# 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

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

Initialisation

Start of Loop

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

```
49      INSERT INTO Fact.ISPDailySpeed
50      (
51          ISPLocationID,
52          CapturedDateID,
53          DownloadKbps,
54          UploadKbps,
55          TotalTests,
56          DistanceMiles
57      )
58      SELECT
59          d.ISPLocationID,
60          CAST(CONVERT(VARCHAR(8), i.CapturedDate, 112) AS INT) AS CapturedDateID,
61          i.DownloadKbps,
62          i.UploadKbps,
63          i.TotalTests,
64          i.DistanceMiles
65      FROM Staging.ISPDailySpeed i
66          INNER JOIN Dim.ISPLocation d
67              ON d.ISPName = i.ISPName
68              AND d.City = i.City
69              AND d.RegionCode = i.RegionCode
70              AND d.CountryCode = i.CountryCode
71      WHERE i.ISPDailySpeedID BETWEEN @StartID AND @EndID
72
```

Core Logic

End of Loop

```
76
77          WAITFOR DELAY @WaitTime
78
79          SET @StartID = @EndID + 1
80
81      END
```

# Paging Function in SQL Server 2012

ORDER BY … OFFSET ... FETCH NEXT ...

```
51  SELECT
52      d.ISPLocationID,
53      CAST(CONVERT(VARCHAR(8), i.CapturedDate, 112) AS INT) AS CapturedDateID,
54      i.DownloadKbps,
55      i.UploadKbps,
56      i.TotalTests,
57      i.DistanceMiles
58  FROM Staging.ISPDailySpeed i
59      INNER JOIN Dim.ISPLocation d
60          ON d.ISPName = i.ISPName
61          AND d.City = i.City
62          AND d.RegionCode = i.RegionCode
63          AND d.CountryCode = i.CountryCode
64  ORDER BY i.ISPDailySpeedID
65  OFFSET @StartID ROWS
66  FETCH NEXT @BatchSize ROWS ONLY
```

# Paging Function Performance

| | Manual Batching (2008) | Paging Function (2012) |
|---|---|---|
| Number of rows | 14,528,281 | |
| Batch Size | 500,000 | |
| Iterations | 30 | |
| Average Duration (seconds) | 11.2 | 68.8 |
| Total Duration (seconds) | 336 | 2,064 |

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

# SSIS - Data Flow Task

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

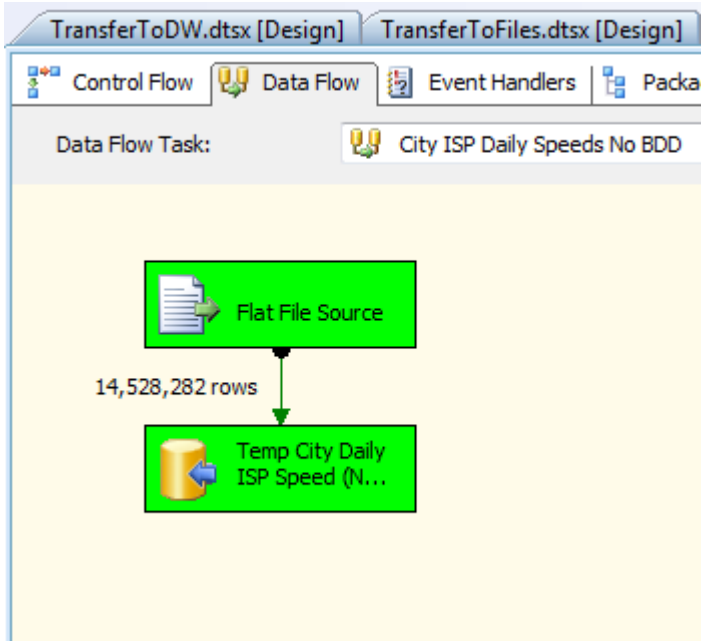
Heavy transformation offloaded to isolated SSIS server

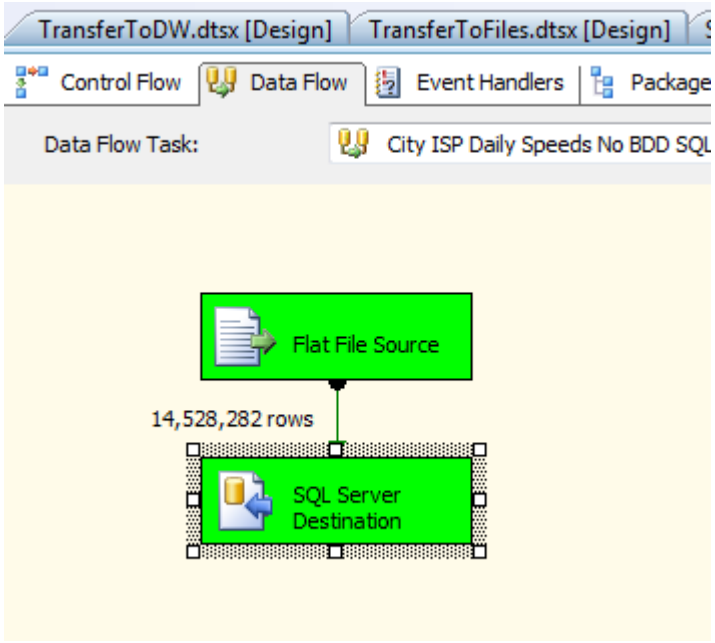
# Minimally Logged Inserts

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



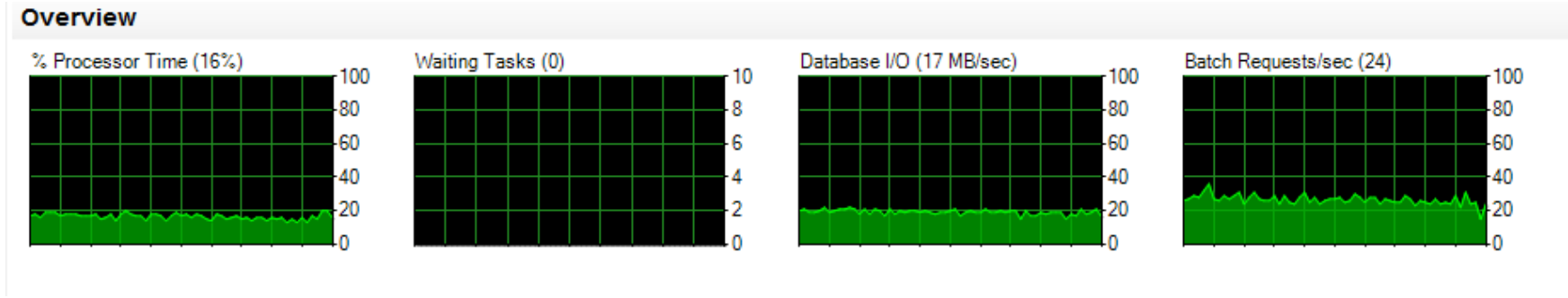




00:03:58
OLEDB

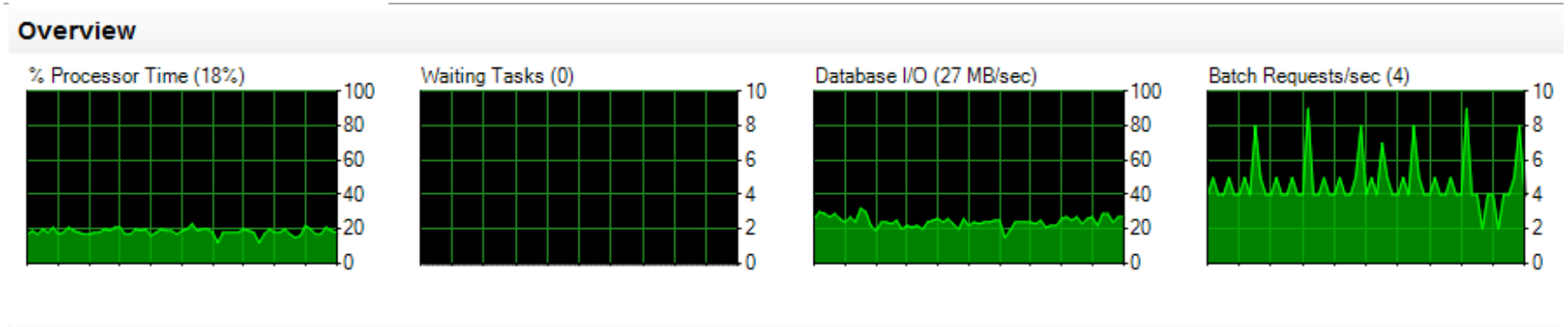
00:02:11
SQL Server Destination

# DEMO

OLE DB Destination – flat table no index, no PK




SQL Server Destination – flat table no index, no PK



# All about Transaction Log

| Existing Data | Clustered Index | Non Clustered Index | Data Page Updates | Index Page Updates |
|---|---|---|---|---|
| No | No | No | Minimally Logged | n/a |
| No | No | Yes | Minimally Logged | Minimally Logged |
| No | Yes | Doesn't matter | Minimally Logged | Minimally Logged |
| Yes | No | No | Minimally Logged | n/a |
| Yes | No | Yes | Minimally Logged | Fully Logged |
| Yes | Yes | Doesn't matter | Fully Logged | Fully Logged |

Source:
http://www.mssqltips.com/sqlservertip/1185/minimally-logging-bulk-load-inserts-into-sql-server/

# 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

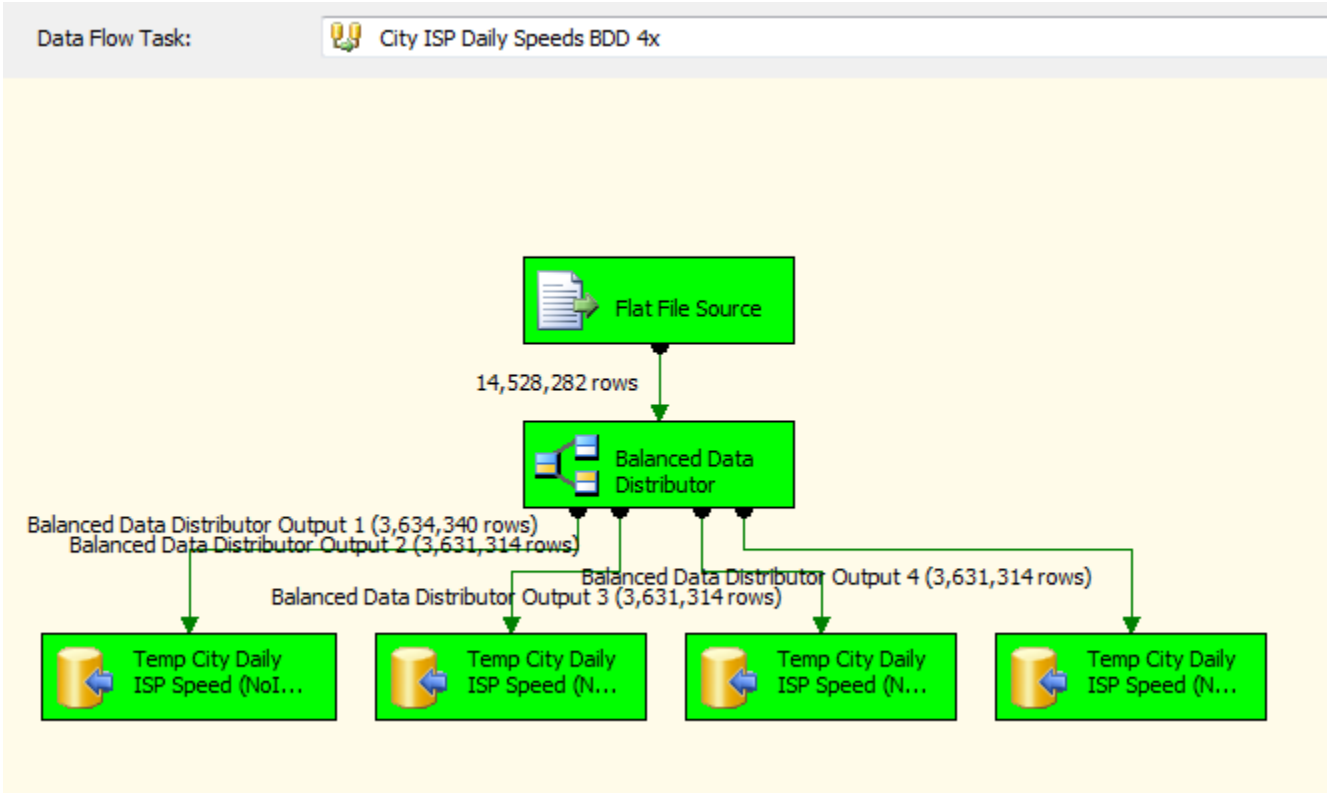
Update & Delete

- Reload Destination table instead of update / delete

# Balanced Data Distributor

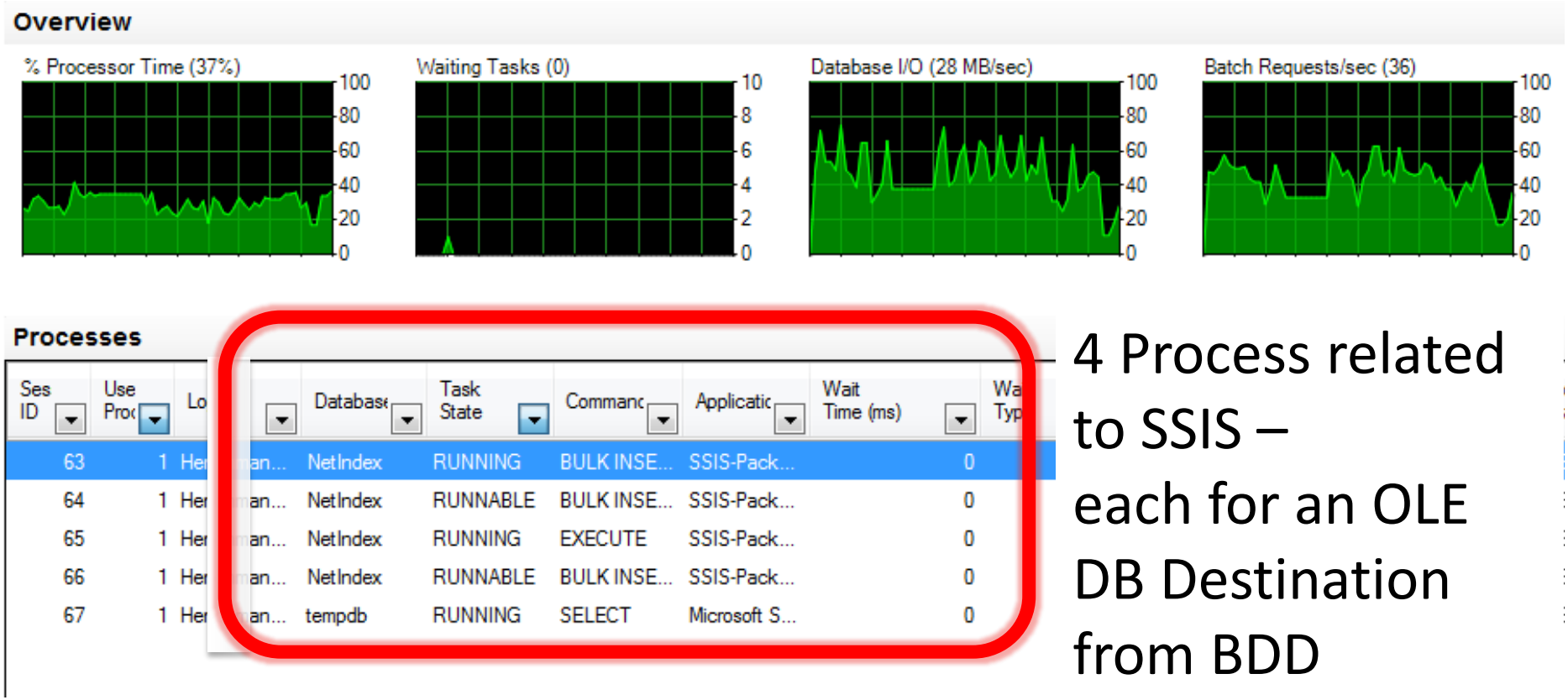
- Parallelism in SSIS Data Flow
- Equal proportion of “pipes” to the same destination
- Currently supported for SQL Server 2008
- Runs on Windows 7 and Windows Server 2008 only

# DEMO




00:01:54 to import 14,528,282 rows into an empty table
Using Balanced Data Distributor with 4 OLE DB Destinations

# DEMO




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 …)
http://msdn.microsoft.com/en-us/library/ms188385%28v=SQL.110%29.aspx

**SSIS**
The Data Loading Performance Guide
http://msdn.microsoft.com/en-us/library/dd425070(v=sql.100).aspx

Minimally Logging Bulk Load Inserts into SQL Server
http://www.mssqltips.com/sqlservertip/1185/minimally-logging-bulk-load-inserts-into-sql-server/

**BDD**
The “Balanced Data Distributor” SSIS
http://blogs.msdn.com/b/sqlperf/archive/2011/05/25/the-balanced-data-distributor-for-ssis.aspx

Download Center – BDD
http://www.microsoft.com/download/en/details.aspx?id=4123

# 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**

# Q & A

Contact details:


Email signal@mssqlgirl.com

Blog http://www.mssqlgirl.com

Twitter @mssqlgirl

LinkedIn http://au.linkedin.com/in/juliekoesmarno

Thank You!