EBOOK

GridGain

Best Practices for Digital Transformation with In-Memory Computing

Part 2: Adding Speed and Scale to Existing Applications

Most companies face an unprecedented challenge around performance and scalability with their existing applications. Over the last decade, there has been a 10-100 times growth on average in queries and transactions. Data has grown 50 times in that period, with as much as 10 times occurring in the last two years alone. Many processes that used to take hours or more must now act as real-time processes that can be completed in seconds (such as one-click shopping).

These performance and scalability challenges are the result of the adoption of new customer-facing Web and mobile channels, of new technologies such as the Internet of Things (IoT), and of new types of data including social and machine data. All of these initiatives have to integrate with existing applications, and use existing data. Their increased adoption has driven up transaction, query, and data volumes, as well as a demand for real-time responsiveness.

IN THIS SERIES

This eBook series outlines the successful digital transformation journeys companies have taken, and some of the common best practices they have followed. The first eBook explained best practices for building a foundation for digital transformation. This eBook explains the best practices for adding speed and scale to existing applications that offer the least disruption and meet the long term goals of transforming the business. Subsequent eBooks cover other project types.

Consult the eBooks in this series in sequence or in random order, depending on your sequence of projects—after all, each digital journey is different.

BEST PRACTICES FOR ADDING SPEED AND SCALE TO EXISTING APPLICATIONS

In this eBook, you will learn about best practices for adding performance and scalability to existing applications:

BEST PRACTICE 1: USE SCALE-UP HARDWARE MONEY TO INVEST IN THE RIGHT ARCHITECTURE

Many companies try to support their growth by scaling the databases that support existing applications. There are three problems with this approach:

1. Most new initiatives that add loads to existing apps by accessing their data need to merge the data across apps. This merge, which happens outside the database, is also part of the performance problem.
2. No matter how fast you make the database, it cannot lower end-to-end latency that is outside the database, which is also a major part of the problem.
3. Scale-up hardware is VERY EXPENSIVE! It is also only a short-term band-aid, not a long-term solution.

Regardless of the cost, trying to support growth by scaling the underlying databases vertically with bigger hardware does not work in the long run. The performance and scalability of a single server might grow as fast as Moore's Law (about two times every 18 months), but not as fast as needed, which is about 10 times or greater every few years. And adoption of all these technologies has only just started – we can expect this rate of growth to continue over the next decade.

Even though it is well known that vertical scaling is an eventual dead end, it is often hard to find an alternative architecture for several reasons:

- The application architecture makes changing it too difficult, too time-consuming, or flat-out impossible.
- The focus is just on fixing the app in the short term, not the long-term.
- The root causes of performance and scalability issues reside outside of the group's control.

Spending all your money on vertical scaling that does not address your long-term performance and scalability issues is a waste. That money is better spent on an architecture that DOES address performance and scalability issues. The question is, what is the right architecture?

GridGain

BEST PRACTICE 2: PLAN AHEAD FOR THE RIGHT ARCHITECTURE—HTAP AT SCALE WITH IMC

There is only one way to deliver the speed, scale, and real-time responsiveness needed to become a digital business. It is implementing what Gartner coined as a Hybrid Transactional/Analytical Processing (HTAP) architecture. To support digital business transformation and other new applications (such as IoT), HTAP must both respond to any individual request at up to 1000 times scale AND execute analytics and automation in real-time. This is only possible by building a horizontally scalable in-memory computing (IMC) architecture that supports general-purpose computing collocated with the data for many different needs.

Move Data into Memory for Speed

As a beginning step, how do you lower latency. Just do the math first:

- How many hops do you add over the network across all your layers?
- How fast is each layer?
- How fast does each layer need to be?
- What new processes do you want to add?
- What new decision automation?

With all this information, now you can ask yourself "How do I deliver sub-second response times, every time, to a customer using their mobile app?"

The only answer is to have all the needed data ready for the app as close to the app as possible: in memory.

Store Data Horizontally for Scale

Now, consider how to scale a database and its transactions to 100 times or more. All the individual scaling issues with each of the apps makes this challenging enough. But when you add merging all this data, because you must, it is even more daunting. You cannot scale to 100 times by scaling each application vertically and throwing hardware at the problem. The only way to handle both the size of the data and the amount of interactions is by scaling horizontally. Apache Hadoop proved that horizontal scalability is possible. Since then, several architectures have proved this approach works for real-time computing, including Apache Ignite.

To scale horizontally with linear scalability, technologies like Apache Ignite use a "shared-nothing" architecture. In this approach, the data is broken up into separate partitions that can be moved to any node in a cluster. The hard part is not partitioning the data, but to partition it in such a way that each node can run SQL or any other computing independently of the other nodes. The key is to partition data across nodes so that it supports distributed transactions or writes and minimizes network traffic. One of the major reasons databases were scaled vertically on a single server was that the network itself was a bottleneck. This changed when software vendors figured out how to collocate code with data and execute massively parallel processing (MPP).

Collocate Data and Compute (MPP) for Speed, Scale, and Automation

Third, look at how to execute real-time computing at scale. Similar to the way Hadoop is architected for batch processing, any real-time architecture also needs to collocate data and compute tasks to ensure speed and scale for real-time computing.

Most companies rely on separate online transactional processing (OLTP) and online analytical processing (OLAP) data pipelines out of necessity. Most existing OLTP technologies only scale vertically and are not well suited for performing OLAP. OLTP applications also do not have all the data needed to analyze the business. Companies turned to efficient approaches to Extract, Transform, and Load (ETL) data from different siloed applications and piece all the data together into a data warehouse, data mart, or data lake that was better suited for analytics.

An ETL-based OLAP architecture meant the data would be out of date by the time it was in the data warehouse. This was good enough until real-time operational analytics were needed. Some companies tried to accelerate ETL and real-time analytics with specialized hardware. But vertically accelerating the wrong architecture using expensive hardware eventually failed, for a couple of reasons:

- First, the data became too big to hold on a single machine. Some companies tried to solve this with Hadoop. But while Hadoop does scale, it was not designed for real-time analytics. It is a batch-based architecture.
- Second, a separate ETL-based pipeline takes a lot of processing and network hops to move the data away from where real-time analytics need to happen. The network has become a bigger problem since the data is often bigger than the network capacity. Most networks handle 1GB per second (at most). But the amount of data needed by many customer-facing and IoT applications reaches gigabytes or more per second at peak times. Many companies have terabytes of historical data about customers. Connected vehicles, for example, now generate terabytes to

petabytes of data. Even if extra bandwidth were added, waiting one second to move the data over each segment of the network can be too long.

The best solution for real-time analytics with large data sets is to partition the data across several machines—or nodes—in a cluster and then move the computing tasks—the code—to the data (since the code size is much smaller). In the case of real-time analytics, for example, this means collocating both the real-time and any needed historical data sets together with the relevant analytical computing to ensure real-time responsiveness. This is called collocated computing, or MPP. Hadoop MapReduce is an example of batch- or micro-batch-based MPP. The Apache Ignite Compute Grid, which also implements MapReduce as well as distributed SQL, is an example of (near) real-time MPP.

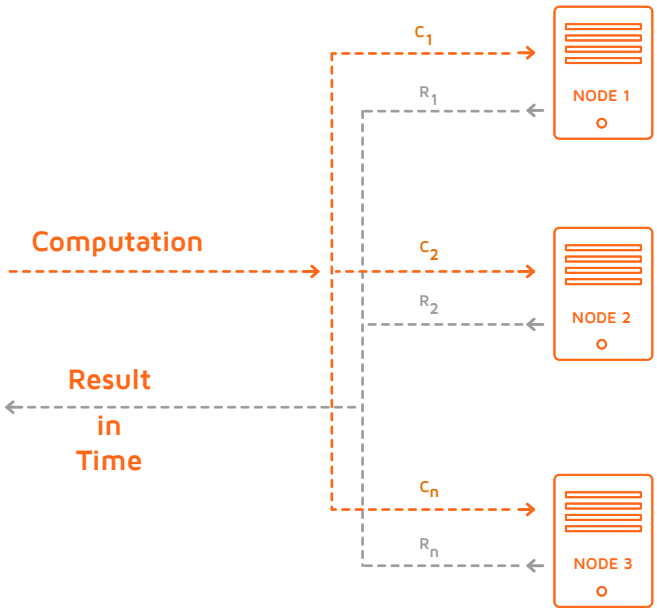


Figure 1. Collocated computing, or MPP

HTAP at Scale with IMC

This ability to perform analytics, or OLAP, in the same place as OLTP is what Gartner calls HTAP.

In-memory computing is widely recognized as the foundation for HTAP. An IMC architecture enables companies to store data in memory, scale data out horizontally, and perform MPP. All three are needed to deliver (near) real-time HTAP at scale. This combination within IMC does not just add 10 to 1000 times speed and scale. The ability to collocate code with the data makes it possible to perform analytics and continuous learning in real-time, during a transaction or interaction. IMC can perform real-time analytics or machine learning—without impacting the performance of the core transactional systems—by scaling horizontally and adding more nodes to spread the load.

HomeAway Delivers a Seamless Personalized Experience

When HomeAway became a go-to site for vacation rentals, network bottlenecks quickly followed. To calculate daily rental rates for vacation homes in real time and for each interested traveler, HomeAway needed to run as many as 2300 batches of calculations per second. Each batch contained 200 calculations and each calculation required 250K of data. 250K multiplied by 200 calculations and then by 2300 batches is 115 GB. HomeAway needed 115 GB of data per second to run their pricing calculations at peak load times.

Moving 115 GB of data per second across a network is not feasible. HomeAway would need over 100 typical corporate networks to handle their peak loads. And even then, they might need to wait one second or more just to get the data across the network—an unacceptable delay when consumers expect total response times of one second or less.

When software architect Chris Berry of HomeAway did this math, he realized HomeAway needed an in-memory data architecture that could collocate processing with data—in other words, send the calculations to the data. The ability to collocate processing with data was one of the key reasons that HomeAway first chose Apache Ignite, the open source project GridGain is built on. With Apache Ignite, HomeAway's network bottleneck went away.

Watch the HomeAway In-Memory Computing Summit Presentation

It is very important to recognize that HTAP is different from data warehousing, business intelligence, and ad hoc analytics. HTAP does not replace traditional analytics—at least not for some time. Both are needed today, and each have their own strengths. But only HTAP can deliver real-time analytics that are fast enough to impact a transaction or interaction.

BEST PRACTICE 3: BUILD AN IMDG TO ADD SPEED AND SCALE TO EXISTING AND NEW APPS, APIS, AND ANALYTICS

The early adopters of in-memory computing realized that if they were to succeed in adding speed, scale, and real-time intelligence as part of their digital transformation and other

initiatives in the long run, then they needed to add in-memory computing up front. The main question was how to add in-memory computing and HTAP without having to rip out and replace existing applications or databases. The first step for most was IMC technology that could act as in-memory data grid (IMDG).

Add Speed and Scale to Applications with an IMDG

An IMDG adds speed and scale to applications by sitting in-between an application and the underlying database. It implements a read-through/write-through cache pattern by sitting in the path of all queries and transactions. For all writes, or transactions, it writes to memory and to the underlying database. It can often be write-through where this is done synchronously (as a pessimistic transaction), or write-behind where the write to the database is done asynchronously (as an optimistic transaction). This approach keeps the in-memory layer completely in-sync with the database. It allows the IMDG to directly handle all queries, thereby offloading all reads from the database. This gives the existing database a lot of room for future growth since most of the load for most customer-facing applications come from reads, and an IMDG can deliver unlimited horizontal read scalability.

There are a few critical capabilities an IMDG needs to help minimize any rework.

- SQL Support. Ideally the existing application does not need to know the IMDG exists. For SQL-based applications, that means the IMDG should be able to fully support SQL. The application should be able to take a new JDBC or ODBC driver as the new integration point. The IMDG would then intercept, process and manage any SQL.
- Distributed pessimistic ACID Transactions. The IMDG must be able to support ACID transactions for any writes. It needs to act as a transaction coordinator to ensure write consistency between the in-memory layer and the database, to keep both in sync. It needs to support synchronous, pessimistic transactions because applications generally expect those from the existing RDBMS. If the data is scaled horizontally, that means it must be able to support distributed in-memory transactions.

There are a host of other capabilities needed for an IMDG. But if an IMDG does not support SQL and distributed pessimistic ACID transactions, the application needs to be rewritten. If the IMDG does not support SQL, all the SQL needs to be replaced (usually with key-value operations), which takes a significant rewrite of the logic around the data as well. If the IMDG does not support pessimistic transactions, compensating transactions are needed. However, applications that rely

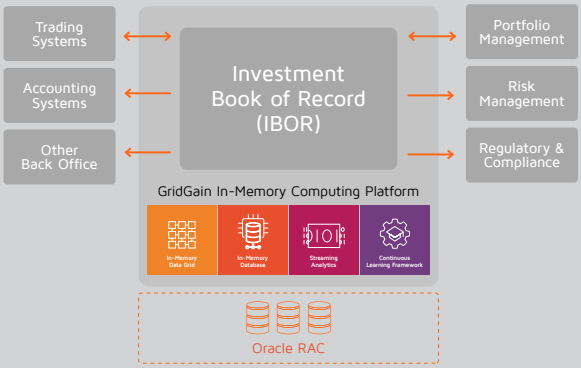
Adding Speed, Scalability and Analytics at Wellington Management

Wellington Management is one of the top 20 global asset management firms in the world, with more than $1 trillion in client assets under management.

Wellington had three major challenges:

1. Its current systems were no longer scalable due to an exploding growth of financial data. It needed horizontal scalability to handle the long-term growth.
2. The 2008 financial crisis resulted in a wave of new financial regulations that resulted in more complexity and risk in existing systems.
3. Many more new and complex asset classes have been introduced in the last few years based on customer demand, and there is a big need to release new asset classes faster

Wellington's solution was to deploy its investment book of record (IBOR) on the GridGain in-memory computing platform. The Wellington IBOR serves as the single source of truth for investor positions, exposure, valuations, and performance. All trading transactions and account and back office activity flow through the IBOR in real time.



- **Horizontal Speed and Scalability**: Wellington's IBOR has unlimited horizontal scalability. It uses GridGain's SQL support to add speed and scalability by sliding in-between Oracle, its system of record, and the applications. The result is at least 10 times faster performance by adding in-memory computing on top of its Oracle database deployment.
- **Use of HTAP**: The IBOR is an HTAP system that is used by portfolio management teams for real-time position, market value, exposure, and performance analytics; by risk management teams for risk analytics and overall risk management; and by compliance teams to ensure, in real-time, that all regulatory requirements are met.

Why Wellington chose GridGain

- In-memory computing
- Horizontally scalable
- Supports distributed SQL
- ACID compliant (consistent data)
- Collocates data and computing
- Combines operational and analytical workflows (HTAP)

on exact values—bank account balances, inventory levels, etc.—usually require pessimistic transactions.

Companies that adopted an IMDG up front generally found it less expensive in the short term. It often paid for itself because it was cheaper than buying new scale-up hardware. Said another way, a good IMC best practice is to first choose projects where performance and scalability challenges are very expensive to fix with hardware. Each project should be able to show a positive ROI.

Build a Bridge to the Right Architecture

Many of these companies also looked to the future, and realized that, while IMC is an investment, adopting IMC one project at a time was the most cost-effective, lowest-risk approach. But it did require leadership to guide investments across projects. Beyond picking the right order for projects, it meant making decisions within each project on what additional longer-term investments to make.

This cross-project approach to IMC meant they needed to choose the right IMC technology that could act as more than just an IMDG. It needed to provide a bridge to the right long-term architecture and support other types of projects needed for real-time business.

One reason IMC needs to span projects and applications is that it needs to bring together all real-time data. An IMDG unlocks any data in existing applications. Once the data is in-memory, it can be merged with other data and used by other applications, APIs, or analytics. Any new workload – including creating new APIs or adding analytics and decision automation – can be handled by adding more nodes to the cluster without impacting the performance of the existing applications.

What early adopters of IMC realized was that if the IMDG is part of a broader IMC architecture or common platform that can be used across projects, then it is possible to build a common real-time data, application, API, and analytics layer over time.

Add Real-Time Analytics and Decision Automation

Beyond application speed and scale, one of the more common IMC uses is augmenting existing applications with real-time analytics, or using various forms of decision automation including machine and deep learning.

Real-time analytics requires analyzing real-time and historical data together in (near) real-time. The optimal place to do this is within the operational applications. But most applications were not built to support these capabilities. An IMDG combines real-time and other data with MPP that can run real-time analytics against all the data at scale. It makes an IMDG a great option for augmenting existing applications with analytics. (This will be explained later in more detail in the eBook on real-time analytics projects.)

Add Speed and Scale to Existing and New Real-Time APIs

The ability to combine data also makes an IMDG ideal for APIs. An API by design is meant to provide a single, simple view of information about a customer, an account, or a service, along with the operations to use the data. This data usually resides across many disparate legacy systems. It is not possible to first access the data in each system via middleware layers that in turn access each application, that in turn access each database; then merge all these results together across applications in a new layer AND get fast performance. The fastest way is to have all the data already together, merged in memory, ready for each API to consume.

BEST PRACTICE 4: MAKE SURE YOU CAN IMPLEMENT A HORIZONTALLY-SCALABLE IMDB

While an IMDG that offloads reads solves many companies' initial challenges with speed and scale, eventually higher write throughput is needed.

First, there are many new types of data that are much bigger data sets than the traditional data. Transactional data is perhaps only 10% of the total data stored by a company today. Web session data, social data, and machine data from IoT can be terabytes or petabytes. They result in much higher data ingestion rates. They also lend themselves to horizontal scalability. An IoT network or mobile applications can range from hundreds to millions of connections. The ideal architecture for ingesting these new types of streaming data is to scale out and ingest horizontally, not vertically.

Second, existing applications can also have performance and scalability issues with writes. In that case an IMDG architecture on its own is not enough. They need a horizontally scalable database architecture.

An in-memory database (IMDB) solves both issues. But you need to plan ahead and make sure you have an easy transition from an RDBMS to an IMDB. Otherwise you end up with a major code rewrite as you rip out and replace an existing IMDG and/or database with data infrastructure.

Make sure your IMC architecture supports an easy transition. The best approach is to implement an IMDG that includes the following capabilities:

- Persistence that can be turned on any time for any data stored in an RDBMS, and that can inherit its schema. This makes it easier to decommission an RDBMS that may already be underneath an IMDG.
- The ability to merge data in memory across multiple data stores. Most databases do not provide this capability.
- Transactional consistency that ensures data is never lost. While that may seem simple and works properly for many IMDB-based technologies, not all IMDG technologies are fully ACID-compliant. This means that will not support pessimistic transactions.
- Persistence that allows only a subset of data to be in memory, with most data stored on lower-cost storage. While it is possible to hold just about any data set in memory with a horizontally-scalable IMDG, it is not always cost-effective.

The last capability is to make sure you can perform general-purpose MPP that allows any computing to be run locally against the data on each node.

BEST PRACTICE 5: ENSURE GENERAL-PURPOSE MPP

One of the most important attributes of any IMC architecture is the "C": the computing. As mentioned earlier, the only way to add both speed and scale is to move data into memory for speed, partition data horizontally, and then collocate data with all of the relevant real-time computing. If you cannot collocate specific code with the data, you are limiting your ability to reuse data.

There are many deployment architectures that work. Most companies using Apache Ignite, for example, rely on Docker to package APIs and IMC technology together on each node, along with the Ignite Compute Grid to execute collocated code.

Regardless of which technology, it is important that you have a general-purpose MPP framework. It is necessary when using MPP to add speed and scale to a host of different code bases over time:

- Web and Mobile code: REST APIs have become the main way to access any capabilities in web and mobile apps. Therefore, it is critical to have support for invoking collocated functionality via REST.
- Data access, analytics and transactions: SQL is the most popular data language by far, and the second most popular language overall after JavaScript. But other models are also important, such as key-value APIs.
- Applications and APIs: There are almost too many languages to count, and most should be supported at least as clients. Evaluate which languages are most important candidates for MPP across projects. Java, .NET and C++ are common candidates.

BEST PRACTICE 6: KEEP PRIORITIZING REAL-TIME PROJECTS BASED ON A GREATER PLAN

Just a continual reminder. As mentioned throughout, plan to use the same in-memory computing technology across applications. It enables you to add speed and scale to existing applications. It also helps you succeed with digital transformation and becoming a real-time business because it:

- Opens up and combines data across existing applications for new uses, such as supporting new APIs that can be consumed by anyone
- Supports new workloads by scaling horizontally, which makes it easier to deliver new capabilities in days without having to change existing applications.
- Allows you to ingest massive amounts of data in real-time and combine it with existing data, during transactions and interactions, to streamline, react to, and improve each step of the customer's experience.

Keep in mind that this is a journey that requires careful planning. You might need to implement projects in a certain order, so that for each step you have already built out the parts to make a given project succeed and to get a positive return on each investment.

GridGain

Keep in mind the same guidelines discussed in part 1 of this eBook series:

- Review any requests for hardware and additional software licenses. Ask whether in-memory computing would make sense. Many such projects are adding vertical scalability to handle increasing loads in the short term, when in fact the better longer-term answer is in-memory computing.
- Review any database purchases or re-platforming projects that rip out and replace an existing database. They are often being done to improve performance and scalability. In-memory computing might be the better answer, especially if performance and scalability issues are due to large read- and data-intensive computing workloads.
- Review all the backend sources for new API development. Determine what additional loads will be created and consider whether the use of in-memory computing or other technologies would lead to a better architecture.
- Add speed and scale to existing systems before building new APIs. Existing systems need to be able to handle the additional loads.
- Consider grouping projects together that access the same systems. Many companies have not sufficiently enforced a common access layer and implemented a single in-memory computing layer that controls all write access for applications from different parts of the company. This can create more work in the end trying to maintain consistency across applications. Commitment to a common layer is a lower-cost and lower-risk approach (if possible).
- Order your projects by understanding your data dependencies for each project, and knowing what data needs to be in-memory for each project to succeed. If a project is dependent on other data being in-memory, either work on moving that data into memory first, or combine the projects.

NEXT STEPS: PLAN AHEAD TO SUCCEED WITH OTHER TYPES OF REAL-TIME PROJECTS

The focus of this eBook was on how to add speed and scale to existing applications as a first step in a longer journey involving digital transformation and other major initiatives. The next eBooks are guides for each type of project you might tackle in your journey. Different topics include:

- Building an in-memory computing foundation and roadmap
- Adding speed and scale to existing applications (current eBook)
- Developing new apps and APIs
- Leveraging and integrating with cloud services
- Implementing HTAP
- Building event-driven applications and streaming analytics
- Adding machine and deep learning to help automate decisions

Go to the resource section at www.gridgain.com for more information on any of these eBooks, related webinars, or customer examples. For more information on this ebook series or any other GridGain ebook, visit https://www.gridgain.com/resources/ebooks.

GridGain

Contact GridGain Systems

To learn more about how GridGain can help your business, please email our sales team at sales@gridgain.com, call us at +1 (650) 241-2281 (US) or +44 (0)208 610 0666 (Europe), or go to complete our contact form at www.gridgain.com/contact and we will contact you.

About GridGain Systems

GridGain Systems is revolutionizing real-time data access and processing with the GridGain in-memory computing platform built on Apache® Ignite™. GridGain and Apache Ignite are used by tens of thousands of global enterprises in financial services, fintech, software, e-commerce, retail, online business services, healthcare, telecom and other major sectors, with a client list that includes ING, Raymond James, American Express, Societe Generale, Finastra, IHS Markit, ServiceNow, Marketo, RingCentral, American Airlines, Agilent, and UnitedHealthcare. GridGain delivers unprecedented speed and massive scalability to both legacy and greenfield applications. Deployed on a distributed cluster of commodity servers, GridGain software can reside between the application and data layers (RDBMS, NoSQL and Apache® Hadoop®), requiring no rip-and-replace of the existing databases, or it can be deployed as an in-memory transactional SQL database. GridGain is the most comprehensive in-memory computing platform for high-volume ACID transactions, real-time analytics, web-scale applications, continuous learning and hybrid transactional/analytical processing (HTAP). For more information on GridGain products and services, visit www.gridgain.com.