Techniques Optimizing the Number of Processors to Schedule Multi-Threaded Tasks

Geoffrey Nelissen, Vandy Berten, Joël Goossens, Dragomir Milojevic
• What can we parallelize?
  – Traditional approach: execution of many **sequential** tasks
    • But... #CPU is increasing -> Systems with more CPUs than tasks
  – New approach: execution of **parallel tasks**
    • Solid background for parallelization in the “non-real time” world
    • Explicit parallel coding: MPI, PVM, OpenMP, multi-thread
    • Compiler (automatic) parallelization

• Should we develop two different scheduling models?
• If not: How could we efficiently use the existing theory?
Generalization of the Fork-Join model

- Any number of threads in each segment

- Sporadic multi-threaded tasks with constrained deadlines
How could we use the existing scheduling theory with such a model?

- Adding artificial intermediate deadlines

  ➔ Each **thread** can be modeled by a **sequential sporadic task**

Any algorithm scheduling sequential sporadic tasks with constrained deadlines can be used.
Density of a segment

Density of a segment $\tau_{i,j} : \delta_{i,j} = \frac{C_{i,j}}{d_{i,j}}$

Density: minimum # needed CPUs.

Techniques Optimizing the Number of Processors to Schedule Multi-Threaded Tasks
Density of a task system

Maximal density:
\[
\max_j \sum_i \max_j \delta_{i,j}
\]

Worst situation:
\[
\max_j \delta_{1,j}
\]
How do we minimize the number of processors?

• A sufficient schedulability test for PD², DP-Wrap, **U-EDF**:  
  \[ m \geq \delta(t), \ \forall t \]

• Minimizing \( m \) \( \Rightarrow \) minimizing the total maximal density
  \[ \sum_i \max_j \delta_{i,j} = \sum_i \max_j \frac{C_{i,j}}{d_{i,j}} \]

• Terms are independent \( \Rightarrow \) minimizing the maximal density for each task
  \[ \max_j \frac{C_{i,j}}{d_{i,j}} \]
Density of a segment

Techniques Optimizing the Number of Processors to Schedule Multi-Threaded Tasks
Density of a segment

Segment’s density upper-bound

Density of a segment

Techniques Optimizing the Number of Processors to Schedule Multi-Threaded Tasks
This Problem can be expressed as a Linear Optimization Problem

- **Minimize:** Maximum density reachable by the segments
- **Subject to:**
  - Intermediate deadline ≥ Largest thread execution (for each segment)
  - Sum of intermediate deadlines ≤ Task deadline
Let us try to understand the optimal solution

• The deadline of a segment cannot be smaller than the WCET of its threads

→ The segment density is upper-bounded

• An **optimal** solution can be found by **comparing** the **upper-bound of all segment densities** with the **average density** of the task
Let us try to understand the optimal solution

- **Rule 1:** Keep the density constant if possible

![Diagram showing density upper-bound and average density]
Let us try to understand the optimal solution

- **Rule 1**: Keep the density constant if possible

![Diagram showing rules and calculations related to WCET, deadlines, and average density.](image)
Let us try to understand the optimal solution

- **Rule 2**: Minimize impact of segments with small densities

![Diagram showing density and WCET of segments]
Let us try to understand the optimal solution

- **Rule 2**: Minimize impact of segments with small densities
A simple algorithm to find an optimal solution

- Sort segments by their density upper-bounds

- **For each** segment in the list **DO**
  - **IF** Maximum density < average density **THEN**
    - Assign maximal density to the segment
    - Recompute average density of remaining segments
  - **ELSE**
    - Assign average density to all remaining segments
    - Break
  - **END**

- **END**

- Complexity : $O(n_i \times \log n_i)$ for each task

- Allow to dynamically add tasks online
What about performances?

Techniques optimizing the Number of Processors to Schedule Multi-Threaded Tasks
• Threads shared by several segments

→ We reduce it to the previous problem

• The Optimization problem becomes non-linear
Extension to a More General Model of Multi-Threaded Tasks

• New difficulties:
  – More variables
    • segment deadline
    • repartition of WCET between segments
  – Decision taken in one segment strongly impacts other segments in $G$

$\Rightarrow$ We need a tool taking into account the properties of many segments
Extension to a More General Model of Multi-Threaded Tasks

- Many parallel branches
Conclusions

- Provide a way to « convert » a parallel task in a set of sequential task, by adding intermediate deadlines
- Optimal assignment (given we use intermediate deadlines)
- Low computational complexity ($O(n_i \times \log n_i)$ for each task)
- In average, we need 20% less processors than previous work
- Augmentation bound: 2
- Extension to multi-branch tasks
Questions?
Manage the evolution of the task density by fixing the intermediate deadlines

- Number of threads varies ➔ Task density changes with time
- By imposing intermediate deadlines
  ➔ We can compute the density of each segment independently

Intermediate Deadlines
Let us try to understand the optimal solution

- **Rule 2:** Minimize impact of segments with small densities

![Diagram showing WCET of segment 1 and segment 2, deadline of segment 1, average density, and maximum on sum of segment deadlines.](image)
Let us try to understand the optimal solution

- **Rule 2:** Minimize impact of segments with small densities