



Microsoft Fabric

This presentation is the property of Microsoft and is intended for informational and educational purposes only. You may use, copy, and distribute this presentation for your personal, non-commercial purposes. You may not modify, alter, or create derivative works from this presentation without the prior written consent of Microsoft. You may not use this presentation to misrepresent, defame, or disparage Microsoft or its products, services, or affiliates. You may not use this presentation to endorse or promote any other products, services, or organizations without the prior written consent of Microsoft.

By using this presentation, you agree to abide by these terms. If you do not agree, you must not use this presentation. Microsoft reserves the right to change these terms and conditions at any time without notice. Microsoft disclaims any and all warranties, express or implied, relating to this presentation, including but not limited to the accuracy, completeness, timeliness, or suitability of the information contained herein. Microsoft is not liable for any damages, losses, or liabilities arising from your use of or reliance on this presentation.

Please review the terms of use posted in the content library.

#FABCONSQLCON2026

FABCON

Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON

Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026

Performance Tuning for Fabric Data Factory

Mohan Sankaran

VP Engineering

Microsoft Data Integration

Chris Webb

Principal Program Manager

Fabric Customer Advisory Team

Pipelines

Pipelines Tip

Parallelism vs. Sequential Execution

Multiple levels of parallelism available to you as a pipeline developer in Data Factory

The image displays the Microsoft Azure Data Factory pipeline editor interface. It is divided into two main sections illustrating different execution models.

Parallel Execution (Top): A pipeline is shown with three activities: 'Copy data', 'Dataflow', and 'ForEach'. A green callout box with an arrow pointing to these activities states: "These activities will all run in parallel". Below this, the 'Settings' tab for the 'ForEach' activity is open, showing the 'Batch count' property set to a value, with a tooltip that reads: "Maximum number of parallel runs of inner activities".

Sequential Execution (Bottom): A pipeline is shown with three activities connected in a sequence: 'Copy data' → 'Dataflow' → 'ForEach'. A green callout box with an arrow pointing to this sequence states: "These activities will all run in sequence".

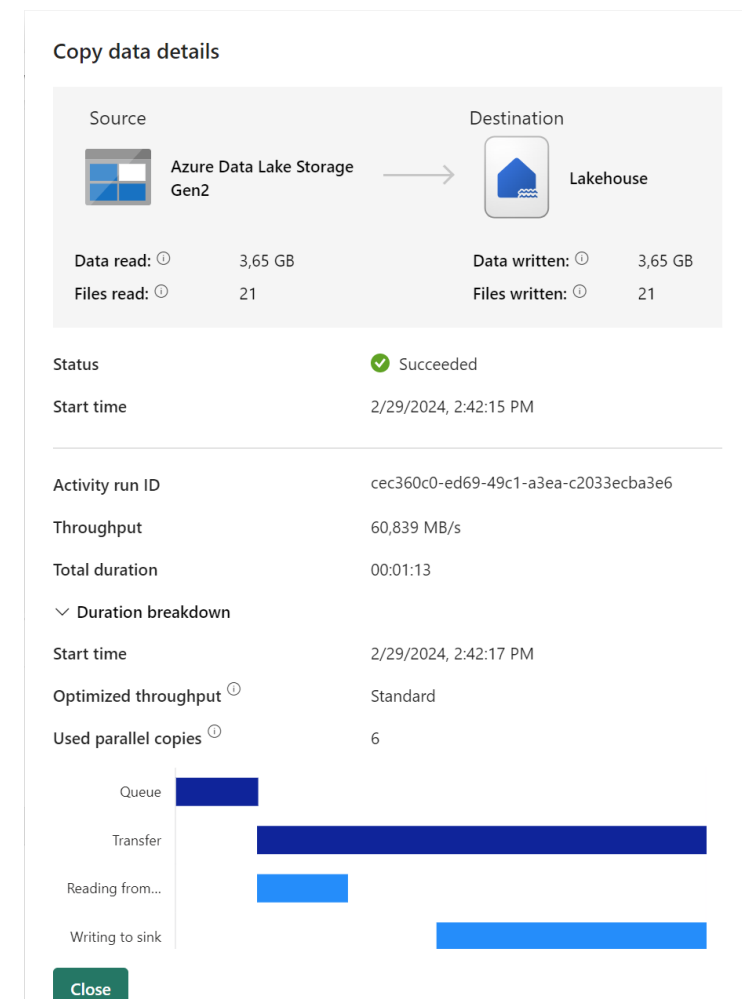
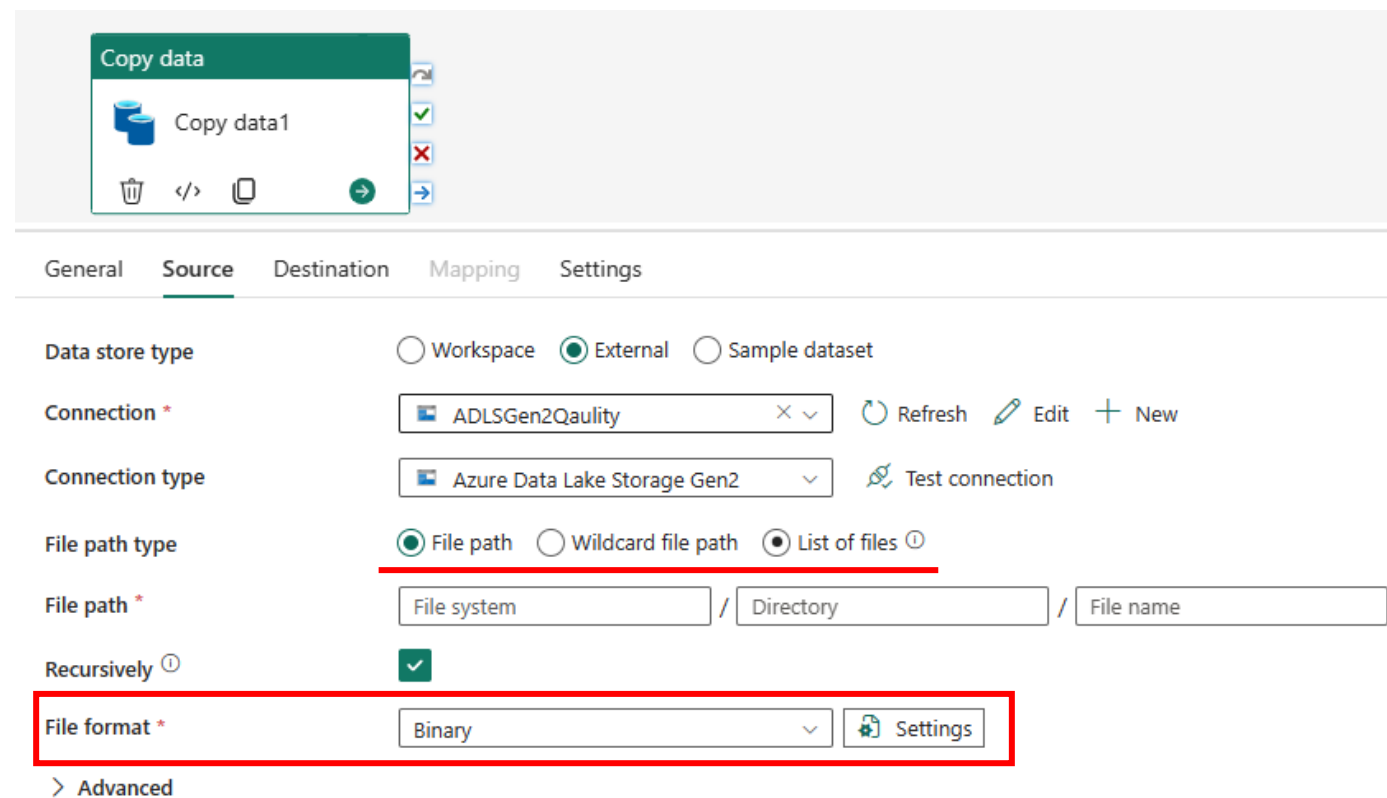
Configuration Details (Right): The 'Settings' tab for the 'Copy data' activity is shown, with the following options:

- Intelligent throughput optimization: Auto
- Degree of copy parallelism: Specify the degree of parallelism that data loading would use.
- Fault tolerance: [Dropdown]
- Enable logging:
- Enable staging:
- Data store type: Workspace External

Concurrency (Bottom Left): A tooltip for the 'Concurrency' property states: "The number of simultaneous pipeline runs that are fired".

Pipelines: Copy Activity

- Utilizing binary copy can improve load times to LH Files or storage accounts by 60-70%
- New Fabric Copy optimization for binary to improve data movement speeds 2x
- Tip1: Wildcards, list of files, and folders inside Copy source rather than iterate over many files using For-Each at pipeline level
- Tip2: Use For-Each with multiple Copy activities at pipeline level when
 - Single Copy activity uses up all its resources (maximum I/O/DIU with CPU/memory-intensive ops)
 - The data source is not supported to be partitioned and read in parallel by Copy activity



LFMM for a Single Copy Activity

- **Largest:** 0.18 PB
- **Fastest:** 6.02 GB/s
- **Most Files:** 21.38 Million
- **Most Rows:** 710.29 Billion

CopyJob

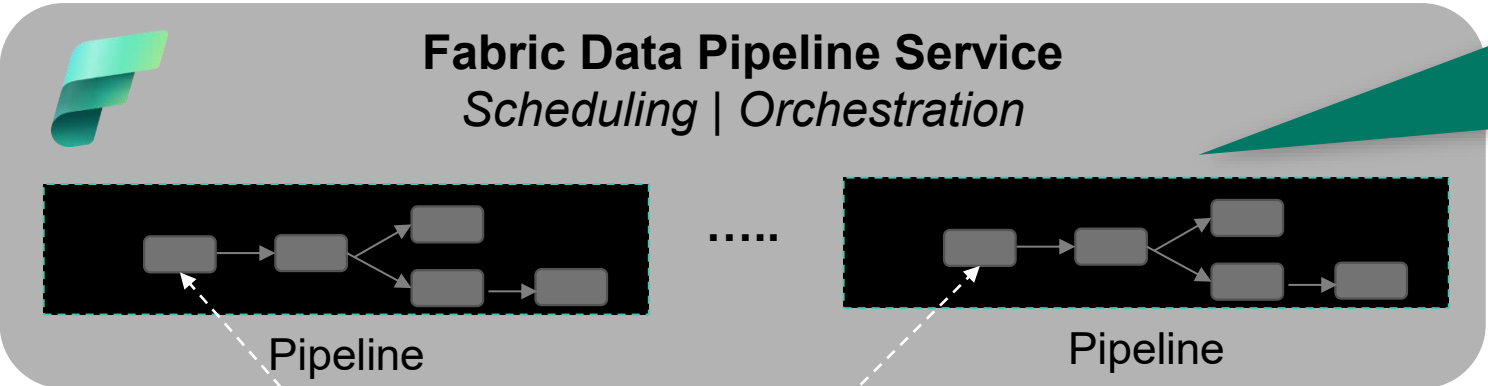
- One-stop solution for all data movement needs
- Bulk Table Selection and Mapping to Sink
- Column Mapping Rules for flexible schema transformations
- Batch, Incremental (Watermark) and CDC load capabilities
- Intelligent Auto-Scaling for optimal resource utilization
- High-performance engine for large-scale data movement
- Seamless integration with Data Factory Pipelines

The screenshot displays the Microsoft Fabric CopyJob interface. At the top, there's a navigation bar with 'Microsoft', 'CopyJob_2', and 'Confidential\Internal Only'. A search bar is on the right. Below the navigation bar, a warning message states: 'Your free Microsoft Fabric trial ends in 0 days. After the trial ends, you won't be able to work with Fabric items you created during the trial unless you buy Microsoft Fabric or move them to a Power BI Premium capacity. Learn more'. The main area shows a pipeline configuration with a 'Source' box labeled 'akv2 mgangavaram Azure SQL Database' and a 'Destination' box labeled 'lakehouse2e Lakehouse'. An arrow labeled 'Incremental copy' connects them. Below the configuration, a 'Results' section shows a table of job execution details.

Source → Destination	Status	Rows read	Rows written	Duration	Run start	Run end	Load type	Lower bound	Upper bound
dbo.lhtablest_1 → lht...	✓ Succeeded	3	3	38 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:08 AM	Full load	-	4
dbo.genericcdc_try2 → ...	✓ Succeeded	7,212	7,212	39 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:08 AM	Full load	-	2025-03-11T14:55:30.5...
dbo.cdf_try1 → cdf_try1	✓ Succeeded	5	5	1 min 0 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:30 AM	Full load	-	6
dbo.genericcdc → gene...	✓ Succeeded	8	8	37 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:07 AM	Full load	-	8
dbo.genericcdctry3 → ...	✓ Succeeded	7,227	7,227	41 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:10 AM	Full load	-	2025-04-29T14:13:41.2...
dbo.lhtablest → lhta...	✓ Succeeded	3	3	38 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:08 AM	Full load	-	6
dbo.lhtablest_cdctest...	✓ Succeeded	3	3	38 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:07 AM	Full load	-	5
dbo.gbqtest → gbqtest	✓ Succeeded	2	2	39 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:09 AM	Full load	-	45
dbo.Produce_gbq → Pr...	✓ Succeeded	2	2	39 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:08 AM	Full load	-	45
dbo.movies → movies	✓ Succeeded	2,180	2,180	36 sec	3/17/2026, 8:09:29 AM	3/17/2026, 8:10:06 AM	Full load	-	108583

Understand How Pipeline Scales

←----- Command and Control
 ←----- Data

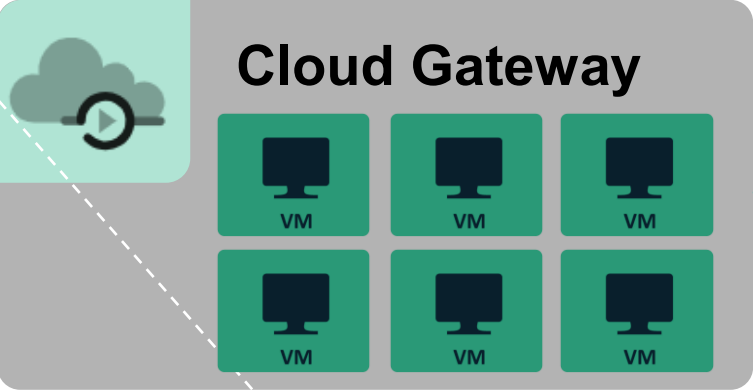


Flexible control flow & scheduling to scale out.
(multiple activities, concurrency, partitions)

Cloud



Cloud Data Stores

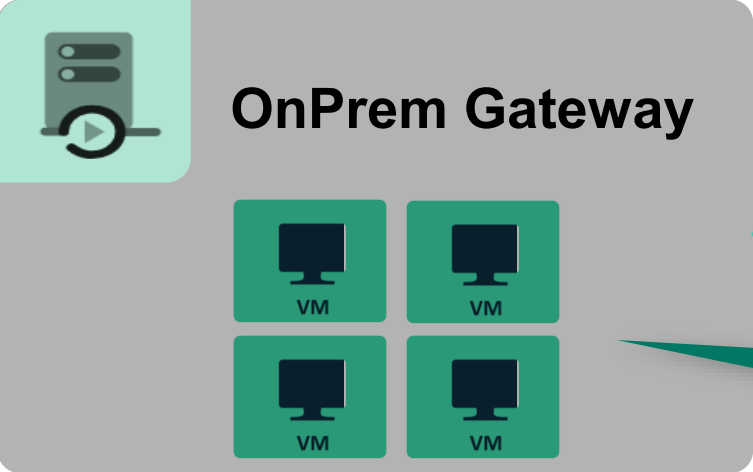


Azure Data Stores

Elastic managed infra to handle data at scale.
(configurable ITO per run)

On-prem

On-prem Data Stores



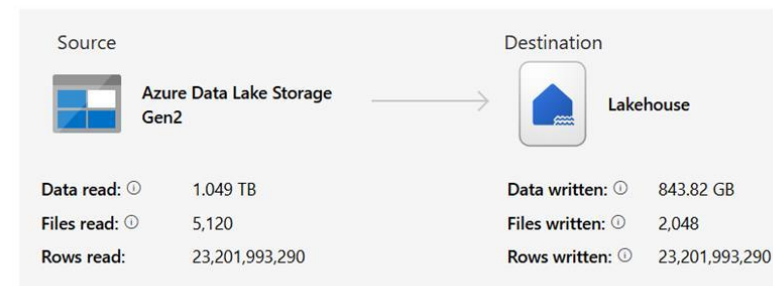
Customer managed infra with scaling options.
(powerfulness, concurrency)

Copy Performance Metrics (Parquet to Lakehouse)

Parquet -> LH (Normal)

Copy data details

loading 1t parquet to lh table



Status: ✔ Succeeded
Start time: 2/5/2026, 8:45:13 PM

Activity run ID: a785546f-d87f-430a-9402-510d59393019
Throughput: 558.176 MB/s
Total duration: 00:31:24
Duration breakdown: Start time: 2/5/2026, 8:45:16 PM

558 MB/s
31 min 24 sec

Parquet -> LH (Columnar) *(when schema mismatch)*

Copy data details

loading parquet to lh table columnar



Status: ✔ Succeeded
Start time: 2/13/2026, 2:34:07 PM

Activity run ID: f63a0d01-08be-4f0c-b46f-c4cc7f3d9d3e
Throughput: 1.635 GB/s
Total duration: 00:10:24
Duration breakdown: Start time: 2/13/2026, 2:34:10 PM

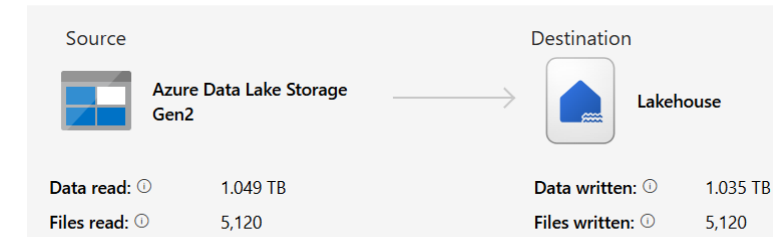
1.635 GB/s (3x)
10 min 24 sec

Parquet -> LH (V-Order) *(when schema match)*

Copy data details

loading 1t parquet to lh table binary

Performance tuning tips
To achieve better performance, you could disable 'Apply V-Order' which skips the V-Order optimization to your parquet files written in the destination.



Status: ✔ Succeeded
Start time: 2/2/2026, 11:00:10 AM

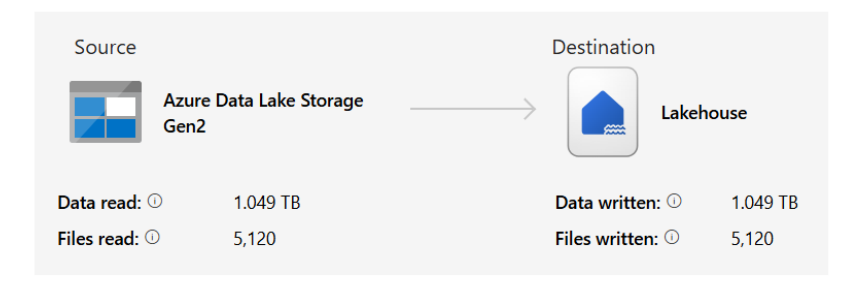
Activity run ID: f2250774-39c7-41db-b685-5a516062a79c
Throughput: 2.596 GB/s
Total duration: 00:06:51
Duration breakdown:

2.596 GB/s (5x)
6 min 51 sec

Parquet -> LH (Binary) *(when schema match)*

Copy data details

loading 1t parquet to lh table binary



Status: ✔ Succeeded
Start time: 2/12/2026, 4:17:08 PM

Activity run ID: 8a410f20-0024-4a37-90cf-df3286f81db7
Throughput: 5.192 GB/s
Total duration: 00:03:26
Duration breakdown: Start time: 2/12/2026, 4:17:10 PM

5.192 GB/s (10x)
3 min 26 sec

Before

After

Copy Performance Metrics (CSV -> Lakehouse)

CSV Folder -> LH

Copy data details

Loading 1T csv to LH table



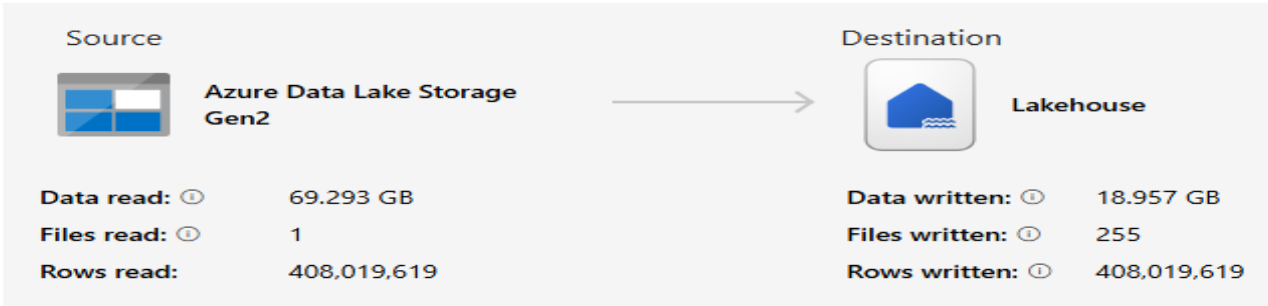
Status	✔ Succeeded
Start time	1/16/2026, 9:46:19 PM
Activity run ID	16a01a76-aac1-41df-bdbc-7c4239a16953
Throughput	1.541 GB/s
Total duration	00:11:25
Duration breakdown	
Start time	1/16/2026, 9:46:22 PM

1.541 GB/s
11 min 25 sec

CSV File -> LH

Copy data details

Loading single large CSV file to LH



Status	✔ Succeeded
Start time	2/13/2026, 1:56:17 PM
Activity run ID	77ed1024-b97a-4aab-8f14-cf4e3b46ae3b
Throughput	753.185 MB/s
Total duration	00:01:36
Duration breakdown	
Start time	2/13/2026, 1:56:20 PM

753.18 MB/s (50x)
1 min 36 sec

Throughput & Total Duration

Copy Performance Metrics with Warehouse

Parquet -> Warehouse (Direct)

Copy data details

loading 1t parquet to dw directly



Status: ✔ Succeeded
Start time: 1/17/2026, 11:45:17 AM

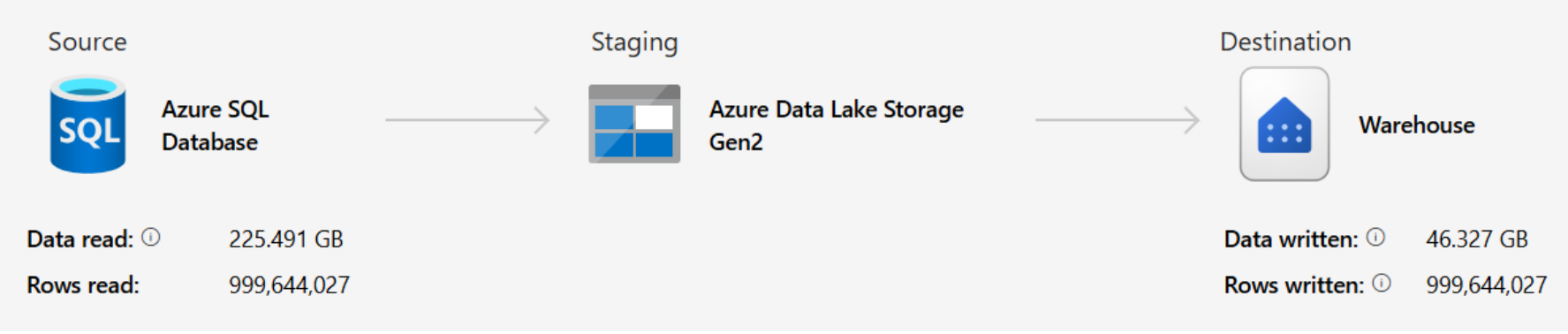
Activity run ID: 0eccbd5f-77ef-439a-a30b-41d96e75797f
Throughput: 1.702 GB/s
Total duration: 00:10:22
Duration breakdown:
Start time: 1/17/2026, 11:45:19 AM

1.702 GB/s
10 min 22 sec

SQL -> Warehouse (Staging)

Copy data details

Loading 1 billion rows from SQL to DW



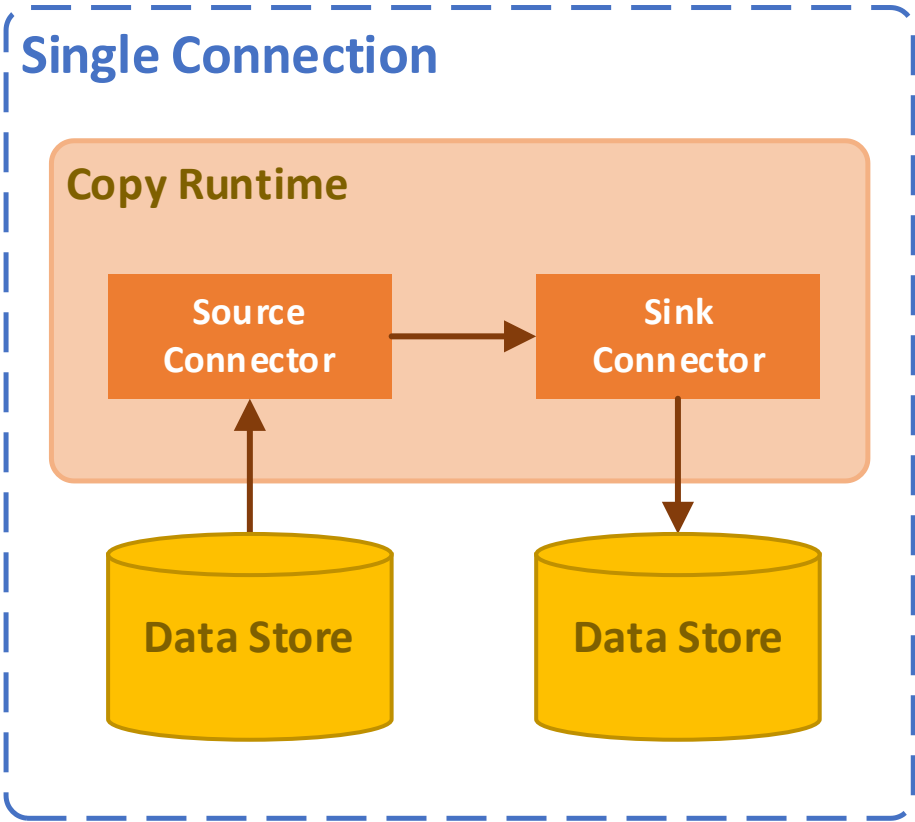
Status: ✔ Succeeded
Start time: 2/13/2026, 2:30:18 PM

Activity run ID: a414ca18-30ac-402d-bdf9-365101d8ea15
Throughput: 1.427 GB/s
Total duration: 00:02:46
> Source to Staging duration: 00:01:46
> Staging to destination duration: 00:00:59

1.427 GB/s
2 min 46 sec

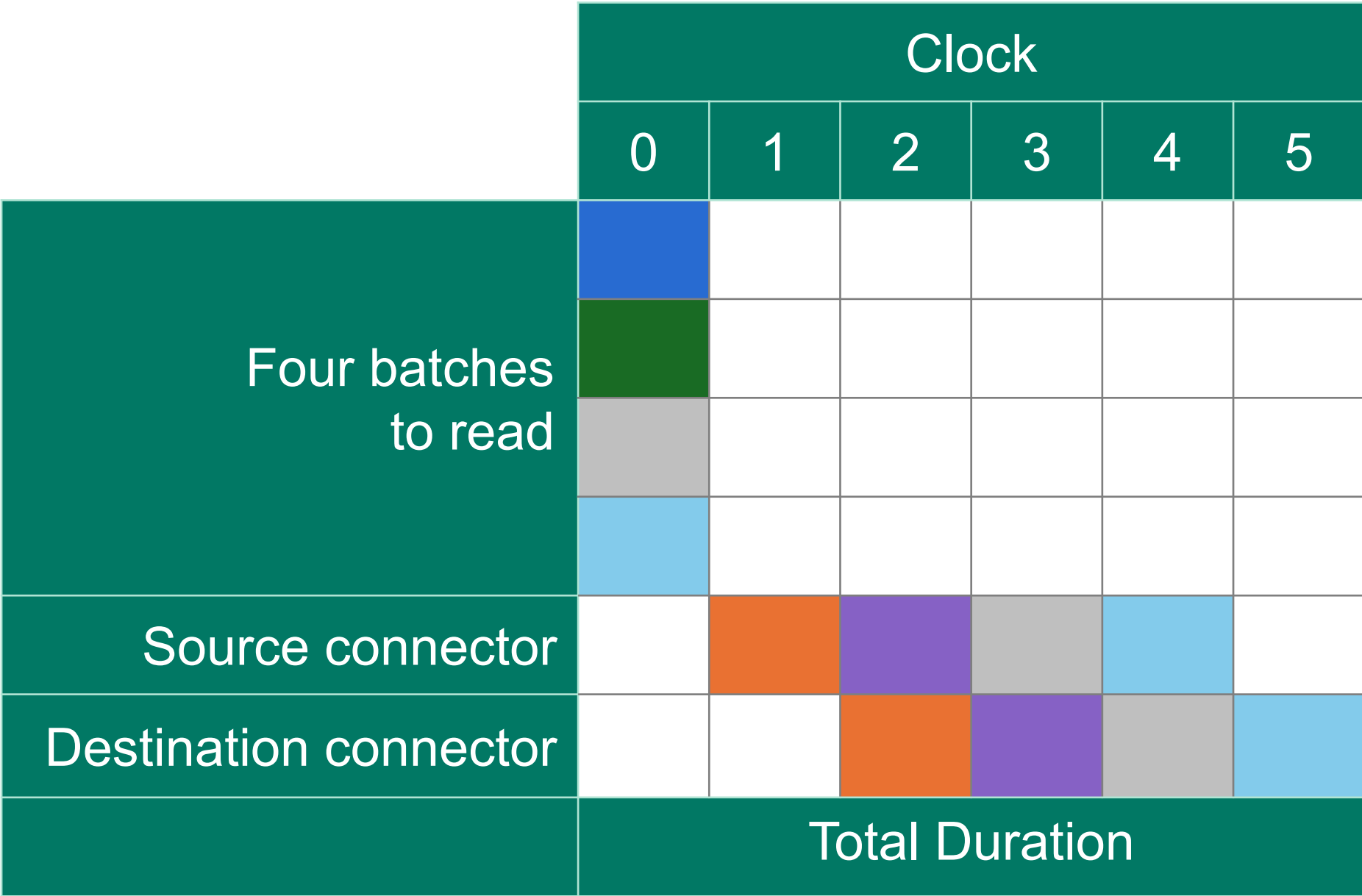
Throughput & Total Duration

How Copy Scales – Connection Level



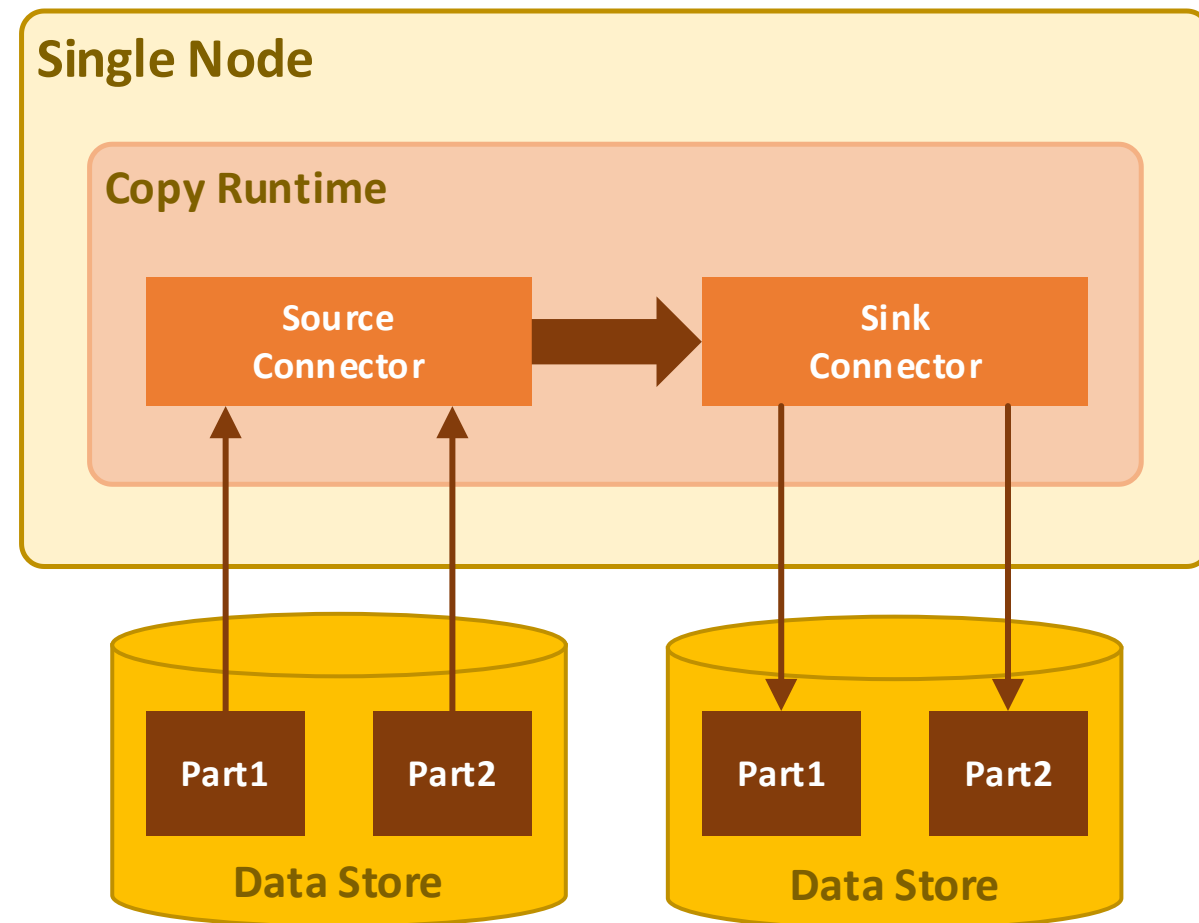
Pipeline processing

- **Less memory:** No need to load everything in memory and then write
- **Less total duration:** Read and Write are in parallel



How Copy Scales – Node Level

Multi-Connection on Single Node



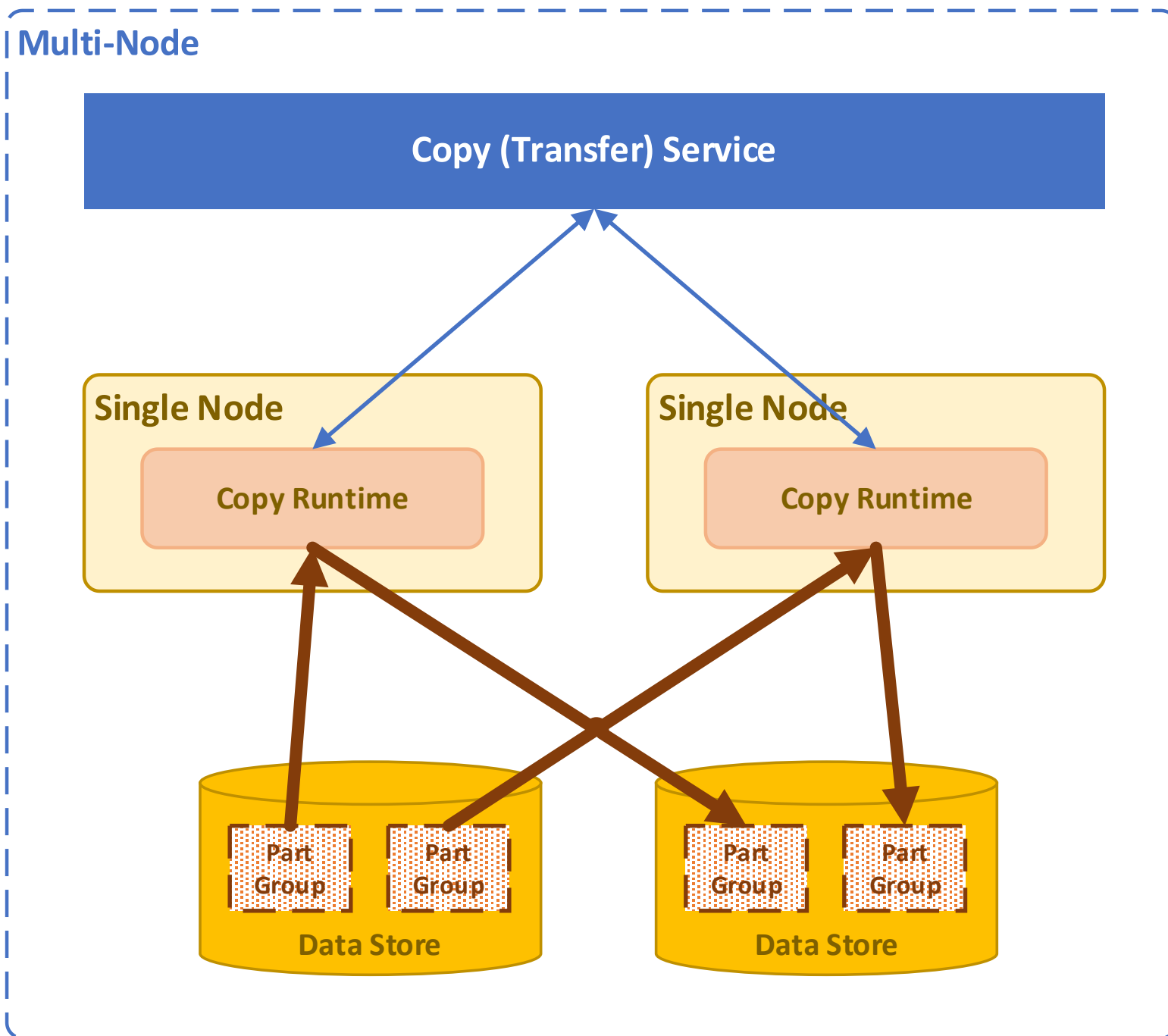
Producer & Consumer design

- Data is partitioned for multiple concurrent connections (even for single large file)
- Full node utilization: Data can be partitioned differently between source and destination to avoid any starving / idle connection

Partitions come from

- Physical partitions setup on DB side
- Dynamic partitions from different queries
- Multiple files
- Multiple parts from a single file

How Copy Scales – Multi-Node Level



Copy(Transfer) Service: Manages Copy activities

- Copy State Management / Monitoring
- Cross Machine Parallelism
- Billing
- Manages Compute Clusters and tasks running on them

Easily scale out to multiple stateless nodes when available partitions are more than what one node can afford.

Theoretically no upper limit on the performance.

Copy Design Considerations

Concepts

- **Intelligent Throughput Optimization (ITO):** A measure that represents the power (a combination of CPU, memory, and network resource allocation) used for a single Copy activity.
- **Parallel Copy:** The maximum number of threads within the Copy activity that read from source and write to destination in parallel.
- **Max Concurrent Connections:** The upper limit of concurrent connections established to the data store during the activity run. (Usually used to avoid throttling)

Connector Partitioning Behavior

- Files based connectors
 - **CSV files:** Data is split into partitions of 256 MB each
 - **Other file formats:** One file = one partition
- Tabular connectors with partition read support
 - MSSQL family, Oracle, Teradata, Azure PostgreSQL, Netezza, SAP HANA, SAP Table, SAP Open Hub
- Tabular connectors with auto-partition support
 - Lakehouse Table, Salesforce
- **Default perf settings**
 - Intelligent throughput optimization: **Auto** (Use maximum available VM resources)

CopyJob Optimizations

- CopyJob supports trickle feed patterns for near real-time data ingestion
- Schedule CopyJob at frequent intervals (e.g., every 5–15 minutes) to continuously move delta changes
- Incremental (Watermark) and CDC modes ensure only new/changed data is picked up each run
- CopyJob now introduces automatic partitioning for tabular and file sources
- No manual partition configuration required — CopyJob intelligently determines optimal partitioning strategy
- Adaptive batch sizing — dynamically adjusts batch sizes based on source throughput and sink write latency
- Enhanced parallelism controls — fine-grained tuning of concurrent table copies for maximum throughput across multiple tables
- Cost-aware scaling — balances performance with capacity unit consumption

Troubleshoot Performance Issues

Common bottlenecks

- Network (cross region/OnPrem)
- Source/Destination data stores
 - Busy neighbors on the same data store
 - Low service/compute tier
 - Throttling mechanism on data store
 - Complex/unoptimized query (huge timeToFirstByte)
- Compute/Memory-intensive Operations
 - Serialization / Deserialization / Compression / Decompression (e.g. gzip)
 - Data Type conversion (e.g. string to number)
 - Row-Column Transpose (e.g. CSV to Parquet)

Troubleshoot Performance Issues – Example

Copy data details

Copy data1

Source: Azure SQL Database → Destination: Lakehouse

Data read:	57.733 MB	Data written:	47.61 MB
Rows read:	232,590	Files written:	1
		Rows written:	232,590

Status: ✔ Succeeded

Start time: 3/6/2024, 7:45:31 PM

Activity run ID: 19f15fea-2021-4be9-a5dd-3d14f81401f9

Throughput: 4.441 MB/s

Total duration: 00:00:27

Duration breakdown

Start time: 3/6/2024, 7:45:34 PM

Optimized throughput: Standard

Used parallel copies: 1

1

Close

Parameters Variables Settings **Output**

Pipeline run ID: 7fdb813b-73c7-470c-bd04-cff8d959d107 🔗 🔄 ⓘ Pipeline status ✔ Succeeded View run detail ⌵ Export to CSV ⌵ ☰ 🔗

Showing 1 - 1 items

Activity name	Activity status	Run start	Duration	Input	Output
Copy data1	✔ Succeeded	3/6/2024, 7:45:31 PM	33s	→	📄

2

Indicating that most time was spent on preparing/running the query on DB side. Optimize the query or upgrade DB tier to eliminate this bottleneck.

Bottleneck at Source

```
usedParallelCopies: 1,
"profile": {
  "queue": {
    "status": "Completed",
    "duration": 1
  },
  "transfer": {
    "status": "Completed",
    "duration": 13,
    "details": {
      "readingFromSource": {
        "type": "AzureSqlDatabase",
        "workingDuration": 10,
        "timeToFirstByte": 9
      },
      "writingToSink": {
        "type": "Lakehouse",
        "workingDuration": 1
      }
    }
  }
},
"detailedDurations": {
```

3

Troubleshoot – Workspace Monitoring Integration

Centralized Monitoring Hub

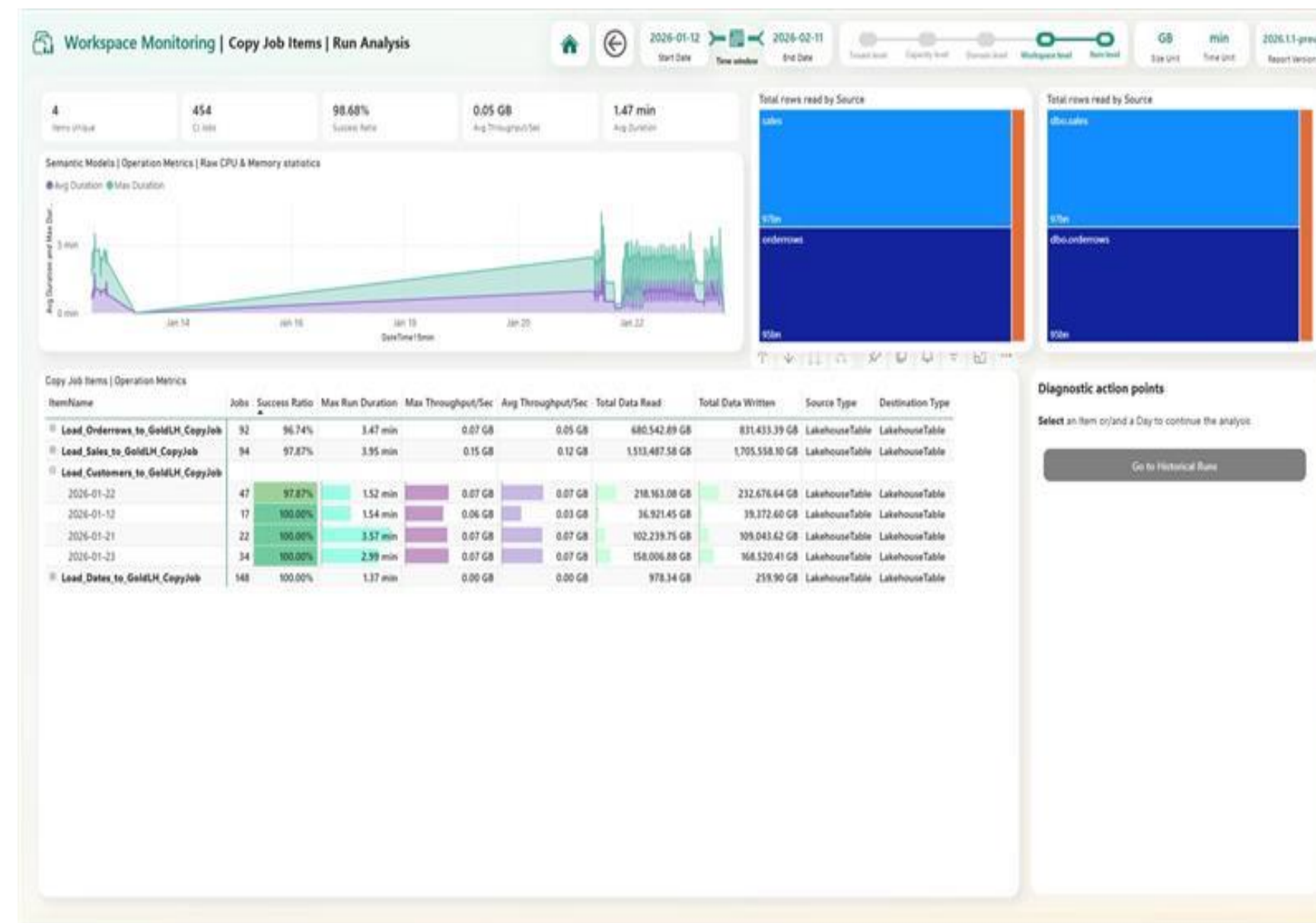
- View all Copy runs across workspaces from a single pane — no need to open the authoring surface
- Filter by status (Running, Succeeded, Failed) to quickly identify issues
- Up to 30 days of historical run data available for trend analysis and root-cause investigation

Rich Data Movement Diagnostics

- **Row-level metrics:** Rows read, rows written, files read, files written
- **Volume & throughput:** Data read/written with average throughput (KB/s) for performance benchmarking
- **Incremental copy tracking:** Load type (Full vs Incremental), lower/upper watermark bounds per table

Troubleshooting Failed Runs

- Drill into any failed run to see exact error messages and failure diagnostics
- Per-table status breakdown — pinpoint which specific table copy failed in a multi-table job
- Compare run start/stop timestamps and durations across runs to detect performance degradation



Dataflow Gen2

Dataflow Versions

- There are three versions of Dataflows:
 - Dataflows Gen1
 - Dataflows Gen2
 - Dataflows Gen2 CICD (replaces Gen2)
- Dataflows Gen2 CICD are where all the innovation is happening – especially around performance
- Upgrade to Dataflows Gen2 CICD by:
 - Right-clicking on an existing Dataflow
 - Using Pat Mahoney's accelerator in the Fabric Toolbox
<https://github.com/microsoft/fabric-toolbox/tree/main/accelerators/DFG2-migration-accelerator>

New Dataflow Gen2

Name

Enable Git integration, deployment pipelines and Public API scenarios

Create Cancel

Preview-Only Steps

- New tool for improving design-time performance:
 - Step 1: Add a filter step ("classic" filter, top rows, etc.)
 - Step 2: Select "Enable only in preview" in the Step menu
- Result: preview queries run over a subset of the data, making them up to orders of magnitude faster

i This data preview uses preview-only steps. Results may differ when running the dataflow. [Learn more](#)

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
1	10248	VINET	5	7/4/1996, 12:00:00 AM	8/1/1996, 12:00:00 AM
2	10249	TOMSP	6	7/5/1996, 12:00:00 AM	8/16/1996, 12:00:00 AM
3	10250	HANAR	4	7/8/1996, 12:00:00 AM	8/5/1996, 12:00:00 AM

Keep top rows ?

Specify how many rows to keep.

Number of rows *

OK **Cancel**

Source

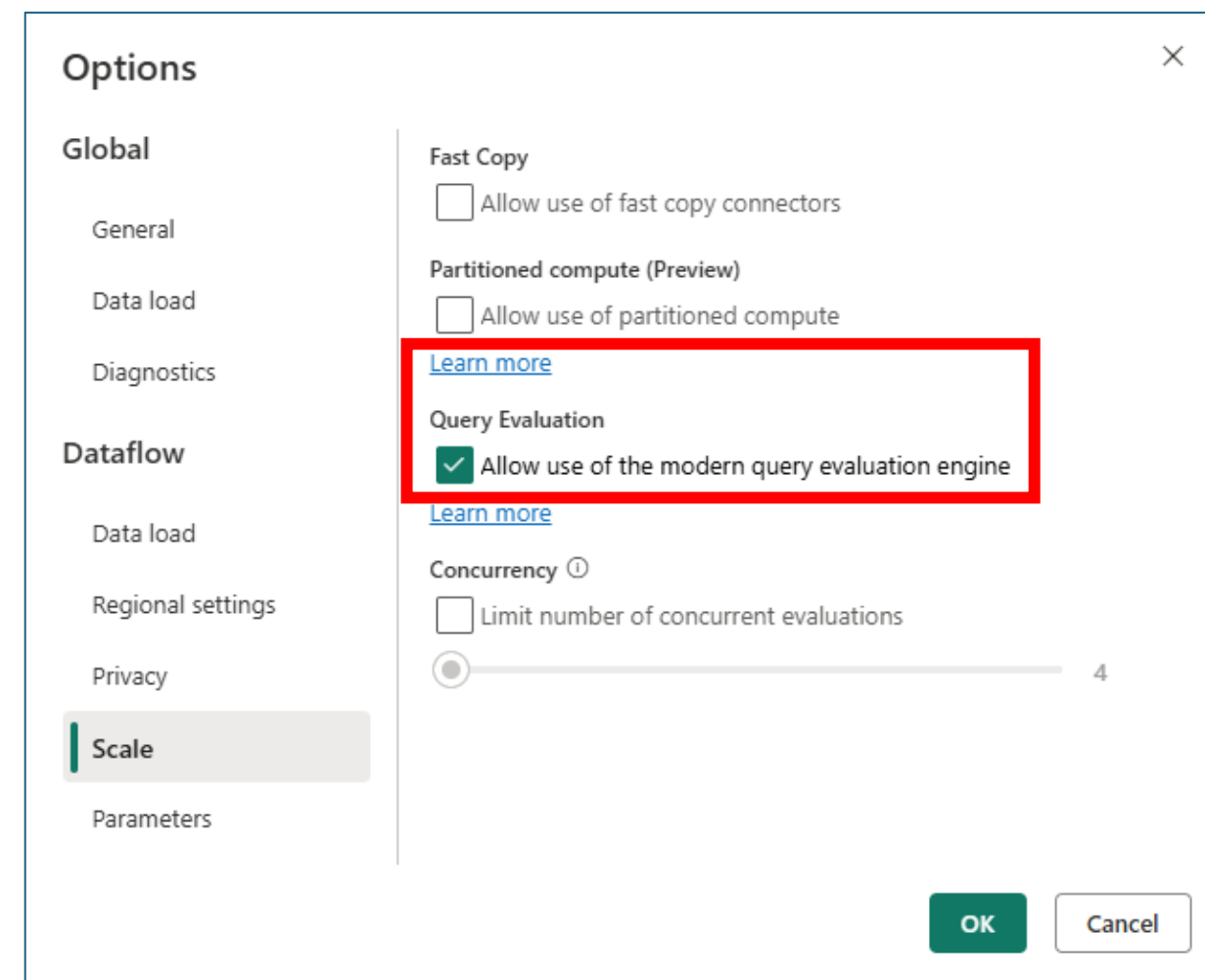
Navigation 1

Kept top rows

- Edit settings
- Explain this step
- Rename
- Delete
- Delete until end
- Insert step after
- Move before
- Move after
- Extract previous...
- View data source query
- View query plan
- Enable only in previews
- Properties...

Modern Query Evaluation Engine

- New “M” processing infrastructure that leverages platform advancements. Significantly improves performance of type conversions and in-memory transformations
- Available only for Dataflow Gen 2 with CI/CD
- Substantially faster than previous M query evaluation engines (almost 2x faster in some scenarios). Note: improvements vary based on data source and transformations.



Dataset	Size	vs. Dataflow Gen1	vs. Dataflow Gen 2 (old engine)
2017 Yellow Taxi Trip Data (1 CSV file)	9.8GB	~60% faster	~40% faster
2017 – 2021 Yellow Taxi Trip Data (5 CSV files)	32.2GB	~65% faster	~45% faster

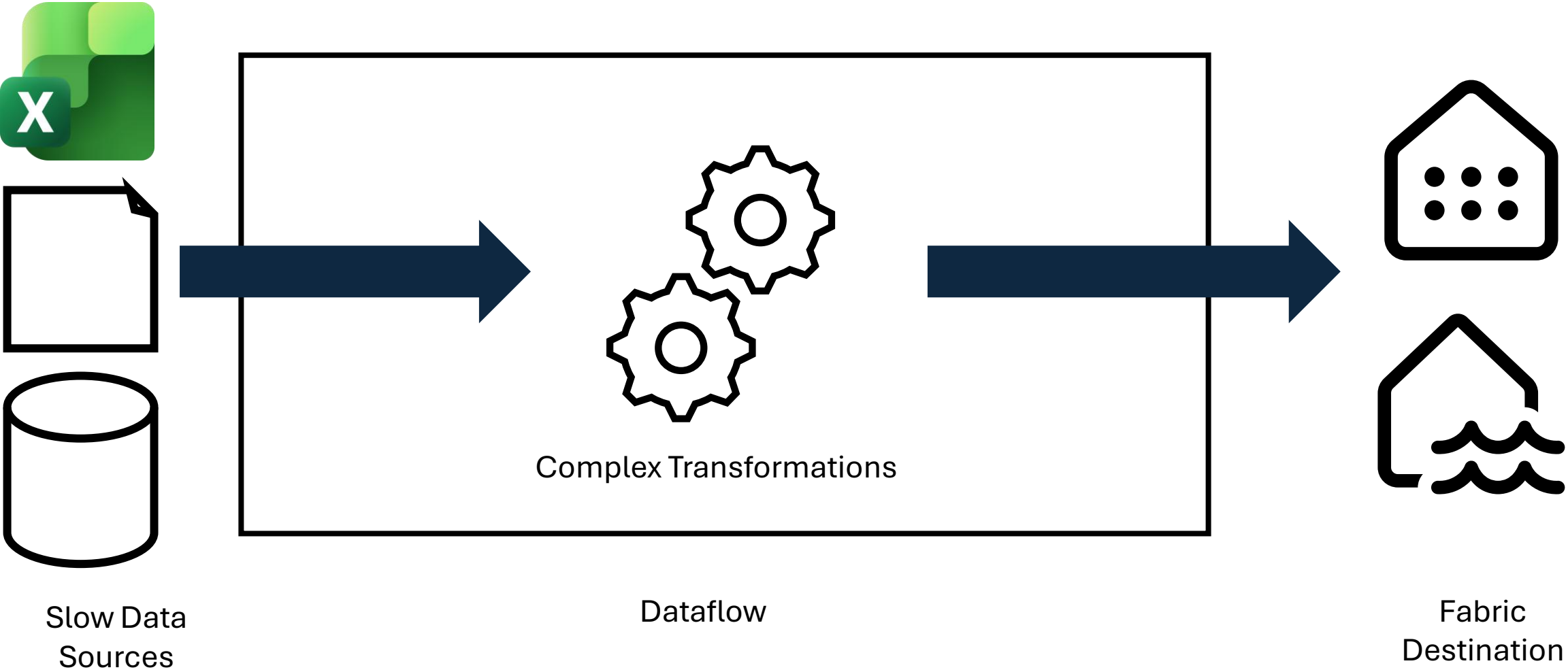
Fast Copy

- Dataflows can sometimes use the Copy Activity engine → “Fast Copy”
- Can result in big performance gains
- Only for:
 - Data sources supported by Copy Activity
 - Limited transformations

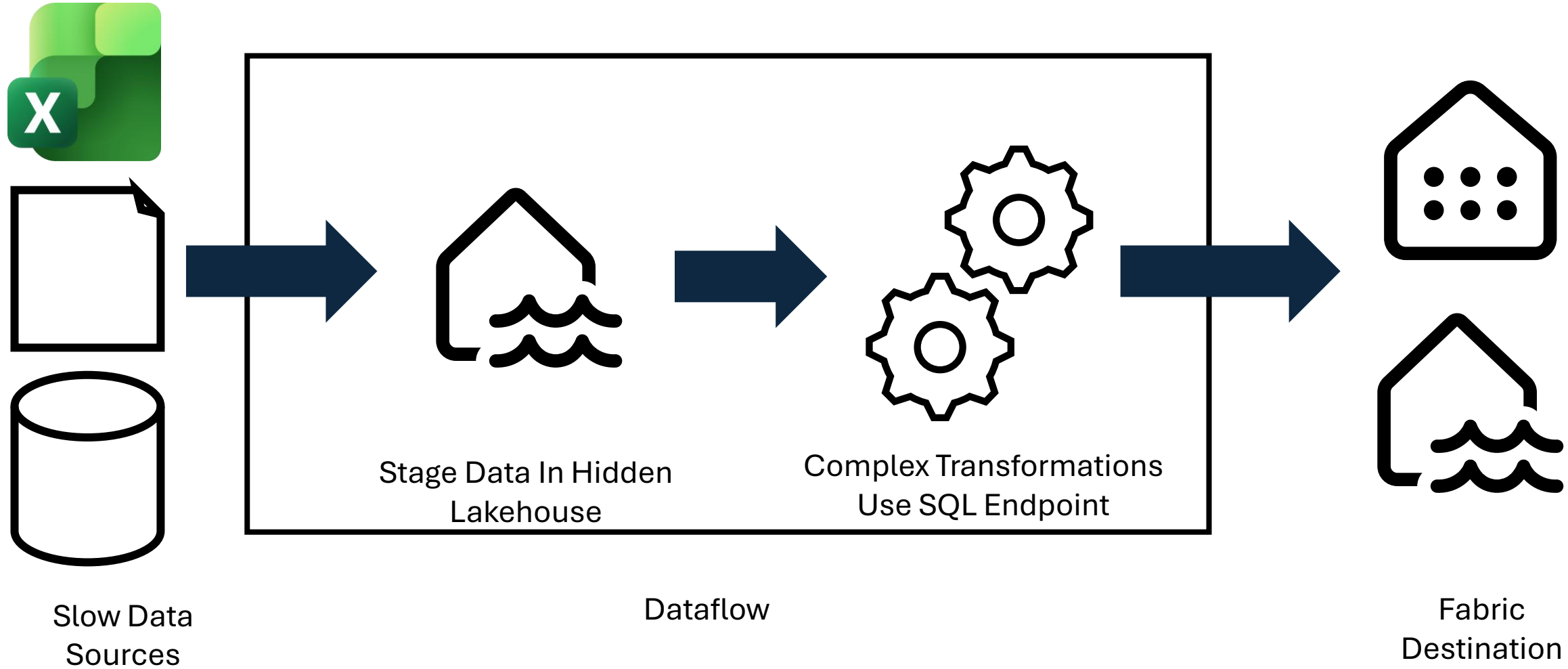
The screenshot shows the Azure Data Factory interface. On the left, a data table is displayed with five columns labeled 1²3 r_1 through 1²3 r_5. The table contains several rows of numerical data. On the right, a task list is visible under the heading 'Applied steps'. The tasks listed are: Source, Filtered hid..., Invoke cust..., Renamed c..., Removed o..., Expanded t..., and Changed c... (with a red 'X' icon). A tooltip is overlaid on the 'Source' task, indicating that it will be evaluated by the data source and that this step is going to be evaluated with fast copy. A 'Learn more' link is also present in the tooltip.

1 ² 3 r_1	1 ² 3 r_2	1 ² 3 r_3	1 ² 3 r_4	1 ² 3 r_5
27450000	3240000	26640000	22770000	2016
1080000	28800000	29070000	15210000	2547
16020000	24210000	26460000	15750000	846
2070000	24840000	12150000	3150000	1926
30870000	9720000	18270000	8010000	594
4320000	17640000	21420000	18990000	1458

Staging



Staging



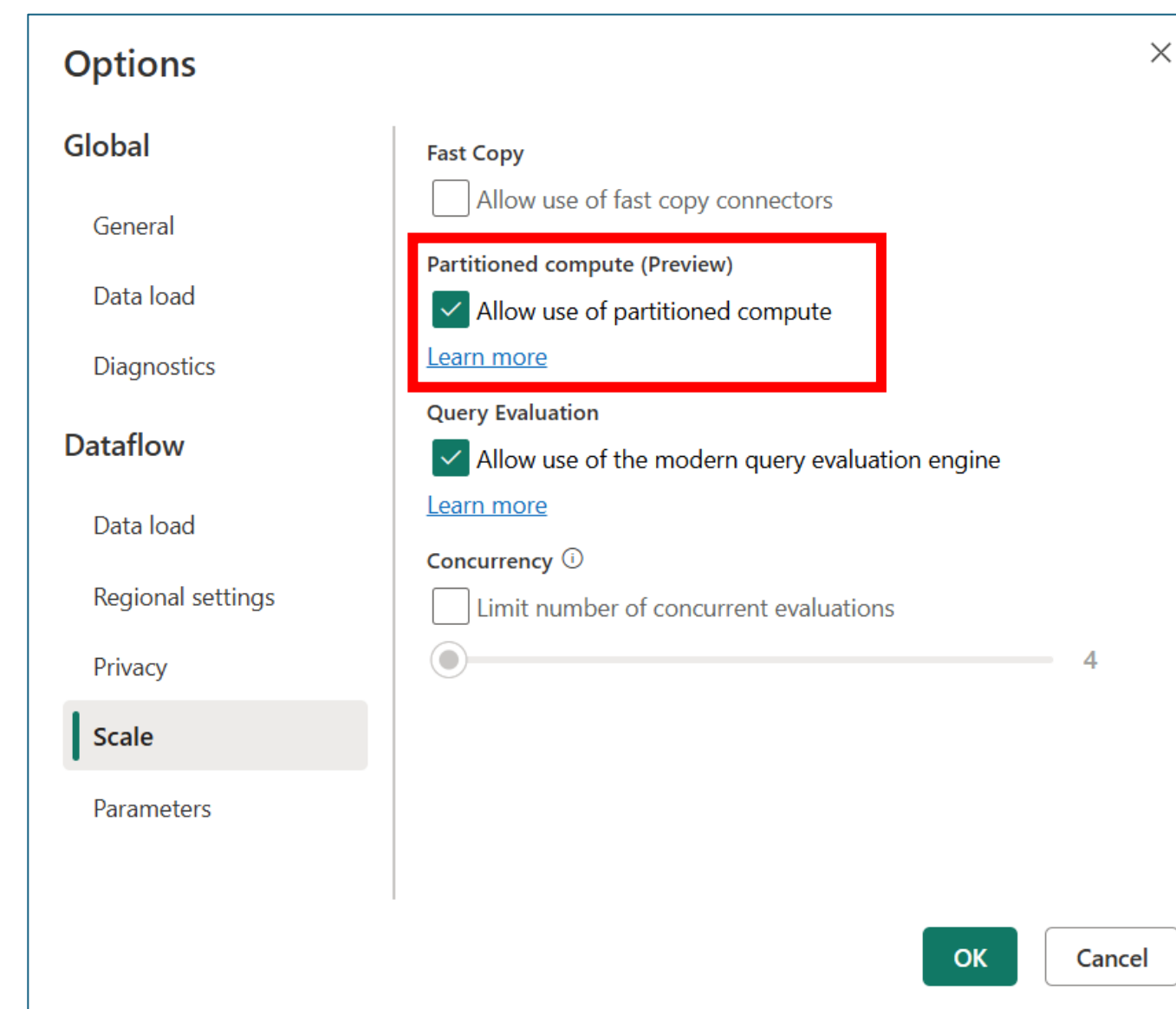
Partitioned Compute

- New Partitioned Compute capability enables parallelized evaluation of the rows in a table
 - Step 1: Enable Partitioned Compute Option
 - Step 2: Use the standard Combine Files experience

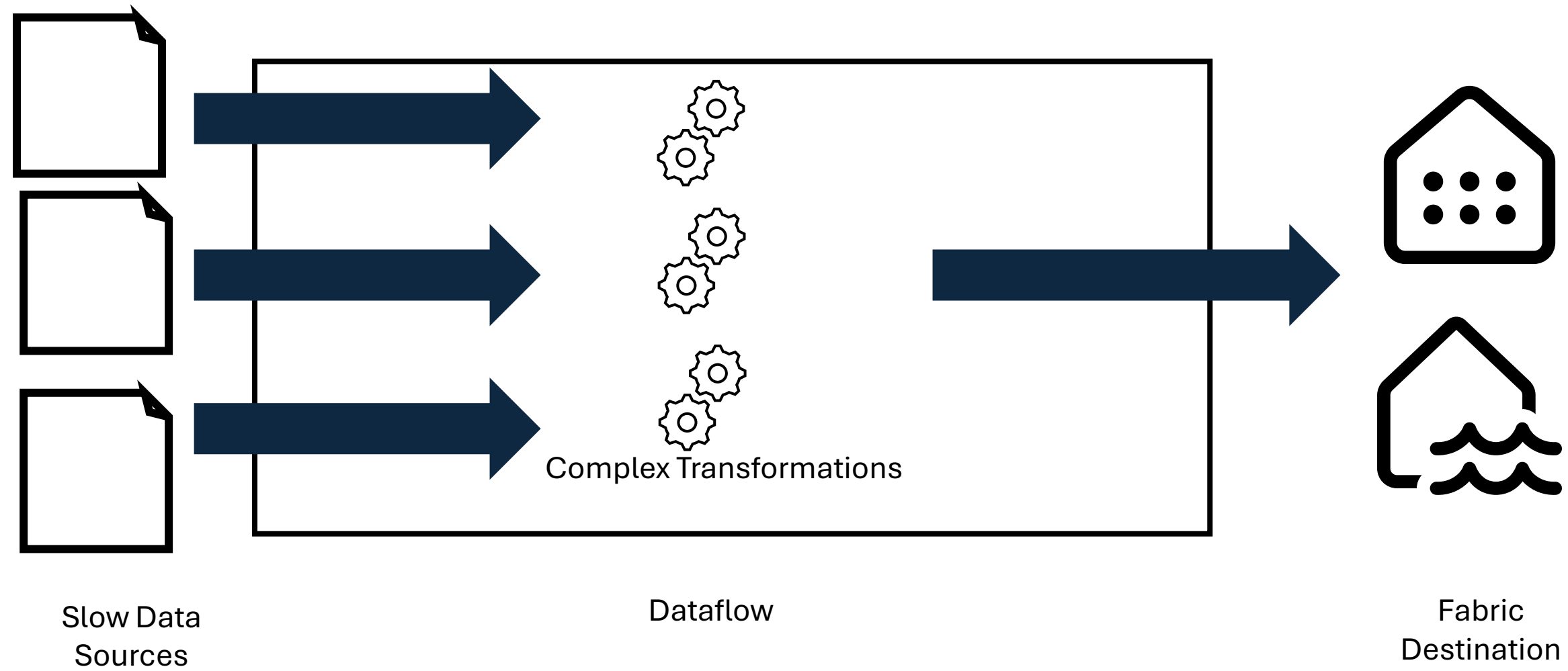
- Result – generated M will now include this:

```
Table.ReplacePartitionKey(  
withRelativePath, {"Relative Path"})
```

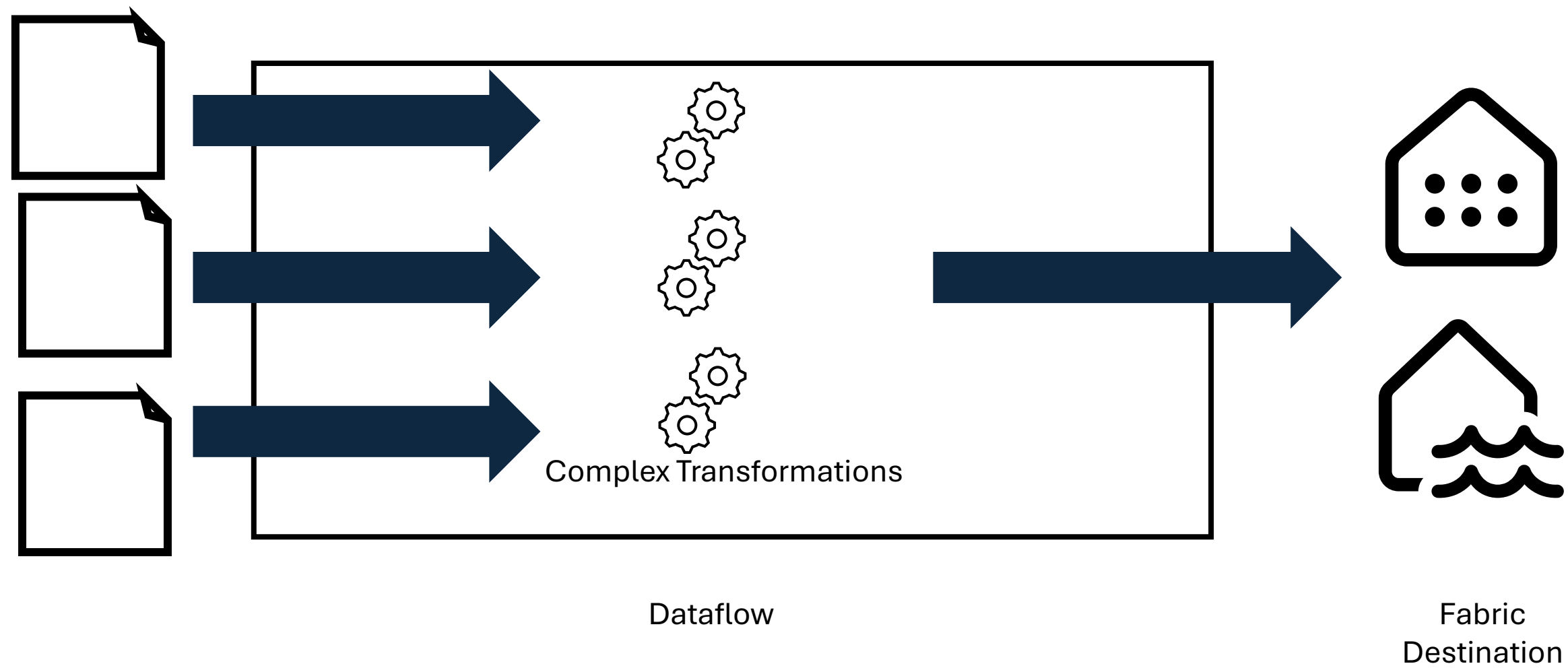
- Custom specification of partition key can be used for advanced scenarios beyond files



Partitioned Compute



Partitioned Compute



Copilot

- Different ways of solving the same problem in Dataflows can result in different refresh performance
- Copilot inside the Dataflows editor cannot answer questions about optimizing performance
- However M365 Copilot (or your favorite AI tool) can look at the M code for your queries and suggest optimizations



Reasoning completed in 3 steps >

Yes — the big slowdown in your current query is the **full sort**:

```
1 #"Sorted rows" = Table.Sort("#Filtered rows", {"ACTUAL_COST", Order.Descending}),  
2 #"Drill down" = #"Sorted rows"{0}[ACTUAL_COST]
```

On a very large CSV, `Table.Sort` forces Power Query to **materialize and sort the entire filtered table** just to get one value. A single-pass aggregate (`max`) is much cheaper. The Power Query engine evaluates steps and can sometimes reorder/optimize, but a `Sort` is inherently expensive because it requires a full ordering rather than a scan. [Architecture: Power q...](#)

Below are two faster rewrites.

Option A (usually fastest + simplest): `Table.Max` after filtering, with minimal columns

`Table.Max` returns the row with the largest value in a specified column, without requiring a full sort.

[Table.Max - PowerQue...](#)

This version also improves performance by:

- **Filtering before type conversion** (so you parse fewer rows)
- **Selecting only the columns you actually need** (so you carry less data through the pipeline)

Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel



Share your feedback directly with our
Fabric product group and researchers.

<https://aka.ms/JoinFabricUserPanel>

Join the SQL User Panel



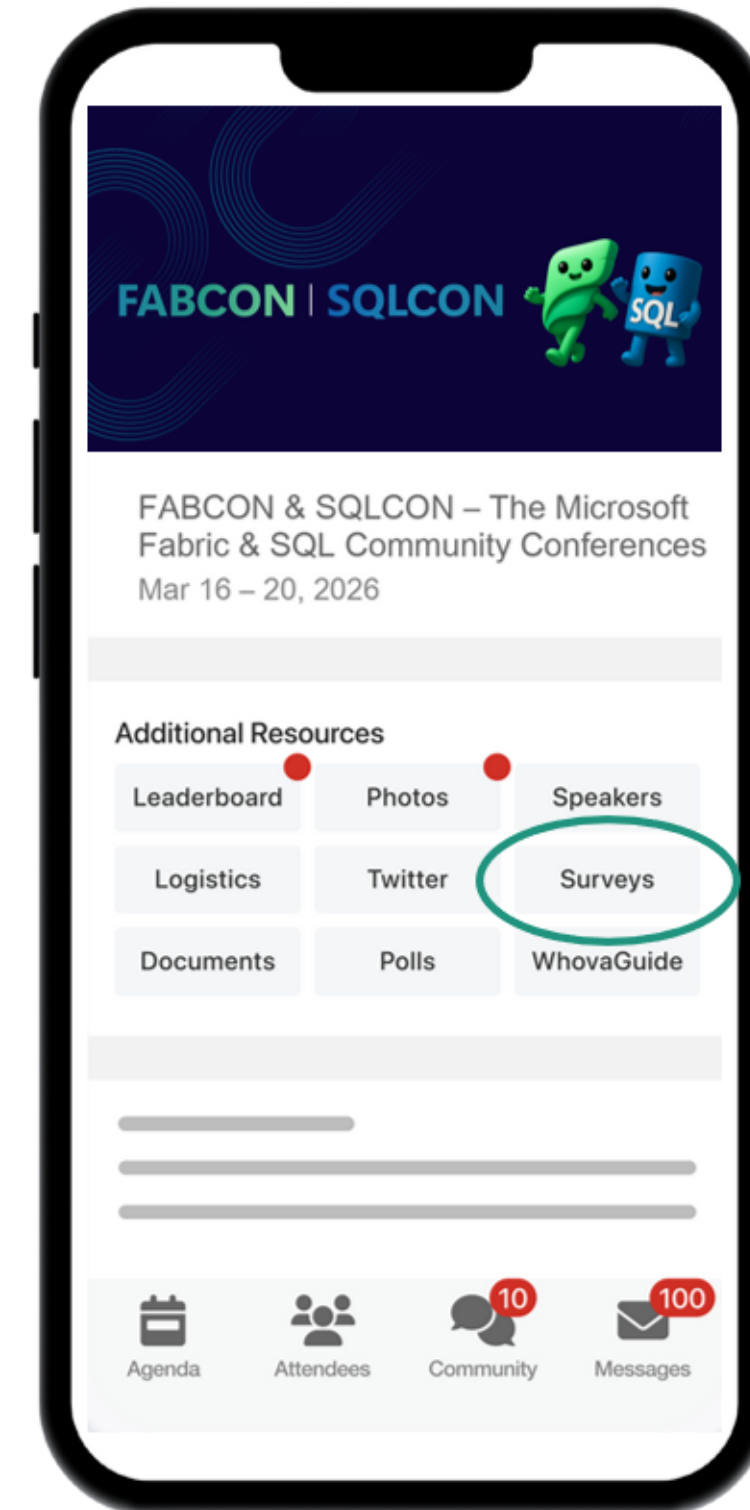
Influence our SQL roadmap and ensure
it meets your real-life needs

<https://aka.ms/JoinSQLUserPanel>

How was the session?



Complete Session Surveys in
Whova for your chance to WIN
PRIZES!



Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

Request your voucher by March 23, 2026.

<https://aka.ms/fabcon/cert100>

