High Volume Data Processing Techniques
Without Driving Your DBA Crazy

Julie Koesmarno
MCITP Database Administrator 2008 | Database Developer 2008 | Business Intelligence 2008
Agenda

High Volume Data Processing with:
• Batching technique in T-SQL
• SSIS
• Balanced Data Distributor

Out of scope:
• Partitioning
Caveats

- Every environment is different – requires different configuration
- Test in DEV / UAT
- These are options to other known alternatives such as Partitioning
Defining the problem

• High Volume Data
  – Size is relative

• Business Requirements
  – Duration
  – Data Availability
  – Complex business logic
  – Scalability
  – Configurable
What drives DBAs crazy about it?

= what are our constraints (environment wise)?
  • Disk space
  • Memory usage
  • CPU usage

Why?!
  • Response time / availability
  • Control
Details, details, details…

SQL Server environment

– Recovery Model
– Utilisation
– Hardware environment: number of processors, memory, network speed, disk configuration, and many more!

What the business wants to achieve

= **Great Solution** = **Happy Users** = **Happy DBA**
Useful techniques

The key is to maximise throughput

Reliable pipe
  Batch processing using T-SQL

Fast pipe
  SSIS

Divide and conquer
  Balanced Data Distributor with SSIS
  Partitioning (not in scope)
Batching in TSQL

Breaking the problem into smaller pieces
1. Table Lock duration reduced
2. Less rows to rollback
3. Less TempDB consumption
4. Control on error handling
5. Customised visibility on progress
DEMO

--- Initialisation variables
21 SET @BatchSize = 500000
22 SET @WaitTime = '00:00:00.100' -- delay time after each insert
23
24 --- Get min and max IDs to be processed
25 SELECT
26 @StartID = MIN(i.ISPDailySpeedID),
27 @MaxID = MAX(i.ISPDailySpeedID)
28 FROM Staging.ISPDailySpeed i
29
30

--- Run until all IDs are processed
35 WHILE @StartID <= @MaxID
36 BEGIN
37
38 -- End ID calculation
39 SET @EndID = CASE
40 WHEN (@StartID + @BatchSize - 1) > @MaxID
41 THEN @MaxID
42 ELSE (@StartID + @BatchSize - 1)
43 END
44

Initialisation

Start of Loop
INSERT INTO Fact.ISPDailySpeed
(
  ISPLocationID,
  CapturedDateID,
  DownloadKbps,
  UploadKbps,
  TotalTests,
  DistanceMiles
)

SELECT
d.ISPLocationID,
CAST(CONVERT(VARCHAR(8), i.CapturedDate, 112) AS INT) AS CapturedDateID,
i.DownloadKbps,
i.UploadKbps,
i.TotalTests,
i.DistanceMiles
FROM Staging.ISPDailySpeed i
INNER JOIN Dim.ISPLocation d
  ON d.ISPName = i.ISPName
  AND d.City = i.City
  AND d.RegionCode = i.RegionCode
WHERE i.ISPDailySpeedID BETWEEN @StartID AND @EndID

WAITFOR DELAY @WaitTime
SET @StartID = @EndID + 1
END
Paging Function in SQL Server 2012

ORDER BY … OFFSET … FETCH NEXT …
### Paging Function Performance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rows</td>
<td></td>
<td>14,528,281</td>
</tr>
<tr>
<td>Batch Size</td>
<td></td>
<td>500,000</td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>Average Duration (sec)</td>
<td>11.2</td>
<td>68.8</td>
</tr>
<tr>
<td>Total Duration (sec)</td>
<td>336</td>
<td>2,064</td>
</tr>
</tbody>
</table>

Paging Function in SQL Server 2012 over **6 times slower**!
Batching T-SQL

Great for:
1. Transferring data in the same instance
2. Low priority data transfer with long duration
3. Batching scripts can be converted to stored proc – ideal for replication
4. TempDB size is limited
5. Transaction Log is generally small for Simple or Bulk Logged recovery mode
Batching T-SQL – Tricky scenarios

• Get a unique list of a number of columns from 100 Million rows
• Joining 100 million rows with another 100 million rows on NVARCHAR(500) columns
• Transaction requirement – i.e. All rows have to be processed and committed as one transaction
Minimally Logged Inserts
1. Only write to MDF file
2. If no sorting required, it won’t use TempDB – i.e. no additional read/write

In Memory pipeline
Fastest when inserting data as Heap
Bulk Upload Lock – multiple concurrent threads
Heavy transformation offloaded to isolated SSIS server
Minimally Logged Inserts

Requirements:
1. Simple or Bulk Logged recovery model on Target database
2. Empty target table
3. Non clustered indexes are disabled
4. Data arriving in the same order of Clustered Index (if any)
5. Table Lock
6. SQL Server Destination (faster) or OLE DB Destination
DEMO

OLE DB Destination – flat table no index, no PK

SQL Server Destination – flat table no index, no PK
# All about Transaction Log

<table>
<thead>
<tr>
<th>Existing Data</th>
<th>Clustered Index</th>
<th>Non Clustered Index</th>
<th>Data Page Updates</th>
<th>Index Page Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Minimally Logged</td>
<td>n/a</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Minimally Logged</td>
<td>Minimally Logged</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Doesn’t matter</td>
<td>Minimally Logged</td>
<td>Minimally Logged</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Minimally Logged</td>
<td>n/a</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Minimally Logged</td>
<td>Fully Logged</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Doesn’t matter</td>
<td>Fully Logged</td>
<td>Fully Logged</td>
</tr>
</tbody>
</table>

SSIS – Data Flow Task Configurations

SSIS:
• Maximum Insert Commit Size
• Buffer Size

Limiting Memory/CPU usage:
• Resource Governor
• SQL Server Max Memory size
SSIS Data Flow Task

Great for:
1. Transferring data from different instances/formats
2. Quick data transfer with short duration and minimal logging*
3. Lots of tuning options (also lots to learn!)
4. Can offload transformation on a separate server
Design Consideration

Insert
• Disable Indexes in Destination table before and rebuild after
• Disable Foreign Keys in Destination table before and rebuild after

Update & Delete
• Reload Destination table instead of update / delete
Balanced Data Distributor

- Parallelism in SSIS Data Flow
- Equal proportion of “pipes” to the same destination
- Currently supported for SQL Server 2008
- Runs on Windows 7 and Windows Server 2008 only
00:01:54 to import 14,528,282 rows into an empty table
Using Balanced Data Distributor with 4 OLE DB Destinations
4 Process related to SSIS – each for an OLE DB Destination from BDD
Balanced Data Distributor

Great for

1. Inserting lots of records
2. Taking advantage of multi-processor and multi-core servers
3. Follow similar rules to SSIS Data Flow Task
4. Data transfer without specific order
5. Bottleneck is in transformation or destination
What’s in common in these techniques?
Further reading

SQL Server 2012
ORDER BY Clause (OFFSET .. FETCH NEXT …)

SSIS
The Data Loading Performance Guide

Minimally Logging Bulk Load Inserts into SQL Server

BDD
The “Balanced Data Distributor” SSIS

Download Center – BDD
Wrap up

Batching in TSQL:
SQL Server source/destination on the same server, Low priority data transfer, Custom batch size and error handling with ease

SSIS Data Flow:
Different source/destination formats, Minimally Logged Insert, Tuning configuration, offload transform to isolated SSIS server

BDD using SSIS:
SSIS Data Flow++, using multithreading multicore to solve bottleneck on transform and destination
Remember!

Always **TEST** first

The Right Techniques for the Right Situation

Great Solution = **Happy Users** = **Happy DBA**
Contact details:

Email    signal@mssqlgirl.com
Blog     http://www.mssqlgirl.com
Twitter  @mssqlgirl
LinkedIn http://au.linkedin.com/in/juliekoesmarno
Thank You!