Zeus: Stream Group Soft Real-time Parallel Scheduler

Raoul Rivas, Klara Nahrstedt
Problem Description

- Distributed Interactive Multimedia Environments (DIME) are becoming ubiquitous (e.g. Online Gaming, 3D Teleimmersion, Interactive TV)

- DIME are systems with correlated multistreams that use Content Delivery Networks to process and disseminate the content among its user

- CPU QoS in terms of jitter, delay and bandwidth are critical in such systems

- Current architectures (VMM and OS) do not provide adequate abstractions and mechanisms to efficiently guarantee QoS to groups of correlated and dependent streams in multicore systems
Challenges

- More streams than cores
- Two type of Dependencies
  - Dependent: Consumer-Producer Relationship (A → B)
    - If A does not happen then B should not either
  - Concurrent: Temporal Correlation (A0, A1, A2, .. An)
    - A0 to An have the same period and should be scheduled concurrently
- Tasks have Period and Bandwidth variation
- Cameras have variable frame rate and variable frame size
- Some tasks in the chain may be optional
Solution Overview

- Restrict process admission to Posets
  - Hasse diagram
- Use process algebra type notation to allow user to describe the dependencies
- Use Liu-Layland model (Period=Deadline)
- Shadow Tasks to allow Temporal Correlated tasks to be scheduled at the same time
- Best Effort is scheduled in place of Shadow Tasks
- Goal is to obtain a Flat EDF schedule
Modified Hasse Diagram

- Each nodes represent a process
- Add a directed edge if there is dependence
- Add a non directed edge between two concurrent nodes

Deadline Constraint: $D(T_A) < D(T_B)$
Processor Demand Constraint: $S(T_A) + C(T_A) + C(T_B) + C(T_C) < D(T_C)$
Release Time Constraint: $S(T_F) + \max[(C(T_F), C(T_G), C(T_H))] < \min[D(T_F), D(T_G), D(T_H)]$
Dependence Specification

• Let the user specify the dependencies using a simple process algebra
  • Allows to model the system
  • Unambiguous
• Simplify the notation
  • $P \parallel Q$: Process $P$ and $Q$ execute concurrently
  • $P[c,d]$: Process $P$ executes for $c$ clock units and has a deadline $d$
  • $P>Q$: $Q$ waits for $P$ to terminate
Hasse Diagram Construction

Specification

$T_A [C(T_A), D(T_A)]$
$T_B [C(T_B), D(T_B)]$
$T_A > T_B$
$T_A > T_D$
$T_D > T_C$
$T_E > T_F$
$T_F | T_G | T_H$

Deadline Constraint: $D(T_A) < D(T_B)$
Processor Demand Constraint: $S(T_A) + C(T_A) + C(T_B) + C(T_C) < D(T_C)$
Release Time Constraint: $S(T_F) + \max([C(T_F), C(T_G), C(T_H)]) < \min[D(T_F), D(T_G), D(T_H)]$
• One partition per core. The use EDF on each partition independently
• Partitioning is an NP complete problem: Need a heuristic!
• Shadow tasks are used for Best Effort Scheduling (promotes work conservation) and suspending the processor (save energy)
Let $T$ be the set of tasks ordered by deadline in the system, and let $T_i$ the task with the smallest deadline that has a concurrent dependence with task $T_{i+1} \ldots T_j$.

1. Use Worst Fit heuristic to assign tasks $T_1$ to $T_{i-1}$.

2. Insert a shadow task $S_k$ in each partition $k$ with deadline $d(T_{i+k})$ and allocation $C_k$ such that the starting time of task $T_i = T_{i+1} = \ldots = T_j$ if each task is placed in different partitions and scheduled under EDF.

3. Insert tasks $T_i$ to $T_j$ each one in a different partition.


---

**Partitioning Heuristic**
Partitioning Heuristic

Let $T$ be the set of tasks ordered by deadline in the system, and let $T_i$ the task with the smallest deadline that has a concurrent dependence with task $T_{i+1} \ldots T_j$

1. Use Worst Fit heuristic to assign tasks $T_1$ to $T_{i-1}$.

2. Insert a shadow task $S_k$ in each partition $k$ with deadline $d(T_{i+k})$ and allocation $C_k$ such that the starting time of task $T_i = T_{i+1} = \ldots = T_j$ if each task is placed in different partitions and scheduled under EDF.

3. Insert tasks $T_i$ to $T_j$ each one in a different partition

4. Continue inserting tasks in set $T$
Let $T$ be the set of tasks ordered by deadline in the system, and let $T_i$ the task with the smallest deadline that has a concurrent dependence with task $T_{i+1} \ldots T_j$.

1. Use Worst Fit heuristic to assign tasks $T_1$ to $T_{i-1}$.
2. Insert a shadow task $S_k$ in each partition $k$ with deadline $d(T_{i+k})$ and allocation $C_k$ such that the starting time of task $T_i = T_{i+1} = \ldots = T_j$ if each task is placed in different partitions and scheduled under EDF.
3. Insert tasks $T_i$ to $T_j$ each one in a different partition.
Partitioning Heuristic

Let $T$ be the set of tasks ordered by deadline in the system, and let $T_i$ the task with the smallest deadline that has a concurrent dependence with task $T_{i+1}$ … $T_j$

1. Use Worst Fit heuristic to assign tasks $T_1$ to $T_{i-1}$.

2. Insert a shadow task $S_k$ in each partition $k$ with deadline $d(T_{i+k})$ and allocation $C_k$ such that the starting time of task $T_i = T_{i+1} = ... = T_j$ if each task is placed in different partitions and scheduled under EDF.

3. Insert tasks $T_i$ to $T_j$ each one in a different partition

4. Continue inserting tasks in set $T$

The admission control needs to consider the maximum $C_k$ for all jobs of $S_k$ in the hyperperiod
Future Work

• Modified Admission control equations
• Compare our heuristic with the optimal schedule (Simulation of random task sets)
• Initial implementation of the Constraint Verifier
  • Modify iDSRT CPU scheduler