Using TPump in an Active Warehouse Environment

Dr. Vincent Hager
Senior Consultant CRM Solutions
Teradata Division – Austria

December 2002

You've never seen your business like this before.

Teradata
a division of NCR

Using TPump in an Active Warehouse Environment

Agenda

- Architecture
- Script Variables
- Data related Variables
- Challenges & Deployment
- Scenarios
- TPump in a Continuous Environment
- Best Practices

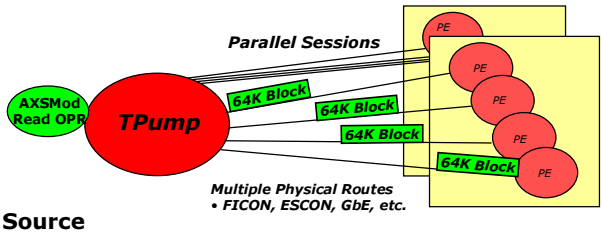
Teradata
a division of NCR

TPump Architecture

- MPP Teradata Loading
  - SQL DML— Active Streams or Batch
- TPump
  - Many client platforms
  - Many client sources
  - Row level locking
- Performance
  - Multi-statement, Multi-session
  - Can saturate: Wire, Client, and/or RDBMS

Teradata
a division of NCR

TPump Architecture



Read any Source
- Disk, Tape
- OLE-DB, ODBC
- Pipes, MQ
- 3rd Party
- Etc.

Utility Processing
- Until EOF:
  - Read input stream
  - Build <pack> exec and rows in buffer
  - Asynch send on any available session
  - Optional checkpoint <frequency>

PE Processing
- Reuse cached plan
- Apply 'pack' in parallel
- Checkpoint

Teradata
a division of NCR

Motivations using TPump in an Active Warehouse

- TPump is suitable, when some of the data needs to be updated closer to the time the event or the transaction took place
- avoids table-level locks (row-hash locks only)
- Concurrent table SQL access during updates
- flexibility in when and how it is executed
- queries can access a table concurrently with TPump
- several TPump jobs can run against the same table at the same time
- sources as varied as MVS, NT or UNIX concurrently

Teradata a division of NCR

TPump Script Variables

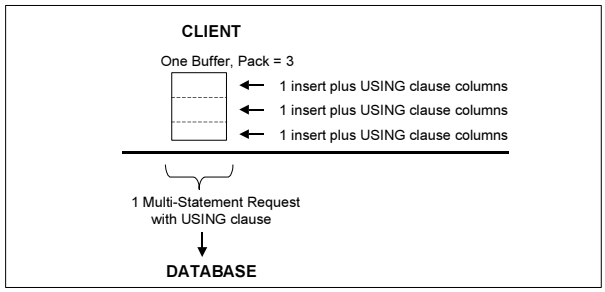
The TPump key variables, in order of greatest potential impact on performance, are the following (**BEGIN LOAD**):

- PACK
- SESSIONS (number)
- SERIALIZE (ON or OFF)
- CHECKPOINT (mins)
- ROBUST (ON or OFF)
- NOMONITOR

Teradata a division of NCR

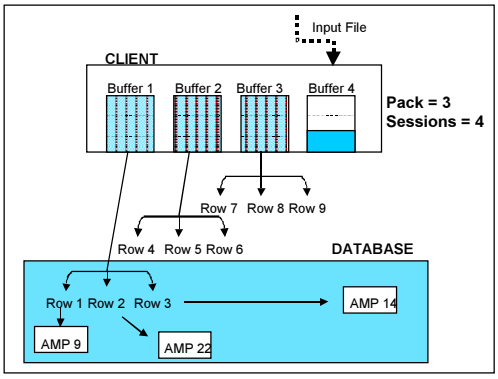
TPump PACK Factor

- The Pack Factor is the number of statements that will be packed together into a TPump buffer and sent to the database as one multi-statement request. Higher is usually better.



Teradata
a division of NCR

TPump Sessions



There is no correlation between TPump Sessions and AMPs. Both the pack factor and the number of sessions contribute to the level of parallelism inside Teradata.

Teradata
a division of NCR

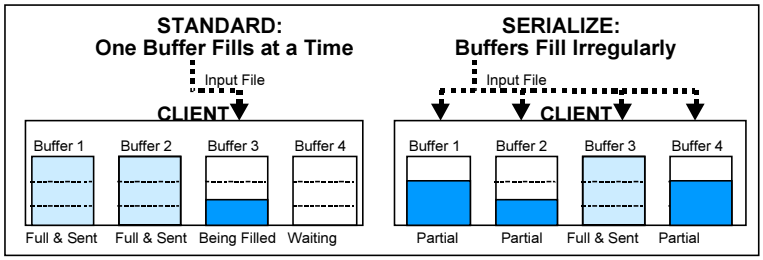
TPump Sessions

- Number of connections to the database that will be logged on for this TPump job. Each session will have its own buffer on the client. The greater the number of sessions, the more work will be required from the client and database.
- In setting the number of sessions, make sure to adjust the pack factor first, then:
  - Start with very few sessions until the application is running as expected.
  - Increase the number of sessions considering the pack factor, server and client power, how many other TPump jobs are likely to be running concurrently.
  - Increase sessions gradually.
  - How many TPumps concurrently?

Teradata a division of NCR

TPump SERIALIZE

- Tells TPump to perform a partitioning of the input records across the number of sessions it is using, ensuring that all input records that touch a given target table row (or that contain the same non-unique primary index value) are handled by the same session.



- With SERIALIZE, all buffers may be partially filled at any point in time; without SERIALIZE, only one buffer will be partially filled at any point in time.

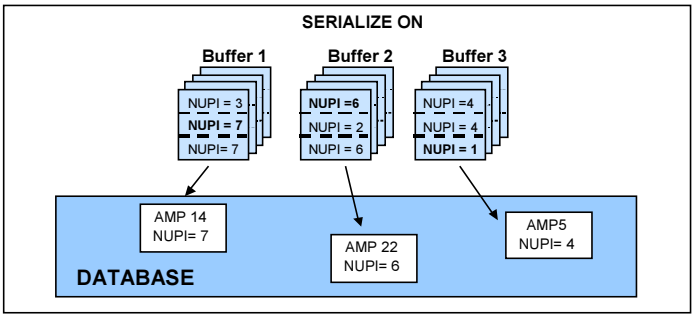
Teradata a division of NCR

Considerations for SERIALIZE

- **Consider SERIALIZE ON for the following situations:**
  - Possibility of multiple updates with the same primary index value in the input file
  - If the order of applying updates is important
  - Recommended with UPSERTs if any single row can be touched more than once
  - To reduce deadlock potential
- **Impacts of serialization:**
  - Some additional work is done by the client when partitioning the input
  - Buffer replenishment is spread out, making stale data a greater possibility
  - Performance may be impacted by sending a frequent number of partial buffers to the database when a checkpoint is taken.

Teradata a division of NCR

TPump SERIALIZE

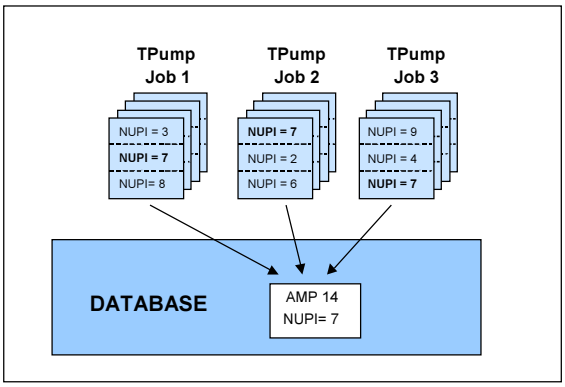


- SERIALIZE ON removes deadlock potential between buffers within the same TPump job, when rows with non-unique primary index values are being processed.
- Manual partitioning is required to do the same between multiple TPump jobs.

Teradata a division of NCR

TPump Deadlock Potential

- Manually partitioning input data across multiple TPump jobs will force the same NUPI values to go to the same TPump job. This eliminates inter-TPump deadlock potential when the same table is updated from different jobs.



Teradata a division of NCR

TPump CHECKPOINT

- Frequency (minutes) between occurencies of checkpointing
- During a checkpoint all the buffers on the client are flushed to the database, and mini-checkpoints (if ROBUST is ON) written since the last checkpoint will be deleted from the log table.

Teradata a division of NCR

TPump ROBUST ON

- ROBUST ON avoids re-applying rows that have already been processed in the event of a restart (data integrity). It causes a row to be written to the log table each time a buffer has successfully completed its updates. These mini-checkpoint are deleted from the log when a checkpoint is taken.
- ROBUST OFF is specified primarily to increase throughput by bypassing the extra work involved in writing the mini-checkpoints to the log.
- When to use:
  - INSERTs into multi-set tables, as such tables will accept re-inserted rows
  - Updates are based on calculations or percentage increases
  - If pack factors are large, and applying and rejecting duplicates after a restart would be unduly time-consuming
  - If data is time-stamped at the time it inserted into the database

Teradata a division of NCR

TPump IGNORE DUPLICATE ROWS

- IGNORE DUPLICATE ROWS means that duplicate inserts, if they are attempted (as they would be on a restart with ROBUST OFF) will not generate a write to the error table. This will add efficiency to a restart, should one occur.

Teradata a division of NCR

TPump Data Variables

- Variables related to the database design and the state of the data itself:
  - How clean is the data
  - Any sequence to the input stream
  - Degree and type of indexes defined on the table being updated
  - Use of Referential Integrity, fallback or permanent journaling
  - Number and complexity of triggers
  - Number of columns being passed into the database per update
  - Ratio of UPDATEs to INSERTs when UPSERT is used

Teradata a division of NCR

TPump Data Variables

- Multi-Statement SQL
- Efficiency by use of „USING"-clause (cached exec-plans increasing throughput)
- „USING" 512 columns limitation
- UPSERT processing: watch out ratio between UPDATEs to INSERTs (the more INSERTs the lower the throughput)

Teradata a division of NCR

Challenges using TPump

- Careful design of insert/update/delete strategy required to make use of effectiveness
- Avoid table locks and FTS (only Single/Dual AMP operations desired)
- Longer runtime (+40%) when data with errors (1%)
- Longer runtime with a NUSI (+50%), worse with 2 NUSIs (+100%)
- Longer runtime (+45%) with fallback
- SERIALIZE adds 30%, ROBUST adds 15%

Teradata a division of NCR

TPump Deployment

- Feeding clean data from your transformation processes into TPump is important for overall performance.
- Pay attention to pack rate and session optimization.
- NUSI and database configuration can have a significant impact on data acquisition throughput.
- Want EAI/ETL tools with continuous data acquisition capability for feeding into TPump.

Teradata a division of NCR

TPump Scenarios

(I) Large, high volume
(II) Small, highly time-critical bursts
(III) Extract from OLTP database
(IV) Multi-Source batch
(V) Variable arrival rates, low volume
(VI) TPump in a Continuous Environment

Teradata
a division of NCR

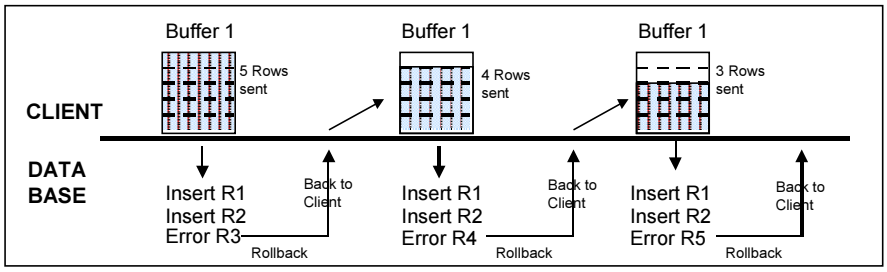
TPump Scenarios (I)

- **Large, high volume**
  - A high volume of transaction data being inserted daily
  - Data must be loaded and available in less than 8 hours from arrival
  - Batches arrive separately from each outlet, and are loaded while queries run
  - Data is clean and a powerful mainframe client initiates TPump
  - Use of Batch Window not feasible anymore

Teradata
a division of NCR

TPump Settings (I)

- Pack factor 31
- Sessions: Same as the number of AMPs
- Serialize OFF
- CHECKPOINT 0
- ROBUST OFF



Upon errors the whole buffer is rolled back and resent

Conclusion (I)

- **Large, high volume**
  - Using TPump in this scenario provides an opportunity to move the load of a given row closer to the actual time the event happened. The site may currently be loading one batch of inserts a day for each outlet, but has the flexibility in the future to do two or three TPump runs per outlet per day, if they choose. Using TPump offers a controllable transition to updating that is closer to real time.

TPump Scenarios (II)

- **Small, highly time-critical bursts**
  - Sporadic, small files, such as reference material or information refreshes
  - Need close to real-time availability of the data, 1 minute or less
  - Arrives from 10-15 different sources throughout a 24-hour period
  - Some sources provide clean data, other sources have occasional errors in the data.

Teradata a division of NCR

TPump Settings (II)

- Pack Factor: 35 (clean data source), 3 (unreliable data source)
- Sessions: 1/4 the # of AMPs (clean data), same as the # of AMPs (unreliable data)
- SERIALIZE OFF
- CHECKPOINT 0
- ROBUST OFF
- NOMONITOR (when jobs are very short: shorten job initialisation time)
- Use Persistent Macros for DML (NAME command)

Teradata a division of NCR

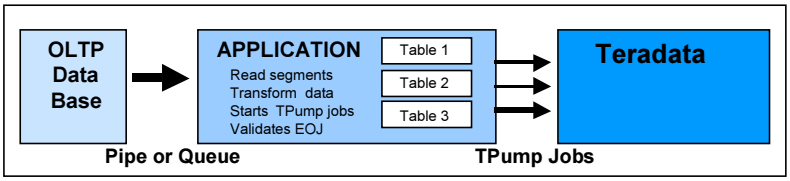
TPump Scenarios (III)

- **Extract from OLTP database**
  - Event or transaction data extracted from the transaction database into queues or pipes
  - 20 minute turnaround from transaction to data warehouse
  - Application reads queue coming from OLTP database, transforms data
  - Multiple, short, demand-driven TPump jobs run concurrently

Teradata a division of NCR

TPump Settings (III)

- Pack Factor: 12
- Sessions: Between 10% to 50% of the number of AMPs
- SERIALIZE: OFF for INSERT jobs, ON for UPSERTs (to bypass Deadlock Potential)
- CHECKPOINT 0
- ROBUST OFF



Teradata a division of NCR

TPump Scenarios (IV)

- **Multi-Source batch**
  - Concurrent TPumps during multi-hour batch window at night
  - Next day turnaround is acceptable
  - Different sources that reside in different locations update the same table
  - Highly-indexed target tables

Teradata a division of NCR

TPump Settings (IV)

- **5 TPump jobs running concurrently**:
  - Pack Factor 5 (TPump jobs less resource-intensive)
  - Sessions: ½ the number of AMPs
  - SERIALIZE OFF
  - CHECKPOINT 15
  - ROBUST OFF (reapplying inserted rows is not an issue, and with a low pack rate, recovery issues are less)

Teradata a division of NCR

TPump Scenarios (V)

- **Variable arrival rates, low volume**
  - Captures movement of an object or a transportation industry carrier
  - Real-time is within one hour of the change in position
  - Duplicate reporting is possible, order of applying updates is important
  - Data contains errors approaching 5%

Teradata a division of NCR

TPump Settings (V)

- Pack Factor 5 (propensity for error conditions and slow-arriving data need to fill the buffers)
- Session: Same as the number of AMPs (more buffers to be sent into Teradata)
- SERIALIZE ON (The order that the updates are applied is important in this application, and there is a possibility that rows with the same primary index value will be inserted through different buffers at the same time)
- CHECKPOINT 20 (large value to counteract the cost of flushing of partial buffers)
- ROBUST ON

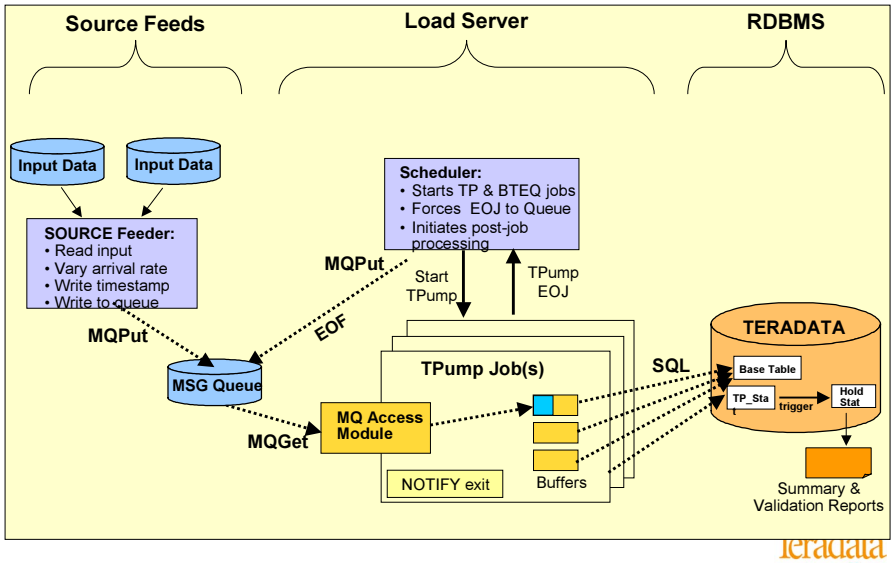
Teradata a division of NCR

TPump Scenarios (VI)

- **TPump in a Continuous Environment**
  - load data through a queue in an ongoing fashion
  - automatic and tightly coordinated rotation of different TPump instances, all fed by the same queue
  - automatic hand-off of control between TPumps, and to handle the end-of-job processing
  - opportunity for tighter management in these areas:
    - Error handling
    - Monitoring statistics
    - Sending alerts
    - Detecting inconsistencies

Teradata a division of NCR

MQSeries Feed into TPump



Teradata a division of NCR

Role of Source Feeder

- Provide application data feeds.
- Read transaction messages from a file.
- Add a timestamp to the message.
- Put message into the queue.
- Arrival rate is adjustable.

Teradata
a division of NCR

Role of Scheduler

- Stand-in for commercial scheduler.
- Start TPump job.
- End TPump job by placing an EOF message to the Queue.
- Launch post-job processing:
  - BTEQ script to consolidate error rows inside Teradata.
- Monitor load process status/results.

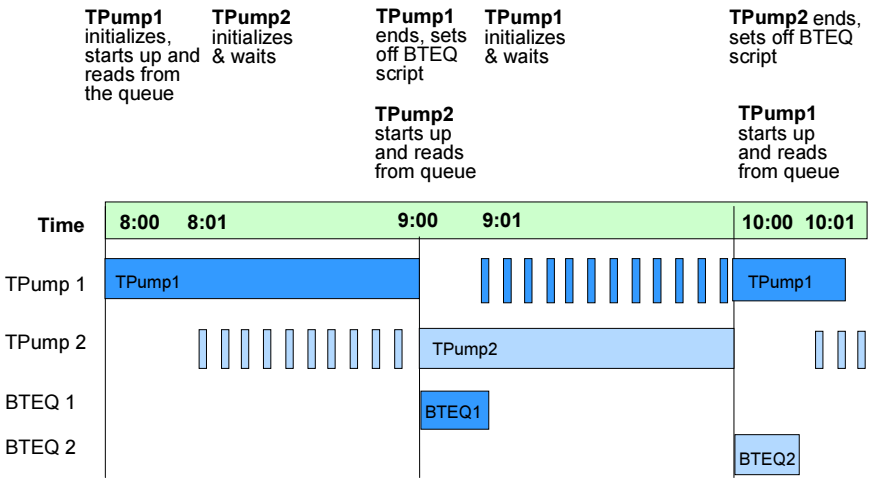
Teradata
a division of NCR

Role of MQSeries AXSMod

- Connect to MQSeries:
  - Local or remote.
- Get messages from MQSeries.
- Add timestamp to the messages.
- Stream messages to TPump.
- Guaranteed reliability:
  - No lost inserts.
  - No duplicate inserts.

Teradata
a division of NCR

Rotating Multiple TPumps



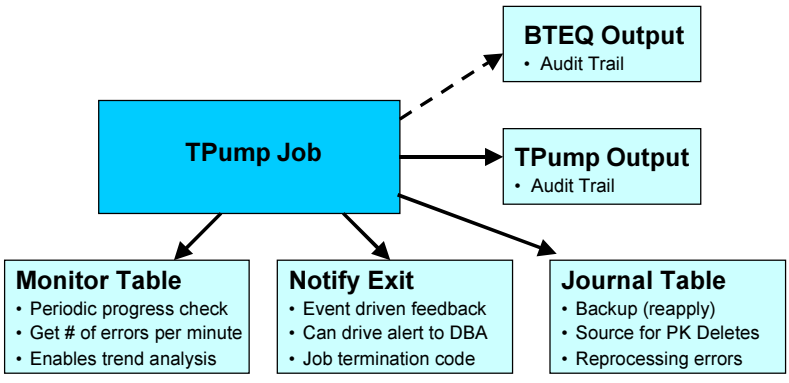
Teradata
a division of NCR

TPump Settings (VI)

- All TPump instances use the same parameters:
  - Pack Factor 10
  - Number of sessions 20
  - Checkpoint 30
  - ROBUST ON
  - SERIALIZE OFF

Teradata
a division of NCR

Real Time Information about a TPump Instance



Teradata
a division of NCR

Error-Handling

Supplementing error row identification:

- Import#, Record# already in the error file row.
- At time of consolidation, add unique TPump name.
- Unique MQ message-ID can also be preserved on each row as it is loaded (or during error processing).

This application will consolidate errors and detail about the row, but will not:

- Reconstruct the error row.
- Re-process the original row, if it has been preserved.

Teradata a division of NCR

TPump Status and Statistics

TPump optionally maintains real time stats in a table:

- UserName, Import Start Date, Time, RestartCount.
- Last Table Update Date, Time
- RecordsRead, RecordsOut
- RecordsSkipped, RecordsRejcted, RecordsErrored

Row exists for life of job:

- Inserted at job start.
- Updated each minute.
- Deleted at job completion.

Teradata a division of NCR

Monitoring Techniques

- Status Table
  - Triggers placed on TPumpStatusTbl.
  - Triggers insert status information into permanent stats table.
- NOTIFY EXIT
  - Initialize, Open.
  - Errors.
  - Checkpoint.
  - Final Stats.
  - Prototype writes delimited final stats record to flat file.

Teradata a division of NCR

TPump Best Practices

- High **pack factors** can increase TPump throughput. When high pack factors cannot be used, **more sessions** are another way to increase TPump throughput if the client can support them.
- To reduce **data load latency** and improve **real-time availability** of single rows, reduce the pack factor
- If input **data contains errors**, a low pack factor will reduce the overhead of **rolling back** the request that contained the error, and re-processing all error-free rows.
- Speed up **TPump startup** by using **persistent macros** and specify TPump's recommended **pack factor** from a previous, similar run.

Teradata a division of NCR

TPump Best Practices (cont'd)

- When selecting the **number of sessions**, consider the total system load at the time the TPump job is run. When multiple TPump jobs are running, consider a number of sessions equal to the number of AMPs in the system, or less.
- If multiple TPump jobs may update rows from the same table with the same primary index value, **manually partition the data** on the primary index of the table, so all rows with the same PI value are directed to the same TPump job. Then also specify **SERIALIZE ON** to force the rows with the same NUPI value to a single session within that TPump job, further **reducing possible contention**.

Teradata a division of NCR

TPump Best Practices (cont'd)

- **Beware of Locking Conflicts**:
  - Locking conflicts on the target table row-hash (insert, update)
    → SERIALIZE ON
  - Locking conflicts in Dictionary at SQL Submission
    → Housekeeping of DBC.AccessRights
  - Locking at TPump startup
    → Pre-build and reuse (persistent) macros for ET-DDLs

Teradata a division of NCR

TPump Best Practices (cont'd)

- Assign the TPump user to a **higher priority** performance group when the TPump job runs at the same time as decision support queries, if the TPump completion time is more critical than the other work active in the system.
- If target table of inserts in the database is part of a **join index**, direct TPump into a **non-indexed staging table**. Insert/select from there into the base table at regular intervals is likely to be a **better-performing approach** to updating a table when a join index is involved. Prior to the insert/select, a UNION can be used to make sure data recently inserted into the staging table is included in query answer sets.
- To ensure that TPump is able to perform **single-AMP operations** on each input record, include the **entire primary index value** for the row being updated **among the columns passed to Teradata**.

Teradata a division of NCR

Thank you !

Teradata a division of NCR