JCC's LogMiner Loader

High Performance JDBC Interface and Other New Features

Thomas H. Musson ♦ Jeffrey S. Jalbert ♦ Cheryl P. Jalbert
Keith W. Hare ♦ Jeff Haidet

JCC Consulting, Inc.

Adgenda

- Introduction to the Loader
- Improvements in the Past Year
- Customer Challenges

JCC LogMiner Loader

- The Loader publishes database changes to a target or targets.
- The Loader is used in mission-critical architectures.
- The source database is, generally, a production database that must not experience a significant impact.
- Targets can diverge significantly from the source.
- Some applications are extremely intensive, involving millions of customers or thousands of transactions per second.

Sources and Targets

- Source
  - Oracle Rdb (any version that supports the LogMiner)
- Target
  - Oracle Rdb (any version that supports multi-statement procedures)
  - Oracle (requires Oracle SQL*net on the system running the Loader)
  - SQL Server via JDBC target
  - Other platforms via JDBC Class 4 drivers (Sun lists 141 platforms with a Class 4 driver. See the documentation for a full list of those tested.)
  - XML (to your own API)
  - File

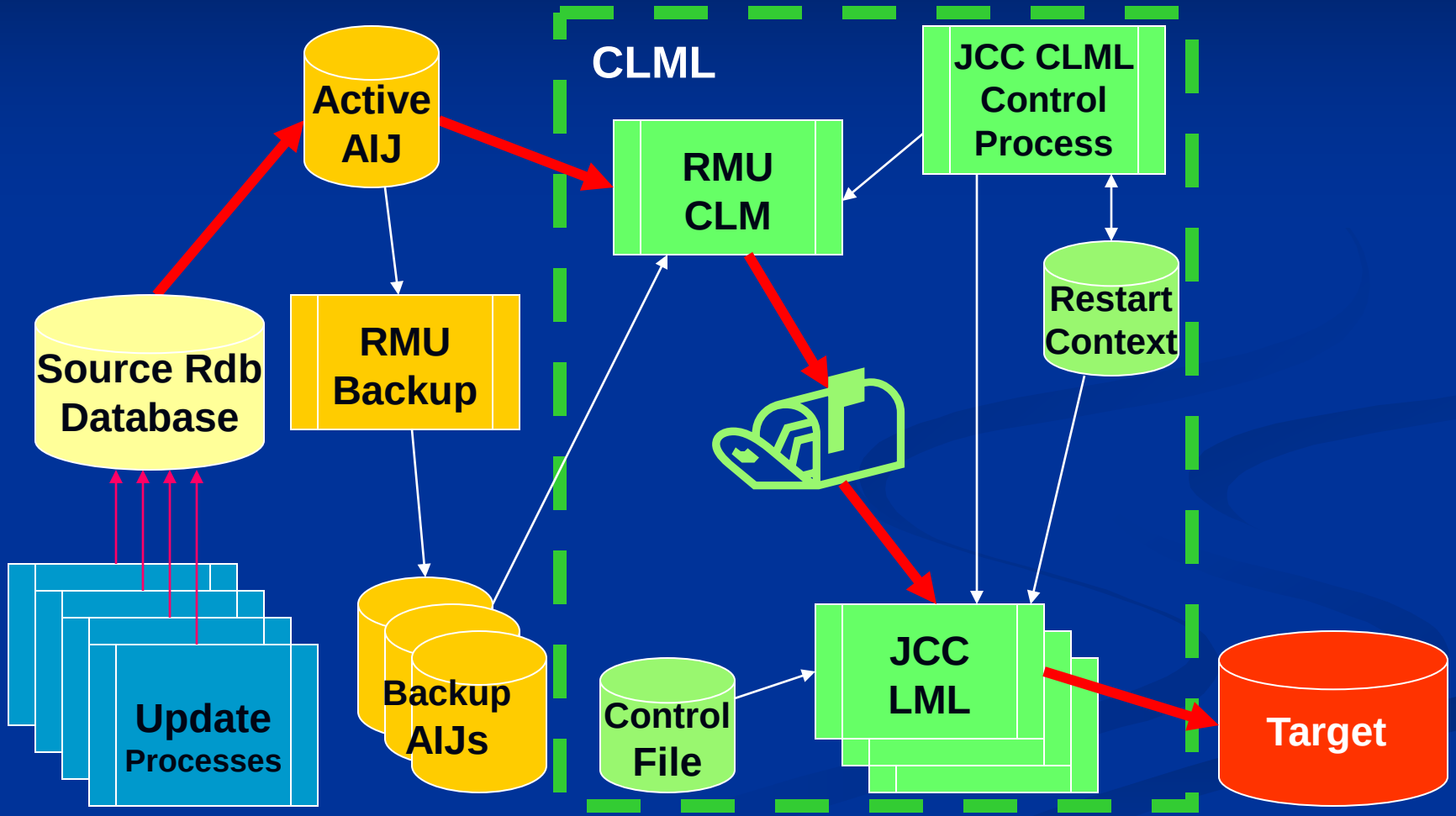
Sources and Targets (cont.)

- The source and target can be different.
  - Format of the target can be completely different
    - Logically
    - Physically
  - Tuning of the target can be different.
    - Indices
    - Placement
    - Buffering
    - Caching
- Multiple targets for one database, table, or column are possible.
- Roll-ups of different sources are possible.
- Subsets of database or table are possible.

Source to Target, Plus

- Continuous is "near realtime"
- Low impact on Mission Critical source systems
- Resilient and resistant to environmental issues
- Restartable, no data loss
- Materialized information, such as commit time or your own constant
- Single or multi-threaded, fast
- Tunable
- Extensively equipped for monitoring and performance analysis
- Data Pump
  - Initial population
  - Correction of downstream difficulties
- Data transforms, Filtering, Data mapping

How The Loader Does It



Development Partnership

- The LogMiner Loader is a partnership development effort.
  - Oracle Rdb LogMiner
  - JCC LogMiner Loader
- Development
  - Began in mid-2000
  - Advanced to Continuous beginning in July, 2001
  - Continues
- Use
  - At multiple sites since 2002
  - Customer questions and experiences have enhanced the product

Recent Releases

- Version 3.2.3, September 2009
  - Java 6.0 support on IPF
  - Fix Alignment Faults on IPF
  - Bug Fixes
- Version 3.2.2, March, 2009
  - Enhanced thread control
- Version 3.2.1, November, 2008
  - JDBC and Tuxedo enhancements
- Version 3.2.0, July, 2008
  - Significantly improved JDBC performance, elimination of restrictions
  - Tuxedo interface re-introduced
- Version 3.1.5, June, 2008
  - Additional LogMiner controls
- Version 3.1.4, March, 2008
  - Oracle target enhancements
- Version 3.1.3, December, 2007
  - Performance enhancements for JDBC target
  - Performance enhancements for Oracle target
- Version 3.1.2, November, 2007
  - Performance enhancements for Oracle target
- Version 3.1.1, August, 2007
  - Option of using 64-bit memory
- Version 3.1.0, March, 2007
  - Improved locking control
  - JAVA and Oracle version updates
  - Improved performance for Data Pump
  - Copy mode for production

Improvements in the Past Year

- JDBC
- Tuxedo
- Oracle

JDBC

- Original Interface
- First improvement – By Record
- Native Interface

JDBC Original Interface

- Single XML document per transaction
- 100% dynamic SQL statements
- Drawbacks
  - Can only handle modest transaction sizes due to memory limitations
    - Uses large amounts of JVM memory for large transactions
    - Pagefaults!
  - No statement reuse
  - High CPU overhead to create and shred XML documents
  - Latency vs. transaction size ratio is far from linear
  - Does not handle binary data within character strings

JCBC Original Interface

- Benefits
  - Fast to code
  - Rapid deployment to customers
  - Easy to improve performance ☺

JDBC By Record

- Still uses XML documents, but one per record instead of one per transaction
- Performance increased 3x for modest transactions
- Drawbacks
  - Still 100% dynamic SQL statements
  - Increases overhead bytes per row
  - Still does not handle binary data within character strings

JDBC By Record

- Benefits
  - Reduces CPU costs for large transactions
    - Creating and shredding smaller XML documents
    - Reduces pagefaulting
  - Uses smaller amount of JVM memory
    - Reduces pagefaulting
  - Transaction size restriction eliminated
    - [Now similar to other interfaces]

JDBC Native Interface

- Complete re-architecture
  - Use prepared SQL statements
  - Use native JDBC data types
    - Support binary data within strings
  - JDBC architecture performance enhancements

JDBC Native Interface - General Architecture

- Calling…
  - C
  - JNI (Java Native Interface)
    - Instantiate JVM (Java Virtual Machine)
    - Allocate variables within JVM
  - Once per table
    - Dynamic SQL with placeholders
    - Arrays of key and non-key columns
  - Each row
    - Convert data into corresponding key/non-key arrays
    - Create NBV (Null Bit Vector) array

JDBC Native Interface - General Architecture

- Called
  - Java static methods
  - Once
    - Attach to database
    - Set buffer sizes
  - Once per table
    - Create a table object
      - Dynamic SQL with placeholders
      - Arrays of key and non-key columns
    - Query target for column data types
  - Each row
    - Buffer data in key/non-key arrays with NBV array
    - Perform insert/update/delete when buffer size exceeded

JDBC Native Interface - HP JNI Issues

- Not really the way JNI is generally used…
- Most common usage
  - Call OUT from Java to other languages
  - Occasionally call back into Java
- Loader use is opposite
  - Implies different memory management
- Calls not implemented
- Work-around interface bugs

JDBC Native Interface - Prepared Statements

- Prepared statements with parameter markers
  - Compile once
  - Pass data for parameters
- Reduces CPU costs on target
  - Compilation
  - Optimization
- Elapsed time significantly reduced

JDBC Native Interface - Batching

- JDBC Batches
  - Allow multiple sets of parameter data to target for a single compiled statement
    - Each set of data values for statement is called a Batch
  - Single network message with all data
  - Target database processes each and returns an array of statuses for success
- Not all drivers support batching
  - [Some driver's support for batching do not adhere to the above description]

JDBC Native Interface - Performance

- Testing to SQL Server 2005
- Regression testing using a single thread
  - Batch size = 1
    - 509 records per second
  - Batch size = 10
    - 596 records per second
  - Batch size = 13
    - 730 records per second
  - Batch size = 20
    - 916 records per second
- Customer reports using a single thread
  - Increased nearly 5x (as compared to by Record)
  - 1000+ records per second

JDBC Native Interface - Specific Driver Issues

- Older JDBC drivers don't support some of these concepts
- Loader has configuration options to disable unsupported features
  - Automatically disabled for drivers known to be deficient
- SQL Server 2000
  - Multiple concurrent prepare statements
  - Newer drivers provide better support
- Oracle
  - Batch support insufficient for Loader requirements
  - Use the Native Oracle (OCI) Loader target

JDBC Native Interface

- Benefits
  - Much better performance
  - "Near Realtime"
  - Handles binary data
- Drawbacks
  - Performance can always be improved… ☺

Oracle Interface Changes

- ANSI vs. Oracle SQL
  - ANSI requires statement closure
  - Increases 'soft' compiles (CPU on server)
- Work-around for Oracle bug
  - UROWID vs. ROWID
    - Use of UROWID causes memory leak
- Disconnect on exception
  - Bugs in Oracle Server cause PGA memory leak
- Insert Optimization
  - Insert first, if exception then update

Tuxedo (Lazarus)

- Non-Support for OpenVMS
  - Announced Q2, 2006 - EOL for OpenVMS 7.x December 31st, 2008
- Support for OpenVMS 8.3
  - Announced in Q2, 2008
  - Same version, just recompiled
- Resurrected Loader Tuxedo interface
  - Only V6.5-518
- No Integrity support…today
- Current Non-OpenVMS version is 10

Customer Challenges - Giving The Loader Some Latitude

- …and Longitude…
- Transcontinental problem
  - Latency in a record-based application (e.g. Loader)
  - As network latency increases, Loader latency increases (per record in the transaction)
  - New JDBC interface can reduce this effect with Batching

Customer Challenges - Giving The Loader Some Latitude

- Intercontinental problem
  - Bandwidth in a data intensive application (e.g. Loader)
  - High latency links are a problem, but intercontinental links are more likely to have limited bandwidth
  - Hardware devices to batch and compress network messages
- Testing network issues without leaving the building
  - Network delay device
    - Allows modification of network latency and bandwidth to determine bottleneck
  - WANem - http://wanem.sourceforge.net/
  - Price is right

Futures – Possibilities

- Oracle as a source
- Configuration GUI
- Automation improvements in responding to changes in the source database metadata
- Expanded ETL (schema change) support for weaving back together enterprises with fractured information architectures
- Always examining and expanding
  - Performance improvements
  - DBA tools

Availability

- Kit is available at FTP.JCC.COM
  - Documentation
  - Kit
- Evaluation license available on request
  - Send mail to info@jcc.com
- Find descriptions of the LogMiner Loader and other information at http://www.jcc.com/LML.htm

Acknowledgements

- Thanks to Rdb engineering for their support and counsel
- Thanks to our Customers for sharing their experiences with the Loader

Questions

http://www.jcc.com/LML.htm

Join the worldwide Rdb community. Send mail to

OracleRdb-request@JCC.com with "SUBSCRIBE" in the body of the message.

For more information send mail to info@jcc.com