EBOOK

GridGain

Best Practices for Digital Transformation with In-Memory Computing

Part 3: Building New Apps and APIs

Most companies face an unprecedented challenge around performance and scalability with their new and 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).

This growth in data has created a problem: all these new channels must be integrated with the existing IT systems that support customers and the business, and the existing systems cannot handle the required levels of the speed and scale. This new growth requires a new, real-time business layer that meets the demands of digital business to:

- Open up business assets as APIs.
- Add real-time analytics and decision automation.
- Enable much greater agility.
- Deliver unlimited speed and scale.
- Allow incremental IT change instead of requiring a "Big Bang" approach.

Fortunately, several early adopters of in-memory computing have successfully implemented an API-centric architecture that works. This eBook summarizes some of their best practices.

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. Part 1 of this eBook series, Building an In-Memory Computing Foundation and Roadmap, explained best practices for building a foundation for digital transformation.

Part 2 of this series, Adding Speed and Scale to Existing Applications, focused on the right long-term approaches for adding speed and scale to existing applications being overwhelmed in part by new digital channels.

This eBook explains the best practices for building new apps and APIs to ensure they perform, scale, and are flexible to change.

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 NEW APPS AND APIS

In this eBook, you will learn about best practices for adding speed and scale to new apps and APIs using in-memory computing:

BEST PRACTICE 1: DESIGN APIS AROUND THE CUSTOMER EXPERIENCE

Solving most IT challenges requires a combination of the right people, processes, and technology. Digital business is no different. Even if you put in place the right technology and architecture, you will probably fail if you do not put in place the right people and processes. If you read nothing else in this eBook, closely examine the first two best practices. In many ways, they are the most important. If you understand these first two points, the others will make more sense (even if you do not review and use them for a while). This first best practice is mostly about people, not technology. But it should help you understand how to justify any digital business-related project.

Digital Business initiatives have some of the largest IT budgets. It used to be that only 20% of a typical IT budget went to delivering new applications or infrastructure. According to some estimates, digital business-related initiatives are now using as much as 40% of the IT budget.

Why? The answer is simple. On average, digital business can deliver more to growth and the bottom line than any other initiative. The reason is that the purpose of digital business is to improve the customer experience. A Harvard Business School Study found that with an increased focus on the customer, the top 25% of digital leader enterprises outperformed the bottom 25%, with 50% greater earnings. The full benefits of improving the customer experience are even greater: it can help double revenues and more than double profits within ten years.

"40 percent of all technology spending will go toward digital transformations, with enterprises spending in excess of $2 trillion in 2019"

IDC

But to reap the benefits, you need to focus on the customer experience. As explained in Part 1 of this eBook series, Building an In-Memory Computing Foundation and Roadmap, you need to design inward from the customer's perspective. To put it into API management terminology, you should design from the outside in. Companies such as ING Bank that have succeeded in these initiatives were completely aligned from the CEO to the engineers around the customer experience.

This means making sure the organization structure and roles are in place, usually within an API management or digital business group that can cut across different parts of the company. This structure is sometimes called a center of excellence (COE). There should also be executive sponsors and champions of the overall initiative who have the authority to realign siloed organizations around customer-centric processes.

To ensure this group succeeds, it is a good idea to define a few other roles. Within the group you should see the following roles and responsibilities defined:

- **Chief customer officer**: someone usually outside of IT who is responsible for the customer experience.
- **Chief digital officer**: a leadership role, ranging from an IT director up to CIO, who is responsible for leading the digital business initiative.
- **Process architects**: architects focused on designing and implementing customer processes that improve the customer experience.
- **System architects**: architects responsible for the overall architecture of the new real-time business layer, which is more than API management by necessity
- **Data architects**: architects who are often in a data governance or integration group responsible for helping ensure data quality and consistency.

Goal of Digital Business: Improve the Customer Experience

Leading digital companies generate better gross margins, better earnings and better net income than organizations in the bottom quarter of digital adopters.

Improving the customer experience by becoming more digital is worth more than your entire IT budget, not just the project, if you focus on the customer.

PERFORMANCE METRIC	DIGITAL LAGGARDS (BOTTOM 25% OF ENTERPRISES)	DIGITAL LAGGARDS (TOP 25% OF ENTERPRISES)
3-Year Average Gross Margin[1]	37%	55%
3-Year Average Earnings Before Taxes	11%	16%
3-Year Average Net Income	7%	11%

The leaders seem to be 50% more profitable than the laggards, pulling in 5% more. And even they are not realizing the full benefits. As we went through in the first webinar, the total benefits can be doubling revenues and more than doubling profits over a ten year period or less.

It is also about survival. If you do not succeed in delivering a great customer experience, you will probably die as a company.

What that means for your projects is that you need to be aligned with an executive sponsor, and your digital business initiatives somehow tied to improving the customer experience as well.

Leaders post a three-year average gross margin of 55 percent, compared to just 37 percent for the laggards. Leaders also outstrip laggards in three-year average earnings 16 percent to 11 percent. And in three-year average net income, leaders have the advantage 11 percent to seven percent.[1]

"It's a pretty substantial gap and it correlates with performance in significance ways," says Iansiti, a professor of business administration at HBS, who collected his research from more than 300 senior business and technology decision makers from large enterprises.

1 https://www.cio.com/article/3122806/it-industry/digital-laggards-must-harness-data-or-get-left-behind.html

GridGain

- **API Product Manager(s)**: the people that own the APIs and define them based on how consumers use them as products, and how they impact the end customer experience.
- **Project architects and developers**: staff meant to lead or augment various projects led by other groups as needed. These architects and developers help provide the core expertise across the company.

BEST PRACTICE 2: BUILD A NEW REAL-TIME API LAYER ON IN-MEMORY COMPUTING

The second best practice requires much more explanation and patience. What several early adopters came to realize is that the right long-term architecture for API management should be built on in-memory computing (IMC). To understand why, you first need to understand the goals of real-time digital business, and how the underlying challenges with speed, scale, and agility make IMC the only answer.

There are three basic goals of API management as a part of any digital business initiative:

- Open existing systems as APIs.
- Deliver APIs in days, not in months or years.
- Improve the real-time customer experience and business operations.

There is a fourth goal that many companies unfortunately realize mid-way through their digital journey:

- Deliver a 10-1000 times increase in speed and scale.

The last point is the easiest place to start to describe the right architecture.

Deliver a 10-1000 Times Increase in Speed and Scale

Existing systems that can only scale vertically will not support the 10-100 times growth in interactions and transactions. Part 2 of this series, Adding Speed and Scale to Existing Applications, explained that the right architecture for adding the needed speed and scale is in-memory computing (IMC). What became clear to people implementing API management at scale is that IMC must be part of the API itself, not just layered in between existing applications and databases.

Do the math regarding speed and latency. People expect that any Web or mobile application must respond in under one second, all the time. API calls go over the Internet and across multiple internal network hops, processing layers, security, and policy management before the actual API call is made. In the case of ING Bank, that travel time left only 100 milliseconds for the API call itself. This included four internal layers of APIs, then a call over middleware to access multiple applications, which in turn accessed multiple databases. In short, it was not possible to get a response in an acceptable time frame. The only way to deliver an acceptably fast response time was to already have the data in-memory within the API.

To scale IMC with APIs, IMC must be collocated with each node. The most common deployment architecture with APIs is a combination of Docker and Kubernetes, where the node of an in-memory data grid (IMDG) is included in each Docker container. As the API instance initializes, so does the node. It attaches to the existing cluster, brings in any relevant data, and begins to handle requests.

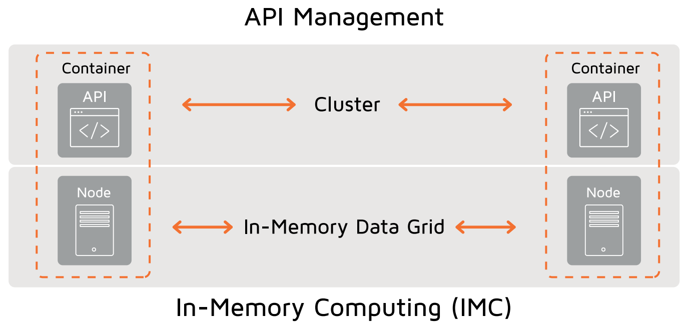


Figure 1. API-Centric Architecture on IMC

Open Existing Systems as APIs

For most companies, API management involves exposing existing systems instead of building new ones from scratch. The principles of being API-first, designing APIs from the outside-in, treating an API as a product, and managing the API lifecycle and API community are all part of API management. So are the best practices for layering API security, identity, policy management, and orchestration. Eventually, API design works its way down to the existing systems.

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.
- Then access each application.

- Then access each database.
- Then merge all these results together across applications in a new layer AND get fast performance.

The fastest way is to locate all the data together, merged in memory, ready for each API to consume. An IMDG as part of a data service keeps the most up-to-date version of the data in memory, which allows it to support all queries directly within the API.

To support data services, beyond providing horizontally scalable in-memory data management, any IMDG architecture must support two key features to make it easier to adopt:

- **SQL Support**. Most data services rely on SQL access. An existing API should be able to take a new JDBC or ODBC driver as the new integration point with an IMDG. Developers working on new APIs should be able to leverage SQL instead of writing extra code to translate the data when accessing an existing database.

"WE WANT TO BE A TECH COMPANY WITH A BANKING LICENSE."

— Ralph Hamers, CEO, ING Group

How to Create The Right Real-Time API Architecture

ING's use of IMC within their API architecture is a great example of what a scalable, real-time, and flexible API architecture with IMC should look like.

On top of their existing applications and their existing channels, ING built two API layers.

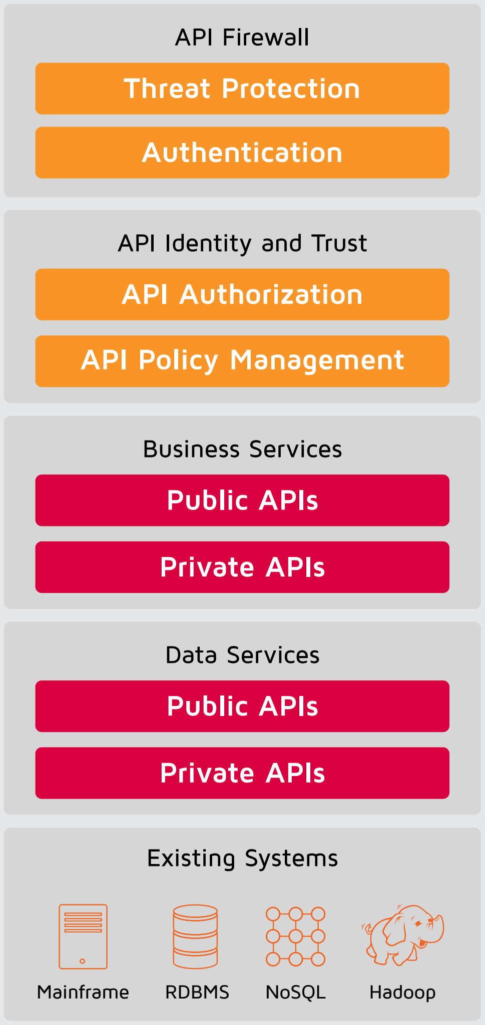
- The first layer is data services, which bring together data from different databases and applications into a single, simple view of a customer or account. By embedding an IMDG node that holds all the relevant data in memory within each API, the APIs can deliver sub-100 millisecond response times every time.
- The second layer is focused on the customer experience within each channel, and across them. IMC is used to store the customer's session state and other information. This in turn is used to improve the responsiveness of the experience. Since there was no other place for holding the current state, IMC became the system of record for the current experience.

Two important notes. First, each API layer is in fact two layers: a set of private APIs used to expose specific capabilities, and a set of public APIs that are consumed by others. This combination is a best practice that makes the APIs much more flexible to underlying change. For each underlying application change, one to three private APIs might change. Another private or public API might be changed to add new functionality. But the public API interface, which is a public contract with anyone using it, can stay the same.

Second, there are at least two layers of security needed on top of the APIs. The outermost layer often performs application-level attack prevention and authentication. The layer below it usually performs authorization and enforces data security and business policies including audit and compliance. These two layers are critical because APIs are used by the outside world—as a mobile app, a cloud app, or an external partner.

This API architecture is at the core of ING's innovation. It helped cut ING's end-to-end latency to less than 100 milliseconds. It also helped the bank rapidly create new APIs from microservices for consumers and third parties, and be first-to-market with new services for the revised Payment Services Directive (PSD2), the Single Euro Payment Area (SEPA) initiative, and instant payments.

Watch the ING In-Memory Computing Summit presentation.



- **Distributed Pessimistic ACID Transactions**. An API expects SQL to just work. When an API makes a distributed write across existing system to commit changes to a customer or account, the developer should not have to worry about the complexity. The IMDG needs to act as a transaction coordinator to ensure write consistency across existing applications. Developers expect distributed, synchronous, pessimistic transactions to just work.

Deliver APIs in Days, Not in Months or Years

For enterprises embracing digital transformation, the goal is to be as agile as the new Internet startups—such as Amazon, Expedia, eBay or PayPal—that have disrupted their respective industries. This is not easy for existing enterprises because they must build on top of existing IT systems that cannot be changed in days.

An IMDG also helps solves another major challenge: how to open data for use by new apps and APIs to become more agile. While a private/public API architecture helps APIs be much more flexible to change than the underlying applications, it is still hard to access existing data. Beyond integration, each application needs to ensure they can handle the additional load.

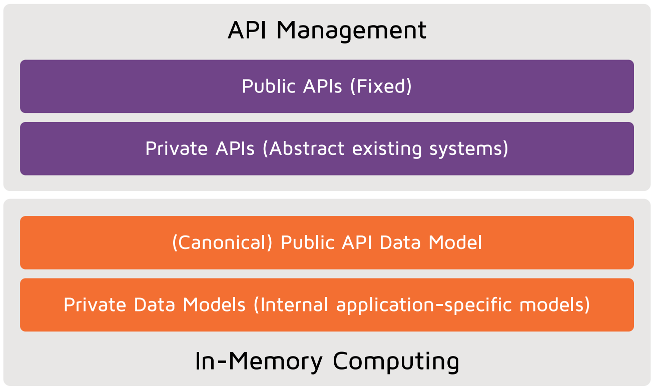

API Management
Public APIs (Fixed)
Private APIs (Abstract existing systems)
(Canonical) Public API Data Model
Private Data Models (Internal application-specific models)
In-Memory Computing

Figure 2. Public and Private APIs

Whenever an IMDG is used with an existing application, unlocking the data becomes easy. An IMDG provides linear horizontal scalability when implemented properly. By provisioning copies of the data on additional nodes as needed, an IMDG helps ensure that increased loads do not impair the performance of existing applications. The IMDG becomes the fastest way to access any data an application needs.

Improve the Real-Time Customer Experience and Business Operations

Improving the customer experience is the true goal of digital business. Beyond just making integration easier, it requires adding real-time intelligence and decision automation. The same APIs initially used to open up operations also have to be extensible to allow real-time analytics and automation using technologies like machine and deep learning. This part of intelligent decision automation often comes last. The first step of improving the experience is often just implementing a better process, not automating it. But the architecture you choose needs to support this longer term goal or the business benefits will not be realized in the long term. This requires in-process Hybrid Transactional/Analytical Processing (HTAP), which as explained in Part 1 of this eBook series, Building an In-Memory Computing Foundation and Roadmap, is the ability to perform analytics in-process, during the transaction or interaction. In other words, the IMC layer that functions as an IMDG should also be able to support in-process HTAP. More on this later.

BEST PRACTICE 3: BUILD A NEW REAL-TIME LAYER ON TOP OF EXISTING SYSTEMS

Another point that was made in Part 1 was that digital business needs to happen in real-time. However, many of the existing systems are not real-time, cannot support real-time interactions, and cannot provide real-time visibility either. The only way to support real-time transactions and analytics is to build a new real-time layer using in-memory computing that processes real-time data and reacts in real-time. This is usually provided by an API management layer, that in turn has its own real-time data managed by an IMC layer.

The question becomes how to keep this real-time data in sync with batch and other non-real-time systems. The answer, which is the same answer for most integration architectures, is to trick the existing systems into operating as-is. The architecture works as follows. First, send real-time transactions to the newly built real-time layer. As each real-time transaction happens, it will pass the transaction to the existing system (at some point). The new layer passes the transaction through to the existing system the old way, transparently, as if the new system had done nothing yet. It is up to the IMC layer to determine whether to coordinate the transaction in both systems synchronously or asynchronously. For example, for existing batch-based systems, the pass-through transaction is typically "fire and forget"—meaning it is simply passed onto the system without requiring any transaction acknowledgment.

Second, for any changes that occur in the existing systems, a synchronization layer built outside the application propagates any changes that occur to the real-time layer. There are two approaches you can refer to as sync or CDC.

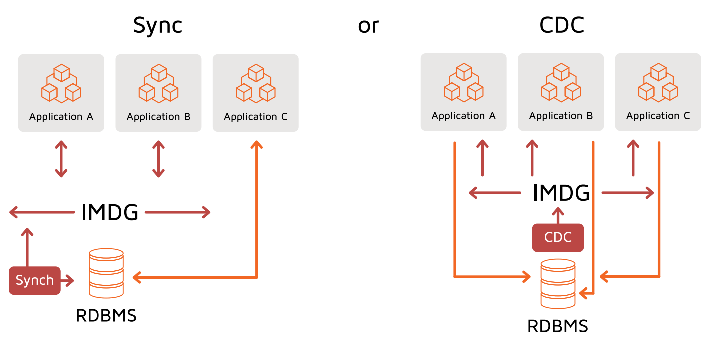


Figure 3. Sync or CDC

Sync refers to the preferred IMDG architecture. You slide an IMDG in-between the application and database in the path of all reads and writes. The IMDG is then able to keep the data in memory in sync with the database. If another application can write to the underlying database without passing through the IMDG, you can build an integration layer based on triggers against the database or new logic that pushes any changes to the IMDG synchronously or asynchronously. One leading fitness company takes this approach, pushing any updates from their backend system to the new app.

If you cannot control all writes to the database via applications a second approach is to leave writes as-is (going directly to the database), but then change all reads to point to the IMDG. You then use change-data-capture (CDC) to continually update the in-memory data with any changes as they occur. CDC typically adds a 2-5% load on the database, which is much more efficient than using triggers to capture any changes. CDC captures all writes exactly once off the write-ahead logs (WAL) used for transactions, and streams them out to a target operational data store. GridGain supports this use case with a connector for Oracle GoldenGate.

Both sync and CDC are perfectly valid, broadly-used architectures.

An IMC layer not only improves responsiveness. It helps with availability as well. As explained in more detail in Part 1 of this eBook series Building an In-Memory Computing Foundation and Roadmap, having the data already merged and up-to-date means that for any queries, the existing systems do not have to be available.

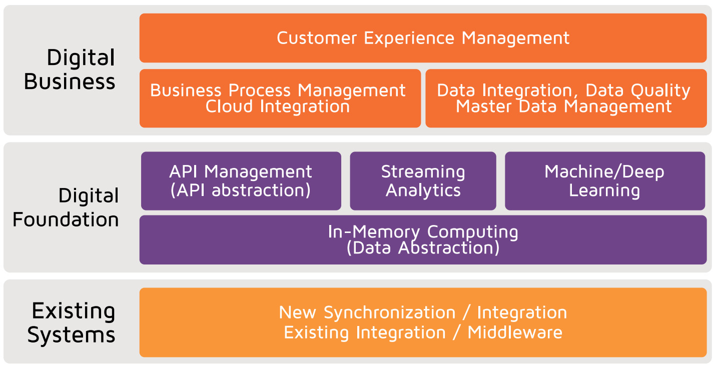

Digital Business
Customer Experience Management
Business Process Management Cloud Integration
Data Integration, Data Quality Master Data Management
Digital Foundation
API Management (API abstraction)
Streaming Analytics
Machine/Deep Learning
In-Memory Computing (Data Abstraction)
Existing Systems
New Synchronization / Integration Existing Integration / Middleware

Figure 4. Digital Transformation Layers

BEST PRACTICE 4: BUILD REAL-TIME DATA, EVENTS, TASKS, AND PROCESSES

Building a real-time business layer does not just require implementing API management. APIs are basically just steps in a process. You are also implementing new, real-time processes that are composed of real-time:

- Data
- Events
- Tasks
- Other processes

So, it follows that you have to practice all these disciplines together. Successful digital transformation initiatives often have architects who help design all aspects of processes, including how to react to important events. The trick is to add different tools and capabilities only when each capability is truly needed and will deliver a positive return on investment.

Generally, to make digital business work, there are a number of layers that get built on top of existing middleware, and alongside or on top of API management. You need to identify when and how you build out these layers across projects over time, based on when you need the capabilities.

Digital Foundation

First, there is the digital foundation. This includes the core APIs that provide access to tasks and data. All the APIs rely on in-memory computing.

Over time streaming analytics and machine and deep learning get added to augment the APIs with real-time intelligence and decision automation. These help improve customer experience and critical business operations initiatives. The need for both real-time analytics and real-time decision automation requires that they run in the same IMC platform that supports the APIs.

While API management is more task-oriented and part of a structured process, streaming analytics is not generally built on the same types of APIs you expose to customers. Streaming analytics involves building compute tasks to process and react to streams of events such as customer interactions, IoT, or other real-time activities. This might be built on streaming API technology or messaging such as Apache Kafka.

Solving a Persistent Customer Service Problem with In-Memory Computing

A leading health and fitness company had a challenge with monthly membership payments. Whenever a member paid online, the payment was submitted to a batch-based system that processed the payment overnight. As a result, it could take nearly a day to update the customer's balance. To the customer, it looked like the payment was not accepted, and they kept trying to pay. This glitch increased costs, because someone had to go in and reverse the extra payments. It also led to dissatisfied customers who would sometimes cancel their memberships.

The company used in-memory computing to fix this problem. Now whenever a payment is made, an API receives the request, updates the balance information in memory, and then passes the transaction to the back-end system. Whenever a customer checks to confirm that their payment was processed, the system looks up the account status in memory, which immediately shows the payment as received.

Doubling the loyal customer base can double revenues, and more than double profits. It is important to remember that the goal of digital transformation is to improve the customer experience and to help increase customer satisfaction, retention, and spending—which in turn increases revenues and profits. The best way to monitor and improve every experience is with real-time event processing or streaming analytics. This is accomplished by monitoring and reacting to key events to improve each customer's experience. If you can identify and fix all issues with customers, you can reduce attrition.

You might use the following setup for streaming analytics which is increasingly common and often referred to as KISS:

- Apache **K**afka for event streaming
- Apache **I**gnite for data management and collocated computing
- Apache **S**park for stream processing
- **S**pring (including Spring Boot or Data) for developing APIs or code to process the data

Many companies use different combinations for their applications. Machine learning and deep learning takes analytics one step further by identifying and automating the responses to various important events. It is implemented as collocated compute against any streaming or other data about interactions. Model training, testing, and execution can be run in parallel against a combination of streaming and stored data. IMC makes this possible by scaling out to support the parallel workloads.

Digital Business Layer

On top of the foundation are the new applications and processes that deliver the customer experience, as well as where the customer experience is monitored and managed. There should exist some version of business process management (BPM) driven by process architects. Processes must be modeled, implemented, and continually improved, like any other BPM initiative. You can also expect cloud integration as you consume external APIs and combine them with your own APIs.

You also need real-time data management and governance. With traditional data management, it used to be that individual systems were responsible for their own data quality. Master data management (MDM) and data quality controls were mostly implemented to help merge data for a data warehouse, and build a single view of the business across disparate systems outside of the transactional applications (after the fact). Any errors would usually get flagged as exceptions and then fixed manually by people.

However, once you rely on real-time processes, bad data in transactions is no longer acceptable. It simply takes too long to have people manually fix exceptions. Data must be accurate and consistent from the beginning to help ensure customers always get quick responses.

The best approach is to validate data at the point of entry with some combination of data quality and MDM. Some best practices include more self-service steps to confirm data is correct, rather than rely on internal corrective procedures.

The need for real-time data management is the reason why a chief data officer and data architects eventually get included: data management and data governance must change to support real-time business. You can expect to implement real-time data integration, data quality, and even MDM because data consistency is critical. Several in-memory computing initiatives have started implementing real-time MDM and providing master data in memory as one of their first projects. It helps guarantee the system can support real-time data validation. MDM is also a relatively low-risk, low-volume first project for in-memory computing, with a fast payback for customer-facing applications.

There should also be clear focus on the customer experience. Customer experience management is not a single tool. Rather, it is a set of tools that support the process of continuously monitoring and improving the customer experience. Your business process improvement and your analytics and machine/deep learning should really be driven by this discipline. The measured customer experience improvement is the justification you need to pay for these projects. This is why you need to be aligned with the executive sponsors responsible for the customer experience.

BEST PRACTICE 5: FREE YOUR DATA! NOW!

This is a simple rule. To have data readily available for use in a given project, you first need to unlock it from existing applications. So, plan ahead and make sure certain projects that unlock the data happen in advance, before the data is needed. The alternative is that you open up the data and build out the project that uses that data at the same time.

Another important consideration: make sure you consider where to implement the IMC architecture to solve any challenges with speed and scale. If you can solve all the challenges by implementing IMC with the APIs, for example, and as a result do not need to implement it in between an existing application and database, then just implement IMC with the APIs. This saves time and cost for the rollout of new business initiatives. The API layer can then be used to unlock the data for other needs, such as HTAP collocated with the APIs.

BEST PRACTICE 6: DESIGN WITH HTAP IN MIND

Hopefully it is clear by now that you will need to extend APIs using HTAP. Therefore, any new APIs must be designed with HTAP in mind. But there is more than one style of HTAP to consider.

IMC technologies like Apache Ignite are often used for real-time analytics as part of a transaction, for streaming analytics, or even stand-alone analytics independent of any events or transactions that could be triggered at any time. All three can be considered HTAP. In general, HTAP means you run your analytics in the same place as the transactions.

Some people refer to in-process HTAP, the ability to execute analytics and automation during the transaction or operation/ interaction to influence that interaction. This is the true purpose of HTAP. Otherwise you could build a separate real-time streaming pipeline and deliver relatively low latency real-time analytics. Examples of in-process HTAP including fraud detection, personalization and real-time cross-selling/ up-selling.

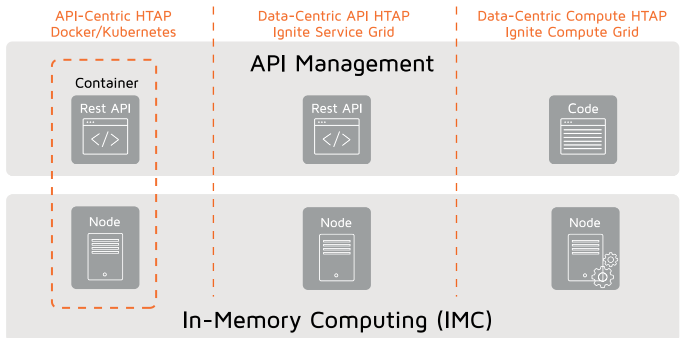


Figure 5. API-Centric HTAP

HTAP can vary based on how you need to collocate the transactions and analytics, and the resulting deployment model that is required. There are at least three different models:

- API-centric HTAP where you need to add speed and scale to an API, but it is all about running something as part of the API. In this case the API should determine the deployment model. This is usually Docker and Kubernetes.
- Data-centric API HTAP where you need to run analytics built around the data distribution and scalability or data lifecycle, but you need to access it as an API. In this case the data grid should control the API deployment lifecycle. This may or may not be the same deployment model as existing APIs.
- Data-centric compute HTAP which is raw compute. Machine and deep learning is an example. It may not be "an API" a consumer invokes. This may be continuously running model training, testing and execution in parallel in the background.

Make sure to look ahead and consider which models you need. In general many companies only use API-centric HTAP for their APIs, and the compute grid for collocated computing at scale for streaming analytics and machine and deep learning. But the service grid pattern is used as well.

BEST PRACTICE 7: KEEP PRIORITIZING REAL-TIME PROJECTS BASED ON A GREATER PLAN

As mentioned throughout, plan to use the same in-memory computing technology across existing and new applications, APIs and analytics. 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.

Remember the same guidelines discussed in Part 1 of this eBook series Building an In-Memory Computing Foundation and Roadmap:

1. 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, perhaps even outside the applications.
2. Review any database purchases or re-platforming projects that rip out and replace an existing database. New purchases 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 coming from new APIs and apps.
3. 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.
4. Consider whether to add speed and scale to existing systems before building new APIs. Existing systems need to be able to handle the additional loads. But as mentioned, it is also possible the best place to add the speed and scale is only with the APIs, not the existing applications.
5. 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).
6. 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 TOWARDS REAL-TIME/ STREAMING ANALYTICS AND DECISION AUTOMATION

The focus of this eBook was on how to add speed and scale to new apps and APIs to support mainstream digital transformation and other major initiatives. The next eBooks are guides for each type of project you might tackle in your journey. What often comes after implementing APIs is adding real-time analytics or decision automation using newer technologies such as machine and deep learning.

Different project types include:

- Building an In-Memory Computing Foundation and Roadmap
- Adding Speed and Scale to Existing Applications
- Adding speed and scale to new apps and APIs (current eBook)
- Building real-time and streaming analytics
- Implementing HTAP
- Leveraging and integrating with cloud services
- Adding machine and deep learning to help automate decisions

For more information on any of these eBooks, related webinars, or customer examples, visit the In-Memory Computing Best Practices for Digital Transformation Resource Center.

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.