Microsoft reinvents sales processing and financial reporting with Azure

Core Services Engineering (CSE, formerly Microsoft IT) is moving MS Sales, the Microsoft revenue reporting platform, from on-premises datacenters to the cloud with Microsoft Azure. But we view this as more than just a move to the cloud—it's an opportunity to re-imagine and redesign the way MS Sales infrastructure functions. To prepare for the migration, we examined several options for hosting MS Sales in Azure, and came away with a clear direction for transition design, planning, and deployment.

MS Sales for revenue and sales data

MS Sales manages Microsoft product and service sales data. Transaction data is conformed, aggregated, and enriched by MS Sales to provide accurate revenue reporting and analysis. MS Sales gives a consistent view of Microsoft businesses and production revenue, and it enables better, faster strategic decisions. People can query purchase, sales, budget, and forecast data, and drill down to see more transaction details.

The MS Sales environment includes:

- More than 30,000 users at Microsoft, which includes people in finance, sales, marketing, and executives.
- More than 1,450 external data sources and more than 65 internal data sources.
- Twenty-one years of sales data (10 past years, this year, and 10 projected years), depending on the reporting pivot.
- A daily average of 2.6 million transactions.

MS Sales publishes data that's aligned with the Microsoft financial calendar. The publishing processes include daily, weekly, and—the most critical—fiscal month-end (FME) data for restatement, forecasting, and budgeting. FME data is time-sensitive because MS Sales data is growing so fast. And the system needed more processing capacity to keep pace with the expanding number of revenue records and details.

We've been experiencing several challenges with the on-premises MS Sales environment, including limited scalability and agility, a complex ecosystem, cumbersome data processing models, and increasing costs. Our goal in migrating MS Sales to Azure is to address these challenges and position MS Sales for success well into the future.

MS Sales now

The MS Sales system was built 20 years ago to report Microsoft revenue. It's the company's standard revenue reporting system, and is pivotal to strategic business, management, and financial decisions. MS Sales helps employees report on and analyze revenue accurately and consistently—and on schedule. Timely, accurate sales data is crucial to assessing Microsoft performance and maintaining a competitive position.

The original solution for MS Sales is hosted in on-premises datacenters and includes:

- Fifty-three servers.
- Thirty Microsoft SQL Server databases.
- Approximately 35 TB of data storage.

MS Sales components and functionality

The original MS Sales architecture ingests data, processes and transforms significant data, creates star schema data marts, and distributes these marts for querying and consumption by other systems. Querying is primarily via

Microsoft Reporting Analytics (MSRA), an Excel add-in that generates and executes SQL queries based on the user's definitions. The architecture supports five major functions:

- **Ingestion.** MS Sales ingests data from more than 1,450 sources. Most of the sources are external partners, such as manufacturers, distributors, and retailers. Data is ingested through a process that's supported by SQL jobs, and Windows services data is batch-loaded into the MS Sales Warehouse (a SQL database). Some basic data validation is performed as part of these processes, mostly with text files or Excel files. The most complex data ingestion process is multi-threaded and handles channel files, customer files, and organizational taxonomy, and it applies parenting, merging, and other complex business rules. One of our focus points in the next phase is to stream partner and customer data for ingestion.
- **Warehouse.** Ingested data ends up in the MS Sales Warehouse SQL Server database. Data from other sources—such as forecasting, budgeting, and planning data—and SAP are loaded directly here. The warehouse holds approximately 5 TB of data. The warehouse performs several key data processing jobs, such as:
    - Key matching between sales and customer data.
    - Allocations, deferrals, and recognitions.
    - License channel and license customer generation.
- **Factory.** Data from the warehouse is log-shipped to the factory server, where all major business data is processed, and the final MS Sales database is produced for reporting. Total factory data is about 4.5 TB of uncompressed data. Jobs are scheduled via NT Batch, and 15 complex stored procedures are executed. The factory server requires a high level of processing capability. For example, in a continuing push to adopt hardware that can support MS Sales functionality, we recently moved the factory to a two-server architecture, using two 72-core servers with 2 TB of RAM to run factory processes in parallel.
- **Distribution.** As the MS Sales database is created, it's cloned to 18 reporting servers. Each reporting server hosts just over 500 GB of compressed data. One server distributes the MS Sales database to downstream consumers. MS Sales data is copied to more than 220 more environments, where it's merged with other data for various reporting needs.
- **User experience.** The MSRA tool offers ad-hoc query definition in a pivot table. Users are offered a business-friendly view of the data fields with organized hierarchies and simple names. They can immediately schedule or execute queries. MSRA generates Transact-SQL (T-SQL) based on source-specific metadata. The T-SQL is executed by a middle tier server, and results are stored or retrieved immediately. Results are presented in an Excel PivotTable, where users can take full advantage of Excel capabilities. We have 25,000 MSRA users who can run ad hoc queries against MS Sales. Approximately three million queries were run in the last fiscal year.

Figure 1 illustrates the MS Sales on-premises infrastructure.

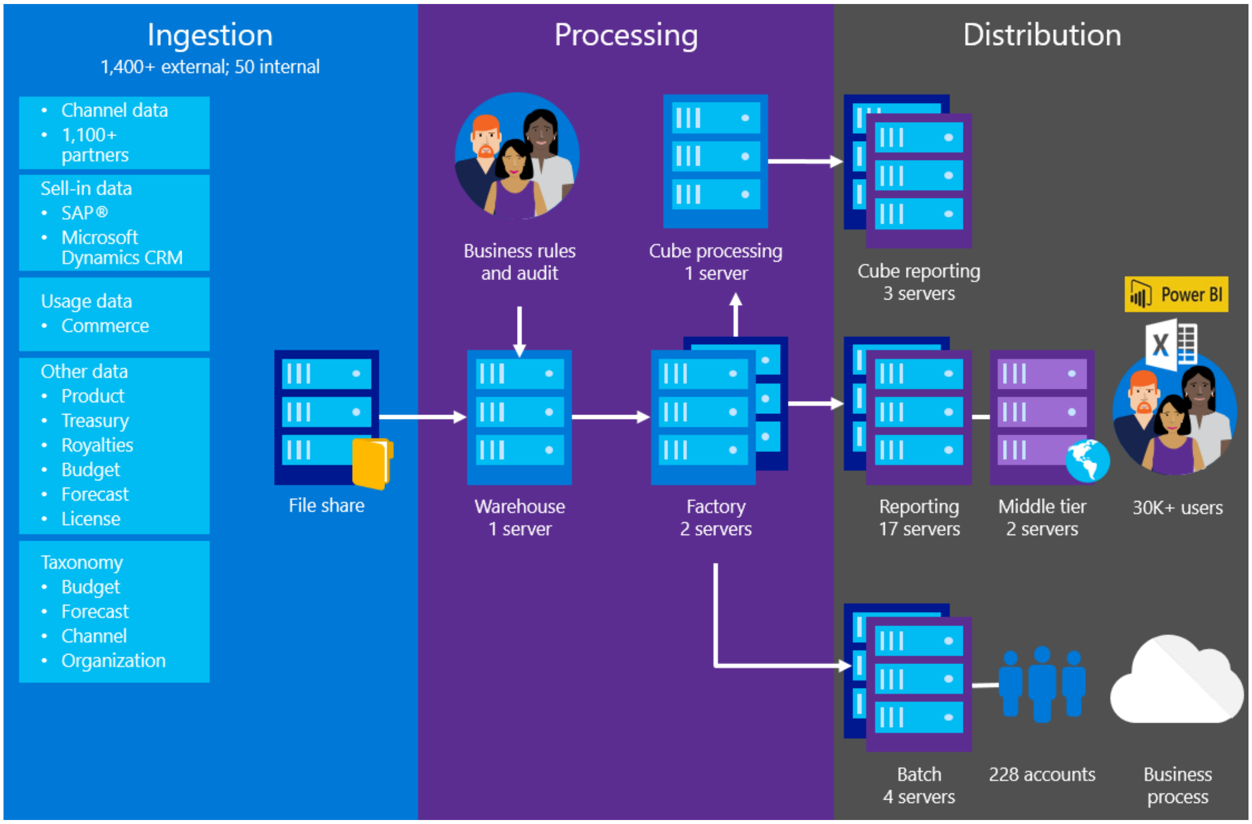

Figure 1. On-premises infrastructure for MS Sales

Redesigning MS Sales in the cloud

CSE is in the midst of a massive migration of our IT infrastructure to Microsoft Azure. Azure is the default target for our new solutions, and soon we'll have over 90 percent of our IT infrastructure hosted in Azure. We are also following this cloud approach with MS Sales in Azure. Our MS Sales cloud-based solution is based on several key goals:

- **Scale.** As our MS Sales environment and data grow, and our business model changes, we want an Azure-based solution that can scale to meet future data growth and changing business models.
- **Speed.** We want to increase the speed for MS Sales to ingest and generate data and process transactions. The time it takes to generate data needs to be significantly reduced so that our users have the most recent data possible to make business decisions.
- **Agility.** Incorporate new business models, acquisitions, and divestitures without slow and cumbersome engineering efforts.
- **Complexity.** We want to reduce the behavioral complexity of MS Sales and make the infrastructure and logical layout less complicated and easier to run.
- **Combinability.** We want to modernize the infrastructure to open new capabilities. We want to combine big data (such as marketing and usage) with core financials and use machine learning to give deeper insights into our sales.

Examining MS Sales architecture in Azure

Azure allows us to rethink data distribution and consumption in MS Sales and redefine what the data flow looks like. We are using many Azure-native services for data processing, so we can generate more granular processing and reporting components. We envision most of MS Sales being developed and hosted in Azure big data handling components. This greater level of granularity and native support for data manipulation leads to more parallel processes and quicker data delivery. The data flow components in the cloud include:

- HD Insight and Spark processing
- Azure Event Hubs
- SQL Azure Database
- Azure Data Lake
- Azure Data Warehouse

Our MS Sales data flow is based on three primary components: data ingestion, business rules, and pipeline processing.

Data ingestion

We ingest data from several internal and external sources. This, in itself, is a challenge because our ingestion methods need to be compatible with our data sources.

We use Event Hubs as the primary method for data ingestion in MS Sales. Event Hubs allow us to process transactions from multiple sources and scale to handle transaction input. After pulling transactions, we use Data Validation Services to ensure that the data coming into MS Sales corresponds with what we expect from our data providers. Data is compared to templates that we receive from our data providers and send through to Event Hubs if the data is valid.

We're working on a new component of MS Sales that uses Spark to process incoming transactions and match them with the list of data providers so we can confirm where the data is coming from. The Spark matching process has three primary components:

- **Domain match and attribution.** Some transactions don't identify the partner source for a transaction. Domain match and attribution uses partial transaction information to try to identify the transaction source.
- **Org resolution.** The org resolution process associates a known partner identity with a transaction so it can be processed.
- **Error correction loop.** The error correction loop feeds the information from the first two processes back to the data providers whenever we can't resolve the identity source for a transaction. This way, they can ensure that their data output is corrected or normalized for future transactions.

Business rules

Business rules play a critical role in MS Sales functionality. Business rules define how data is represented in MS Sales. We took the opportunity to evaluate and optimize business rules management within MS Sales, and adopted these best practices:

- All rules can be updated in a prompt and inexpensive manner.
- Each rule is documented in a single, natural language statement (although it may be implemented using more than one programming language or declaration).
- Each natural language rule statement uses business terminology rather than database table or column names or program module names.
- Each natural language rule is unambiguous and succinct (uses no more words than necessary).
- All natural language rules are consistent in terms of vocabulary (terms used) and syntax (structure).

Incorporating these best practices into rules we build using Drools and JBoss makes it easier to check for rule conflicts, negative rule effects, and confirm data relationships within the ruleset taxonomy. Our rules are authored in Drools, published to our business rules Git repository, which is managed through Visual Studio Team Services (VSTS) until the publishing phase, and then Spark processes them.

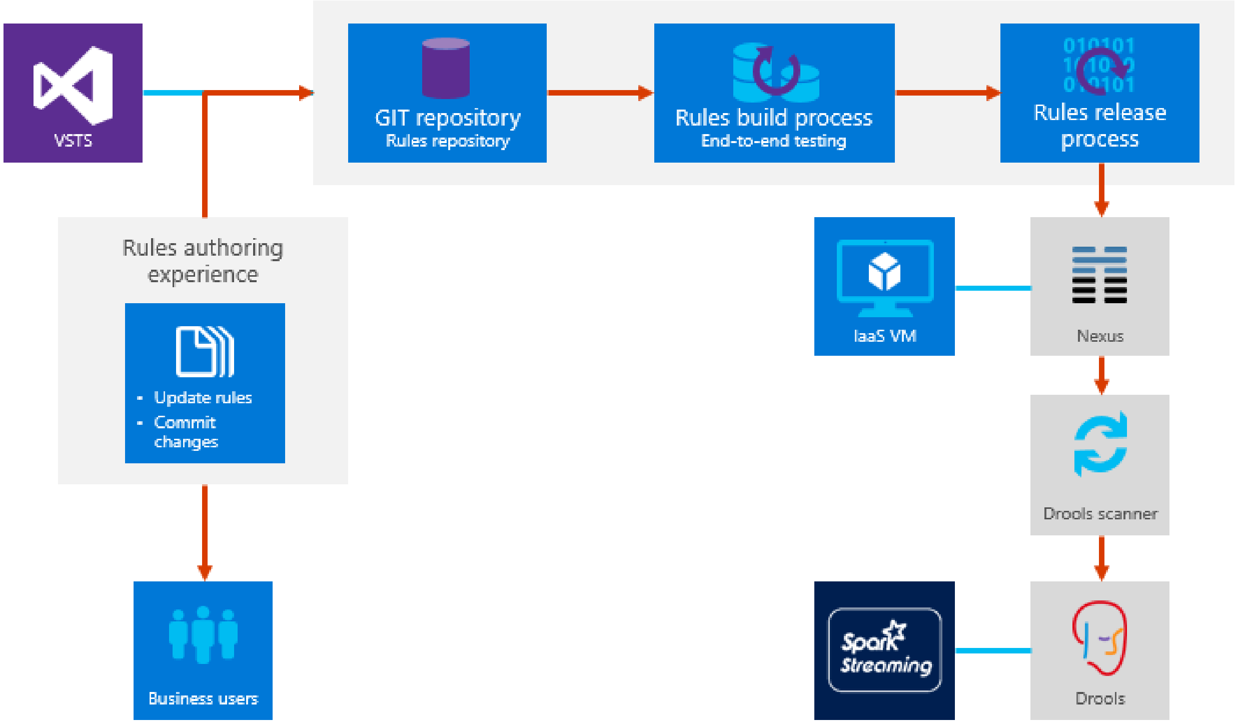

Figure 2. The business rules update flow

Using a declarative business rules approach

Our business rules are created using a declarative approach, which makes the rules implementation process more flexible and makes it easier to create or redefine rules. Our natural language rules are accessible to all our stakeholders, which makes them easy to review, approve, or change, and keeps our rules in a standard format that uses business terminology rather than database object names and obscure variables.

Pipeline processing

We use Apache Spark to drive our big data processing tasks in MS Sales, and a lot of our performance gains have come from converting our data pipeline.

Spark processing

Spark streaming processes as much MS Sales data as possible, instead of our old batch processing method.

Pipeline output

Our pipeline output is housed in Azure SQL databases. We use SQL Azure to replicate our data in multiple instances of Azure across several regions. This gives us redundancy and global distribution as well as load-balancing to improve performance. Our reporting tool, MSRA, is configured to access the right database instance.

MS Sales workarounds

We’re also using some workarounds in the original version of MS Sales, which allows us to focus on improving overall performance and scalability in MS Sales without having to specifically convert data storage mechanisms, which require more time to migrate to Azure. In Figure 3, red numbers indicate the shortcuts.

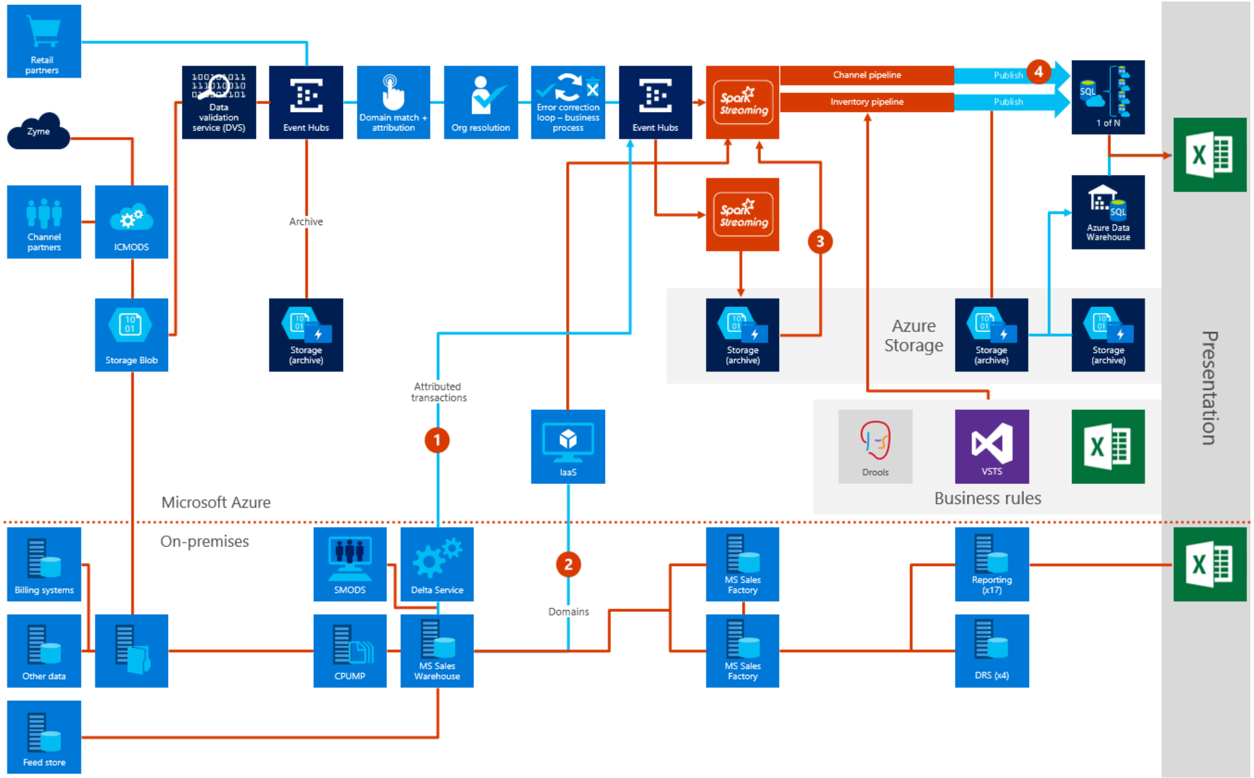

Figure 3. MS Sales architecture in Azure

Testing and release

Our testing and release process for MS Sales has three primary phases to ensure that all functionality works as expected before full release. Our three phases are:

1. **Testing and pre-production.** In testing and pre-production, we analyzed the primary functions of MS Sales in Azure and focused on any areas where significant change has occurred. We focused on:
    - Data ingestion patterns and use.
    - Testing comparison data between the on-premises and Azure versions.
    - Load testing and performance testing of the Azure solution to confirm initial estimates and look for optimization.
2. **Pilot phase.** The pilot phase includes a limited release to a group of users who are familiar with MS Sales and have a better-than-average understanding of its architecture and functionality. These users are helping us test real-life usage patterns and find any issues that arise from day-to-day use.
3. **Public release.** We are preparing for public release in July or August 2017.

Benefits and best practices

Although MS Sales is still migrating, we've already realized several benefits on the new platform. Many Microsoft business groups have adapted their business processes to the schedule and workflow in the original MS Sales version. With the new version, they're finding that the faster processing time enables them to re-examine their business processes and redefine them to fit business demands, rather than technical limitations. Here are the other benefits that we've experienced:

- **Reduced end-to-end latency.** End-to-end latency has been drastically reduced. We've moved from a 24-hour window to less than 45 minutes, which is an improvement of over 95 percent. As MS Sales continues to evolve in Azure, we expect that number to shrink even further.
- **Demonstrated scalability.** The Azure components we're using scale naturally and automatically to adapt to demand and volume. As a result, MS Sales can handle larger transaction volumes with a sub-linear correlation in end-to-end time. For example, we ran 1,000 percent more transactions through MS Sales for testing, and end-to-end processing went from 42 minutes to 52 minutes, an increase of only 24 percent.
- **Increased agility.** We have moved away from the monolithic nature of the earlier code base, and we incorporate modern engineering practices into development of the next version of MS Sales. The distinct components of the new MS Sales can be modified apart from the rest of the solution. We can get new features incorporated more quickly using continuous deployment and continuous integration.
- **Easy-to-deploy rules publishing.** By creating the rules management process as a discrete component of MS Sales rather than part of the monolithic whole, our rules management processes are much simpler. Rules changes can be made and deployed in production with zero downtime.
- **Easy-to-manage rules definitions.** Using natural language and declarative rules has opened the rules management process to all our business teams. New users can quickly get up to speed on rules management, and they no longer need the 18 to 24 months of training that was needed to work with the original solution.

For more information

Microsoft IT Showcase

microsoft.com/ITShowcase

© 2017 Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.