Dynamic Scheduling for Real-Time CORBA

Christopher D. Gill, David L. Levine, and Douglas C. Schmidt
Washington University, St. Louis
{cdgill,levine,schmidt}@cs.wustl.edu

www.cs.wustl.edu/~cdgill/research/scheduling/OMG-dynsched.ps.gz

11 January 1999

Research Supported by The Boeing Company and DARPA
Contents

- Hard real-time domain requirements
- Static/dynamic scheduling tradeoffs
- Bringing dynamic scheduling into RT-CORBA
- TAO scheduling service design
- TAO performance
- Concluding remarks
- Future work
- For further information
Hard Real-time Domain Requirements

- Support hard real-time behavior
  - Critical deadlines are met
  - Critical processing is predictable
- Utilize scarce resources efficiently
  - Conserve scarce resources
  - Minimize debugging costs
- Readily support platform (hardware/OS) upgrades
  - Ability to scale and distribute
- Reuse designs and implementations
  - Control testing, certification costs
Static and Dynamic Scheduling Approaches

- **Criticality** is an application-specified operation significance
- **Schedulability** indicates whether there are sufficient resources to perform all expected operations
- Operations are dispatched in order of **urgency**

<table>
<thead>
<tr>
<th>Scheduling Algorithm</th>
<th>Static Priority Levels</th>
<th>Schedulability Basis</th>
<th>Urgency Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Urgency First</td>
<td>one per criticality level (at least two: high, low)</td>
<td>critical set</td>
<td>“real” priority = criticality + laxity</td>
</tr>
<tr>
<td>Minimum Laxity First</td>
<td>single</td>
<td>all tasks</td>
<td>laxity</td>
</tr>
<tr>
<td>Earliest Deadline First</td>
<td>single</td>
<td>all tasks</td>
<td>deadline</td>
</tr>
<tr>
<td>Rate Monotonic</td>
<td>one per rate</td>
<td>all tasks</td>
<td>rate</td>
</tr>
</tbody>
</table>

Washington University, St. Louis
Operation Characteristics

- **Criticality** is an application defined significance of the operation missing its deadline.
- **Period** is the time interval between arrivals of dispatch requests for the operation.
- **Execution time** is the longest time used by one execution of the operation.
- **Importance** is a weaker secondary indication of the operation’s significance.

```c
struct RT_Info
{
    Time worstcase_exec_time_;  
    Period period_;            
    Criticality criticality_;  
    Importance importance_;    
    Dependency_Info dependencies_; 
};
```
Maximum Urgency First

- Static priority is assigned by criticality, dynamic priority by laxity, and static subpriority by importance.

- Urgency is an ordered tuple with static and dynamic components.

Related work:
- Stewart and Khosla ’92
- Levine, Gill, Schmidt ’98, DASC
- Gill, Levine, and Schmidt ’98, submission to IJTCCS
Sources of Schedule Variability

- Phase differences occur when rates are not harmonic
- Variable computation times are produced by operations themselves
- Variable loading is due to variation in system load or to cross-rate dependencies
Limitations of Purely Static Scheduling

- Non-periodic processing is handled inefficiently
- Time cannot be reassigned if an operation is not called, or does not use its worst case computation time
- These limitations can be addressed by pre-allocating sufficient resources to cover worst case behavior
- However, for hardware dependent resources (e.g., CPU time), software may out-pace hardware life-cycles
Limitations of Purely Dynamic Scheduling

- Imposes additional overhead for totally ordering operations
- For under-loaded systems, static approaches may offer comparable average-case behavior
- Canonical dynamic approaches (i.e., EDF and MLF) cannot isolate which operations will meet/miss their deadlines under conditions of overload
A Solution: Hybrid Static/Dynamic Scheduling

- Goal: achieve higher *utilization* by scheduling more of the unused time
- Goal: preserve *stability* of the schedule under load by isolating missed deadlines to non-critical operations
- Goal: let *applications* specify which operations are critical
- Hypothesis: with *hybrid* scheduling techniques we can achieve these goals without undue overhead or schedule instability under load
Proposed Metrics to Test the Hypothesis

- **Laxity**: time from operation completion to deadline (laxity < 0 if deadline missed)
- **Latency**: time from operation invocation (arrival of dispatch request) to operation completion
- **Jitter**: latency variation
- **Total CPU utilization**: efficiency
- **Scheduler/ORB CPU utilization**: overhead
The Cost of Dynamic Scheduling

- Solaris 2.5.1/Ultra 30
- Server and client on same CPU
- Real-Time Scheduling class
- One high priority (20 Hz) client, varying number of low priority (10 Hz) clients
- Small (< 10 percent) overhead for dynamic (MUF) vs. static (RMS) scheduling

TAO Event Channel Request Latency
## Bringing Dynamic Scheduling into RT-CORBA

### Research Issues

- Specifying QoS requirements through operation characteristics
  - Static scheduling requires *priority*
  - Also need *run-time* characteristics for dynamic scheduling: deferred decisions require more information be preserved
- Enforcing QoS requirements through scheduling mechanisms
  - *Configuring* specific enforcement mechanisms for a given policy
  - Using *factories* to produce the correct set of enforcement mechanisms for a given policy over an entire system

```c
struct RT_Info {
    Time worstcase_exec_time_;  
    Period period_;  
    Criticality criticality_;  
    Importance importance_;  
};
```
Bringing Dynamic Scheduling into RT-CORBA, Cont’d

- **Research Issues, Cont’d**
  - Reduce multiplexing and demultiplexing latency
  - Reduce layer-to-layer overhead
  - Minimize latency and latency jitter for distribution
  - Eliminate sources of priority inversion
    * Unmediated resource contention
    * Hidden scheduling

- **Related work**
  - Zinky, Bakken, and Schantz, ’95
  - Lee, Yoshida, Mercer, and Rajkumar ’96
Solution Approach

- At configuration time, the scheduling service generates static information according to specific scheduling policy
  * Assigns a static (thread) priority to each operation
  * Defines priority and sub-priority enforcement configurations
- At initialization, factories configure components according to the configuration information
- At run-time, configured components enforce priority and subpriority
The Design of TAO’s Strategized Scheduling Service

1. SPECIFY RT_OPERATION CHARACTERISTICS AND DEPENDENCIES
2. POPULATE RT_INFO REPOSITORY
3. ASSESS SCHEDULABILITY
4. ASSIGN STATIC PRIORITY AND STATIC SUBPRIORITY
5. MAP STATIC PRIORITY, DYNAMIC SUBPRIORITY, AND STATIC SUBPRIORITY INTO DISPATCHING PRIORITY AND DISPATCHING SUBPRIORITY
6. ASSIGN DISPATCHING QUEUE CONFIGURATION
7. SUPPLY DISPATCHING QUEUE CONFIGURATION TO THE ORB
8. CONFIGURE QUEUES BASED ON DISPATCHING QUEUE CONFIGURATION
9. SUPPLY STATIC PORTIONS OF DISPATCHING PRIORITY AND DISPATCHING SUBPRIORITY TO THE ORB
10. DYNAMIC QUEUES ASSIGN DYNAMIC PORTIONS OF DISPATCHING SUBPRIORITY (AND POSSIBLY DISPATCHING PRIORITY)
TAO Performance on a Commercial RT Platform

- LynxOS 3.0.0/MCP750
- Server and client on same CPU
- One high priority (20 Hz) client, varying number of low priority (10 Hz) clients
- TAO preserves high priority client response latency and jitter with increasing number of low priority clients
Distribution with TAO on Commercial RT Platform

- LynxOS 3.0.0/MCP750/CompactPCI
- Server and client on different CPUs, and communicate via TCP/IP over Fast Ethernet
- High priority client response unaffected by distribution
- Low priority client response improves due to parallelism
- Distribution has small impact on jitter
Concluding Remarks

- Hybrid static/dynamic scheduling supports higher utilization while preserving the hard real-time behavior of critical operations under load.
- Dynamic scheduling overhead appears sufficiently small.
- Priority must be supplemented with additional information to achieve dynamic or hybrid scheduling.
Future Work

- Future scheduling work:
  - Support for distributed scheduling
  - Support for additional mechanisms, policies, and operation characteristics
  - End-to-end priority preservation along a request-response path

- Future performance evaluation work:
  - Investigate effects of priority preservation mechanisms on end-to-end performance
  - Investigate performance of various pluggable network protocols
  - Investigate performance tradeoffs of static, dynamic, and hybrid scheduling strategies and enforcement mechanisms
For Further Information

- **TAO Scheduling:**
  - www.cs.wustl.edu/~schmidt/dynamic.ps.gz
  - www.cs.wustl.edu/~schmidt/DASC-98.ps.gz
  - www.cs.wustl.edu/~schmidt/TAO.ps.gz

- **TAO RT Performance Results:**
  - www.cs.wustl.edu/~schmidt/RT-OS.ps.gz
  - www.cs.wustl.edu/~schmidt/words-99.ps.gz
For Further Information, Cont’d

- **TAO:**
  
  www.cs.wustl.edu/~schmidt/TAO.html

- **ADAPTIVE Communication Environment (ACE):**
  
  www.cs.wustl.edu/~schmidt/ACE.html

- **These slides:**
  
  www.cs.wustl.edu/~cdgill/research/scheduling/OMG-dynsched.ps.gz