A Stretching Algorithm for Parallel Real-time DAG Tasks on Multiprocessor Systems

Manar QAMHIEH    Laurent George*    Serge Midonnet

University Paris-Est Marne-la-Vallée, France
LIGM / ESIEE*

Versailles France    8-10 Oct. 2014
Presentation Outline

- General Introduction
- Related Work & Problem Description
- DAG Stretching Algorithm
- Resource Augmentation Bound for global preemptive EDF
- Simulation-based Evaluations
- Conclusion and Perspective
General Introduction

- **Hard Real-time Systems**
  - Failure in respecting timing characteristics of the real-time system leads to catastrophic results.

- **Multiprocessor Systems**
  - execution platform consists of homogeneous unit-speed processors.

- **Global Preemptive Scheduling**
  - the execution of a job can be interrupted by higher priority jobs, and job migration is allowed between processors.
Parallel programming is proposed as an evolution of software so as to get advantage of multiprocessor architecture.

Parallel programming APIs: OpenMP, pThreads

Many real-time parallel task models: Fork-join (FJ) and Multi-Threaded Segment Model, Directed Acyclic Graphs (DAG).
Fork-join task sets on multiprocessor systems can have schedulable utilization bounds slightly greater than and arbitrarily close to uniprocessor schedulable utilization bounds. From the perspective of schedulability, it is therefore desirable to avoid such task structures as much as possible.¹

¹ “Scheduling Parallel Real-Time Tasks on Multi-core Processors”. Lakshmanan et al., ECRTS 2010
Related Work: FJ-Stretching Algo.

For each FJ task before scheduling, apply FJ-Stretching Algorithm.

[1] “Scheduling Parallel Real-Time Tasks on Multi-core Processors”. Lakshmanan et al., ECRTS 2010
Related Work: FJ-Stretching Algo.

For each FJ task before scheduling, apply FJ-Stretching Algorithm.

A set of independent threads.
Contribution: DAG-Str Algorithm

We propose a DAG-Str (DAG Stretching) Algorithm.
Contribution: DAG-Str Algorithm

We propose a **DAG-Str (DAG Stretching) Algorithm**

- A DAG task (periodic, implicit-deadline):
  - Set of subtasks with specific WCETs
  - Set of precedence constraints between subtasks
  - timing parameters: offset, relative deadline, period
DAG-Str Algorithm

Multi-Threaded Segment Form of DAG task

Parallel Segment

$L_1 = 6$

$S_{11} = 4$

$S_{12} = 2$

$S_{13} = 1$

$S_{14} = 2$

$S_{15} = 1$

$C_1 = 14$

$D_1 = T_1 = 10$

Segment Index

Number of threads in segment
DAG-Str Algorithm

Calculate a unit factor $f_1$ of the DAG task

$L_1 = 6$  $S_{l_1} = 4$

$\tau_{11} (3)$  $\tau_{14} (1)$  $\tau_{16} (2)$

$\tau_{12} (3)$  $\tau_{17} (1)$

$\tau_{13} (2)$

$\tau_{15} (2)$

$S_{11}$  $m_{11} = 4$

$S_{12}$  $m_{12} = 2$

$S_{13}$  $m_{13} = 1$

$S_{14}$  $m_{14} = 2$

$S_{15}$  $m_{15} = 1$

$C_1 = 14$

$D_1 = T_1 = 10$

$f_1 = S_{l_1}/(C_1-L_1) = 1/2$

Uniform slack filling on a Thread-level
DAG-Str Algorithm

Calculate a unit factor $f_1$ of the DAG task

$L_1 = 6$  $SI_1 = 4$

$\tau_{11} (3)$  $\tau_{14} (1)$  $\tau_{16} (2)$

$\tau_{12} (f)$  $f$  $\tau_{17} (f)$

$\tau_{13} (2)$  $f$

$\tau_{15} (2)$  $f$

$S_{11}$  $m_{11}=4$

$S_{12}$  $m_{12}=2$

$S_{13}$  $m_{13}=1$

$S_{14}$  $m_{14}=2$

$S_{15}$  $m_{15}=1$

$C_1 = 14$

$D_1 = T_1 = 10$

$f_1 = SI_1/(C_1-L_1)=1/2$

$\leftarrow$ Segment Index

$\leftarrow$ Number of threads in segment

Uniform slack filling on a Thread-level
DAG-Str Algorithm

Calculate a unit factor $f_1$ of the DAG task

A lot of thread migration and preemptions are caused.

Solution: fill slack time based on a Segment-Level.
DAG-Str Algorithm

Calculate a segment factor $f_{1j}$ of the DAG task

$L_1 = 6$  $S_{l1} = 4$

$C_1 = 14$
$D_1 = T_1 = 10$
$f_1 = S_{l1}/(C_1-L_1) = 1/2$

$f_{11} = 3/2$
$f_{12} = 1/2$
$f_{13} = 0$
$f_{14} = 1/2$
$f_{15} = 0$

$f_{1j} = f_1(m_{1j}-1)$
DAG-Str Algorithm

Calculate a segment factor $f_{1j}$ of the DAG task

\[
L_1 = 6 \\
S_1 = 4
\]

\[
\tau_{11} (3) \\
\tau_{14} (1) \\
\tau_{16} (2)
\]

\[
f_{11} = \frac{3}{2} \\
f_{12} = \frac{1}{2} \\
f_{13} = 0 \\
f_{14} = \frac{1}{2} \\
f_{15} = 0
\]

\[
f_1 = \frac{S_1}{C_1 - L_1} = \frac{4}{14 - 6} = \frac{1}{2}
\]

\[
C_1 = 14 \\
D_1 = T_1 = 10
\]

\[
(f_{1j} \times c_{1j}) \text{ execution units added to master thread from each segment } S_{1j}
\]

One thread/segment migrates between processors
DAG-Str Algorithm

Stretched DAG task

$L_1 = 6$

$S_{l_1} = 4$

$(1+3/2)2 = 5$

$(1+1/2)1 = 3/2$

$(1+1/2)1 = 3/2$
DAG-Str Algorithm

Stretched DAG task

L₁ = 6
S₁,1 = 4

(1+3/2)² = 5
(1+1/2)¹ = 3/2
(1+1/2)¹ = 3/2

Intermediate offsets and deadlines are assigned to parallel threads
DAG-Str Algorithm

Results of DAG-Str algorithm when applied on a DAG task whose:

- **Utilization > 1:**
  - Fully-Stretched threads with utilization equal to 1 → Assigned to dedicated processors.
  - Periodic independent constrained-deadline threads, with intermediate offset and relative deadline. → Scheduled using any multiprocessor scheduling algorithm.

- **Utilization == 1** → Fully-stretched thread which is assigned to a dedicated processor.

- **Utilization < 1** → Periodic independent implicit-deadline thread scheduled using any multiprocessor algorithm.

We consider **Global Preemptive Earliest-Deadline First (GEDF) Scheduling algorithm.**
Resource Augmentation Bound for GEDF

We prove a resource Augmentation Bound of global preemptive EDF equal to $(3 + \sqrt{5})/2$

for all task sets with $n < \varphi \times \overline{m}$, where:

\begin{itemize}
  \item $n$: number of tasks in the set
  \item $\varphi$: golden ratio where $\varphi = (1 + \sqrt{5})/2$
  \item $\overline{m}$: number of processors after stretching
\end{itemize}

The same value was proved for Global EDF for DAG tasks for large m in [1].

[1] “Analysis of Federated and Global Scheduling for Parallel Real-time Tasks”. Li et al., ECRTS 2014
Simulation Results

![Graph showing simulation results]

The graph illustrates the schedulability success percentage for different system utilization levels. Each line represents a different value of $m$: $m=2$, $m=4$, $m=8$, and $m=16$. As the system utilization increases, the schedulability success percentage decreases for all values of $m$. The graph helps in understanding the impact of system utilization on the schedulability of tasks.
Simulation Results

Compare with Decomposition Algorithm (DCMP)\(^2\) for DAG tasks.

Assigns slack time for segments based on a density threshold.

Simulation Results

![Graph showing simulation results](image)

- **Success rate %**
- **U (%) of number of processors m**

Lines and markers represent different conditions:
- **DAG-Str**
- **DCMP**
- **DAG-Str, m=2**
- **DCMP, m=2**
- **DAG-Str, m=4**
- **DCMP, m=4**
- **DAG-Str, m=8**
- **DCMP, m=8**
- **DAG-Str, m=16**
- **DCMP, m=16**
Simulation Results
**Conclusion & Perspectives**

**Conclusions:**
- DAG-Str algorithm which is more general than FJ Stretching algorithm
- Resource Augmentation Bound for GEDF scheduling algorithm.
- Simulation-based evaluations of the stretching algorithm.

**Perspectives**
- Optimize DAG stretching algorithm to reduce forced migrations and preemptions.
- Analyze Fixed Priority Scheduling Algorithm.
Thank You

Questions?