Evaluating synchronization mechanisms

Categorizing resource synchronization methods

Accepting or Avoiding?

Handling requests by type

Handling requests by parameters

References for this chapter

[Mercer97]
Clifford W. Mercer
real-time and multimedia applications
Proceedings of the seventh ACM symposium

[Bloom79]
Toby Bloom’s evaluation criteria for synchronization primitives

[Murthy2001]
C SR Murthy, G Manimaran
Resource Management in Real-Time
(often: by their
parameters, and the
priority
Synchronization state
of the resource
States which refer to the actual contents and available resources –
relative time
(One_Task, All_Tasks);

 ALT j=0 FOR max
modify [j] ? object

procedure
protected
package

pragma
protected
entry

Resource_Manager;
end

body

Resource_Manager;
end

function

Higher_Queues_Empty (Priority : Priorities)
return
Boolean;
end

of

Instances_Of_Resource

Allocate
Free

end

resource_control

is

CONSTANT

Instances_Of_Resource

is

procedure

Allocate
Free

end

is

procedure

Allocate
Free

end

is

procedure

Allocate
Free

Evaluating synchronization mechanisms

Handling requests by type and parameters

List of expected power might happen.

Discrete Interactions

- Register all requests that then are either individual types in a global order.
- Remove other parameters from the resource. Each parameter in a type-based on

Partial implications:

- Register all requests that then are either individual types in a global order.
- Remove other parameters from the resource. Each parameter in a type-based on

Expand Sign In (Amount: Resource Range);

Allocate Sign In (Amount: Resource Range);

Free (Amount: Resource Range);

Lack of expressive power might lead to:

- Ways out:

  - Assume or verify that the client will never die during a double interaction sequence.
  - Eliminate the double interaction by means of a attributed, single request type and requeuing.

Motivation for resource reclaiming [Murthy2001]

Complexity shall be bound by a constant.

• Correctness:

  - Resources reclaiming need to maintain the consistency.
  - Resource reclaiming overhead need to be small in comparison to the possible gains.

• Bounded complexity:

  - Improved resource reclaiming can be included in the overall task's worst case performance if the overhead shall be bounded to an acceptable constant.

• Effectiveness:

  - Improved resource reclaiming can be implemented, e.g., more failures can be handled by applying resource reclaiming.

Greedy method

Postrun schedule

S

Task constraints:

- The total time is a constant.
- The total time is a constant.

Further constraints:

- Task constraints can be handled by applying resource reclaiming.
Resource Reclaiming [Murthy2001]

Resource reclaiming for interdependent tasks

Restriction vector (RV) (static processor assignment):

No task is delayed by swapping the dispatching queues with migration algorithm.

Proof of correctness:

Assume the task set is scheduled at dispatching time $t$. If task $j$ is non-independent and task $i$ is independent, then we have $j = i$. Thus, the task set is scheduled at dispatching time $t$ and task $j$ is non-independent.

Resource reclaiming for interdependent tasks

Restriction vector (RV) (static processor assignment):

No task is delayed by swapping the dispatching queues with migration algorithm.

Proof of correctness:

Assume the task set is scheduled at dispatching time $t$. If task $j$ is non-independent and task $i$ is independent, then we have $j = i$. Thus, the task set is scheduled at dispatching time $t$ and task $j$ is non-independent.
Resource reclaiming evaluated

Practical measurements:
• There is a continuous improvement in terms of gained resources by applying:
  - basic early start
  - RV reclaiming
  - RV reclaiming with task migration,
• Incase of RV reclaiming with task migration,
the extended communication overhead can raise noticeable levels.
• There need to be a high degree of dependencies in the task-set (Pp),
in order to push through RV reclaiming with task migration.

Reclaiming in the introduced sense is applicable only to real-time systems which:
• Allow for earlier task start times.
• Allow for task migration.
• And where all dependencies can be expressed in terms of the introduced formalism.

Real-time Resource Control [Mercer97]

Resource Control Issues

Policies:
• Priority assignment problem
  - The mapping of the known and arising timing constraints and reliability considerations to linear priorities.
• Overload problem
  - Predicting and protecting the system from overload conditions.
• Flexibility problem
  - Locally adjusting the system behaviour to the current timing constraints.

Run-time environment:
• Enforcement problem
  - Handling tasks and resources which exceed their anticipated worst case limits.
• Measurement problem
  - Recording all relevant information in a sufficient resolution and frequency.
• Coordination problem
  - Synchronizing system components which are organized according to different policies.

Summary

Resource Control

- Resource synchronization primitives
  - Evaluation criteria for resource synchronization methods.
- Resource reclaiming schemes
  - Basic reclaiming
  - Early start algorithms
  - Basic reclaiming with task migration.
- Real-time resource control
  - Policies and control issues to be considered.