAILURE

2 **else** Compute m_L and D_L // Solve quadratic, linear equations – $\Theta(1)$ time

RUNTIME(m_L, D_L, m)

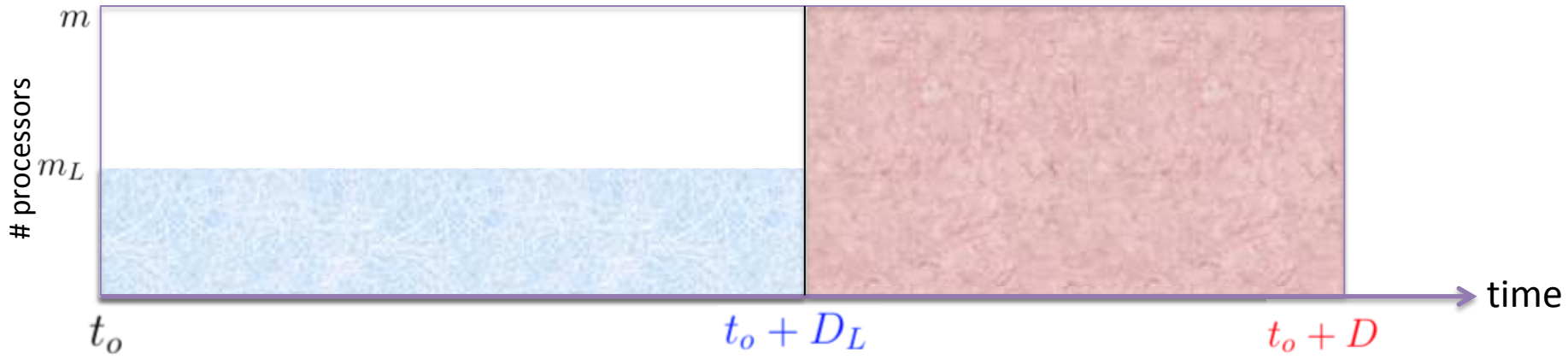
// An instance arrives at time-instant t_o

1 LIST-SCHEDULE upon m_L processors

2 **if** the instance has *not* completed by time-instant $(t_o + D_L)$

then // *Less conservative assumptions do not hold*

3 LIST-SCHEDULE upon all m processors



Proposed model – scheduling algorithms

PREPROCESS($\langle work_H, span_H, work_L, span_L, D \rangle, m$)

1 i

2 e

RUN

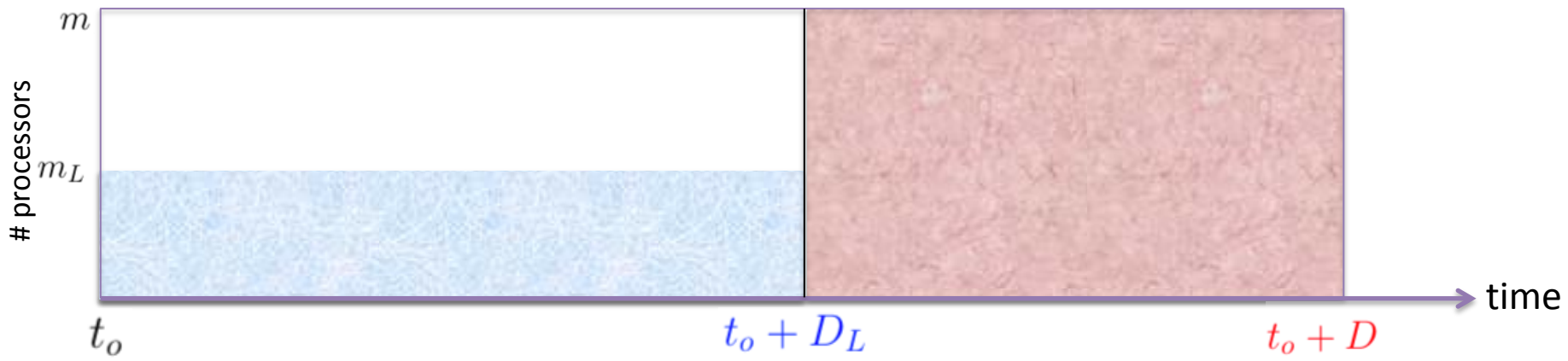
1 L

2 i

3

PROPERTIES

1. **Safety** is guaranteed (provided preprocessing succeeds)
2. **Efficiency**: expect m_L processors to be used "almost always"
   - The other $(m - m_L)$ processors may be asleep/ execute less-critical workloads
3. **Dynamically tunable**: m_L and D_L may be recomputed during run-time



Summary

Task models that expose intra-task parallelism

- Shortcomings of current models

Proposed model

(*work*, *span*) characterization of DAGs

+ measurement-based estimation of parameters

+ multiple estimates of each parameter

A **scheduling algorithm** that

guarantees correctness

and aims for efficiency

and is tunable during run-time