Adaptive QoS Management in Middleware

Christopher D. Gill
Center for Distributed Object Computing
Department of Computer Science
Washington University, St. Louis

cdgill@cs.wustl.edu

Friday, April 21, 2000

Research supported by: Boeing, BBN, DARPA
Motivation: DRTMCS

• Distributed Real-Time Mission Critical Systems:
  – Perform both critical and non-critical operations with timing constraints
  – Must adapt to rapidly changing environment
  – Timing characteristics and requirements may vary with environmental factors
    • Dynamic Scheduling, Admission Control
• Additional Requirements
  – Cost containment through portability, reuse
• Proposed Solution:
  – Adaptive QoS Management in Middleware
Why Middleware?

- OS kernel has a direct view of system resources
  - Rajkumar et al. (CMU) Resource Kernel
  - Peterson et al. (Arizona) Scout OS
- However, DRTMCS cost requirements may indicate heterogeneous COTS OS components
- Furthermore, may need end-to-end agreement at higher layers (e.g., priority lanes)
  - Clark: End-to-end argument
- Application layer is end-to-end, offers flexibility
- Yet, decoupling application itself from QoS management promotes reuse, less error-prone
Hybrid Static/Dynamic Scheduling

- Hybrid static/dynamic scheduling techniques preserve critical resource availability while redistributing additional resource availability for beneficial non-critical tasks.

- Original hybrid scheduling approach consisted of two phases:
  - Off-line schedulability assessment, static priority assignment, and dispatcher configuration.
  - On-line static and/or dynamic dispatching within each static priority level.

- Run-time changes in static scheduling parameters such as operation rate were not addressed.
Results: Static RMS Local Adaptation

- The set of critical operations must be feasibly schedulable each time frame
- Adding non-critical operations is useful to increase both functionality and total CPU utilization
- Popular RMS strategy does not protect critical operations under conditions of overload
- As the number of non-critical operations increases, impact on critical operations can worsen

- Data from a real-time avionics application executing on realistic hardware
- Transition from an already overloaded environmental state into another with even higher load
  - Due to admission of additional high rate non-critical operations into the schedule
- Empirical results show missed deadlines for critical operations with lower rates
Results: MUF Local Adaptation

- Maximum Urgency First is a hybrid static/dynamic scheduling strategy
- MUF protects critical operations under conditions of overload
- Even as the number of non-critical operations increases, impact on critical operations remains negligible
- Without operation cancellation, operations are dispatched whether or not they will make their deadlines

- Data from the same experiment and same state transition shown previously for RMS
- Here, all critical operations make their deadlines
- Some non-critical operations make their deadlines, while others miss theirs
Results: MUF + Cancellation

- Operations that miss their deadlines have diminished (e.g., zero) value
- Canceling operations that cannot meet deadlines reduces wasted CPU usage
- Cancellation decision was made at the point of dispatch to the application component, via an Adapter
- Trade-off between the overhead of the cancellation decision and the additional control over wasted work

- Data from the same experiment and same state transition shown previously for RMS and MUF
- Slightly fewer non-critical operations make their deadlines than without cancellation
  - Due to pessimistic cancellation using worst-case rather than best-case execution time
- Very few dispatched operations subsequently missed their deadlines
Results: Dispatching Overhead

- Time probes were used to measure time spent in dispatching code
- Overall, dynamic dispatching overhead for MUF was higher than, but comparable to, static dispatching overhead for RMS
- Anomalous spikes in static and dynamic measurements attributed to non-determinism in experimental setup (i.e., network interrupts)
- Heavier loading of dynamic queues may occur with more diverse non-critical utilization in current work
- Hypothesis: operation cancellation should help mitigate loading effect
Supporting Adaptive Admission Control

- The original TAO off-line scheduler has been redesigned to provide efficient on-line schedulability assessment and priority assignment.
  - Design patterns such as Visitor and Strategy were used to reduce algorithmic overhead and improve flexibility
- This allows its use in adaptive admission control decisions by HTC’s RT ARM, in response to environmental changes
Enhancing Adaptive Admission Control

- However, strict layering between scheduler and RT ARM gives quadratic $O(n^2)$ admission control performance.
- Increasing integration between RT ARM and scheduler supports admission control as part of $O(n \cdot \log(n))$ schedulability sorting pass.
- RT ARM can plug policy for admission control into generic TAO scheduling framework.
Conclusions

• Our empirical results show hybrid static/dynamic scheduling strategies can be employed effectively in a realistic real-time avionics application
• Operation cancellation can reduce the amount of wasted work, providing additional adaptive control in overload
• Optimized scheduling service provides efficient on-line adaptive resource analysis and allocation capabilities
• Increased integration of the TAO Scheduling Service and RT ARM, using the Strategy design pattern, can yield performance improvements over strictly layered integration
Related Work

- DARPA Quorum Integration Research Efforts
  - (BBN) QuO
    - Highest Level Adaptive Management Middleware
  - (Honeywell HTC) RT ARM
    - Adaptive Services: Resource Managers
  - (Washington U) TAO
    - Adaptive Services and Core Middleware
- Technology Transfer and Integration Research
  - (Boeing St. Louis) Applications and Testbeds