White Paper
Data Center
May 2023

IT@Intel: Minimizing Manufacturing Data Management Costs

Intel IT developed a custom benchmark to evaluate MPP databases and identify a cost-optimized solution that meets our technical requirements

Intel IT Authors

Erick Carvajal Gonzalez
Data Architect, Manufacturing IT

Miroslav Dzakovic
Principal Engineer, Manufacturing IT

Nghia Ngo
Cloud Software/Platform Architect, IT Data Platform

Ramesha Bhatta
Principal Engineer, Manufacturing IT

Table of Contents

Executive Summary 1
Business Challenge 2
Our Manufacturing Data Characteristics........................ 2
Data Complexity 2
Data Volume and Ingestion Velocity with Low Latency........... 2
Consumption Model 2
Our Unique RDBMS Evaluation Considerations........................ 3
Determining Cost Is Our Benchmark Goal 3
Throughput Is More Important Than Elapsed Time 3
Solution................................ 3
Completing the Request for Information (RFI) Process........... 3
Creating Our Custom Benchmark . . . 4
Evaluating Vendors' Products Using Our Custom Benchmark...... 6
Solution Architecture................ 8
Summary 8
Related Content........................ 9

Executive Summary

As Intel manufactures hundreds of millions of complex products every year, Intel IT collects and stores terabytes of manufacturing data to support continual engineering data analysis. As the volume, velocity and complexity of the data increases, it is imperative that we maintain this decision support system at the lowest possible cost. Additionally, we need to be able to assess the cost for future scaling needs. Therefore, we decided to evaluate the scalability, performance and cost of several Intel® architecture-based massively parallel processing (MPP) relational database management systems (RDBMS). We found that industry-standard benchmarks did not closely resemble our manufacturing data and did not measure the metrics that were important to us. Therefore, we created a custom MPP RDBMS benchmark that helped us choose a cost-optimized solution.

We used this custom benchmark to complete a comprehensive technical proof of concept (PoC) with several industry-leading MPP RDBMS vendors whose products run on Intel® architecture. We are confident that this benchmark enabled us to choose the best Intel® Xeon® processor-based MPP RDBMS solution while keeping manufacturing data management costs under control. Also, based on the evaluation results, the vendors we worked with have improved their products, strengthening the entire industry ecosystem. And, with the release of the 4th Gen Intel® Xeon® Scalable processors and associated accelerators, we're expecting that RDBMS vendors will make their products even more cost competitive. By sharing our benchmark methodology, we hope to help other companies to understand their data better and select a data management system that meets their needs.

Contributors

Joe Sartini, Global Domain Lead, Industry 4.0
Ram Varra, Sr. Principal Engineer, IT Infrastructure DevOps Architect

Acronyms

CART	Classification and Regression Tree
CP	critical path
CSI	critical success indicator
ER	entity resolution
ETL	extract-transform-load
MIDAS	Manufacturing Integrated Data Analysis System
MPP	massively parallel processing
PoC	proof of concept
RDBMS	relational database management system
RFI	request for information
SMP	symmetric multiprocessor

Business Challenge

Since its founding in 1968, Intel has been an integrated device manufacturer (IDM)—a company that both designs and builds its own semiconductor chips. With the announcement of "IDM 2.0,"[1] Intel is undergoing significant manufacturing expansion around the globe. In parallel, Intel's products are becoming increasingly complex (more transistors, more features, smaller footprint).

The manufacturing, testing and assembly of every instance of every product generates vast amounts of data. Intel IT has seen Intel's manufacturing data increase by more than 20x over the past decade. While a majority of this data can be stored in an open-source, NoSQL database like HBase, a key portion of it may be stored in a massively parallel processing (MPP) relational database management system (RDBMS) to create a decision-support system for engineering data analysis. (See the sidebars, "Why Not Just Use NoSQL?" and "MPP versus SMP" for a closer look at our custom-built Manufacturing Integrated Data Analysis System (MIDAS) and why an MPP RDBMS is required in addition to HBase.)

Our Manufacturing Data Characteristics

Our manufacturing data ingestion process has several characteristics that guide our choice of a cost-effective MPP RDBMS:

- Complexity (a dozen ID types in a record)
- High volume of existing data
- High velocity of incoming data
- Low latency expectations (integration must complete within five minutes)

In addition to these ingestion challenges, we also have a dynamic and mixed consumption model. When you combine all these characteristics, choosing the right MPP RDBMS can be challenging.

Data Complexity

Each chip can have one or more unique IDs of different ID types (for example, visual markers and lot/unit numbers). This is similar to how a person can have an SSN, a personal email address, or a personal cell phone number—each of which can uniquely identify that person. There are about a dozen ID types, some of which are applicable to only certain chips. Also, it is rare that a chip will have every type of applicable ID value. Whenever data is collected on a chip, typically only one or two ID types are present. Some are missing because data is not collected, is not available or is incorrect. Since data analytics engineers query the database using these IDs, an ingestion process called entity resolution (ER) must reconcile partial ID information into comprehensive and consistent ID data for each chip.[2]

Data Volume and Ingestion Velocity with Low Latency

Each year, Intel manufactures hundreds of millions of silicon chips that power Intel® processors. In some cases, one or more components of a chip are manufactured in non-Intel factories. For engineering analytical purposes, we gather and store data about every component and every chip—meaning that ingestion of manufacturing data occurs at extremely high volumes and high velocity. For ER, approximately 200,000 records must be integrated into RDBMS tables with billions of records within a five-minute period. As mentioned earlier, each record can have any of the dozen IDs populated, each of which must be checked individually against the existing billions of records in the database. The matching algorithm used for ER must identify and report any conflicting IDs. It must also use approximate matching logic to reconcile differences in values. (Shipping companies might use a similar algorithm to reconcile minor variations in street addresses, such as "1001 Evelyn Terrace" and "1001 EVELYN TER.") This complex ingestion process must finish within five minutes to keep up with the velocity of the incoming data, as it is part of the critical path (CP) of parallelized extract-transform-load (ETL) ingestion into MIDAS. On average, a total of 100 GB of data is loaded every fifteen minutes.

Consumption Model

Engineering data analysis at Intel occurs non-stop around the globe. The analytic process consumes both fresh data (such as from the most recent hour) and old data (such as one year old). We provide Intel's thousands of engineers with documentation about the data model so they can write the ad hoc SQL queries they need. Queries vary widely and some can take a few seconds to complete; others may take up to five minutes. This usage model, characterized by wide query variation, makes our SQL workloads unpredictable, dynamic and mixed.

[1] "Intel's 'IDM 2.0' Strategy Defined in 60 Seconds," https://www.intel.com/content/www/us/en/newsroom/news/idm2-strategy-defined-60-seconds.html

[2] Entity resolution is the process of working out whether multiple records are referencing the same real-world thing, such as a person, organization, address, phone number, bank account or device. [source: https://www.quantexa.com/entity-resolution]

Our Unique RDBMS Evaluation Considerations

With extreme data growth and significant data complexity, reducing manufacturing data management costs is challenging. Lowering data management costs is a critical factor in continuing to keep Intel's product pricing stable (as it has been for decades). The MPP RDBMS that we choose plays a crucial role in data management costs. Choosing the wrong platform could seriously compromise its business value and end up increasing costs, not lowering them.

However, comparing alternative RDBMS platforms is not straightforward. Standard benchmark tools' datasets do not closely resemble Intel's manufacturing data or its ingestion and consumption model. Therefore, they are not ideal for helping us choose the right solution. Also, we have some unique approaches to how we measure database performance, driven by our usage model.

Determining Cost Is Our Benchmark Goal

Typically, the word "benchmark" suggests some sort of performance metric, such as new orders per minute or I/O operations per second. However, our benchmark goal is primarily to estimate the cost of the solution, not the platform performance. In an MPP RDBMS, performance can theoretically always be improved by adding more nodes or clusters. Thus, performance is a function of the underlying platform capability and the amount of resources provided to it. To model the cost, we establish performance on a given hardware/software platform, and we measure the performance boost by increasing hardware resources. Once we have those two elements, we can then extrapolate the cost for our target solution.

Throughput Is More Important Than Elapsed Time

Elapsed time (that is, query response time) is typically how IT professionals gauge performance. We take a different approach: we assess database performance by measuring throughput—the number of queries completed in a given time frame—for various query categories. Although higher throughput and lower elapsed time are usually correlated, this is not always true in a time-bound benchmark. Figure 1 illustrates this concept by representing throughput within a specific workload category for a 10-minute period.

To illustrate the difference between these two metrics, consider five concurrent users who are each running multiple queries in a ten-minute test. In one scenario, only two queries finish, and their average elapsed time is about six minutes. In another scenario, five queries finish and their average elapsed time is about eight minutes. Clearly, the second scenario in which queries 1-3 also finish in less than ten minutes is preferred; focusing on the average elapsed time could lead us to choose the wrong platform, which is why the average elapsed time is only a secondary consideration for us.

Because standard benchmarks were not appropriate given our data characteristics, usage model and benchmark goals, we decided to create our own benchmark tool and dataset that would help ensure a reliable recommendation and lowest cost in time to meet a strategic decision deadline.

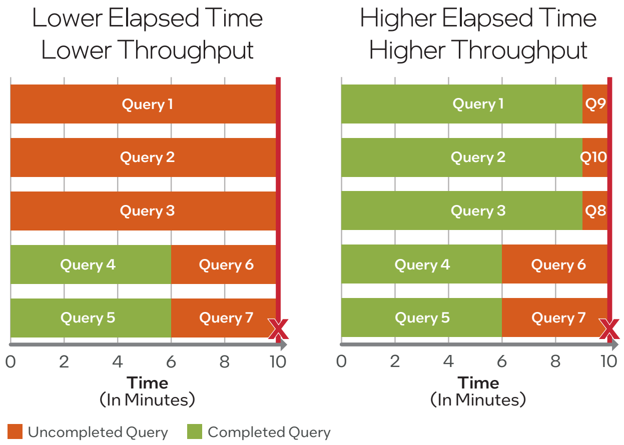


Figure 1. In a time-bound benchmark, we focus on throughput, not average elapsed time.

Solution

Creating our own benchmark tool and dataset was a multi-step process.

Completing the Request for Information (RFI) Process

We started with requirements gathering, a market scan and a paper study. This step took about three months, including a full-engagement RFI process with commercial off-the-shelf, industry-leading MPP RDBMS vendors whose products run on Intel® architecture. The following list provides a high-level view of our critical success indicators (CSIs):

- **Query capacity utilization.** A single query execution should be able to use all the available CPU capacity.
- **Linear scalability.** In general, MPP RDBMS scale by adding more nodes, rather than upgrading nodes. This ability to scale is critical to meet our business's growing data and usage characteristics.
- **Robust workload management.** Dynamic allocation of resources, prioritization, quotas, and throttling should all be available.
- **Flexibility.** The platform should support high concurrency and a mix of simple, medium and complex queries or transactions, as well as high-volume, high-velocity, low-latency ingestion.
- **Ingestion throughput.** Frequent mini-batch SQL DML and ER operations are critical to meet our throughput needs.
- **Table maintenance.** Table maintenance must be easy and fast. We cannot afford any downtime during table and database maintenance operations such as creating indexes or gathering statistics.
- **Security and business continuity.** Security and disaster recovery functionalities are essential for Intel's high-volume manufacturing environment, where downtime can cost millions per hour.

Using our requirements list, we identified multiple commercial off-the-shelf, industry-leading MPP RDBMS vendors based on our assessment of their responses to the RFI. Next, we embarked on a two-part effort.

- **Created a custom benchmark** using a representative subset of data, obfuscated to protect intellectual property, and bloated it to the desired dataset size. The custom benchmark emulates the ingestion process as well as the mixed workload of the consumption through SQL. This process took about six months.
- **Conducted a complex and comprehensive PoC** with standard RFI and Technical Review Committee approval with multiple MPP RDBMS vendors and studied each product's capability, scalability, workload throughput and performance to see if it would work for our manufacturing data. This process took about five months.

The following sections provide more detail on these efforts.

Creating Our Custom Benchmark

Our comprehensive PoC benchmark (see Figure 2) emulates our typical data management workload—from the data itself to the query types.

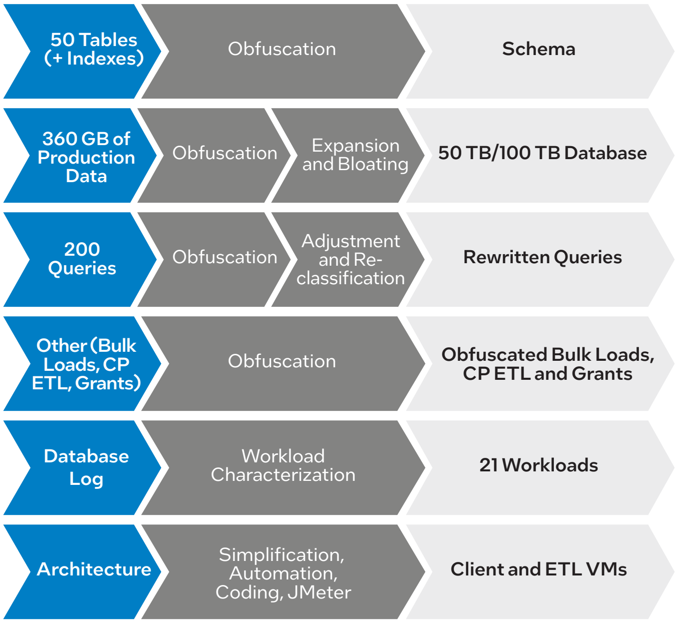

50 Tables (+ Indexes)
Obfuscation
Schema
360 GB of Production Data
Obfuscation
Expansion and Bloating
50 TB/100 TB Database
200 Queries
Obfuscation
Adjustment and Re-classification
Rewritten Queries
Other (Bulk Loads, CP ETL, Grants)
Obfuscation
Obfuscated Bulk Loads, CP ETL and Grants
Database Log
Workload Characterization
21 Workloads
Architecture
Simplification, Automation, Coding, JMeter
Client and ETL VMs

Figure 2. An illustration of the creation process of our custom benchmark for MPP RDBMS evaluation.

The following sections provide more detail on these aspects of the benchmark:

- Creating, obfuscating and bloating the dataset and queries
- Building the workload
- Workload characterization using *k*-means clustering and Classification and Regression Trees (CART)

Creating, Obfuscating and Bloating the Dataset

To provide useful MPP RDBMS recommendations, our benchmark dataset needed to closely resemble our production data. To achieve this goal, we chose 360 GB of seed data for a random sample of material extracted from 50 of the core MIDAS tables. We then expanded it with redundant and mocked-up data to 2.3 TB of seed data.

MPP versus SMP

Data usage models dictate system architecture.

Parallel processing relational database management systems (RDBMS) can be either symmetric multiprocessor (SMP) systems or massively parallel processing (MPP) systems. At a high-level, the difference is in system design. Processors in SMP systems share access to memory and storage drives. In an MPP system, each processor has its own allocated, fixed portion of the storage drives and memory. Most importantly, execution of a single query in an MPP system involves many threads that each work on its own chunk of data in parallel and can therefore use all of the system resources. By simply adding hardware capacity to an MPP system, almost any query can run faster.

That is also the primary reason that our Manufacturing Integrated Data Analysis System (MIDAS) uses an MPP RDBMS. Scalability is one of our top priorities to meet service-level agreements and concurrency/query performance expectations. Currently, the amount of data in the MPP RDBMS is about 100 TB, but we expect the data to continue to grow. Plus, our data consumption model is highly diverse. Manufacturing engineers write many ad hoc queries, ranging from very simple to very complex. An MPP RDBMS provides excellent scalability and the ability meet performance expectations for both simple and complex queries. Although adding more nodes increases the cost, it allows us to meet any future increase in demand. Another reason for focusing on MPP systems is that MIDAS already uses an MPP RDBMS. Switching to an SMP RDBMS would require substantial rearchitecting effort and could introduce new risk.

Although we focused on MPP RDBMS evaluation for these reasons, we are continuing to evaluate possibilities of using an SMP RDBMS in the future.

Intel's manufacturing data is one of the company's most valuable assets. Therefore, in order to share the dataset with MPP RDBMS vendors, we obfuscated table definitions and their data. We wrote an algorithm that replaced table and column names and data values with a random phrase; we also scrambled numbers. We modified the timestamp values only in the minutes-seconds portion of the data so that the queries on date ranges would return the same number of rows. The synthesized and obfuscated data preserves all PK/UK/FK[3] constraints, as well as data distribution properties.

Next, we bloated the dataset to match the volume of data in our production environment. We created a 50 TB and a 100 TB version of the dataset, varying a substring in key columns (for example, lot AAAA000, AAAA001, AAAA002 and so on).

Obfuscating Queries

Besides obfuscating data and table/column names, we also needed to obfuscate the queries. We used the same obfuscation algorithm, which replaced all words and numbers in a query—except the SQL keywords—with random phrases that matched the data and schema obfuscation. Additionally, we generated multiple versions of the same query by varying the same substring in literal values for the key column (for example, lot AAAA000, AAAA001, AAAA002 and so on). This allowed us to expand our benchmark query set from a little over 200 queries to more than 10,000.

Building the Workload

To ensure the benchmark represented a typical production consumption workload, we identified top queries to include in the benchmark based on the following characteristics:

- Usage model (ETL or Select)
- CPU usage
- I/O usage

Queries were subdivided into query buckets (such as very simple, simple, medium, complex, very complex and super complex; see the next section for explanation of how we categorized the buckets). We also chose a random sample of 20 queries per query bucket. This process culminated in a little over 200 rewritten queries that worked with all the key tables in the database. A similar process was done for ETL statements. Some of the ETL statements also created locking conflicts, reflecting the need for the MPP RDBMS to manage concurrent ETL operations on the same tables.

With queries and ETL statements defined, we built the benchmark workload, which consisted of four main elements:

- Downstream application and other user queries (178 queries)
- ETL statements (including massive deletes) and typical bulk-loads (50 concurrent ETL loadings)
- CP ETL, to be finished in five minutes (124 mixed-complexity queries)
- SQL GRANT statements maintaining user roles and privileges (2,000 queries)

The completed benchmark consisted of a total of 21 workloads, each 15 minutes long, and each varying in the number of concurrent sessions per bucket. Of the 21 workloads, one had an average number of sessions for each bucket; the other twenty had varying peaks and valleys in the number of sessions for each bucket.

Workload Characterization Using Machine Learning

We used modern machine learning techniques to help characterize our production workload and build a benchmark that is as real-world as possible. Specifically, we used *k*-means clustering and CART, which is a simple but powerful method to determine cluster characteristics. While the details of this process are beyond the scope of this paper, the high-level steps are outlined in the following list. The characterization process was performed separately for user queries and for ETL statements, and it focused on the CPU time, I/O and running time, as well as other data (such as skew).

1. We performed *k*-means clustering to separate all queries into a desired number of query buckets. Initially, we decided to use four query buckets, named 1, 2, 3, 4. Each query was assigned, using a cluster ID, to one of these query buckets.
2. For each cluster ID, we created an indicator variable (1/0), which is set to 1 if a query belongs to that query bucket.
3. We used these indicator variables (four, initially) as response/output variables, and ran a simple CART algorithm (see Figure 3 on the following page) to predict each indicator variable, based on all input characteristics (CPU time, I/O, etc.). The goal was to establish a criterion that closely describes each query bucket. For example, based on the decision tree output from the CART algorithm, "OverallCPUTime > 771.75" was a good fit for the indicator variable in query bucket #3 (see the left side of Figure 3). For some query buckets, we had to use multiple criteria, such as "OverallCPUTime <= 789.3 and runningSec > 37.44" for query bucket #4 (see the right side of Figure 3).
4. If we couldn't easily identify criteria for a query bucket, we repeated the process from step 1 with a different number of target query buckets or sub-divided a single query bucket using steps 1-3 to create new indicator variables.

[3] PK/UK/FK = primary key/unique key/foreign key.

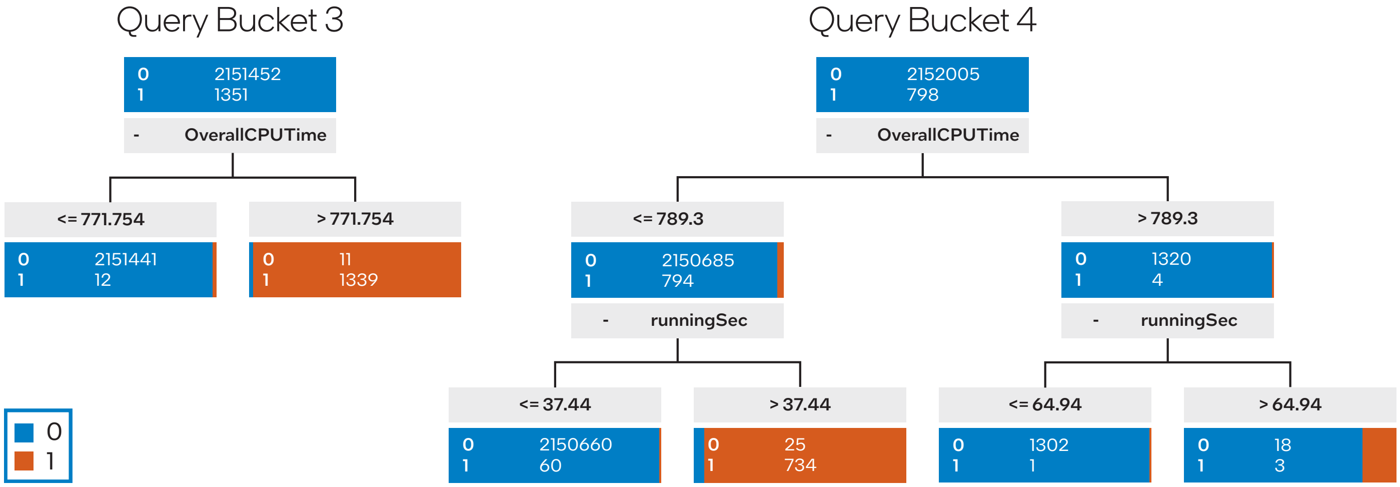


Figure 3. Examples of how we used Classification and Regression Trees (CARTs) to characterize the workloads.

Our analysis of production ETL operations and SQL queries revealed seven ETL buckets (based on CPU time, highest thread I/O, and running time) and six SQL query buckets (based on CPU time).

Next, we used historical query metadata to identify workload signatures. To start, we counted the total executions in each query and ETL bucket per hour. We then ran a clustering algorithm to cluster different one-hour slots and identified a few peak hours in each cluster as representative hours of that cluster. For each of the cluster representative hours, we divided the hour into 10-minute slots, which helped us to determine the average number of concurrent sessions for each query and ETL bucket. This resulted in workload signatures where each signature indicates how many concurrent sessions are running statements in each bucket.

Finally, we computed the expected number of executions in each bucket by considering several months' worth of production utilization metadata. We counted how many queries were executed in each bucket during that large time period and then averaged it down to a 15-minute interval. For example, we expected to finish 6,000 very simple queries in 15 minutes, but only nine very complex queries. We validated these targets by considering the concurrency for each bucket and the typical running time of each query.

Evaluating Vendors' Products Using Our Custom Benchmark

We worked with commercial off-the-shelf, industry-leading MPP RDBMS vendors identified by the RFI process to set up the PoC environments, bloated datasets and converted queries. As each vendor completed our custom benchmark, we assessed the vendors against the expected requirements and CSIs, and estimated the cost of each tested product. We documented the key findings and gaps, sharing feedback with vendors for potential future product enhancements. The following sections describe the test approach and our performance and cost calculations in more detail.

Why Not Just Use NoSQL?

Not all data is stored and accessed in the same manner. NoSQL is not cost-efficient for some data usage models. Additionally, NoSQL does not support transactions that modify multiple tables.

Our custom-built Manufacturing Integrated Data Analysis System (MIDAS) architecture cost-effectively supports both large and small tables.

Large tables. Some manufacturing data is stored in large tables with a single access path and don't require multi-table transactions. To reduce the cost of managing that data, we recently expanded the MIDAS architecture and migrated these large tables to an HBase cluster. Using the open-source NoSQL HBase removed more than 90% of data from the massively parallel processing relational database management system (MPP RDBMS).

Small tables. We also have data that is stored in small tables with multiple access paths. This data must reside in an MPP RDBMS that supports transactions and updates. NoSQL is not a good fit for this more complex data environment. At best, HBase supports fast access by a primary index (row key) but must scan full tables for other queries. This makes it prohibitively expensive for joins that do not use the primary index. In addition, by default, HBase does not support joins.

Test Approach

We originally planned for a six-week PoC. However, during query optimization we encountered an issue with meeting a key CSI, which caused us to extend the PoC to ten weeks for each vendor. Each vendor evaluation was an intensive process of preparing and porting the code for the benchmark (resolving both syntactic and semantic differences), then running the tests multiple times to tune the performance. We uploaded our custom dataset to each vendor's cloud subscription bucket, who then ran the workload in the cloud. All vendors used the same cloud service provider, and all cloud instances were based on Intel® architecture using Intel® Xeon® processors (see the sidebar, "New Generation of Intel® Xeon® Scalable Processor for MPP RDBMS Workloads," for additional information). In some cases, vendors made hot fixes to their product (deployed only to the PoC environment) to improve performance. Some vendors also changed the physical schema of our PoC data, such as index or table organization.

The benchmark consisted of the following tests (see Figure 4):

- Run the benchmark workload on the initially defined hardware. Each vendor ran the workload multiple times to fine-tune the performance so that they could meet the target throughput threshold. The workload manager allocates resources so that the maximum percentage of the workload finishes within a given time. More specifically, since the workload consists of many buckets, each of which can have a different percentage of completion, the goal is to increase the smallest completed percentage as much as possible (see Figure 5).
- Test scalability (scale-out) by doubling the hardware while keeping the workload and dataset the same.
- Double the dataset to compare different vendors when both data and capacity grows.

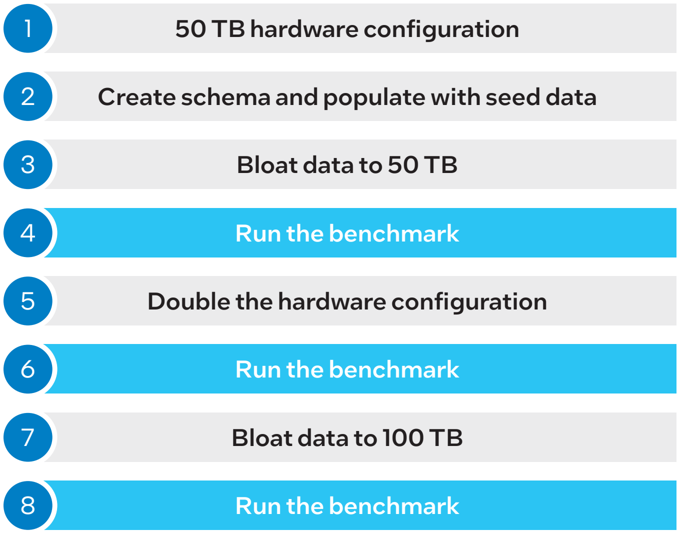

1
50 TB hardware configuration
2
Create schema and populate with seed data
3
Bloat data to 50 TB
4
Run the benchmark
5
Double the hardware configuration
6
Run the benchmark
7
Bloat data to 100 TB
8
Run the benchmark

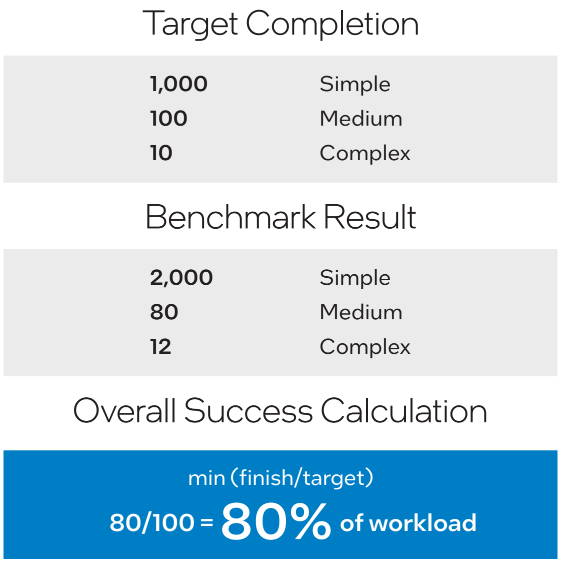
Figure 4. Benchmark process.

We also assessed the ability to actively manage the workload to protect the CP ETL work. A robust workload management tool is a necessary capability of an MPP RDBMS so that it can meet the mixed workload demands of our manufacturing data environment. Other elements that we assessed included the impact of deletes/updates, latency, throughput and the ability to meet the unique needs of simple/medium/complex queries. All this data is necessary to calculate the price/performance ratio, which was our main focus.

Calculating Performance and Cost

We defined the following performance metrics:

- **Consumption.** For SQL queries, the platform must complete a certain number of queries in a bucket (of a certain complexity) within 15 minutes. For example, it must complete 1,000 simple queries and 10 very complex queries within the 15-minute period. A secondary measure is the average response time for a query bucket (such as simple or complex). Figure 5 shows how we calculate consumption performance.
- **Ingestion.** For CP ETL, we must be able to process 200,000 incoming identities and integrate the data within five minutes. We also measured completion percentage for each ETL bucket together with the consumption buckets in the way described.



Target Completion

1,000	Simple
100	Medium
10	Complex

Benchmark Result

2,000	Simple
80	Medium
12	Complex

Overall Success Calculation

min (finish/target)

80/100 = **80%** of workload

Figure 5. Consumption performance calculation example.

The calculation for a product's total cost of ownership included the following components:

- Hardware (such as servers, network, data center, power and racks or, alternatively, the comparative equivalent cost of using the cloud)
- Software (license, support, consultation and training)
- Head count (system engineer, database administrator and others)
- One-time migration cost from existing solution to new solution

We estimated the hardware configuration and the corresponding software cost based directly on the benchmark results. If, for example, a vendor satisfied 50% of the target throughput with a configuration X, we estimated we would need twice that configuration to satisfy 100% of the target. That projection is based on perfect linear scalability; our benchmark also included scalability testing, which could influence the estimate.

Solution Architecture

The architecture for our custom benchmark (see Figure 6) is simple, consisting of the database under test, a Linux client virtual machine running bulk-load jobs and a Windows client virtual machine running Apache JMeter.[4] We wrote a custom Perl script to throttle bulk-load jobs on the Linux client, and a custom Python code to exercise array inserts on the Windows client.

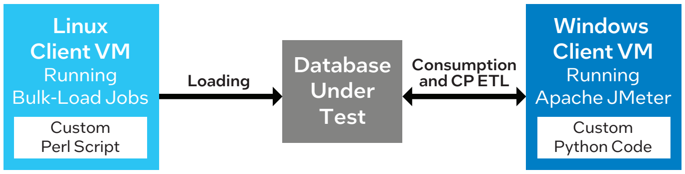


Figure 6. MPP RDBMS custom benchmark architecture.

Custom Benchmark Well Received by the Ecosystem

Even vendors that couldn't meet the needs praised the high quality of the benchmark itself.

The quality of work and transparency of our benchmarking efforts resonated beyond executive leaders in IT. Both Intel leaders and the vendors themselves praised the benchmark. Even vendors who failed on one or another critical success indicator (CSI) did not complain that the benchmark and PoC were poorly designed. Some vendors are driving significant improvements in their product based on the PoC. This sort of feedback confirms the value of our benchmark and PoC.

"The project team served as an exemplary model for the way we would like to conduct every vendor evaluation. They demonstrated exceptional expertise in managing a highly technical evaluation, utilizing data-driven insights to draw significant conclusions and make recommendations for the future. In addition, the team shared their findings with each vendor, enabling them to improve their products over time."

— Jeffrey Walsh, VP, Manufacturing IT, Intel

New Generation of Intel® Xeon® Scalable Processor for MPP RDBMS Workloads

Intel® In-Memory Analytics Accelerator Architecture helps improve performance.

4th Gen Intel® Xeon® Scalable processors have a new hardware accelerator that can boost massively parallel processing relational database management system (MPP RDBMS) workload performance. The accelerator logically contains three main functional blocks: Compression, Encryption and Analytics. The Analytics pipe contains three sub-blocks: Decrypt, Decompress and Filter. These functions are tied together so each analytics operation can perform any combination of decrypt/decompress/filter (for example, decrypt-filter). Alternatively, one can compress or encrypt the input. Compression and encryption cannot be linked with any other operations.

The accelerator allows columnar databases to be stored in compressed form, decreasing memory footprint. In addition to increased effective memory capacity, this also reduces memory bandwidth by performing the filter function used for database queries "on the fly," thereby avoiding the use of memory bandwidth for uncompressed raw data transfer. Read "Intel® In-Memory Analytics Accelerator Architecture Specification" for more information.

Summary

We successfully conducted a thorough and highly technical PoC and learned much about different products' capabilities, scalability, workload management and cost. Besides using the PoC findings to minimize the present cost, we will use that information in future to identify the best cost-efficient choice as our needs grow and evolve. Additionally, the MPP RDBMS product evaluations revealed that some vendors' products lacked capabilities that we needed. In response, vendors have begun to make improvements in their products based on their learnings from the benchmark. This will help Intel in the long term by influencing the industry to deliver better solutions that meet our needs.

We believe the outcome of our benchmarking efforts has resulted in a better understanding of the best Intel architecture-based MPP RDBMS solution that meets our technical requirements while helping keep manufacturing data management costs low. We hope that sharing our experience in creating a custom benchmark can help other companies better evaluate their data management needs.

[4] Apache JMeter is an open-source solution using Java to load test functional behavior and measure performance.

Related Content

If you liked this paper, you may also be interested in these related stories:

- Optimizing Factory Performance with Digital Twin Technology white paper
- Transforming the Factory Floor with Software-Defined Networking white paper
- Accelerated Analytics Solution Drives Breakthroughs in Factory Equipment Availability white paper

For more information on Intel IT best practices, visit intel.com/IT.

IT@Intel

We connect IT professionals with their IT peers inside Intel. Our IT department solves some of today's most demanding and complex technology issues, and we want to share these lessons directly with our fellow IT professionals in an open peer-to-peer forum.

Our goal is simple: improve efficiency throughout the organization and enhance the business value of IT investments.

Follow us and join the conversation on **Twitter** or **LinkedIn**. Visit us today at **intel.com/IT** if you would like to learn more.