Real-Time Databases

CSC 436 – Fall 2002

Outline

- Introduction and definitions
- Overview of real-time systems
- Requirements of a real-time database
- Real-time database models
- Current research

Introduction

- Definition: A real-time database manages time-constrained data and time-constrained transactions
- System uses environmental data as input and must produce output to control its environment
- Component of a real-time system

Real-Time Systems

- A *real-time system* is a system in which the time at which the output is produced is significant.
- The input corresponds to some "movement" in the physical world, and the output has to relate to the same movement.
- Not necessarily “fast”

Some Examples

- Flight control in avionics
- Process control in industrial plants
- Robotics
- Patient monitoring
- Command and control

Some Definitions

- *job* - unit of work scheduled and executed
  - every job executes on some resource
- *task* - set of related jobs
- *timing constraint* - constraint imposed on behavior of a job
  - *ex*: move robot arm within 3 seconds to pick up item on conveyer belt

Types of Timing Constraints

- *release time* - instant in time job becomes available for execution
- *deadline* - instant in time by which job's execution is require to complete
- *relative deadline* - maximum allowable response time

Modes of Real-Time

- *Hard real-time constraint* - failure to meet it is considered a fatal flaw
  - *ex:* submarine maneuver to avoid a torpedo by some deadline

Modes of Real-Time

- *Soft real-time constraint* - late computation is undesirable, but not catastrophic
  - late result usually better than none, but has less value
  - *ex:* read item from a database by some deadline

Modes of Real-Time

- *Firm real-time constraint* - late computation is useless, but not catastrophic
  - *ex:* robot arm maneuver to pick up object from a conveyer belt by a deadline
  - some literature only discusses soft and hard

Hard Real-Time Systems

- Require guarantee that all timing constraints will be met.
  - By proof OR by exhaustive simulations
- Generally static - all tasks known a priori
- Scheduling is static - done ahead of time

Soft Real-Time Systems

- No guarantee required
- Best-effort approach
- May be dynamic - tasks enter system at any time
- Scheduling usually dynamic - schedule tasks as they enter system.

Example Hard Real-Time System

- **Automatically controlled train:**
- For train to STOP - use current speed and safe deceleration rate to compute stop time
- Impose constraints on response time of jobs that sense and process stop signal and activate brake.
- Without guarantee, train could crash if timing constraints missed.

Example Soft Real-Time System

- **Telephone Network**:
- Make call
  - sequence of jobs route signal through switches
  - we expect put through in short amount of time
- Probabilistic timing constraint
  - jobs must complete < 10 sec for 95% of time
  - < 20 for 99.95% of time

Predictability

- Behavior must be predictable to *guarantee* all timing constraints are met
- Accurately analyze timing behavior
  - Resource utilizations
  - Worst case, average case, etc.

Imprecision

- May need to be allowed to meet timing constraints
- Trade-off between timeliness and precision
- To meet timing constraints, may need to relax serializability constraints

Scheduling

- Scheduler
  - allocates resources to jobs
- schedule
  - assignment of all jobs on available resources (processors)

Hard Real-Time Scheduling

- Feasible schedule
  - all timing constraints are met
  - set of jobs is *schedulable* if there is at least one feasible schedule of those jobs
- Optimality
  - a scheduling algorithm is *optimal* if it always produces a feasible schedule when one exists

Hard Real-Time Scheduling

- Different approaches to hard real-time scheduling
- Priority-driven scheduling
- Optimality and non-optimality
- Scheduling with resource contention
  - Priority inheritance

Soft Real-Time Scheduling

- Performance measures
  - lateness - difference from deadline
  - tardiness - how far after deadline
  - miss rate - percentage of missed deadlines
  - loss rate - percentage of discarded tasks

Example Priority Driven Scheduling Algorithms

- Rate-monotonic (RM)
  - highest priority to task with shortest period
- Earliest deadline first (EDF)
  - highest priority to task with closest deadline
- Least slack time (LST)
  - highest priority to task with shortest slack time
  - slack time = relative deadline - exec time
- Latest release time (LRT) (reverse EDF)
  - highest priority to task with latest release time

Optimality

- EDF - When preemption is allowed and jobs do not contend for resources, EDF is optimal.
- LRT and LST - also optimal under same conditions

Scheduling with Resource Contention

- When non-preemptible resources are used by processes along with CPU (semaphore for instance), blocking can occur.
- *priority inversion*: if a low priority task blocks a higher priority task
  - bad in real-time
  - need to bound it to be able to predict amount of time this will occur

Priority Inversion Example

- Task T3 holds resource S
- Task T1(higher priority) wants resource S
  - non-preemptible, so must wait
- Task T2 enters system and takes over CPU
  - an intermediate task that does not use S
  - causes T3 to block T1 for longer time
  - number of such intermediate tasks is unbounded

Priority Inheritance - A Solution

- Allow T3 to “inherit” the priority of T1 while it holds the resource
- T3 runs at priority 1, so T2 will not preempt it on the CPU
- Bounds priority inversion

Real-time Database Requirements

- Consistency maintenance
- Bounded imprecision
- Predictability
- Transactions

Consistency Maintenance

- Four forms of consistency:
  - Transaction logical
  - Data logical
  - Transaction temporal – treat transactions as real-time tasks
  - Data temporal – constrains how old data item can be and still be valid

Bounded Imprecision

- Conflict between temporal constraints and logical constraints
  - T1 updates data item X
  - T2 reads X
  - If T2 is reading X, T1 not allowed to update
    - Logical constraint
  - If X may become old, T2 should update
    - Temporal constraint
  - May not be able to maintain both
    - Trade off one for the other

Predictability

- Required for hard real-time, desirable for soft and firm
  - Bound wcet for all database primitives
  - Bound sizes of tables and data structures
  - Bound waits for buffers
  - Bound blocking time due to concurrency control
  - Bound transaction aborts
  - Bound indexing for locating data items
  - Use real-time scheduling

Transaction

- Three types of transactions
  - Sensor (write-only)
  - Read-only
  - update

ACID Properties Redefined

- *Atomic* – selectively applied to parts of transactions that need consistent data
- *Consistent* – includes all 4 forms – need trade-off
- *Isolated* – no longer independent – must allow for communication and synchronization
- *Durable* – still persistent – but may become old and then thrown away

Real-Time Database Model

- (Ramamritham)
- Real-time data: d: (value, avi, timestamp)
  - Absolute temporal consistency:
    - $| t - d.timestamp | \leq d.avi$
  - Relative temporal consistency:
    - $\forall d1, d2 \in R, |d1.timestamp - d2.timestamp| \leq R.rvi$

Real-time Database Model

- Real-time transactions
  - Characterized along 3 dimensions:
    - How data is used
      - Read-only, Sensor, Update
    - Origin of timing constraint
      - Data temporal consistency or system
    - Model of real-time
      - Hard, soft, firm

Current Real-time Database Research

- University of Virginia
- University of Massachusetts – Amherst

UVA

- **StarBase Overview**
  - built on top of the RT-Mach operating system
  - supports real-time transactions with firm deadlines
  - seeks to minimize the number of high-priority transactions which miss their deadlines
  - uses no *a priori* information about the transaction workload
- **Problems Faced by Real-Time Databases**
  - resource contention
  - data contention
  - specifying/enforcing timing constraints

UVA

- **Dealing with Resource Contention**
  - RT-Mach provides:
    - priority-cognizant thread scheduling and a real-time thread model
    - *Basic Priority Inheritance* synchronization for non-preemptible resources
  - StarBase employs these features to:
    - service transactions in priority order
    - ensure transactions progress according to priority
    - ensure transactions access resource managers in priority order

UVA

- **Dealing with Data Contention**
  - WAIT-X(S) optimistic concurrency control
  - priority-based commit test
  - Precise Serialization to reduce unnecessary aborts
- **Enforcing Deadlines**
  - RT-Mach provides:
    - real-time thread model
    - real-time clocks and timers
    - StarBase uses these features to abort transactions and reply at or before deadline
  - StarBase:
    - avoids race conditions between deadline handler and transaction

UMass

- **Real-Time Active Database Experimental Research**
- RADEx is the first real-time, active, object-oriented, temporal database simulator. It is being used to study
  - Priority assignment and real-time transaction scheduling in active real-time databases
  - Real-time logging and recovery
  - Consistency and scheduling in temporal databases
  - Multimedia databases

OMG Data Distribution Service for Real-Time Systems

- OMG – Object Management Group
  - Standards organization
    - CORBA
    - UML
- Currently working on standard for delivering distributed real-time data

OMG DDS

- Distributed Shared Memory
  - classic model to provide data-centric exchanges
  - hard to implement over internet
- Data-Centric Publish-Subscribe (DCPS)
- Higher level data model
  - aggregation and coherence relationship
  - updates to sub-elements

OMG DDS

- Publish-Subscribe (PS) system
  - concept of publishers and subscribers
  - information posted by publisher automatically delivered to subscriber of related topic
- DCPS adds data model to PS to express
  - types and relationship among data-items
  - aggregation & consistency relationships
  - QoS requirements
- DCPS in a way counterpoint to Notification Service

DCPS vs. Notification Service

- domain of disclosure – data
- "channel" functions as data-stream
- events notify of data and data-stream QoS change
- definition, configuration, QoS relate to data
- data structured to provide refinement in data-channels
- many data-stream channels no/or few filters

- domain of disclosure - events
- "channel" functions as event-stream
- data piggybacked to events
- QoS relate to events
- consumers use filters to select events of interest
- few event channels & filters

DCPS vs. Notification Service

- efficient real-time delivery mechanism of frequent data updates
- scales to many suppliers and even more recipients
- accommodates variety of QoS settings

- filtering and distribution streams of uncorrelated and often asynchronous events to consumers w/ interest on subset of these events

Data Model

- tree structure
  - node or leaf identified by topic or key and has associated data
  - branch – sequence of topics started from root
  - publishers provide values for tree elements
  - subscribers register their interest in individual elements or complete branches
  - Air-traffic control systems (Flight Plan: the route, the aircraft identification, etc.)

Data Model

- shared data implies issue of policies & multiple writers (Data Ownership)
- policy to obtain, retain, yield ownership
- granularity of ownership
- lifecycle of ownership
- “permanent” or “leased” ownership

Data Distribution Server

- structural model to represent aggregation relationships
- publish – subscribe interface that allows to subscribe to subsets of data
- quality of service model tailored to data-centric model, where data-delivery can be customized
- data ownership model

EMPRESS RDBMS

- **The Embedded Real-Time Database**
  (http://www2.empress.com/)
- full-featured database engine designed for embedded, real-time applications
- provides total control to the developer delivering high-performance, deterministic data management
- compact, agile and maintenance-free and is suited for embedded systems, real-time, communications, military & defense, process control and scientific & engineering applications
- runs on Unix, Linux, Windows and Real Time systems.