WINTER CORPORATION

WINTERCORP REPORT

SPONSORED RESEARCH PROGRAM

The InterSystems Speed Test

A Comparison of Performance and Data Latency in Operational Cloud Database Systems

EXPERTS IN DATA MANAGEMENT AT SCALE

The InterSystems Speed Test

A Comparison of Performance and Data Latency in Operational Cloud Database Systems

RICHARD WINTER
NORBERT KREMER
ATIF MAJID
TENNYSON YESUPATHAM

WinterCorp
www.wintercorp.com
42 DERBY LANE, TYNGSBORO, MA
617-695-1800

CONCLUSION: ON THE BASIS OF ITS INDEPENDENT RESEARCH, WinterCorp recommends that companies with needs for low latency, high performance transactional-analytic data management software take a close look at InterSystems IRIS, which compares favorably to all alternatives tested on AWS on a single node and in 1- to 4-node clusters. Compared to alternatives, InterSystems IRIS exhibits significant advantages in query throughput, insert throughput, data latency and query efficiency without special tuning or configuration efforts. •

Summary

AN IMPORTANT CLASS OF DATABASE APPLICATIONS must satisfy multiple critical performance requirements concurrently, including:

- High Volumes of Transactional Processing and Data Ingestion;
- High Volumes of Queries; and,
- High Consistency, including retrieving records immediately after insertion with very low latency.

For example, there are databases used to monitor, and rapidly respond to price changes in publicly traded securities, where there are billions of trades per day. These companies want to monitor their portfolios and market data in order to compute their exposure to risk and decide what to buy or sell and how much. If they can make these decisions before other traders, they will have a powerful competitive advantage that will make a big impact on their business. The new transactions must often be visible in the database within milliseconds of their occurrence in the public markets. Many other database applications have similar, near real time requirements in industrial, commercial and engineering applications.

In these applications, transactions and record structures are relatively simple, but there is nothing simple about satisfying the demanding requirements involved, in which high throughput and low data latency must be delivered consistently throughout the day.

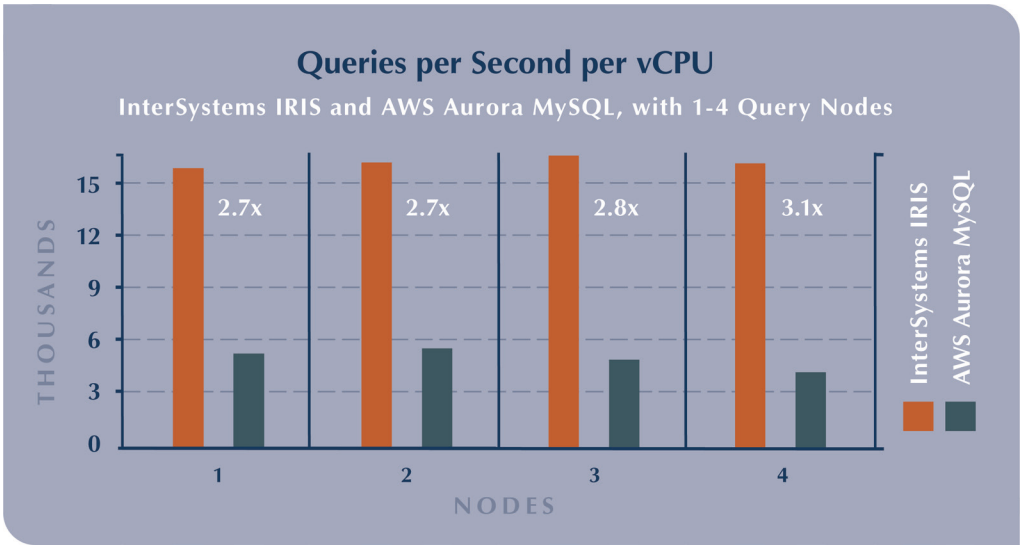

Figure 1: Queries per second per vCPU, InterSystems IRIS and AWS Aurora MySQL, with 1–4 query nodes

Wanting to demonstrate capabilities of this sort to a customer, InterSystems, a data management software provider, devised the InterSystems Speed Test, a straightforward way to measure the performance of data management software for these criteria.

This report covers our experience running the InterSystems Speed Test in single node and clustered configurations; presents our findings; and summarizes our conclusions.

Our testing resulted in five principal findings:

1. **The InterSystems Speed Test is a valid measurement of a key building block of many operational database applications.**
2. **Our independent, single node test results demonstrated a large advantage in performance for InterSystems IRIS when compared to AWS Aurora MySQL, MariaDB, Microsoft SQL Server, Oracle and PostgreSQL.**
3. **The insert rate for InterSystems IRIS was between 1.7 times and 9 times faster than for the other systems. The data query rate for InterSystems IRIS was between 1.1 times and 600 times faster than for the other systems.** ***No other system came close to matching InterSystems IRIS in the combination of insert rate and query rate*****.**
4. We completed cluster testing comparing one of these systems, AWS Aurora MySQL, to InterSystems IRIS. In those tests, we chose a configuration in which AWS Aurora MySQL could roughly match the insert rate of InterSystems IRIS: this required that AWS Aurora MySQL run on servers with four times the processor power and four times the memory that IRIS had. We then compared query rate per virtual CPU (a measure of compute power on AWS). By this measure (see *Figure 1,* on page 3), ***InterSystems IRIS was between 2.7 and 3.1 times more efficient than AWS Aurora MySQL and the efficiency advantage grew larger as the number of nodes in the cluster increased.*** That is, we saw linear scaling with InterSystems IRIS and less-than-linear scaling with AWS Aurora MySQL.
5. **InterSystems IRIS demonstrated low data latency throughout the three- to five-minute clustered tests we ran.** That is, records inserted by InterSystems IRIS showed up in queries very quickly after insertion. However, as shown in *Figure 2*, there was a substantial delay between insertion of a record by AWS Aurora MySQL and its retrieval in subsequent queries. This delay persisted throughout a seven-minute test.

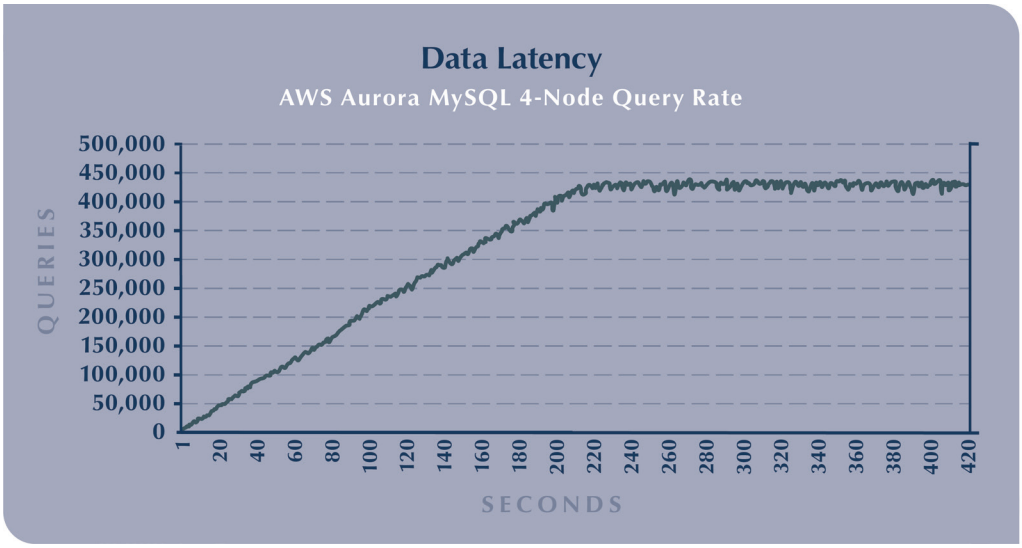

Figure 2: Persistent data latency was apparent throughout our seven-minute test.

Low data latency is essential in all near real time database applications. For example, a system monitoring price changes in securities must react rapidly when a trade is made. If there is a substantial delay before newly inserted data can be retrieved, large losses may result.

6. Having AWS Aurora MySQL successfully execute the test in a 5-node cluster configuration was a challenge overcome only by artificially increasing the number of test clients and test threads to unusually large numbers, suggesting that **business solutions with requirements similar to those that we were testing would require a complicated application architecture with** AWS Aurora MySQL. When our earlier attempts to get AWS Aurora MySQL to run the test with more than two reader nodes failed, AWS support was unable to help us find a solution. By contrast, **InterSystems IRIS worked immediately in configurations of one- to five-nodes**.

1 High Performance Operational Database Challenges

Databases are the beating heart of virtually every modern enterprise. While some databases exist only to support analytics and decision making, many others actually record, control and manage the moment-to-moment operation of the business. These operational databases often face the most demanding requirements for performance: they must respond instantly to queries; they must incorporate new data as it arrives via a veritable firehose; and they must always return the latest information, often within milliseconds of its receipt. Required insert, query and update volumes can be very high.

There are many examples of such requirements in manufacturing, distribution, e-commerce, facilities management and transportation. But the most widely understood example is in financial services, where many companies manage a portfolio of securities, the value of which can change in a fraction of a second. Often a price change will trigger programmed trading to balance a portfolio of securities or maintain its relationship to a published index or other criteria. Such databases must incorporate new transactions immediately and respond within milliseconds to queries that rely on having current information.

Customers with requirements like these need a way of validating which database product best meets their needs. The InterSystems Speed Test was designed for this purpose.

2 The InterSystems Speed Test

In the InterSystems Speed Test, the database under test is fed a steady stream of new records to be stored. At the same time, a steady stream of queries for these records is fed to the system. Each new record has a unique key, which can be used to retrieve it.

The queries and insertions cycled through one hundred thousand values of these keys. The test harness counted the number of completed queries that successfully retrieved a record. By default, the test runs for five minutes and produces a report showing:

1. the number of records inserted;
2. the number of queries that retrieved a record successfully; and,
3. a graph of these values, at the end of each second, over the period of the test.

The test harness includes two types of "user" processes: writers submit inserts to the database and readers submit queries and process the responses. Both writers and readers can have multiple "threads." Each thread simply repeats its task: a writer thread will insert a record into the database; wait for a response; then immediately insert another record. The insert rate and query rate can be increased incrementally until the database under test is "saturated" that is, until it is doing all the work it can handle. These rates are first increased by adding threads, but at some point, the server on which a client process is running exhausts its resources. At that point, the rate can be further increased by adding clients, also called workers.

Thus, under some wide range of conditions, the workload applied to a database can be increased to extremely high levels — until the database under test is doing all the work it can perform.

For a given period of time, *Item 1* above — the number of records inserted — is a measure of database ingest throughput. *Item 2* above — the number of queries that retrieved a record successfully — is an measure of query throughput and latency. *Item 3* above — the graph produced by the test program — shows how throughput and latency progress over the test period. These items are discussed in some detail and illustrated in the subsequent sections, as we discuss the results of each test.

3 WinterCorp Test Process

WinterCorp performed its testing in the Amazon cloud. We employed two senior database engineers with cloud experience for the testing. Both of the engineers performing our hands-on testing have years of experience implementing on, and conducting performance tests with, a variety of commercial database products. Both engineers have in depth experience with database implementation on AWS.

This testing activity was overseen by two senior WinterCorp architects, both of whom have in depth experience in database performance and one of whom is a certified cloud architect, implementor and trainer.

WinterCorp, an independent technology consultant, has been involved in database performance testing, database architecture, database implementation and database performance and scalability analysis for over 20 years. We have been retained by leading enterprises — users, vendors and system integrators in over 50 separate engagements for this purpose.

Our approach was to use the test harness developed by InterSystems, but to configure and deploy it independently according to our views of what would make the most reasonable test of InterSystems IRIS and the products to which we were comparing it. Our work proceeded in two phases.

In the first phase, we replicated the tests InterSystems had performed on single server (also called single node) configurations of the databases under test. The purpose of this phase was twofold: first, we wanted to gain hands-on experience with the test harness, test process and outputs; second, we wanted to produce independent, validated results for the simplest form of the test. In this first phase, we tested InterSystems IRIS, AWS Aurora MySQL, Microsoft SQL Server, MariaDB, Oracle and PostgreSQL.

In the second phase, we tested clustered configurations of InterSystems IRIS and AWS Aurora MySQL with one writer node and up to four reader nodes.

In all cases we used default setups for the database products under test. We did not involve the product vendors except to access support if we ran into difficulty getting the test to run. When we did use support, we did not ask the vendor for performance advice — we only sought help with getting the test to run.

Below we provide some additional information on specific tests and test issues.

4 Single-Node Test Results

For the single node tests, we used the default hardware configuration as created by InterSystems. The test harness ran on AWS *c5.xlarge* instances[1]. The database products under test ran on an AWS *m5.xlarge* instance[2] or the closest analog available for the product.

For InterSystems IRIS, we used one query worker and one ingestion worker. The ingestion worker was configured with 15 threads and the query worker was configured with 10 threads. The key count was 8,

[1]Amazon AWS *c5.xlarge* instance consists of 4 vCPUs, 8 GiB RAM, up to 4.75 Gbps EBS bandwidth and up to 10 Gbps network bandwidth.

[2]Amazon AWS *m5.xlarge* instance consists of 4 vCPUs, 16 GiB RAM, up to 4.75 Gbps EBS bandwidth and up to 10 Gbps network bandwidth.

meaning that the inserts cycled through 8 distinct key values and all of the queries were of those 8 keys. The ingestion batch size was 1000 records. The test was run for 20 minutes.

With InterSystems IRIS, AWS Aurora MySQL and Microsoft SQL Server running on the similar configurations, the results were:

PRODUCT	INSERTS (PER SECOND)	COMPLETED QUERIES (PER SECOND)	AWS SERVER TYPE
AWS Aurora MySQL	12,985	15,640	db.r5.xlarge
MS SQL Server	15,569	3	db.m5.xlarge
InterSystems IRIS	128,322	19,379	m5.xlarge

Table 1: Single-node tests of AWS Aurora MySQL, SQLServer and InterSystems IRIS

The insert rate for InterSystems IRIS was almost 10 times the insert rate for AWS Aurora MySQL and over 8 times the insert rate for MS SQL Server. The completed query rate for InterSystems IRIS was about 24% faster than for AWS Aurora MySQL and over 6000 times faster than for SQL Server.

The other three systems we tested required a larger server to run, using the AWS marketplace. So, for these systems we ran InterSystems IRIS on a *m5.2xlarge* server, the most comparable choice. The results for these tests are shown in *Table 2*:

PRODUCT	INSERTS (PER SECOND)	COMPLETED QUERIES (PER SECOND)	AWS SERVER TYPE
Maria DB	32,908	11,362	db.m5.2xlarge
Oracle	50,360	42,378	db.r5.xlarge
PostgreSQL	110,299	12,872	db.m5.2xlarge
InterSystems IRIS	193,013	47,537	m5.2xlarge

Table 2: Single-node tests of MariaDB, Oracle, PostgreSQL and InterSystems IRIS

So, in these tests, the insert rate for InterSystems IRIS was between 1.7 times (PostgreSQL) and 5.9 faster (MariaDB) faster than the other systems. The query rate for InterSystems IRIS was between 1.1 times (Oracle) and 4.2 times (MariaDB) faster than the other systems.

In none of the tests was another system able to come close to matching InterSystems IRIS on both insert rate and query rate, illustrating how InterSystems IRIS is uniquely suited to applications which require high performance on both measures. Clearly, the securities price/portfolio monitoring application requires both, as do many other transactional-analytic database applications with high volume, near real time requirements. •

5 Cluster Test Process and Results

While the single node tests demonstrate a large advantage for InterSystems IRIS, we recognized that some operational database products can also be run in a clustered configuration, which should increase throughput.

We therefore proposed to InterSystems that the testing be extended to test clustered configurations; InterSystems agreed.

We chose AWS Aurora MySQL as the first system to compare to InterSystems IRIS in this way, thinking that it would be the simplest because it is native to AWS, where we were performing our testing.

We first ran a series of cluster tests with InterSystems IRIS to establish a baseline.

NODES	INSERTS (PER SECOND)	COMPLETED QUERIES (PER SECOND)	COMPLETED QUERIES PER SECOND, PER NODE
1	99,515	60,189	60,189
2	100,356	123,819	61,910
3	94,076	186,387	62,129
4	101,876	248,910	62,228

Table 3: Clustered test of InterSystems IRIS, 1–4 query nodes

For these tests, we set up InterSystems IRIS with one node processing all the inserts and 1– 4 nodes processing queries. The insert rate for all four tests was in the range of 100,000 per second (as shown in the table, it ranged between roughly 94k and 102k per second). However, as we increased the number of nodes processing queries, the query rate increased slightly faster than the number of nodes. Thus, InterSystems IRIS generally sustained its high insert rate while exhibiting better than linear scaling in query rate, a nearly ideal pattern.

AWS Aurora MySQL uses "replicas" to implement its closest analog to a cluster architecture. We set up *Aurora* with one node for insert and 1– 4 replica nodes for query.

We decided to test the way we thought a customer would, if the customer needed to support a given rate of inserts. So, we progressively increased the size of the AWS Aurora MySQL node until we could roughly match the insert rate for InterSystems IRIS. We reached this point with AWS Aurora MySQL running on *db.r5.4xlarge* nodes, each of which has 16 virtual CPUs and 64 GiB of RAM. (Compare this to the *m5.xlarge* nodes InterSystems IRIS was running on, which have 4 virtual CPUs and 16 GiB of RAM; thus **the AWS Aurora MySQLnodes had roughly *four times* the compute and memory resources of the InterSystems IRIS nodes**). With these larger nodes, we got the AWS Aurora MySQL insert rate up to about 94k/second — within 5% of the InterSystems IRIS insert rate. Our plan was then to test the AWS Aurora MySQL query rate as we increased from 1 to 4 reader nodes.

What followed was weeks of difficulty. We got test results fairly quickly for one reader node and two reader nodes. However, we could not get AWS Aurora MySQL to distribute queries correctly in a three-node or four-node configuration. After trying a variety of configurations for the test harness and the AWS Aurora MySQL system under test, we filed a ticket with AWS support. We got no meaningful response. Eventually we upgraded our support agreement so that we could get a video call with an AWS Aurora MySQL expert. This call went on for hours but also produced no solution. Finally, our most senior engineer on the project decided to try greatly increasing the number of concurrent sessions in the test harness, so that we ultimately were running the tests with 28 client processes (8 ingestion workers and 20 query

workers, each with many threads). This resulted in apparently correct operation of the AWS Aurora MySQL cluster, as shown in *Table 4*:

NODES	INSERTS (PER SECOND)	COMPLETED QUERIES (PER SECOND)	COMPLETED QUERIES PER SECOND, PER NODE
1	93,976	88,586	88,586
2	93,668	181,276	90,638
3	93,185	262,294	87,431
4	92,304	324,673	81,168

Table 4: Clustered test of AWS Aurora MySQL, 1– 4 query nodes

Here, AWS Aurora MySQL is exhibiting both efficiency and scaling problems, as can be seen in *Table 5*.

NODES	*AWS Aurora*		*InterSystems IRIS*		ADVANTAGE TO *IRIS*
	vCPUs	QUERIES PER SECOND, PER vCPU	vCPUs	QUERIES PER SECOND, PER vCPU	
1	16	5,537	4	15,047	2.7x
2	32	5,665	8	15,477	2.7x
3	48	5,464	12	15,532	2.8x
4	64	5,073	16	15,557	3.1x

Table 5: InterSystems IRIS delivers 2.7 to 3.1 times more queries per vCPU than AWS Aurora MySQL — and scales better

The superior ability of InterSystems IRIS to use hardware efficiently, combined with its advantage in scalability, yields an extraordinary advantage of 3.1x in query processing efficiency, as measured in completed queries per virtual CPU (vCPU), with four nodes. The advantage to IRIS also grows as the number of nodes increases, a critical issue, since virtually all customers face processing requirements that grow over time.

Figure 6 (see next page), in which the orange bars represent InterSystems IRIS throughput and the dark green bars represent AWS Aurora MySQL throughput, shows this graphically. In addition, the large numbers ranging from 2.7 to 3.1, show the size of the advantage exhibited by InterSystems IRIS as the number of nodes is increased from 1 to 4.

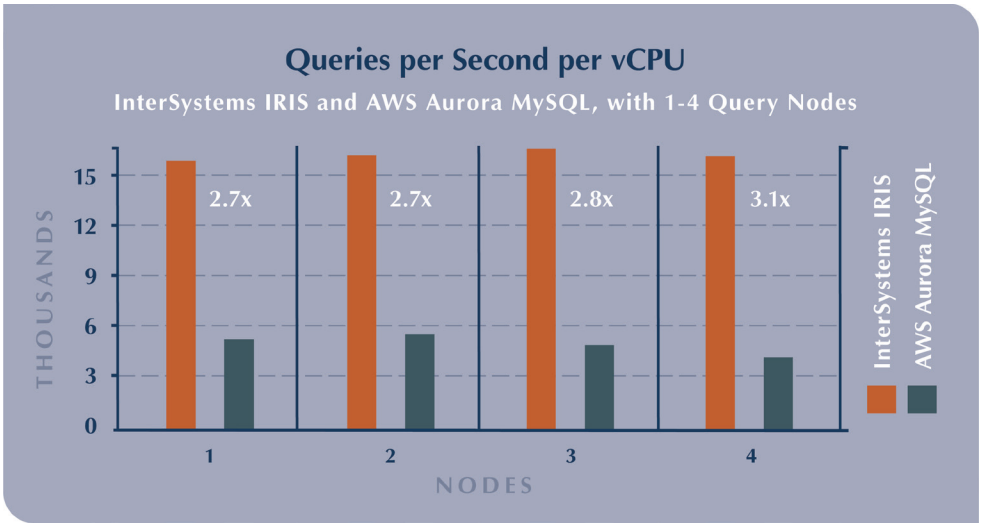

Figure 6: InterSystems IRIS delivers 2.7x – 3.1x more queries per vCPU than AWS Aurora MySQL, an advantage which grows as the number of nodes increases

6 Data Latency

Our testing brought out one more difference between InterSystems IRIS and AWS Aurora MySQL, which will be the most significant difference for many customers: **InterSystems IRIS features much lower data latency**. This is most evident in the 4-node test as shown in *Figures 7 and 8*. The term "completed queries" refers to queries that find the record they are seeking in the database. Queries that do not find the record are regarded as "incomplete."

For our cluster tests, we increased the number of distinct keys in the inserted records to 100,000 to more closely represent a large universe of publicly traded securities. We know from the insertion rates that 100,000 keys generated in the test are present in the database after the first 2 seconds of the test.

With InterSystems IRIS (the orange curve in *Figure 7*), the rate of completed queries reaches a maximum early and the rate remains at or near the maximum throughout the test. This indicates that data latency is much lower: that is, records are available for retrieval immediately after insertion so virtually all queries find the record they are looking for.

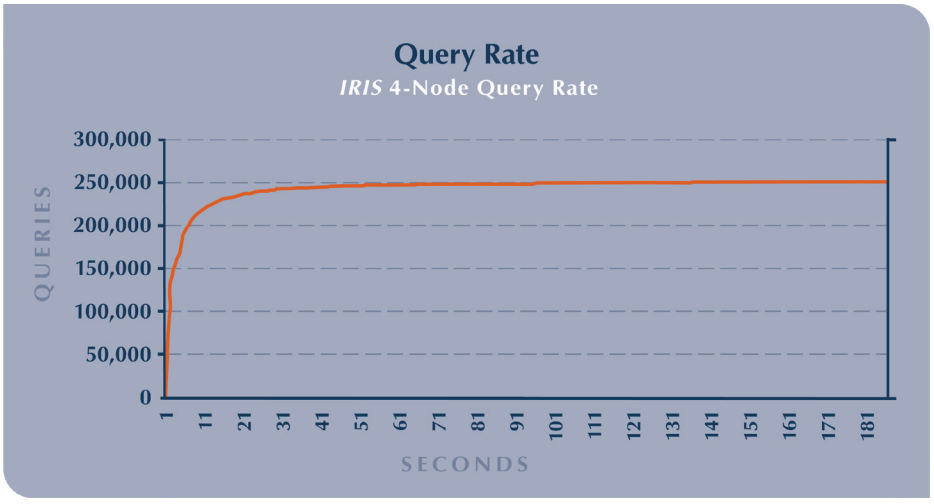

Figure 7: Query rates for a 4-node InterSystems IRIS cluster

AWS Aurora MySQL's rate of completed queries, indicated by the curve in *Figure 8*, below, consistently lags the InterSystems IRIS query rate. This is an indication of two problems. First, AWS Aurora MySQL cannot insert records fast enough while also performing queries. This is a fundamental problem in the database architecture. Second, inserted AWS Aurora MySQL records are not available for retrieval until some time AFTER they have been inserted or after new values for them have been inserted. We believe this is because AWS Aurora MySQL must actually replicate the records in data storage after insertion. The higher the insertion rate with AWS Aurora MySQL, the larger the replication lag.

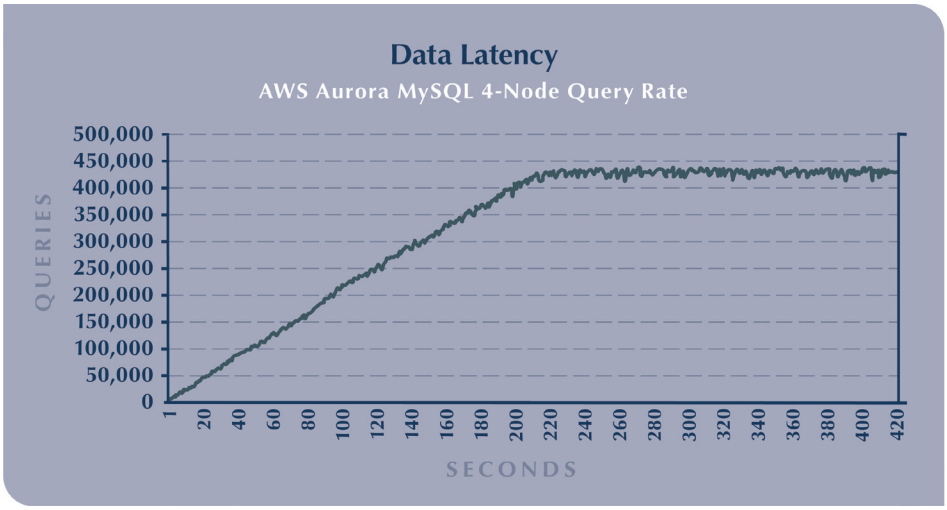

Figure 8: Query rates for a 4-node AWS Aurora MySQL cluster

InterSystems IRIS is able to deliver much lower latency than AWS Aurora MySQL — that is, queries see a new record or new value in IRIS much more quickly after it is stored in the database.

As shown in *Figure 9*, this is because InterSystems IRIS implements a distributed in-memory cache, using its distinctive Enterprise Cache Protocol (ECP), in which live data is maintained in an in-memory cache, and is fetched on demand if not present in the cache. If the data is uploaded in the database, its old version

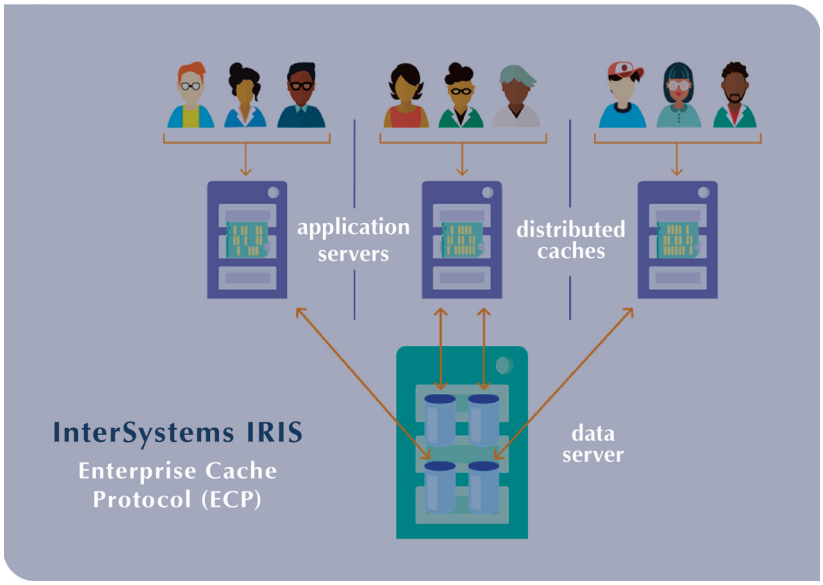

Figure 9: The distributed cache architecture of InterSystems IRIS

is purged from all nodes where it was cached as part of the transaction. The caching and on-demand data fetching is what provides the guarantee of consistency in the presence of high rates of ingestion and query. By contrast, AWS Aurora MySQL employs a conventional disk-based replication scheme, in which it must do more reads and writes to data storage as inserts are made.

7 Conclusions and Recommendation

OUR TESTS INDICATE InterSystems IRIS is uniquely able to deliver high throughput and low latency for both queries and inserts concurrently on a single database. InterSystems IRIS demonstrates significant advantages on single-node tests on AWS when compared to AWS Aurora MySQL, MariaDB, Microsoft SQLServer, Oracle and PostgreSQL.

In addition, InterSystems IRIS demonstrates a significant advantage over AWS Aurora MySQL in these same areas as well as in data latency and application simplicity in clustered tests of 1–4 nodes.

On the basis of its independent research, WinterCorp recommends that companies seeking high performance transactional-analytic data management software for the cloud look closely at InterSystems IRIS. •

WinterCorp is an independent consulting firm expert in the architecture and strategy of the modern database management ecosystem.

Since our founding in 1992, we have architected and engineered solutions to some of the toughest and most demanding database management challenges, worldwide.

We help customers define their data-related business interests; develop their data strategies and architectures; select their data platforms; and, engineer their solutions to optimize business value.

Our customers, with our help, create and implement cloud, multi-cloud and hybrid cloud architectures; they create the data foundation needed for data science, artificial intelligence and machine learning.

Our customers get business results with analytics in which their return is often ten or more times their investment.

When needed, we create and conduct benchmarks, proofs-of-concept, pilot programs and system engineering studies that help our clients manage profound technical risks, control costs and reach business goals.

We're expert with structured data, unstructured data, and semi-structured data — with the products, tools and technologies of data management for data analytics in all its major forms.

With our in-depth knowledge and experience, we deliver unmatched insight into the issues that impede scalability and into the technologies and practices that enable business success.

WinterCorp
www.wintercorp.com
42 DERBY LANE, TYNGSBORO, MA
617-695-1800