SSIS Scaling and Performance

Erik Veerman
Atlanta MDF member
SQL Server MVP, Microsoft MCT
Mentor, Solid Quality Learning

Agenda

• Buffers
• Transformation Types, Execution Trees
• General Optimization Techniques
• Scaling Sources and Destinations
• Execution location
• Monitoring and Logging
Pipeline Buffers

- Data extracted into the Data Flow is passed into data buffer groupings
- Transformations logic flows over buffers for optimal performance
- Buffers allow a “streaming” process
- Each Data Flow can have multiple buffer profiles defined for different types of data
Data Flow Architecture

- Buffers based on design time metadata
  - The width of a row determines the size of the buffer
  - Smaller rows = more rows in memory = greater efficiency

- Memory copies are expensive!
  - Pointer magic where possible
  - E.g. Multicast – logical vs. actual
Column <LineageID>

- Each buffer profile assigns a <LineageID> to each column (not to be confused with ETL Lineage tracking)
- A single column coming in from a source can end up having multiple LineageIDs through its life in the pipeline
- LineageIDs are viewable in the Advanced Input and Output Properties

Component Types

- **Streaming (synchronous)**
  - Logically works at a row level
  - Buffer Reused
  - *Examples*: Data Convert, Derived Column, Lookup

- **Partially blocking (asynchronous)**
  - May logically work at a row level
  - Data copied to new buffers
  - *Examples*: Pivot, Un-pivot, Merge, Merge Join, Union All

- **Blocking (asynchronous)**
  - Needs all input buffers before producing any output rows
  - Data copied to new buffers
  - *Examples*: Aggregate, Sort, Row Sampling, Fuzzy Grouping
Synchronous and Asynchronous

- Think transformation communication
- Definitions apply to transformation outputs
- Synchronous transformation outputs
  - Same buffers immediately passed onto next transformation
  - No rows added, no rows removed
- Asynchronous transformation outputs
  - Data is copied to new buffer
  - Downstream transformation works independently of upstream asynchronous transformation

A Transform is not limited to a single synchronous output
- Multicast and Conditional Split have multiple synchronous outputs

Synchronous outputs preserve the sort order of the input rows

Identifying Synchronicity
- See SynchronousInputID property in Advanced Editor | Output property
- Entry of 0 identifies an asynchronous transformation
Execution Trees and Threads

- **Execution Trees**
  - Start from a source or an asynchronous output
  - Ends at a destination or an input that has no sync outputs
  - Different buffer profiles per tree

- **Execution Threads**
  - Each Source can get a thread
  - Each Execution Tree can get a thread

- **Use EngineThreads to control parallelism**
  - Value applies to Execution Trees, not Sources

---

Execution Trees and Threads

- Each component within an execution tree applies work on the same set of buffers
- Data in a new execution tree requires existing buffer data to be copied into new buffers
**General Optimization**

- Increase Engine Threads
- Breakup large Execution Trees with a Union All transformation to allow a more process threads to handle operations
- Remove columns in pipeline not used downstream (avoid pipeline warnings)
General Optimization

- Limit Rowbased operations
- Limit Blocking Transforms (presort if possible)
- Perform data correlation in the Data Flow
- Handle staging requirements with a Multicast transformation
- Use strategic staging to optimize pipeline
- Optimize temporary storage locations

General Optimization

- Manage and monitor memory!
- Isolate long running task durations
- Tuning buffers
  - DefaultBufferMaxRows
  - DefaultBufferSize
  - Buffer size is the lesser of...
    - DefaultBufferMaxRows * row-width
    - DefaultBufferSize
When to Stage Data

- Restartability requirements
- Process window times and precedence
- Intense downstream transformations causes source back pressure (slows extraction)
- Data Flow optimization
- Eases complexity

Source and Destination Optimization

- Drop indexes, load, re-add indexes
- Increase destination DB filegroup size
- FastParse to True in the Flat File Source!
- Use Row Count in various data flow places to understand bottlenecks
- Use Fast Load!
  - MaxInsertCommitSize
    - 0 = all rows committed at one time
    - >0 = each commit will lesser than buffer rowcount or MaxInsertCommitSize... be careful!
Package Execution Location

- Storage location
- Execution on source server
- Execution on destination server
- Execution on third server
- Distributed execution
Monitoring the Data Flow

• Pipeline logging events
  – Pipeline Execution Trees
  – Pipeline Execution Plan

• Performance Monitor counters
  – Object = SQLServer: SSIS Pipeline
    • Blob counters
    • Buffer counters
    • Row counts
Pipeline PerfMon Counters

- **Buffer types**
  - Flat Buffers: primary buffers used in the pipeline
  - Private Buffers: used by individual transformations to perform operations (Sort, Aggregate, Lookup cache)

- **Buffer Counters (by Total, Flat, or Private)**
  - Buffer memory: Amount of memory used by buffers
  - Buffers in use: number of buffers
  - Buffers spooled: # of buffers spooled to disk

KATMAI – SQL 2008

- Buffer threading scheduler changes
- MERGE/UPSERT
- Change Data Capture
- Better lookup component