Online Semi-Partitioned Multiprocessor Scheduling of Soft Real-Time Periodic Tasks for QoS Optimization

By: Behnaz Sanati and Albert M. K. Cheng
bsanati@uh.edu, cheng@cs.uh.edu

April 11-14,
RTAS 2016, Vienna, Austria
Introduction

The Problem / Motivation

- Maximizing the benefit gained by soft real-time tasks in many applications is highly needed to provide an acceptable QoS
- Existing multiprocessor scheduling policies are mostly proposed for minimizing tardiness, and relatively very few studies on benefit-maximization

Objective

Providing an appropriate strategy for better QoS in highly loaded soft real-time multiprocessor systems with periodic tasks, by maximizing total gained benefit while minimizing tardiness, using approximation algorithms in semi-partitioning of the tasks at job-boundaries
Examples of Applications

- Online (and mobile) banking
- Multimedia applications
- Image and speech processing
- Robot control/navigation systems
- Medical decision making
- Body-sensor networks
- Medical monitoring systems
- Cloud computing, and IoT

By Y. Gil, W. Wu and J. Lee


Task Model

- Soft real-time task sets
- Periodic tasks
- Independent in execution
  (No precedence constraints among them)
- Preemption is allowed
- Synchronous and/or Asynchronous
- Each task come with its period, WCET and benefit density function
System Model

- \( m \) identical processors
- Three storage areas for each processor:
  1. **Pool:**
     - for waiting jobs of any tasks (instead of a shared pool)
  2. **Stack:**
     - for the scheduled jobs (preempted or running)
  3. **Garbage collection:**
     - for the jobs that miss their deadlines and gain no benefit for the system

Software Architecture of the System
Methodology (1 of 2) – Hybrid Model

1. Reward-based Priority Setting
   - **Variable priority** assignment over time for ready and waiting jobs
   - **Fixed priority** assignment for scheduled jobs at their start time

2. Hybrid Scheduling
   - **Reward Model**: Schedule high priority jobs
   - **Cost Model**: Gready/Load-balancing Approx. for Partitioning low priority jobs

3. Reward Calculation
   - No reward for incomplete jobs
   - Calculate gained reward by each completed job and add it to the total reward
Methodology (2 of 2) – Hybrid Model

**Benefit Model:**
To maximize *total benefit*

**Cost Model:**
To minimize *makespan*

1. **Job Arrival**
2. **Benefit Based Algorithm**
   - Prioritize the job(s) using their benefit density function and run the highest priority jobs
3. **Greedy / Load Balancing Algorithm**
   - Find a machine with minimum total $w_j$
   - Add the new (waiting) job to its pool
4. **Benefit Based Algorithm**
   - For each machine decide when to move a job from its pool to its stack
5. Compute total benefit of the system
Objective Functions and Solutions

- ** Benefit Maximization 
  - The main goal in a benefit-aware, soft real-time system
  - To gain maximum total value or benefit for the system by the jobs that complete their execution
  - An approximate solution due to multiprocessor scheduling being an NP hard problem

- ** Reducing Tardiness 
  Semi-partitioning approach (Migration at job-boundary)

- ** Overhead Reduction 
  - Reducing Number of Preemptions
  - Limiting Migrations
Summary of Advantages toward QoS Optimization

- more conservative CPU cycles consumption (less idle time)
- Reduces the makespan without compromising on benefit maximization
- Increases the total benefit gained, specially on systems with higher work load, by
- Applicable to broader scope of tasks models, i.e. synchronous and/or asynchronous
- No off-line phase, and no limit on the number of processors for migrating jobs of each task (unlike other semi-partitioning techniques)
- The NP hard problem of multiprocessor scheduling is reduced into uniprocessor scheduling problem by partitioning the tasks at their arrival time (no dualization is needed as in RUN)
Thank You

Questions
or
Comments?