

JBoss Transactions

Dr Mark Little

Red Hat

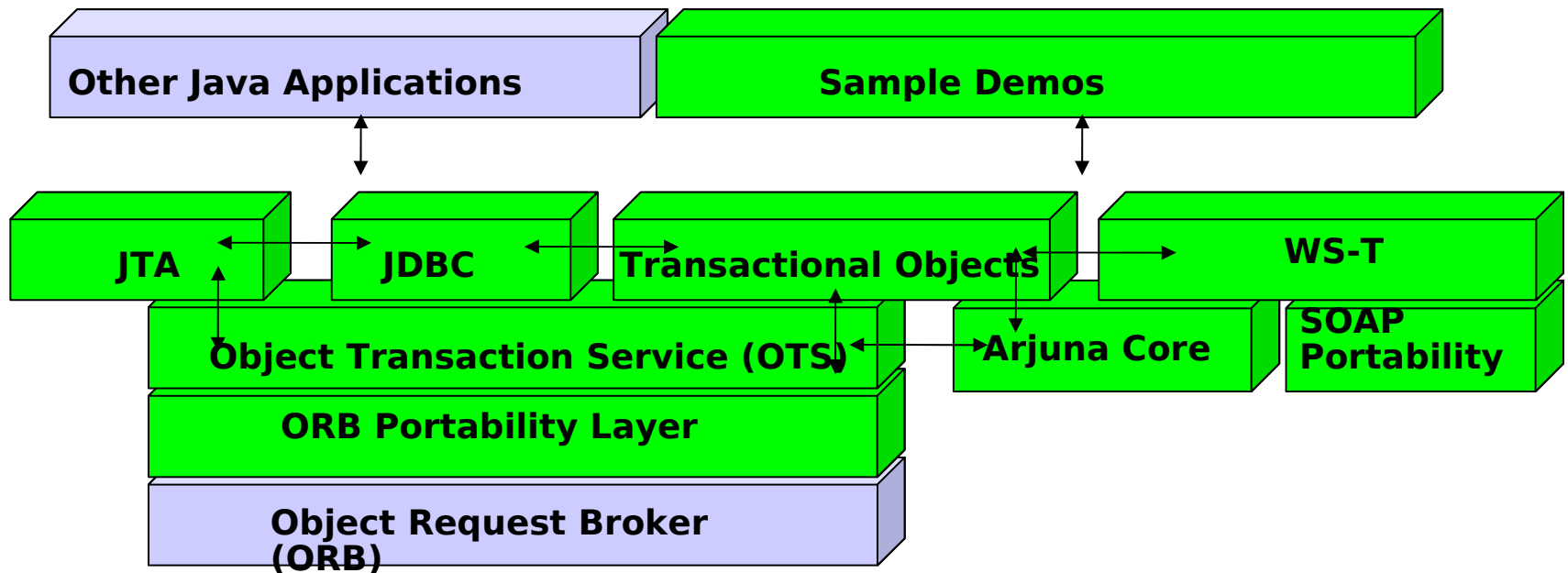
What is a transaction?

- 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
- Complexity is hidden by the transaction system

What is JBoss Transactions?

- JBoss Transactions
 - Default transaction service in JBossAS 4.2
 - Based on
 - JTA 1.0.1
 - JTS 1.0 (OTS 1.4)
 - WS-Coordination, WS-Atomic Transaction, WS-Business Activity
 - Demonstrated interoperability with IBM and MSFT
 - Used at HP World for Web Services seminars
 - Licensed to TIBCO, webMethods, Mizuho and others
 - Does not require an application server to run
 - I18N and L10N

JBossTS Components



 JBossTS Package

←→ Interact with each other

ArjunaCore

- Stand-alone transaction engine
- Full failure recovery
- ACID properties can be relaxed
 - Gray's matrix of transaction models
 - Does not restrict to XA
- Designed to be used stand-alone
 - Own set of APIs
- Similar to what MSFT are doing with Indigo

JEE support

- Local and remote JTA implementations
- World's first JTS implementation
 - Used to push the OTS specification
 - Completely multi-thread aware
- Portable to a number of ORBs
 - E.g., Orbix 2k, JacORB, JDK ORB, ...
- Distributed failure recovery
- Sub-transaction aware resources

What is JTS?

- The Java™ language mapping of the OMG's Object Transaction Service (OTS)
- supports multiple transaction models
 - flat
 - nested (optional)
- interoperability between OTS and X/Open DTP model
- flexible transaction propagation
 - implicit
 - explicit

Why do I need to know this?

- EJB™ architecture mandates JTS for interoperability
 - Actually only on-the-wire message formats required
- The OTS specification is more complete than the JTA
 - Read both together to get an idea of how distributed transactions work

Interposition

- Allows a subordinate coordinator to be created
- Interposed coordinator registers with transaction originator
 - Form tree with parent coordinator
 - Application resources register locally
- JBossTS supports interposition for implicit and explicit propagation

Further features

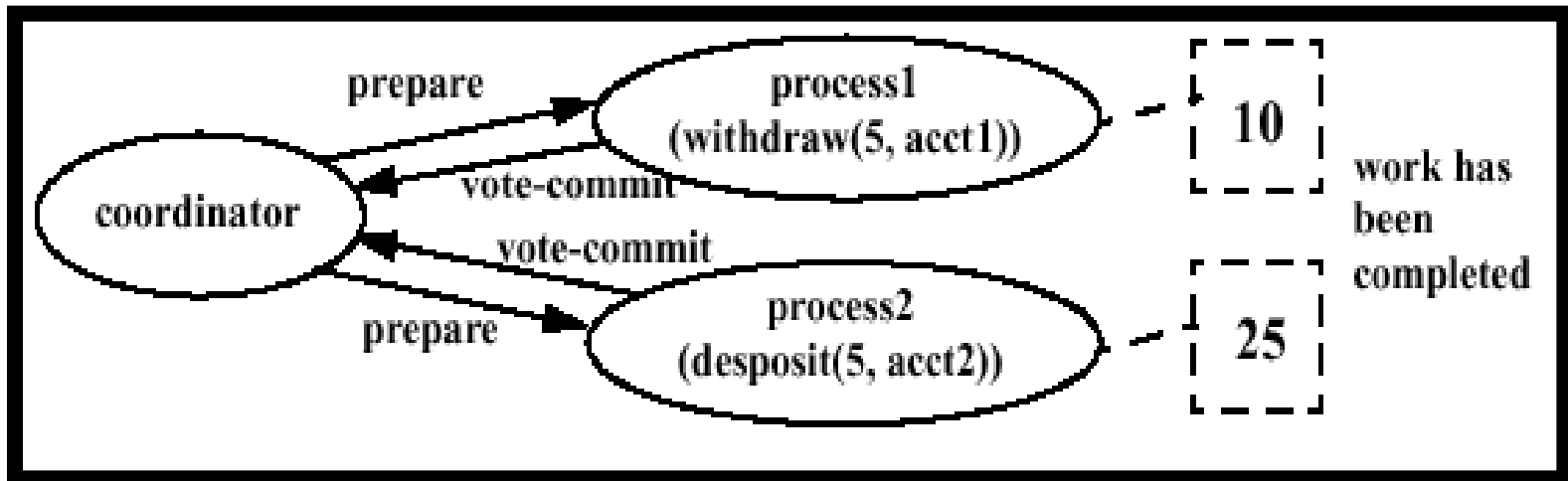
- Checked transactions
 - Per transaction basis
- Last resource commit optimization
- Asynchronous commit protocol
 - Prepare and commit
- Transaction management tools
 - Heuristic resolution

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 ..**
- 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

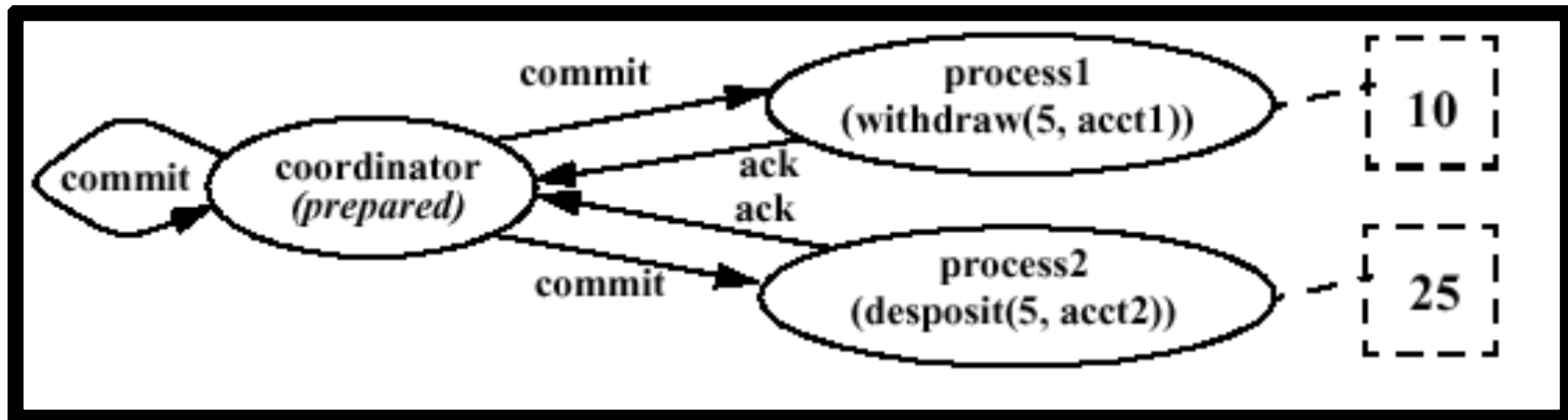
Phase One

- Transaction manager tells resource managers to get ready to commit
- Resource managers log information necessary to ensure they can commit
- If any resource manager cannot, transaction is rolled back
- If all agree to commit, transaction manager logs transaction outcome of commit



Phase two

- If all resource managers vote yes, TM instructs each RM to commit
- If failures occur, TM will contact RM when they are corrected and continue to tell RM to commit



Handling failures

- Presumed Abort Strategy
 - can be stated as « when in doubt abort »
 - any failure prior the commit phase lead to aborting the transaction
- A coordinator or a participant can fail in two ways
 - it stops running (crashes)
 - it times out waiting for a message it was expecting
- A recovered coordinator or participant uses information on stable storage to guide it recovery

Failure recovery

- Automatic failure recovery daemon
 - Runs periodically
 - Can be driven directly
- Recover inflight transactions
- Recover resources
 - different recovery mechanisms required for each resource type
 - different mechanisms can be easily added

Why do I care about recovery?

- Why is it critical?
 - Sh*t happens!
 - Insurance policy
 - Infrastructure uses log to drive atomic outcomes
- Without recovery, a transaction service is of limited use
 - Consensus is easier to achieve

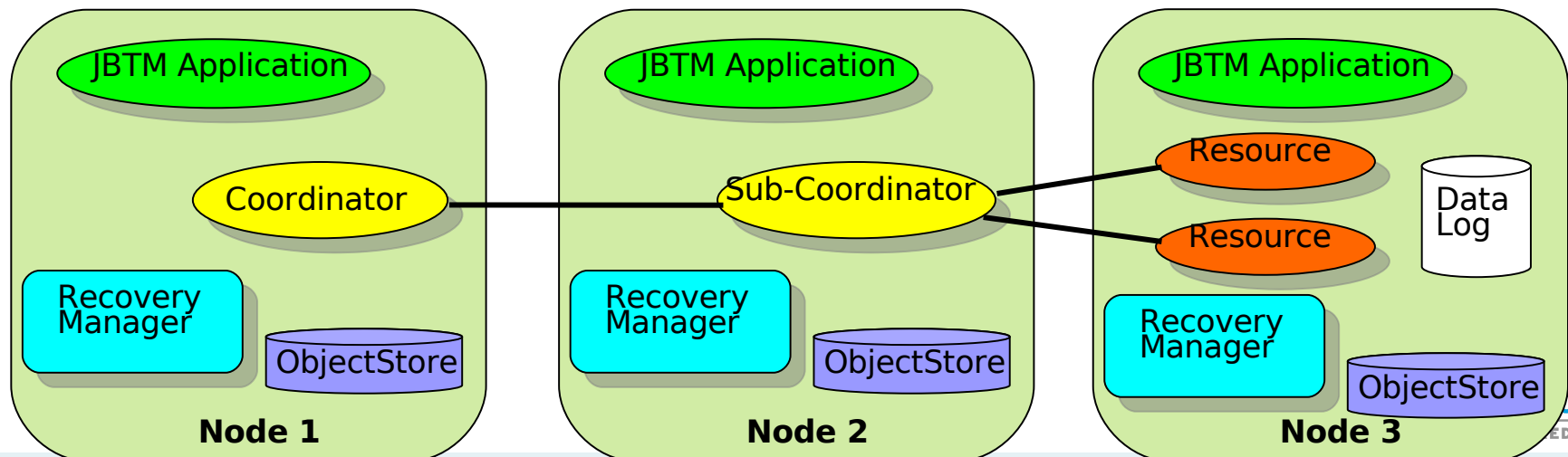
Logs and Recovery

- When a participant node starts, it goes through a crash- recovery phase
 - checks logs for outstanding transactions
- Unresolved transactions leads to recovery work
- Recovery work depends on participant's role

- Within JBossTS
 - Information on transactions and resources are stored in the ObjectStore
 - The Recovery is delegated to the Recovery Manager process

The Recovery Manager

- Recovery requires a Recovery Manager process
 - Stand-alone
 - Embedded (application server)
- To start the Recovery Manager stand-alone
 - `com.arjuna.ats.arjuna.recovery.RecoveryManager`
- Configuration - The Recovery manager reads properties through
 - `RecoveryManager-properties.xml`



RecoveryManager

- Recovery manager drives RecoveryModules
 - Typically one module per type of recoverable resource
 - E.g., file system, XAResources, transactions
- Runs periodically but can be driven directly
 - `com.arjuna.ats.arjuna.tools.RecoveryMonitor`

Periodic Recovery

- Recovery Manager scans the ObjectStore to retrieve transactions and resources which need to be recovered
- Scans made by instances of classes that implements the interface `com.arjuna.ats.arjuna.recovery.RecoveryModule` which defines methods:
 - `periodicWorkFirstPass`
 - `periodicWorkSecondPass`

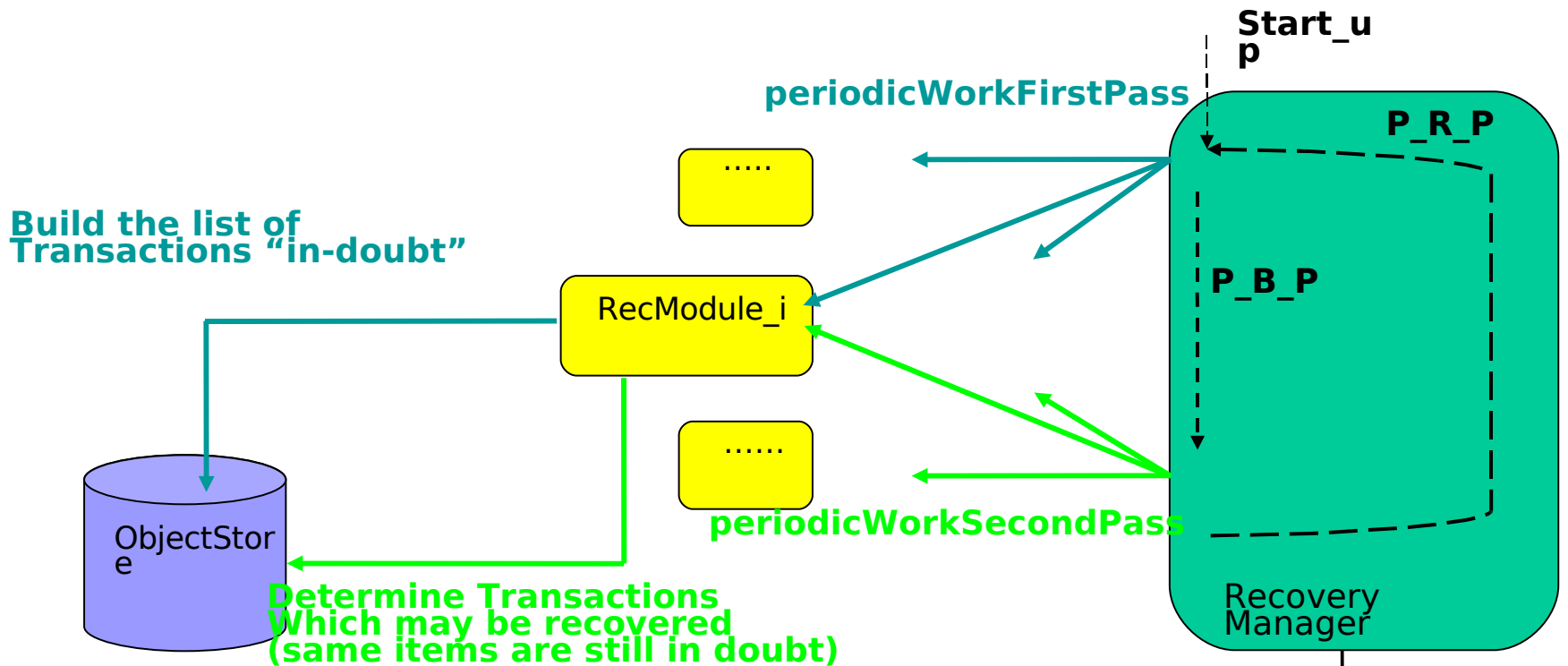
Periodic Recovery

- The Recovery Manager invokes both methods on each module defined in its properties file
 - Methods are invoked at regular intervals
 - `com.arjuna.ats.arjuna.recovery.periodicRecoveryPeriod` (default 120 seconds)
 - To call the first pass. Check transactions “in-doubt”
 - `com.arjuna.ats.arjuna.recovery.recoveryBackoffPeriod` (default is 10 seconds)
 - To call the second pass after the first . Detect transactions to recover.

Recovery Modules

- Recovery modules are registered with the RecoveryManager using properties that start with “com.arjuna.ats.arjuna.recovery.RecoveryExtension”.
- Modules are invoked on each pass of the periodic recovery in the sort-order of the property names
- Advanced users may create their own recovery modules and register them with the Recovery Manager

The Recovery Manager in action



Original process of Transactions "in-doubt" are checked before recovery (com.Arjuna.CosRecovery.Contact)

RecoveryManager_2_1.properties

Recovery Module_1
 Recovery Module_2

 Recovery Module_i

P_R_P: PERIODIC_RECOVERY_PERIOD
P_B_P: PERIODIC_BACKOFF_PERIOD

XA specifics

- XAResources are the participants that represent the backend RMs
- JBossTS supports two types of implementation
 - Serializable XAResources
 - Non-serializable XAResources

Serializable XAResources

- Coordinator serializes reference to participant
- JBossTS XAResource wrapper records
 - Serializes state after successful prepare
 - Information on coordinator too

Non-serializable XAResources

- Coordinator serializes reference to participant
- JBossTS XAResource wrapper records
 - The fact that it needs a new XAResource instance for recovery
 - information on coordinator

Web Services transactions

- 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

Relaxing isolation

- Internal isolation of services should be a decision for the 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
 - Relaxing isolation

WS-AT/WS-BA

- Specifications released by Arjuna, BEA, IBM, IONA and Microsoft
- Separate coordination from transactions
- Define two transaction models
 - AtomicTransaction
 - Closely coupled, interoperability
 - Business Activities
 - Compensation based, for long duration activities

Business Activities

- Workflow-like coordination and management
- Business activity can be partitioned into tasks
 - Parent and child relationships
 - Select subset of children to complete
 - Parent can deal with child failures without compromising forward progress
- Tasks can dynamically exist a business activity
- Tasks can indicate outcome earlier than termination
 - Up-calls rather than just down-calls

Summary

- Product features
 - High performance and reliability
 - Manageability and configurability
 - Standards compliance
 - Modular architecture to optimise footprint
 - Pure Java implementation
- Deployment options
 - Application server agnostic
 - Deployable in or outside a J2EE application server