Real-Time & Embedded Systems

Resource control

References for this chapter

[Ada95RM] (link to on-line version)
Ada Working Group
[Resource Manager] (with C++ version)
Resource Manager Design and Standard Editor
+ Resource Manager Implementation
Honda B20991 3/6/98 with C++ (3.08)
June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]

[Ada95RM]

[Resource Manager]

[Resource Manager Implementation]

[Honda B20991 3/6/98 with C++ (3.08)]

June 2000

[Addenda]
Handling requests by types, attributes, and in a global order

Motivation for resource reclaiming

Resource reclaiming algorithms

Resource reclaiming from independent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks

Resource reclaiming from independent tasks
Resource reclaiming from interdependent tasks

**Proof of Correctness**

Theorem: The RV algorithm gives a correct plan schedule S'

Proof: By the above lemma, passing occurred if S' is incorrect, i.e.

Proof: Assuming no passing occurred,

then all $c_{ij}$ have been computed before $t_i$ and all $r_{ij}$ are only evaluated after $t_i$ completed.

By definition of a feasible schedule if $r_{ij}=0$ then $t_j$ does not interfere with $t_i$, and captured by no measure delay the execution of $t_j$.

Therefore $c_{ij}=0$.

Restriction vector (RV) algorithm (Shen, Ramamritham, Stankovic, '93)

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks

Resource reclaiming from interdependent tasks
Synchronizing system-components which
• overload conditions.
☞ efficient resolution and frequency.
☞ recording all relevant information in a sufficient resolution.
☞ Resource reclaiming schemes exceed their anticipated worst case limits.
☞ handling tasks and resources which exceed their anticipated worst case limits.
☞ the mapping of the known and arising timing constraints.

Resource reclaiming evaluated

Some additional observables:
• Task graph density \( \rho_G \)\(^\dagger\) ∈ \((0, 1]\), where \( \rho_G \) is the density of a fully dependent task-set.
• \( \text{RV-cost} \)\(^\dagger\), \( \text{RV-allocation}\)\(^\dagger\) act as output time rates.
• \( \text{RV-allocation} \)\(^\dagger\): number of checks on dispatch queues by the RV with migration algorithm.
• \( \text{RC-reclaiming} \)\(^\dagger\) (Manimaran, Murthy, Vijay, Ramamritham '97):
RC computational costs (from Manimaran, Murthy, Vijay, Ramamritham '97):
\[\begin{align*}
\text{RC-reclaiming} = \text{RC-reclaiming} \cdot \text{RC-allocation} \cdot \text{RC-allocation}.
\end{align*}\]

Correctness of the migration process
To ensure that the swapping of dispatching queues does not interfere with the correctness of the post-run schedule, the swapping is permitted only if \( C \neq S \).

The unrestricted and executable task, which is next to be scheduled on the idling processor, is the next to be scheduled on the processor.

To ensure that the swapping of dispatching queues is not delayed by swapping these dispatching queues.

Resource reclaiming from interdependent tasks

Restriction vectors (RV) algorithm with task migration (Manimaran, Murthy, '97)

Policies
• Priority assignment problem
• Preempt assignment problem
• Resource reclaiming schemes
• Preempt assignment problem
• Resource reclaiming schemes

Real-time Resource Control (Manimaran '97)

Issues
• Policy and resource control

Resource control
• Resource synchronization primitives
• Evaluation criteria for resource synchronization methods
• Resource reclaiming schemes
• Preempt assignment problem