Transaction Processing in a Service Oriented Architecture

Dr Mark Little
Technical Development Manager, Red Hat
Overview

• Fault tolerance
• Transaction fundamentals
  – What is a transaction?
  – ACID properties
• Distributed transactions
• The SOA effect on transactions
Fault tolerance

• Machines and software fail
  – Fundamental universal law (entropy increases)
  – Things get better with each generation, but still statistically significant

• Failures of centralized systems difficult to handle

• Failures of distributed systems are much more difficult
Fault tolerance techniques

• Replication of resources
  – Increase availability
    • Probability is that a critical number of resources remain operational
    • “Guarantee” forward progress
  – Tolerate programmer errors by heterogeneous implementations

• Spheres of control
  – “Guarantee” no partial completion of work in the presence of failures

• Often a duality
  – “Understanding the Role of Atomic Transactions and Group Communications in Implementing Persistent Replicated”, Proceedings of the 8th International Workshop on Persistent Object Systems, California, USA, 1998
What is a transaction?

• Mechanistic aid to achieving correctness
• Provides an “all-or-nothing” property to work that is conducted within its scope
  – Even in the presence of failures
• Ensures that shared resources are protected from multiple users
• “Guarantees” the notion of shared global consensus
  – Different parties in different locales have the same view of the transaction outcome
ACID Properties

- Atomicity
- Consistency
- Isolation
- Durability
Atomicity

- within the scope of a transaction
  - all changes occur together or no changes occur
- atomicity is the responsibility of the transaction manager
- for example - a money transfer
  - debit removes funds
  - credit add funds
  - no funds are lost!
Two-phase commit

• Required when there are more than one resource managers (RM) in a transaction
• Managed by the transaction manager (TM)
• Uses a familiar, standard technique:
  – marriage ceremony - Do you? I do. I now pronounce ..
  – It is only a consensus protocol
• Two - phase process
  – voting phase - can you do it?
    • Attempt to reach a common decision
  – action phase - if all vote yes, then do it.
    • Implement the decision
Consistency

- Transactions scope a set of operations
- Consistency can be violated within a transaction
  - Allowing a debit for an empty account
  - Debit without a credit during a Money Transfer
  - Delete old file before creating new file in a copy
- Transaction must be correct according to application rules
- Begin and commit are points of consistency

- Consistency preservation is a property of a transaction, not of the TP system (unlike the A, I, and D of ACID)
Isolation

- Running programs concurrently on same data can create concurrency anomalies
  - the shared checking account example
Isolation

• Transaction must operate as a black box to other transactions
• Multiple programs sharing data requires concurrency control
• When using transactions
  – programs can be executed concurrently
  – BUT programs appear to execute serially
Isolation

Begin()
read BAL
subtract 100
write BAL
Commit()

Bal = 100
Bal = 0
Bal = 0

Begin()
read BAL
Not Enough
Rollback()

Oh NO!!
Durability

• When a transaction commits, its results must survive failures
  – must be durably recorded prior to commit
  – system waits for disk ack before acking to user

• If a transaction rolls back, changes must be undone
  – before images recorded
  – undo processing after failure
Transactions for SOA

• Business-to-business interactions may be complex
  – involving many parties
  – spanning many different organisations
  – potentially lasting for hours or days

• Cannot afford to lock resources on behalf of an individual indefinitely

• May need to undo only a subset of work

• So the search has been on, because …
ACID-ic SOA?

• ACID transactions implicitly assume
  – Closely coupled environment
    • All entities involved in a transaction span a LAN, for example.
  – Short-duration activities
    • Must be able to cope with resources being locked for periods
• Therefore, do not work well in
  – Loosely coupled environments!
  – Long duration activities!
However …

• Web Services are as much about interoperability as they are about the Web
• In the short term Web Services transactions will be about interoperability between existing TP systems rather than running transactions over the Web
OASIS WS-TX Goals

• 4th attempt at standardising
• Support range of use cases
• “One-size does not fit all”
  – “Make each program do one thing well; to do a new job, build afresh rather than complicate old programs by adding new features”, Doug McIlroy, inventory Unix pipes
  – Therefore a single protocol cannot cope with all requirements
• Interoperability with existing transaction infrastructures
ACID transaction model

• Assume ACID transactions
  – High degree of trust
  – Isolation for duration of transaction
  – Backward compensation techniques
  – Does not allow heuristic outcomes

• Integration with existing transaction systems
  – Important to leverage investments

• Interoperability between transaction systems
What characteristics are right?

- Need to be able to relax the strict ACID properties
- Need to put control of some into hands of service developer
  - Is consistency (or consensus) important?
- May need a notion of a central coordinator
  - But may not!
  - Or something with a fuzzy idea of what’s going on
Relaxing isolation

• Internal isolation or resources should be a decision for the service provider
  – E.g., commit early and define compensation activities
  – However, it does impact applications
    • Some users may want to know a priori what isolation policies are used
• Undo can be whatever is required
  – Before and after image
  – Entirely new business processes
Relaxing atomicity

- Sometimes it may be desirable to cancel some work without affecting the remainder
  - E.g., prefer to get airline seat now even without travel insurance
- Similar to nested transactions
  - Work performed within scope of a nested transaction is provisional
  - Failure does not affect enclosing transaction
- However, nested transactions may be too restrictive
Structuring transactions

• Could structure transactional applications from short-duration transactions
  – Release locks early
  – Resulting application may still be required to appear to have “ACID” properties
    • May require application specific means to restore consistency

• A transactional workflow system could be used to script the composition of these transactions
Relaxation of consistency

• ACID transactions (with two-phase commit) are all about strong global consistency
  – All participants remain in lock-step
  – Same view of transaction outcome (atomic)
• But that does not scale
  – Replication researchers have known about this for years
    • Weak consistency replication protocols developed for large scale (number of replicas and physical deployment)
    • Merging of caching and replication protocols
      – Local domains of consistency
    • Cannot “stop the world” and enforce global consistency
  – Some transaction research into this, but industry pushing global consistency
    • Starting to see a change
Heisenberg’s Uncertainty Principle

- Cannot accurately measure both position and momentum of sub-atomic particles
  - Can know one with certainty, but not the other
  - Non-deterministic measurements

- Large-scale/loosely-coupled transactional applications suffer the same effect
  - Can know that all services will eventually see same state, just not when
  - Or at known time can determine state within model/application specific degree of uncertainty

- Or another way of thinking about it …
  - No such thing as simultaneity in data space as there isn't in space-time
    - “Data on the Outside vs. Data on the Inside”, by Pat Helland
No global consensus

• **Split transactions into domains of consistency**
  – Strong consistency within domains
  – Some level of (known) consistency between domains
    • See “A method for combining replication and caching”, *Proceedings of International Workshop on Reliable Middleware Systems, October 1999*.
  – Resolve inconsistencies at the business level
    • Don’t try and run consensus protocols between domains

• **Consistency related to isolation**
  – Put into the control of the service and application developers
OASIS Business Process

• All parties reside within *business domains*
  – Recursive structure is allowed
  – May represent a different transaction model
  – No required notion of consistency between domains

• **Business process is split into** *business tasks*
  – Execute within domains
  – Compensatable units of work
    • Forward compensation during activity is allowed
      – Keep business process making forward progress

• **Consistency is application (service) dependent**

• **Atomicity (or lack thereof) in the “large” is taken for granted**
SOA or scale?

- Problems with transactions pre-date SOA
- Current issues with database technologies are not SOA specific either
- Problems are two-fold
  - Scalability (size and geographic distributed nature)
  - Control over the infrastructure/services
    - Trust comes into this too
- Much research in the 1990’s
- SOA (and Web Services) bring this to the foreground
  - REST would be just as appropriate
Future directions

- One size does not fit all!
- Business domains will impose different requirements on implementers
  - Essentially construct domain-specific models
  - Real-time
- The range and requirements for such extended models are not yet known
  - Do not restrict implementations because we don’t know what we want yet
- Still a very active area of research and development