

1. Parallel Computing Overview, Parallel Computing Framework and Architecture

Parallel Computing :

It is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource that has been applied to work is working at the same time.

Advantages of Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of the hardware

Applications of Parallel Computing:

- Databases and Data mining.
- Real-time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality, and virtual reality.

Limitations of Parallel Computing:

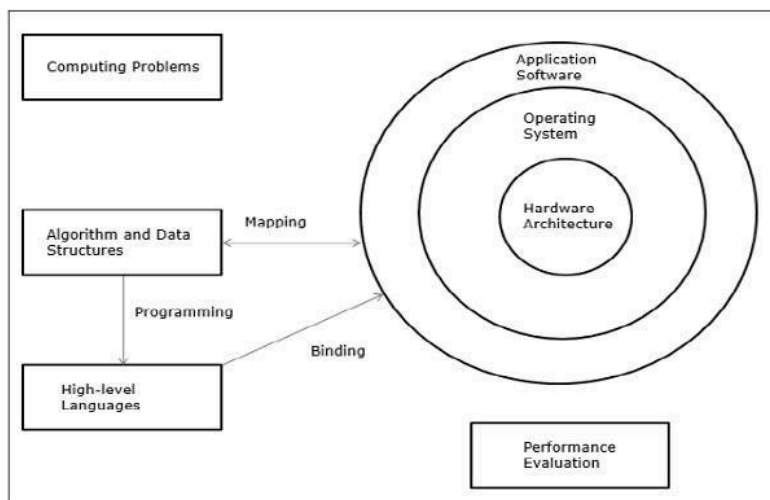
- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
- The algorithms must be managed in such a way that they can be handled in a parallel mechanism.
- The algorithms or programs must have low coupling and high cohesion. But it's difficult to create such programs.
- More technically skilled and expert programmers can code a parallelism-based program well.

Parallel Computer Architecture - Models

Parallel processing has been developed as an effective technology in modern computers to meet the demand for higher performance, lower cost and accurate results in real-life applications. Concurrent events are common in today's computers due to the practice of multiprogramming, multiprocessing, or multicomputing.

Modern computers have powerful and extensive software packages. To analyze the development of the performance of computers, first we have to understand the basic development of hardware and software.

- **Computer Development Milestones** – There is two major stages of development of computer - **mechanical** or **electromechanical** parts. Modern computers evolved after the introduction of electronic components. High mobility electrons in electronic computers replaced the operational parts in mechanical computers. For information transmission, electric signal which travels almost at the speed of a light replaced mechanical gears or levers.
- **Elements of Modern computers** – A modern computer system consists of computer hardware, instruction sets, application programs, system software and user interface.



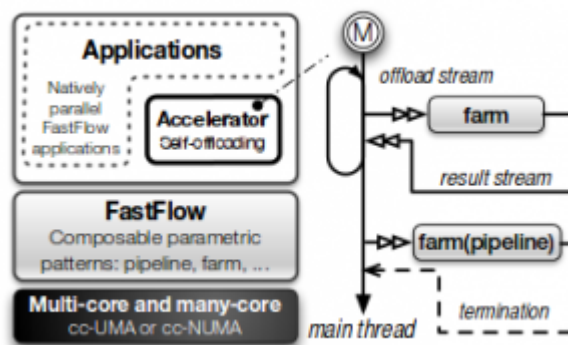
The computing problems are categorized as numerical computing, logical reasoning, and transaction processing. Some complex problems may need the combination of all the three processing modes.

- **Evolution of Computer Architecture** – In last four decades, computer architecture has gone through revolutionary changes. We started with Von Neumann architecture and now we have multicomputers and multiprocessors.
- **Performance of a computer system** – Performance of a computer system depends both on machine capability and program behavior. Machine capability can be improved with better hardware technology, advanced architectural features and efficient resource management. Program behavior is unpredictable as it is dependent on application and run-time conditions

Parallel Computing Framework

Structured Parallel Programming simplifies the development of parallel applications by substantially raising the level of abstraction presented to the parallel programmer. The programmer concentrates on computational aspects, having only an abstract high-level view of the parallel program, while all the most critical implementation choices are left to the programming tool and its run-time support.

- In the years, our group has been actively involved in the development and maintenance of various structured parallel programming frameworks (e.g. P3L, Muskel, Assist). Currently we are involved in the development and maintenance of [Fastflow](#), a C++based parallel programming framework targeting multicores, GPUs and (partially) distributed systems. It is based on **algorithmic skeletons/design patterns** and is composed of a stack of layers that progressively abstracts out the programming of shared-memory parallel applications. FastFlow was used as the main run-time framework in the FP7 projects ParaPhrase and REPARA and also in the H2020 project RePhrase.



2. Business Challenges of Parallel Computing, Business Benefits of Parallel Computing

Parallel Computing for Business Applications

Business applications are very different from engineering or scientific applications. They have the following traits:

- They process transactions.
- They process tasks with mixed workloads. Quite often you can't predict the size of each task, or what the processing requirements might be.
- The workload varies widely throughout a given time period. It might even change from day to day, or from one hour to the next.

- They often have requirements that defy the very concept of performing multiple tasks in parallel. For example, first in/first out (FIFO) transactions (which are very commonly used) must be done in an exact, ordered sequence.
- They almost always use a database or other centralized resource that bottlenecks and caps off transaction throughput.

Up to now, research on parallel computing concentrated mostly on mechanical solutions with limited scalability, or on grid-based scientific and engineering applications that lie outside the business domain. What we need is a new, simpler way to implement parallel computing for businesses. This new approach must support the following requirements:

- It must handle a wide variety of business application needs.
- It must provide ultimate scalability.
- It must maintain critical business requirements.
- It must be easy to implement by corporate IT developers.

1. Parallel computing models the real world

The world around us isn't serial. Things don't happen one at a time, waiting for one event to finish before the next one starts. To crunch numbers on data points in weather, traffic, finance, industry, agriculture, oceans, ice caps, and healthcare, we need parallel computers.

2. Saves time

Serial computing forces fast processors to do things inefficiently. It's like using a Ferrari to drive 20 oranges from Maine to Boston, one at a time. No matter how fast that car can travel, it's inefficient compared to grouping the deliveries into one trip.

3. Saves money

By saving time, parallel computing makes things cheaper. The more efficient use of resources may seem negligible on a small scale. But when we scale up a system to billions of operations - bank software, for example - we see massive cost savings.

4. Solve more complex or larger problems

Computing is maturing. With AI and big data, a single web app may process millions of transactions every second. Plus, "grand challenges" like securing cyberspace or making solar

energy affordable will require petaFLOPS of computing resources [5]. We'll get there faster with parallel computing.

5. Leverage remote resources

Human beings create 2.5 quintillion bytes of information per day [6]. That's the number 25 with 29 zeros. We can't possibly crunch those numbers. Or can we? With parallel processing, multiple computers with several cores each can sift through many times more real-time data than serial computers working on their own.

3. Applications of Parallel Computing

Applications of Parallel Computing:

Historically, parallel computing has been considered to be "the high end of computing", and has been used to model difficult scientific and engineering problems found in the real world.

Some examples:

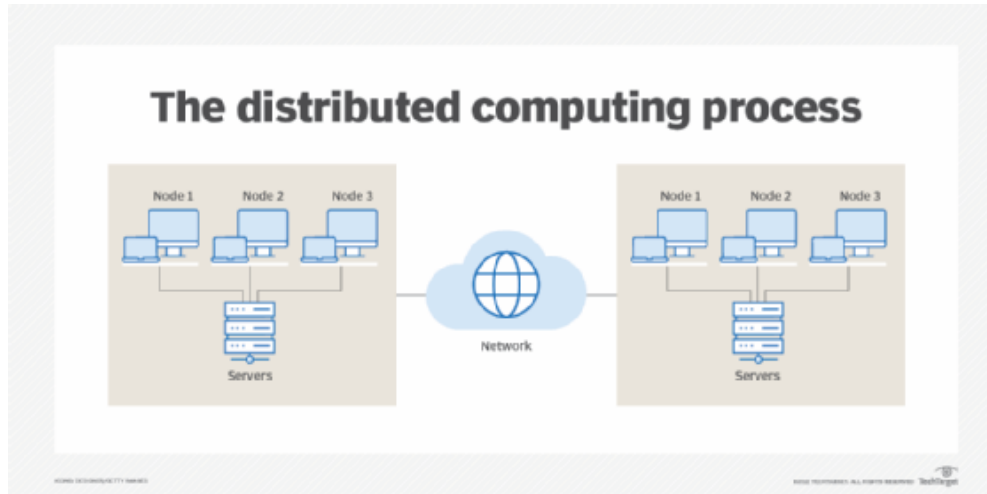
- Atmosphere, Earth, Environment, Space Weather
- Physics - applied, nuclear, particle, condensed matter, high pressure, fusion, photonics
- Bioscience, Biotechnology, Genetics
- Chemistry, Molecular Sciences
- Geology, Seismology
- Mechanical and Aerospace Engineering
- Electrical Engineering, Circuit Design, Microelectronics
- Computer Science, Mathematics

Today, commercial applications provide an equal or greater driving force in the development of faster computers. These applications require the processing of large amounts of data in sophisticated ways. For example:

- Databases, data mining
- Oil exploration
- Web search engines, web-based business services
- Medical imaging and diagnosis
- Pharmaceutical design
- Management of national and multi-national corporations
- Financial and economic modeling
- Advanced graphics and virtual reality, particularly in the entertainment industry
- Networked video and multi-media technologies

- Collaborative work environments

4. Role of Distributed Computing



An example of how networks, servers and computers are structured in distributed computing.

Benefits of distributed computing

Distributed computing includes the following benefits:

- **Performance.** Distributed computing can help improve performance by having each computer in a cluster handle different parts of a task simultaneously.
- **Scalability.** Distributed computing clusters are scalable by adding new hardware when needed.
- **Resilience and redundancy.** Multiple computers can provide the same services. This way, if one machine isn't available, others can fill in for the service. Likewise, if two machines that perform the same service are in different data centers and one data center goes down, an organization can still operate.
- **Cost-effectiveness.** Distributed computing can use low-cost, off-the-shelf hardware.
- **Efficiency.** Complex requests can be broken down into smaller pieces and distributed among different systems. This way, the request is simplified and worked on as a form of [parallel computing](#), reducing the time needed to compute requests.

- **Distributed applications.** Unlike traditional applications that run on a single system, [distributed applications](#) run on multiple systems simultaneously.

5. Role of Parallel Computing in AI

In parallel processing, a complex task is split into multiple smaller tasks that are more proportionate in processing requirements, size, and number to the processing units on hand. After the division, each processor starts working on their part of the task independently from one another, with the exception of constant communication through the software in order to stay up to date regarding the state of other tasks.

After the processing of all the parts is complete—whether the number of processors was the same as the number of tasks and they ended in sync or took turns—the result is a fully-processed program segment. Parallel process can be characterized as being either fine-grained or coarse-grained. In fine-grained parallelism, tasks communicate with each other several times per second to provide results in real or near-real time. Coarse-grained parallel processes are slower because they communicate less often.

Parallel Processing Architectures

- [Multicore](#) -- a device's integrated circuit (IC) has two or more separate processing cores, each of which can execute program instructions in parallel. Multi-core architectures can be homogeneous and have identical cores, or heterogeneous and have cores that are not identical.
- [Symmetric](#) -- two or more independent, homogeneous processors are controlled by a single operating system instance that treats all processors equally.
- [Distributed](#) -- processors are located on different networked devices that communicate and coordinate actions through HTTP or [message queues](#).
- [Massively parallel computing](#) -- numerous computer processors simultaneously execute a set of computations in parallel.
- [Loosely Coupled Multiprocessing](#) --- individual processors are configured with their own memory and are capable of executing some user and operating system instructions independently of each other.

Types of Parallel Processing

Currently, there are mainly three types of parallel processing, categorized according to the source of the data and the type of processing it goes through.

- [Multiple Instruction Multiple Data \(MIMD\) Processing](#)– each set of processors runs data coming from different sources, following instructions and algorithms from the data source.
- Multiple Instruction Single Data (MISD) Processing – multiple processors receive the same data sets but are instructed to process them differently to produce more diverse results.
- Single Instruction Multiple Data (SIMD) Processing – multiple processors run the identical task using the same instructions to validate results.

Parallel vs Serial vs Concurrent Processing

Parallel processing and concurrent processing are often confused with one another because they both work on processing multiple tasks at the same time. Concurrent processing is similar to real-life multitasking, but tasks don't need to be completed at the same time.

Current and Future Applications of Parallel Processing

Parallel processing has a role in countless achievements ranging from scientific discoveries, like [building complex computer models](#) to map out how mass orbits around a black hole, to predictions that aid the economy. In 2019, [researchers at the University of Illinois](#) used parallel to help the US Department of Agriculture more accurately predict yield-boosting crop traits by incorporating more data than before and processing it in record times.

Parallel processing plays a major role in [developing and implementing machine learning algorithms](#) and AI programs because it allows them to run faster, process more data points and generate more accurate (and useful) insights.

6. Unit 7: Overview of Serverless Computing

What is serverless?

Serverless is a cloud computing application development and execution model that enables developers to build and run application code without provisioning or managing servers or backend infrastructure.

Serverless lets developers put all their focus into writing the best front-end application code and business logic they can. All developers need to do is write their application code and deploy it to containers managed by a cloud service provider. The cloud provider handles the rest, provisioning the cloud infrastructure required to run the code and scaling the infrastructure up and down on demand as needed. The cloud provider is also responsible for

all routine infrastructure management and maintenance such as operating system updates and patches, security management, capacity planning, system monitoring and more.

7. Serverless Computing Overview

Serverless Computing Explained

Serverless is a cloud computing model where the service provider dynamically allocates the exact amount of resources needed on-demand. The main advantage of serverless computing is the provider only charges you for the exact machine resources needed, whereas in cloud computing, you pre-purchase units of bandwidth and resources are dedicated to you at all times, whether they're in use.

While the name serverless is not the elimination of servers from [distributed applications](#). The name serverless refers to the fact that the provider manages, provisions, and maintains backend services on an as-used basis. A company that gets computing resources from a serverless vendor is charged only for the resources their application actually consumes. Hence the client does not have to reserve and pay for a fixed amount of bandwidth or a number of servers, as the service is auto-scaling.

How Serverless Works

Serverless architecture relies on functions, or more specifically functions-as-a-service (FaaS). It is a service model that allows developers to run code directly in the cloud without the need to build packages or maintain any infrastructure. Applications are broken up into individual functions that can be invoked and scaled individually. Listen to the [podcast](#), or read the [blog](#) to learn more about serverless stream processing with Apache Kafka®.

Benefits of Serverless

Full scalability – Developers do not have to upload code to servers or do any backend configuration in order to release a working version of an application, nor administrators to upgrade existing servers or add servers. Scalability is automatic and you don't have to worry about the need to provision the underlying infrastructure.

Easy To Deploy – By using a serverless solution you will get a faster deployment of resources than you would by using other cloud computing models. Instead of weeks and months to deploy an app, you can do it within minutes. The main reason behind such a quick and simplified deployment is that you don't have to take care of the infrastructure.

No infrastructure requirements – Since developers will be literally using someone else's computer to execute their serverless functions, there will be no infrastructure to maintain.

No server or software management – Even though 'serverless' computing actually takes place on servers, developers are not required to deal with the servers.

Fault-tolerant – Developers are not responsible for ensuring the fault tolerance of the serverless architecture. The cloud provider provides the IT infrastructure—compute, storage, networking, and databases that will be automatically allocated to account for almost any kind of failure.

Lower cost – The costs related to serverless computing are minimal compared to other cloud services, developers are only charged for the duration of execution and not by the server unit. The serverless model enables customers to avoid the costs associated with operating a server such as access authorization, presence detection, security, image processing, and other costs.

No upfront payments – Since you will only be paying for the running code there will be no upfront investment.

Reduced Latency – Using serverless functions will bring minimal latency to end-users. As the serverless functions do not operate from a designated origin server, there will not be just one location that an end user's traffic has to be directed to. In this case, all of the cloud provider's data centers will be used and the function will be executed by the nearest server, hence greatly reducing the response time.

Simplified backend code – As the app is not hosted on an origin server, its code can be run from anywhere.

8. Serverless Computing Architecture and Framework

What is Serverless Architecture?

Serverless architecture is an approach to software design that allows developers to build and run services without having to manage the underlying infrastructure. Developers can write and deploy code, while a cloud provider provisions servers to run their applications, databases, and storage systems at any scale. In this article, we'll cover how serverless

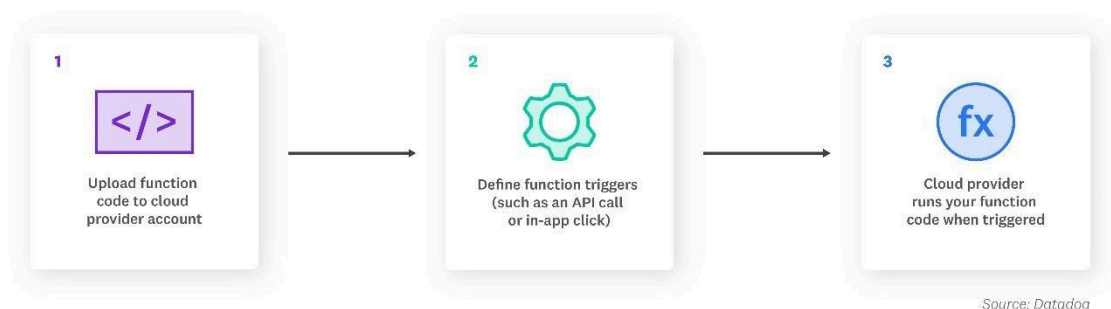
architecture works, the benefits and drawbacks of using it, and some tools that can help you go serverless.

How Serverless Architecture Works

Servers allow users to communicate with an application and access its business logic, but managing servers takes considerable time and resources. Teams have to maintain the server hardware, take care of software and security updates, and create backups in case of failure. By adopting serverless architecture, developers can offload these responsibilities to a third-party provider, enabling them to focus on writing application code.

One of the most popular serverless architectures is Function as a Service (FaaS), where developers write their application code as a set of discrete functions. Each function will perform a specific task when triggered by an event, such as an incoming email or an HTTP request. After the customary stages of testing, developers then deploy their functions, along with their triggers, to a cloud provider account. When a function is invoked, the cloud provider either executes the function on a running server, or, if there is no server currently running, it spins up a new server to execute the function. This execution process is abstracted away from the view of developers, who focus on writing and deploying the application code.

How Serverless Functions Work



Function as a Service (FaaS), a popular type of serverless architecture, allows developers to focus on writing application code.

While serverless architecture has been around for more than a decade, Amazon introduced the first mainstream FaaS platform, AWS Lambda, in 2014. Currently, a majority of developers still use AWS Lambda to build serverless applications, but Google and Microsoft

have their own FaaS offerings as well, called Google Cloud Functions (GCF) and Azure Functions respectively.

Fundamental Concepts in Serverless Architecture

Although serverless architecture eliminates the need for server management, there's still a steep learning curve, especially if you're chaining multiple functions together to create complex workflows in an application. It can therefore be helpful to familiarize yourself with these fundamental serverless terms:

- **Invocation**

A single function execution.

- **Duration**

The time it takes for a serverless function to execute.

- **Cold Start**

The latency that occurs when a function is triggered for the first time or after a period of inactivity.

- **Concurrency Limit**

The number of function instances that can run simultaneously in one region, as determined by the cloud provider. A function will be throttled if it exceeds this limit.

- **Timeout**

The amount of time that a cloud provider allows a function to run before terminating it. Most providers set a default timeout and a maximum timeout.

Keep in mind that each cloud provider may use different terminology and set unique limits on serverless functions, but the list above defines the basic concepts.

9. Business Challenges of Serverless Computing

There has been a significant increase in serverless adoption in recent years, with nearly 40 percent of companies worldwide using it in some form. Small startups and global corporations alike are leveraging serverless architectures for the following reasons:

- **Cost**

Cloud providers charge you on a per-invocation basis, so you're not paying for unused servers or virtual machines.

- **Scalability**

Function instances are automatically created or removed in response to traffic variations, within the boundaries of concurrency limits.

- **Productivity**

Engineers who use serverless can simply deploy their code without having to manage any servers, which helps accelerate delivery cycles and rapidly scale company operations.

10. Business Benefits of Serverless Computing

There are also some challenges associated with serverless architectures:

- **Loss of Control**

In serverless environments, you lack control over the software stack that your code runs on. If a hardware fault, data center outage, or other issue impacts one of your servers, you're dependent on a cloud provider to fix it.

- **Security**

A cloud provider may run code from several of their customers on the same server at the same time. If the shared server isn't configured properly, your application data could be exposed.

- **Performance Impact**

Cold starts are common in serverless environments, adding several seconds of latency to code execution when functions are invoked after a period of inactivity.

- **Testing**

Developers can run unit tests on function code, but integration tests, which evaluate how frontend and backend components interact, are difficult to perform in a serverless environment.

- **Vendor Lock-In**

Large cloud providers like AWS offer several services—such as databases, messaging queues, and APIs—that you can use in harmony to run serverless applications. Although it's possible to mix and match elements from different vendors, services from a single provider are designed to integrate most seamlessly.

Companies that want to minimize their go-to-market time and build scalable, lightweight applications can benefit greatly from serverless. But if your applications involve a large number of continuous, long-running processes, virtual machines or containers may be the better choice. In a hybrid infrastructure, developers may utilize containers or virtual machines to handle the bulk of requests but hand off certain short-running tasks, such as database storage, to serverless functions.

11. Applications of Serverless Computing

Serverless computing has many practical uses and applications. Let's explore a partial list of instances where it fits perfectly.

- **Application Programming Interface (API) Back-Ends.**

You can turn any function or action on a serverless platform into an HTTP endpoint suitable for web clients. These actions, when enabled for the web, are called web actions. You can assemble web actions into a full-featured API with an API Gateway, bringing additional security and other advantages.

- **Data Processing.**

It is ideal for working with structured audio, image, text, and video data. You can also use it to work around tasks like cleansing, enriching, transforming, and validating data.

Additionally, serverless processing is perfect for audio normalization, image processing, optical character recognition (OCR), PDF processing, and video transcoding.

- **Microservices Support.**

Serverless computing often supports [microservice](#) architectures. Microservices focuses on creating small services that perform a single task and use APIs to communicate. Although microservices can be built and run using containers or [Platforms-as-a-Service \(PaaS\)](#), this model has made remarkable inroads in the process.

- **Massive Parallel “Map” and Compute Operations.**

Each parallelizable task results in a single action invocation, making serverless computing a good fit. As a result, you can easily perform tasks like Map/reduce operations, data processing and searches, and hyperparameter tuning.

- **Everyday Applications.**

Increasing numbers of businesses and industries are adopting serverless computing. This chart, supplied courtesy of [IBM](#), breaks down this model into selected applications.

12. Business Driven for Adopting Serverless Computing

So, what are the specific [benefits of opting for serverless](#)? We explore them below.

Pricing

One of the main benefits of going serverless is definitely the pricing.

For mid-sized businesses, the burden of not only provisioning servers but maintaining them 24.7, proved to be a giant hole in a pocket.

“The cost of a serverless design is execution-based. This means that you are charged for the number of executions. You pay for what you use, which seems a lot fairer. As a mid-sized company, it’s only wise that you look for budget-friendly options. In this case, serverless architecture is the clear winner.” — Emma Kovacs, Senior Tech Writer at [IsAccurate](#).

3rd party dependencies

The majority, maybe even all, of your projects, have external dependencies. You’ll know, these rely on libraries that aren’t built into the language or framework that you use.

These libraries can be extremely dense – with cryptography, image processing, and other accessories.

The reason we’re mentioning this is that, as a mid-sized business, serverless still proves to be better than traditional servers.

Applications with few dependencies, it’s a straightforward choice.

No server management necessary

Even though serverless computing does technically take the place of a server, you never have to deal with it yourself.

They are managed by the vendor and reduce the necessary investment.

Obviously, this helps with the pricing point – as expenses are much lower. It also frees up time, allowing you to focus on other tasks and pieces of work.

Lowering employee downtime is, as you know, a highly important aspect.

Also, as your business grows, you don't have to update your server. It automatically does this. This means that you can create and expand your applications, without any kind of constrained by server capacity.

Environments

With mid-sized businesses, nothing is ever really specific. In fact, this is the same for any type of business.

You never know when you'll have to relocate or change your office. Setting up in different environments is super easy, given that its pay per execution.

With serverless architecture, you don't have to set up staging, production machines or dev.

With traditional servers, you risk losing a large amount of your data, especially when you change environments.

Again, this is why medium-sized companies should opt for a serverless design.

Quick updates

With a serverless infrastructure, you don't need to do any kind of backend configuration. This means that no coding or work is had to be done, for you to release a working version of an application.

As it's all done by a third-party, updates are all done, without you lifting a finger.

As a mid-sized business, you might not even have somebody in your team that focuses on updating servers. With a serverless design, this eradicates the need.

Obviously, with updates being easy and fast, this reduces downtime. It's not necessary to make changes to the entire applications – instead, you can update one function at a time.

Scale

Serverless architectures are intrinsically scalable. Imagine if your company could increase the range of their server whenever they need to.

Likewise, your server will decrease when applications reduce.

Applications that are built with a serverless infrastructure will scale automatically – for instance, when the user base grows or as usage increases.

As a result of this, a serverless application will be able to handle an obscure number of requests, just as well as it can prepare an individual request from a user. This is one of the main benefits of this type of server.

Automatic scaling is really the future of servers – it allows your business to grow, without any worries is always a great thing.

Functions as a service (FaaS)

FaaS is an implementation of Serverless design. Engineers deploy an individual function or a piece of business logic, to the architecture.

They start with milliseconds, and individual process requests within a 300-second timeout imposed by most cloud vendors.

FaaS allows complete management of servers, invocation-based billing, and event-driven and instantly scalable designs. FaaS are similar to functions that you might have come across when writing programming.

They're small, separate, units of logic that take input arguments. They operate on input and return the result.

A medium-sized business might just find this incredibly appealing.

Stateless

When you opt for serverless architecture, everything is stateless. This is great for backup purposes and also saving essential files.

For instance, if you lose some of your work on your computer, you can always check your server, and never risk losing your work.

You can also save a file to your desk, and expect it to be there at the next function.

It's an extra bit of safety.

Happier customers

Of course, this depends on the type of company that you have and other variables. However, a serverless architecture can lead to more satisfied customers, especially when executed right. Customers can access new features instantly, without having to wait for updates or small fixes. That's incredibly important, and probably one of the main reasons why big companies, [like Netflix](#), have decided to go serverless.

You don't have to worry about infrastructure, you can release features and logic quickly and easily, that will improve the user experience.

As a company, you should always be looking for new ways to improve customer service, and serverless might just do this for you.

It's greener

A focus for many companies and individuals at the moment is how to reduce their carbon footprint. It's all with good reason, too.

Businesses that host data centers of their own servers need to run their servers continually. This means that you need a huge amount of data and physical resource to build these types of centers. Have you ever considered the energy required to do this?

"Serverless architecture means that you can buy servers on-demand, only when you need it. This reduces the number of resources you need – plus cloud data is known to be more eco-friendly than traditional forms." — James Overbach, tech writer and editor, [WoWGrade](#).

Improved latency

With the serverless design, you can handle users from every corner of the world. This is due to the access points that you can have on a global scale.

This benefit means that you can scale without affecting the performance of your application. Again, this also improves customer service and enhances how many people can use your applications and sites.

You have more time for UX

If your application is customer-focused, you need to remember that your customers do not care about the infrastructure. They just want a simple, secure, and straightforward site to use.

They never see the coding, the graphics, or the work that went into your site. The only thing the customer will see is the interface.

This means that you have more time to focus on keeping your customer happy. Such as allocating resources to certain elements and making necessary changes.

Gain feedback from your users, and focus on making them happy.

Quick to deploy

If you need to create an app, and fast, serverless is your best option.

Traditional servers require weeks, even months to deploy an app.

With serverless, you can do it in a matter of days or hours.

This mainly falls with the fact that you don't have to concern yourself with any kind of infrastructure. You can just focus on the code and release it immediately.

Also, as we mentioned, Scalability is automatic, and there's no need to worry about any kind of provisioning needs.

13. Role of Serverless Computing in AI

The most important benefit of using serverless computing in AI is hassle-free server management. The service consumers do not hold any responsibility to maintain or administer the server, and it's all the service vendor's responsibility. Furthermore, serverless computing offers built-in services provided by Cloud Service Providers, which are fully managed, supports elasticity, fault-tolerant, and offer enterprise-level integrated security along with automatic scaling. Hence, it is more productive concerning developers' activity.

Apart from that, there are the following benefits of using serverless computing in AI development –

Auto-scalability

The serverless architecture leverages flexibility towards scalability and growth. Whether it is a small or big business, the cloud allows users to use what they need. So, they can scale up without worrying about complex and time-consuming data migrations. Furthermore, if the application is deployed correctly, there is no issue with scaling it, even if the workload increases.

Focus on core tasks

With serverless computing, there is no headache of managing servers. Hence, data scientists or machine learning engineers can only focus on AI development and deploying AI models.

Pay only for the functions you use

In traditional application deployment models, you need to pay a recurring and fixed cost for computing resources. This could be more than the amount of computing work is actually being performed by the server. However, in serverless computing deployment, the user needs to pay only for the service. It is only for the number of executions and their duration during development and deployment.

Less interdependency

Machine learning models are treated as serverless functions that you can invoke, update, and delete as required. This can be done without interrupting the rest of the system. Furthermore, different teams can work independently to develop, deploy, and scale their individual AI models.

Abstraction from the users

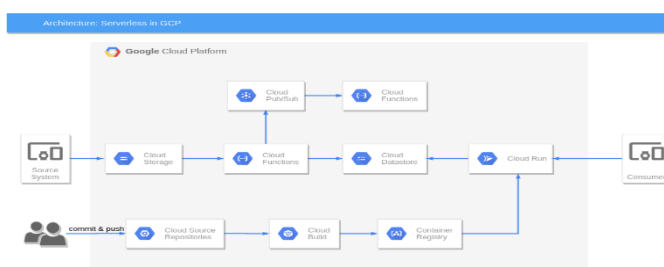
The machine learning model is exposed as a service through API Gateway to the users in the serverless computing model. Hence, you can easily decentralize the backend as serverless creates an abstraction to the users. Simultaneously, this architecture isolates failure on a per-model level while hiding every implementation detail from the final user.

High availability and fault tolerance

Serverless applications have built-in features of availability and fault tolerance by default. Hence, while modeling AI applications, there is no need to architect for these capabilities. Furthermore, Serverless computing facilitates a simpler approach to artificial intelligence as it removes the overload of server maintenance from developers and data scientists.

14. Serverless on Google Cloud

The full stack app using serverless services on GCP.



Cloud Run

Build applications in your favorite language, dependencies, and tools, and deploy them in seconds. [Cloud Run](#) abstracts away all infrastructure management by automatically scaling up and down from zero almost instantaneously—depending on traffic. Cloud Run only charges you for the exact resources you use.

Cloud Functions

Develop faster by writing and running small code snippets that respond to events. Use [Cloud Functions](#) to connect with Google Cloud or third-party cloud services via triggers to streamline challenging orchestration problems. Run functions across multiple environments (local development environment, on-premises, Cloud Run, and other Knative-based serverless environments) and prevent lock-in.

Service integration

When integrating services, it's all too easy to introduce tight coupling, which grows brittle, slow, and difficult to debug over time. Let our [service integration](#) products take care of the connective tissue, so you can do what you do best: building brilliant applications.

What does GCP Serverless Computing Provide?

- No need for Managing server
- Fully Managed Security
- Pay only for what you use
- Flexible environment for developer
- Easy to understand interface
- Various types of options regarding Full-Stack Serverless
- GCP provides with everything one needs to build an end to end serverless application
- It helps you entirely focus your attention to the code only rest everything is taken care of by GCP
- Deployment is quite fast, and you can go global as soon as you complete every formality regarding the system.

15. Serverless on Amazon AWS

Choose from a variety of AWS, open source, and third-party web frameworks that simplify serverless application development and deployment. These popular frameworks are supported across multiple languages.

- [AWS Serverless Application Model \(AWS SAM\)](#) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings.
- [AWS Cloud Development Kit \(AWS CDK\)](#) is an open source software development framework to define your cloud application resources using familiar programming languages.
- [Serverless Framework](#) - The Serverless Framework consists of an open source CLI and a hosted dashboard. Together, they provide you with full serverless application lifecycle management.
- [Chalice](#) is a framework for writing serverless apps in Python. It allows you to quickly create and deploy applications that use AWS Lambda.
- [Arc.codes](#) provides everything you need to build massively scalable serverless apps with low code, clear and terse config, and zero ceremony.
- [Claudia.js](#) makes it easy to deploy Node.js projects to AWS Lambda and API Gateway.

16. Serverless on Microsoft Azure

Azure - Introduction

The Serverless Framework helps you develop and deploy serverless applications using Azure Functions. It's a CLI that offers structure, automation and best practices for deployment of both code and infrastructure, allowing you to focus on building sophisticated, event-driven, serverless architectures, comprised of [Functions](#) and [Events](#).

Core Concepts

Here are the Framework's main concepts and how they pertain to Azure Functions...

Functions

A Function is an [Azure Function](#). It's merely code, deployed in the cloud, that is most often written to perform a single job such as:

- *Saving a user to the database*
- *Processing a file in a database*
- *Performing a scheduled task*

You can perform multiple jobs in your code, but we don't recommend doing that without good reason. Separation of concerns is best and the Framework is designed to help you easily develop and deploy Functions, as well as manage lots of them.

Events

Anything that triggers an Azure Function to execute is regarded by the Framework as an **Event**. Events are platform events on Azure Functions such as:

- *An HTTP Trigger (e.g., for a REST API)*
- *A scheduled timer (e.g., run every 5 minutes)*
- *A Service Bus Queue trigger (e.g. a workitem from another Function)*
- *An IoT/Event Hub message (e.g., a message from a device or service)*
- *A Webhook fires (e.g., Github project update)*
- *And more...*

When you define an event for your Azure Function in the Serverless Framework, the Framework will automatically translate this into [bindings](#) needed for that event and configure your functions to listen to it.

17. Unit 8: In-Memory and Real Time Computing with Python

In-Memory Computing provides super-fast performance (thousands of times faster) and scale of never-ending quantities of data, and simplifies access to increasing numbers of data sources. By storing data in RAM and processing it in parallel, it supplies real-time insights that enable businesses to deliver immediate actions and responses. That's what makes it ideal for implementation in transactional and analytical applications sharing the same data infrastructure (point-of-decision [HTAP](#)) and transactional analytics guided by real-time analytics (in-process HTAP).

What is real-time computing?

Real-time computing, also known as reactive computing, is used to describe a computer system that reacts to events by performing tasks within a specific time interval. The time frame for these actions to be carried out is in the order of milliseconds.

Meeting these specific time intervals are key to the effectiveness of real-time systems. If a real-time system fails to carry out actions in time, it can result in the system being inadequate or useless.

18. In-memory Computing Overview

In-memory computing (or in-memory computation) is a technique based on RAM data storage and indexing, which proposed by the MIT research group, and its main purpose is to accelerate the convolution calculation. We know that convolution calculations can be expanded into weighted accumulation calculations. From another perspective, it is actually a weighted average of multiple numbers. Therefore, the circuit realizes the weighted average of the charge domain. The weight (1-bit) is stored in SRAM, and the input data (7-bit digital signal) becomes an analog signal through the DAC. According to the corresponding weight in the SRAM, the output is multiplied by 1 or -1 in the analog domain, which averaged in the analog domain, and finally read out by the ADC as a digital signal. Specifically, since the weight of the multiplication is 1-bit (1 or -1), it can be controlled by using a switch and a differential line simply. If the weight is 1, the capacitor on the side of the differential line is charged to the required output value. Otherwise, let the other side of the differential line be charged to this value. As for average, connect several differential lines together in the charge domain.

Of course, there is more than one circuit for in-memory calculation, and the calculation accuracy is not limited to 1-bit. However, we can see the above examples that the core idea of in-memory calculations is generally to convert calculations into weighted calculations. Store the weights in the memory unit, then modifications on the core circuit of the memory (such as the readout circuit) are made. So that the process of reading is like a process in which the input data and weights are multiplied in the analog domain, that is, convolution. Because convolution is a core part of AI and other calculations, in-memory computing can be widely used in such applications. In-memory computing uses analog circuits for calculations, which is the difference compared with traditional digital logic calculations.

In more traditional architectures, there are some multiply-accumulate circuits (MAC) for tensor math, especially the matrix multiplication. These architectures attempt to arrange the MAC in a way that moves weights and activations to the appropriate location. Activations are calculated from the previous neural network layer. Multiplication usually involves activations and weights, both must be moved to the place where multiplies them. In-memory computing makes use of it. Therefore, if the weights are stored in memory, the memory can

access through activations to obtain multiplication and accumulation. The only difference from the actual memory is that the in-memory computing concatenates all word lines at once, instead of decoding the input to get one word line only.

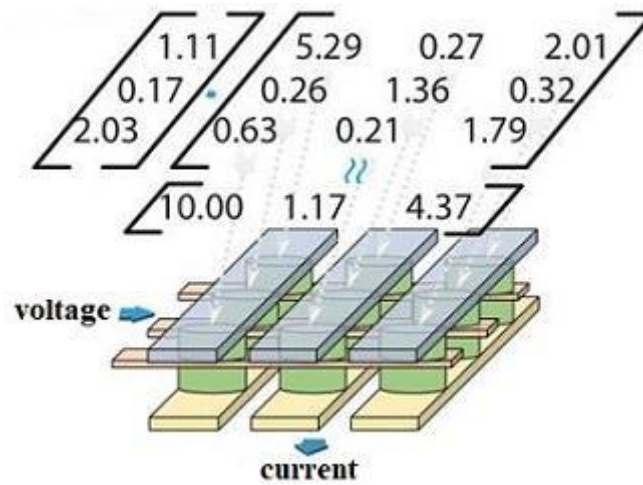


Figure 4. In-memory Computing Diagram

3.2 Four Realization Methods

The attempt is to enter the analog domain and treat the storage unit as an analog unit instead of a digital unit to reduce consumption. We have already got a way to use simulation on the front end of the inference engine. That is in-memory computing. Therefore, we take digital data, using a DAC to convert it to an analog value, and then driving a memory with these analog content to obtain an analog bit-line output, finally using an ADC to convert the result back to a digital format. However, the in-memory computing is still in the exploratory stage, and there are many specific implementation methods to study, currently there are three types: RRAM, Flash, SRAM, and DRAM.

- Based on RRAM

RRAM is the most common method of doing this, because it is easy to use by applying Ohm's law to a series of resistors, but it still has the problem of relying on RRAM. The relationship between programming and resistance is non-linear, which requires more work to be done to make viable calculation circuits in RRAM memory for market. So it is just an idea, and the specific plan is still under study.

- Based on Flash

NOR Flash memory has a more traditional word-line/bit-line structure. It is both resistive and capacitive. Generally, the memory cell is a transistor that is turned on or off. However, if it is partially conductive, it can be used as a resistor. The resistance depends on the amount of

charge on the floating gate of the memory cell (capacitor). When running all the time, the cell will conduct to its maximum capacity. During this process, it does not conduct at all, however, it can be partially programmed. There is a problem is that you cannot precisely control the number of electrons. Moreover, the response to any number will vary with the process and temperature and other variables.

Two companies are studying this method. Microchip owns their memBrain array, thanks to their acquisition of SST, and Mythic is a start-up company dedicated to an inference engine that uses in-memory computing with flash memory. Both companies said that they are using extensive calibration techniques to deal with this change.

Another issue, flash cells will lose electrons over time. Electrons will flow around, which brings up an interesting topic: on this type of memory array, data retention and durability will be like.

From the application point of view, it depends on whether it is to be used in cloud computing or edge inference engine. At the edge, it may perform certain fixed reasoning functions throughout the life cycle of the device. Therefore, if there are enough arrays, then you will load the weights for the first time and don't need to program it anymore (unless you do a update), because the flash memory is non-volatile. Although you still need to move activations, there is no need to move the weights, which will be stored permanently in the array. This would indicate that data durability (number of times the device can be programmed before cumulative damage accelerates electron leakages to an unacceptable level) does not matter, it only need to program once.

In contrast, in cloud applications, the device is likely to be shared as a general-purpose computing resource, so this requires reprogramming for each new application. This means that battery life becomes more important in the cloud. Mythic claims to have a 10K write cycle, and has observed that even if it is reprogrammed every day, it will last for more than 10 years.

If set an analog value for it and use an analog value in the cell, then in theory, each electron is important. However, if there is enough electron migration, you need to refresh the storage unit, or compensate for electrons change in some way. Because the same analog input today will produce different results than a year ago. The calibration circuit can also deal with some aging problems. However, for data retention, Mythic said they do perform regular updates of the weight values stored in flash memory. This will make persistence the main wear-out mechanism rather than data retention. Microchip stated that its data retention time is TBD, but it is likely to reprogram the device quarterly or annually to restore the unit.

So they need a large number of high-quality ADCs and DACs to keep the signal-to-noise ratio (SNR) within a scope of accurate reasoning, which is the focus of designing work. Mythic claims that they provide a novel ADC, so that Microchip can share it to reduce the number required. Although ADC does consume energy, it also greatly reduces overall system consumption.

- Based on SRAM

This idea came from a lecture at Hot Chips at Princeton University. By definition, SRAM is a bistable unit. Therefore, it cannot be in an intermediate state, how should this be handled? And the DACs and ADCs that need to be corrected more over than the array in terms of area and power consumption.

The point of this problem boils down to the question of how to simulate. They explained that this method uses more than one-bit line for calculation. Since the unit is still a digital value, it takes several bit lines to perform a calculation. The bit line can be split, and different groups perform different multiplications. The following figure illustrates it.

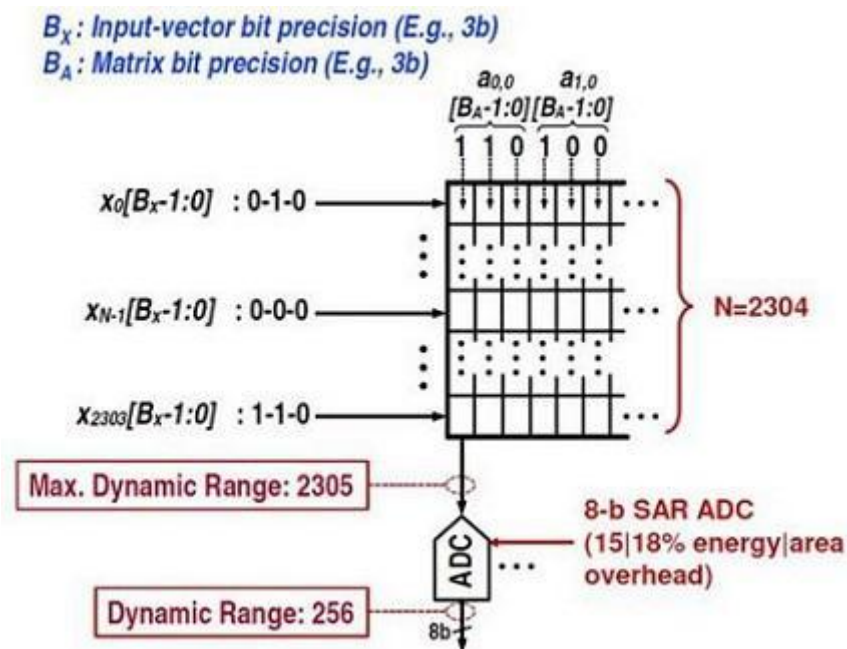


Figure 5. Bit Line

With 8 inputs at a time, so the input vector is sliced and several consecutive multiplications are carried out to obtain the final results. The bit line charge is deposited on the capacitor. When ready to read, the charge is read out and sent to the ADC for conversion back to the digital domain. Their basic unit structure is as follows:

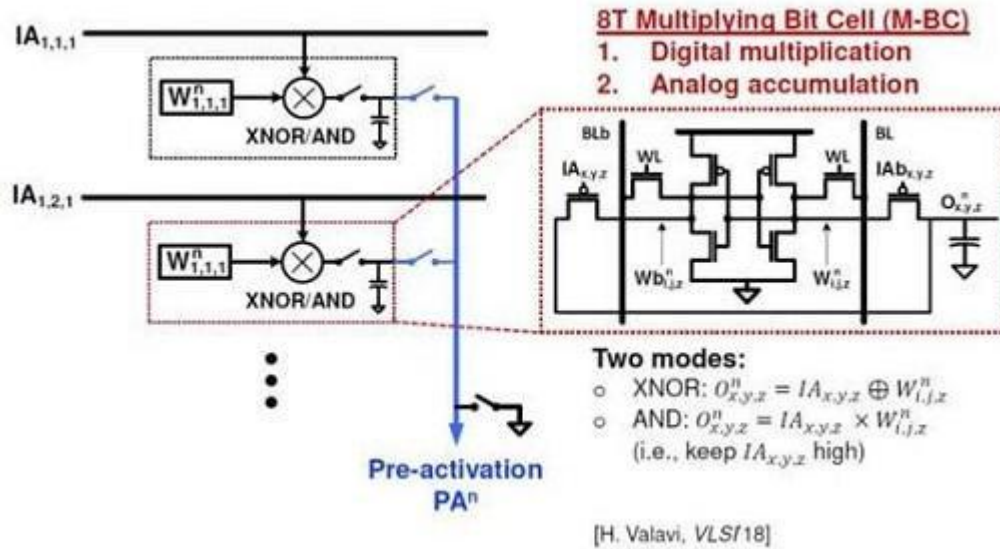


Figure 6. Bit Cell

These capacitors may affect chip size issues, but they said that the metal above the cell can be used. Of course, one cell is now 80% larger than the standard 6T SRAM cell (even without capacitors), but they say that their overall circuit is still much smaller than a required circuit based on standard digital implementation. In addition, since their basic array operations are still in digital form, they are less sensitive to noise and changes, which means their ADCs can be simpler and consume less power.

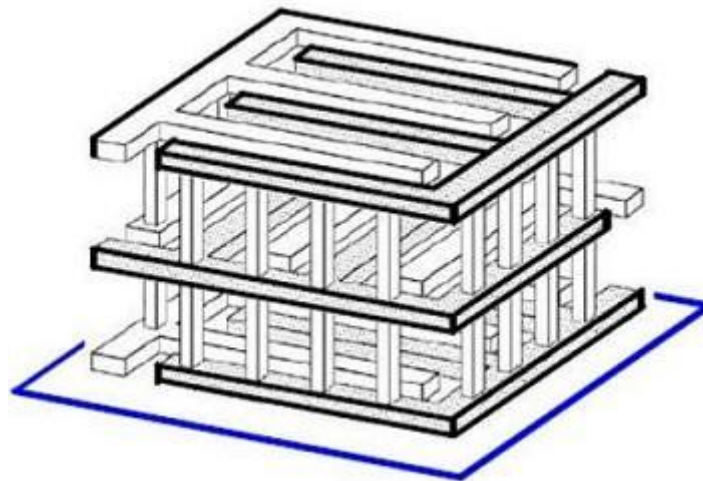


Figure 7. Chip Size

- Based on DRAM

This idea refers to not using a lot of power to obtain DRAM content, and in some way incorporate calculations into the CPU or other computing structures and directly run it on the DRAM die, which is what UPMEM does. A simple processor is built on the DRAM die,

also the architecture will not compete with Xeon chips, they call this set "processing in memory" or PIM.

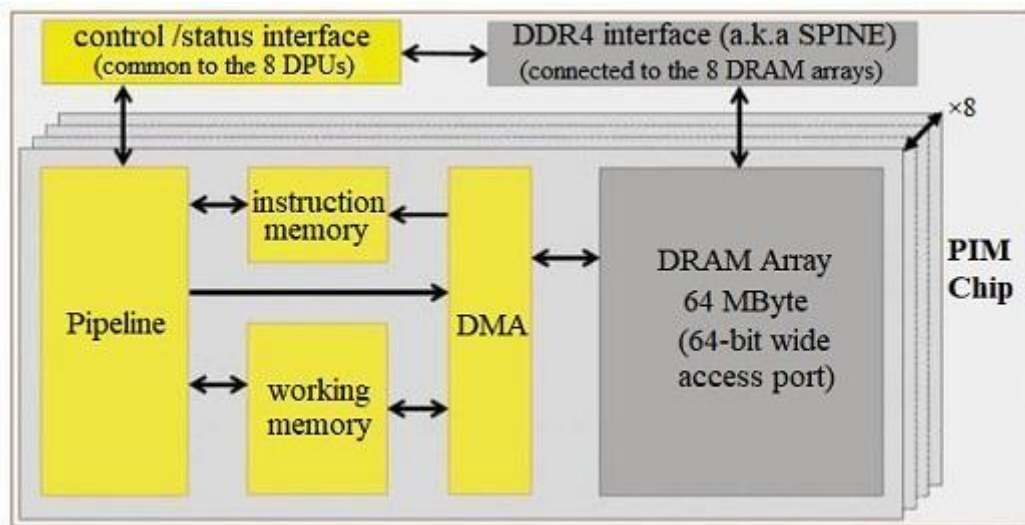


Figure 8. PIM Chip

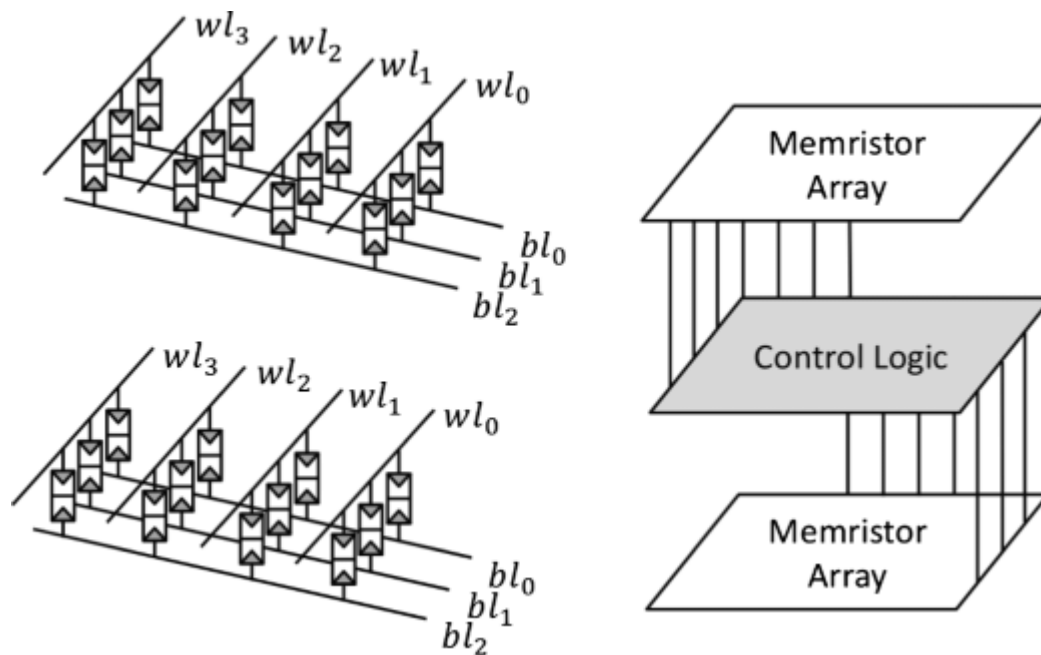
Instead of bringing data to calculations, they bring calculations to data. The runtime is performed by the CPU in DRAM chip. That is, there is no need to move the data to any location outside of the DRAM chip, just send the calculating result back to the host system. Also, since ML calculations usually involve a lot of reduction, less data required for calculations. Although this does require some minor changes to the DRAM, they did not change the manufacturing process. Under this case, a standard DRAM module will provide multiple opportunities for distributed computing. At the same time, it becomes complicated to use this function to write a program.

They said that a server using PIM offload will consume twice as much power than a standard server connected to a DRAM module without PIM. However, with a throughput of 20 times, it still provides them with a 10 times energy efficiency advantage. In addition, this method can help defend against side-channel security attacks. Thus a group of computing threads originally contained in one or more CPUs flows to DRAM. Therefore, it is necessary to check all DRAMs and figure out where thread is in some way, but this will be a difficult task.

19. In-memory Computing Framework and Architecture

In-memory computation works by eliminating all slow data accesses and relying exclusively on data stored in RAM. Overall computation performance is greatly improved by removing the latency commonly seen when accessing hard disk drives or SSDs. Software running on one or more computers manages the computation as well as the data

in memory, and in the case of multiple computers, the software divides the computation into smaller tasks which are distributed out to each computer to run in parallel. In-memory computation is often done in the technology known as [in-memory data grids](#) (IMDG). One such example is Hazelcast IMDG, which lets users run complex computations on large data sets across a cluster of hardware servers while maintaining extreme speed.



20. Business Challenges of In-memory Computing

For the last several decades, gains in computing performance have come by processing larger volumes of data more quickly and with superior precision.

Memory and storage space are measured in gigabytes and terabytes now, not kilobytes and megabytes. Processors operate on 64-bit rather than 8-bit chunks of data. And yet the semiconductor industry's ability to create and collect high quality data is growing more quickly than the ability to analyze it.

Neural networks and the von Neumann bottleneck

Meanwhile, much of the analysis of such large data sets falls to neural networks.

Low precision, analog memory, high accuracy

In an IEDM [presentation](#), Jeff Welser, vice president and lab director at IBM Research Almaden, noted that high computational precision generally is not needed for neural network calculations. A 16-bit computational block uses a quarter of the circuit real estate needed by an equivalent 32-bit block and cuts the required data volume in half. Reduced

precision arithmetic can significantly improve computational efficiency, even with conventional architectures.

Endurance, stability, and repeatability

The large number of training cycles required also [threatens the stability](#) of the stored weight values. In RRAM devices, the conductivity of the filament is defined by the [concentration of oxygen vacancies](#) inside the filament volume. This concentration is in turn controlled by the applied voltage pulses. However, it is not possible to precisely control the locations of individual vacancies.

Beyond neural networks

While the convolutional neural network is the most common machine learning technique, it is not necessarily the best one. The nonlinear, probabilistic behavior of emerging memory devices is a challenge for some algorithms, but can be an advantage for others.

In-memory computing looks to the future

Historically, electronic device research has come first, and then electrical engineers and software developers have learned how to take advantage of the new capabilities.

21. Business Benefits of In-memory Computing

1. Data blending from different sources

Once a company has chosen its platform, it can combine data from various sources and these can include news feeds, data warehouses, spreadsheets, and social media. This simplifies the process of collecting data from different sources.

Data blending is an ideal process that effectively combines data from different sources into an integrated structure. It is a quick and easy way to avoid time consuming and expensive data integration processes.

Depending on the tool an organization, non-technical employees can also easily access and understand the information. Data blending allows for optimal interaction with data to gain valuable insights and generate insightful reports based on real-time data from different sources.

2. Faster access to real-time analytics

In-memory analytics allows for the readiness and availability of data as soon as it is ready. This allows businesses to make informed decisions at a faster rate and this saves a business a lot of time and effort in gathering information.

Businesses can now operate from a more informed standpoint instead of relying on information being processed over a few days leading to lagging indicators. The use of real-time analytics has grown in its popularity and various industries have adopted the tools to streamline their data processing.

In-memory analytics contribute to client relationship management and a business can easily identify fraudulent activity at POS.

3. Working on an easy-to-access dashboard

In the past, it would be difficult to access information outside of the parameters of a business's office because all the information is stored on hardware onsite. However, in house analytics allow a business to be accessible from any device.

This is ideal for any business as the culture of working remotely increases. When building a dashboard, certain elements need to be taken into consideration to create a tool that serves the entire organization.

The first place to begin is with an operational overview of the business needs and key performance indicators. The KPIs will determine the measurements for various operations to create operations.

Once the dashboard is created, its complexity will depend on the number of tools used, the number of people using the platform and the number of data sources.

The display metrics should be in line with company goals to determine whether they are being met and what is outstanding. The dashboard is also a simplified way to generate reports as analysts can download spreadsheets to gain a single view of the data.

4. Simplified working strategy with big data visualization

At times, companies are dealing with large volumes of data that need to be processed at a rapid rate. This can be challenging for some businesses as they may struggle to sift out valuable data.

With data visualization tools, the analyst is able to review large volumes of data through a graph and easily spot trends and market drivers. Ultimately, data serves businesses by providing a wealth of information that helps a business thrive.

Big data visualization tools can easily display correlations and relationships between certain actions and client responses. It also simplifies reporting to management and creates easy to read graphics with the relevant information.

5. Optimal safety and security

One of the main concerns of migrating to cloud-based solutions is the compromise and breach of data; thus, security is extremely important. A business needs to protect its in-memory computing platform by doing the following:

- Installing a security system: as with any asset, a business needs to invest in protecting its information. Depending on the service provider, a security system might already be in place but it is best to purchase security software if not. There is a wide range of DIY security options available.
- Identify crucial data: a business needs to add extra precaution to information that could be targeted by hackers. If a business operates in eCommerce, an extra layer of protection should be added to customer information such as credit cards and contact details.
- Define levels of access: a business should assign access according to roles in the company and the relevance according to the role. Not everyone should have access to everything because data breaches can also happen internally. The business is also easily able to trace events and ensure accountability and transparency across the business.
- Create strong passwords: This is of utmost importance. Everyone in the company should be assigned a password that is strong and is not repetitive. For instance, it is not a good idea to give all users in a specific department the same password. A business needs to ensure that each member has a different password to avoid data breaches.
- Regular updates: One of the ways to ensure optimal operations of cloud-based software is to ensure that it is updated at all times. A business can do this automatically but it is important to also manually check on available updates.

22. Applications of In-memory Computing, Business Drivers of In-memory Computing

Banking

The Customer: A top 10 European banking and financial services company focused on retail banking, corporate investments and global investments and active across Europe, the USA and Asia-Pacific.

The Need: Calculate risks with extremely low latency (sub-second); improve overall performance of hundreds of applications in multiple geographies; achieve ‘full’ automation, consistency and resiliency; and reduce manual intervention to a minimum.

The Challenge: The ingestion of millions of data objects per day and the provision of about half a billion reads with over 100 million notifications; the distribution of referential data from heterogeneous sources to client applications using standardized modeling while preserving data consistency; and the provision of standard services..

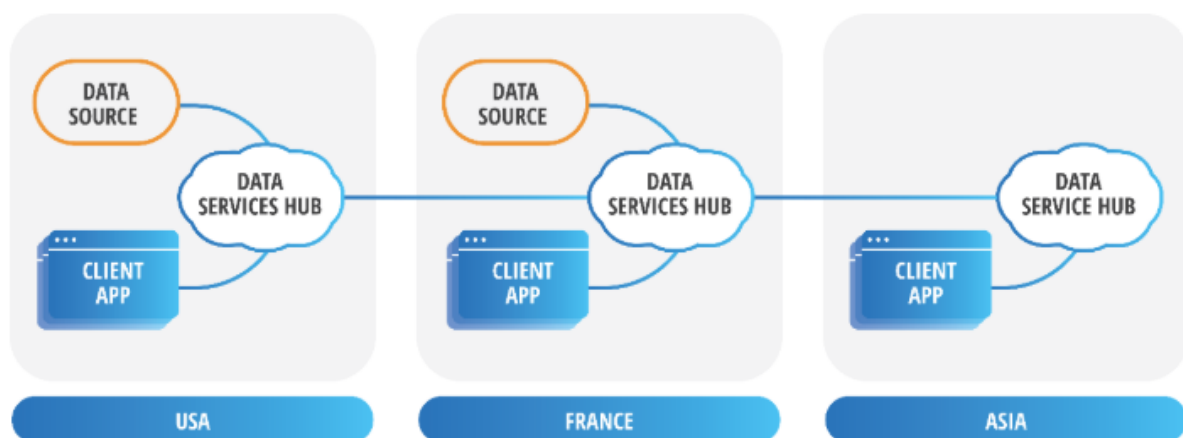


Figure 4: Implementation of InsightEdge Through a Worldwide Platform Deployed in Paris, New-York and Hong-Kong

The Results:

- Access to data in less than 1 second
- Sub-second performance for users across global sites.
- Data consistency for hundreds of services, cache and web applications in multi-geographies.
- Zero downtime

Transportation

The Customer: A leading transportation company focused on rail transportation and real estate, supplying rail-based freight transportation services.

The Need: To run predictive analytics on equipment while ingesting and processing streaming data at scale, in order to monitor and diagnose potential failures to support operators in reducing maintenance costs, improving troubleshooting and redirecting trains in a timely manner.

The Challenge: Process streaming data at scale and query from a live data mart; event-driven analytics and business logic; multiple small, low-volume streams requiring correlation and statefulness (the IoT streaming problem); and real-time analytics leveraging GPS and train sensor data.

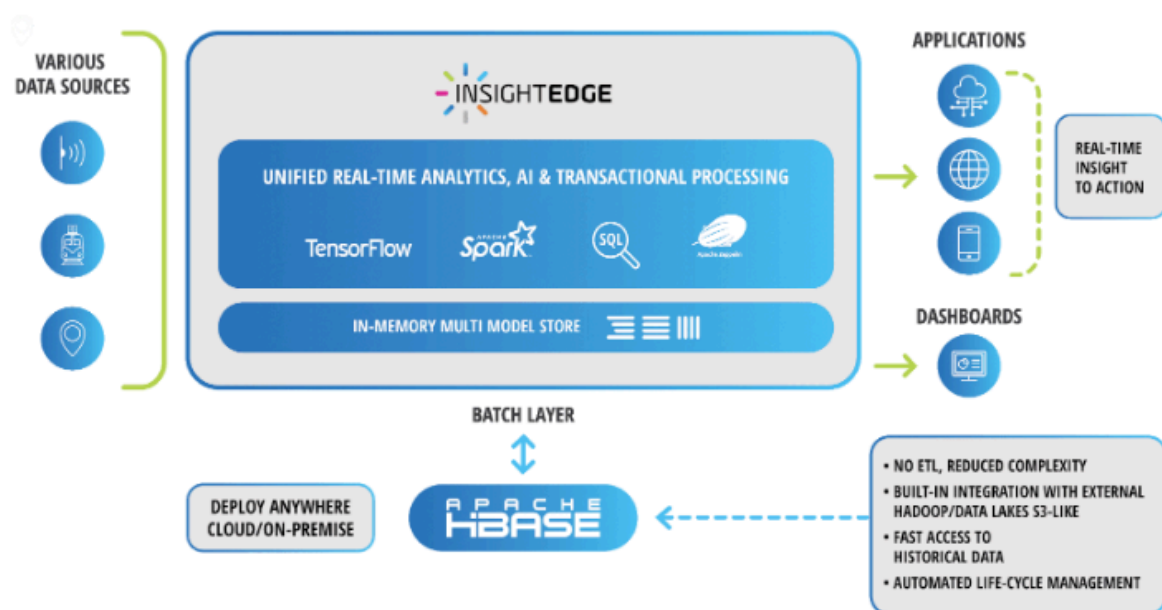


Figure 5: Integration of InsightEdge to Ingest and Process Streaming Data from Millions of Sensors, Provide Real-time Insights and Respond Instantly to Situations

The Results:

- Big data pipeline simplified by high-performance, high availability stream ingestion and processing from multiple sources

- Safety and reliability of journeys improved and cost-saving measures that conserve fuel and increase safe operating speeds achieved by leveraging machine learning on real-time and historical data from train events, fence events and GPS.
- Timely maintenance and redirection of trains through event-based triggers which direct the output to operational workflows and live dashboards.

eCommerce

The Customer: PriceRunner, a leading European shopping comparison site which has 4.4 million unique visitors and receives prices from 18,00 different merchants, every month.

The Need: To ensure the provision of real-time price comparisons, particularly at high peak periods such as the night before Black Friday when traffic increases by up to 20 times normal traffic, ensuring customer experience.

The Challenge: Support scalability requirements for peaks such as Black Friday, without compromising performance; no downtime; real-time analytics on transactional data; event-driven applications powering integrated applications; and microservices architecture for rapid development and deployment.



Figure 6: InsightEdge Provides a Load-balanced Environment that Flexibly and Quickly Scales Out without Compromising Performance and Speed

The Results:

- Scalability while retaining performance – at peak traffic spikes of 20 times normal traffic and updating 100 million prices in parallel, response time rose from 5ms to 8ms for the 95th percentile of the product page endpoint.
- Rapid development and deployment of multiple services through microservices architecture integrated with a big data ecosystem.
- Service never down, ensuring high levels of customer experience and powering PriceRunner's leadership.

23. Near-Memory Computing

Near-memory processing (NMP) is a prospective paradigm enabling memory-centric computing. By moving the compute capability next to the main memory (DRAM modules), it can fundamentally address the CPU-memory bandwidth bottleneck and thus effectively improve the performance of memory-constrained workloads. Using the personalized recommendation system as a driving example, we developed a scalable, practical DIMM-based NMP solution tailor-designed for accelerating the inference serving. Our solution is demonstrated on a versatile FPGA-enabled NMP platform called AxDIMM that allows rapid prototyping and evaluation of NMP's performance potential on real hardware under a realistic system setting using industry-representative recommendation framework. We experimentally validated the performance of a two-ranked AxDIMM prototype, which achieves up to $1.89\times$ speedup in latency and 31.6% memory energy saving for embedding operations. For end-to-end recommendation inference serving, AxDIMM improves the throughput up to $1.5\times$ and latency-bounded throughput up to $1.77\times$, respectively.

24. In-memory Computing vs Near-memory Computing

What is in-memory computing?

In today's systems, the traditional memory/storage hierarchy is straightforward. For this, [SRAM](#) is integrated into the processor for cache, which can quickly access frequently used programs. DRAM, which is used for main memory, is separate and located in a dual in-line memory module (DIMM). And disk drives and NAND-based solid-state storage drives (SSDs) are used for storage.

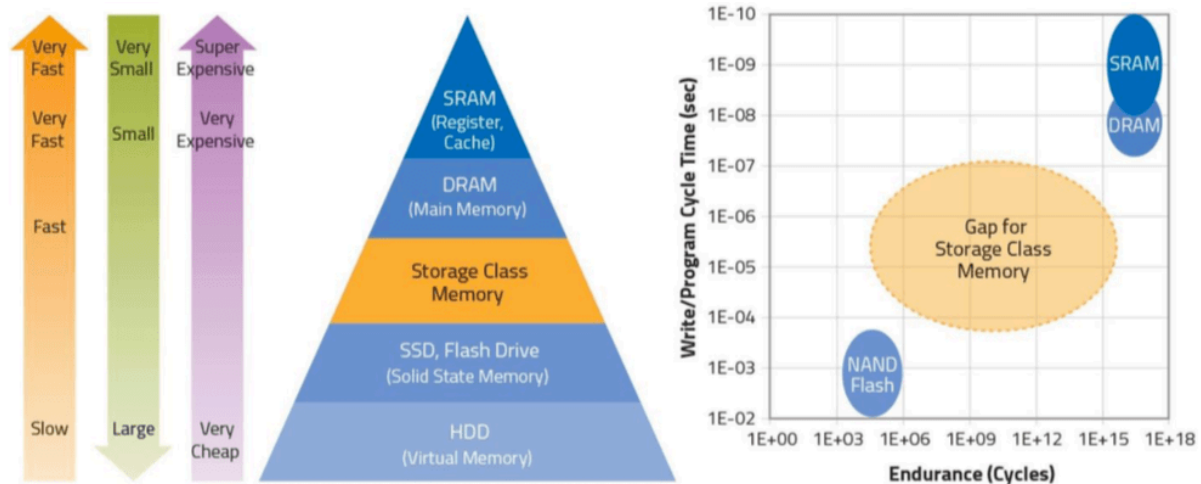


Fig. 1: Memory/storage hierarchy. Source: Lam Research

Based on this hierarchy, systems face an explosion of data on the network. For example, IP traffic is expected to reach 396 exabytes (EB) per month by 2022, up from 122 EB per month in 2017, according to Cisco.

The data growth rates are accelerating. “If you look at some of the drivers, you have mobile applications. There is a need for more data as you go into [5G](#) networks. You have more video and higher screen resolutions,” said Scott Gatzemeier, vice president of R&D operations at Micron, at a panel during the recent IEDM conference. “Then, if you look at some of the AI applications on the phones with facial recognition and authentication, it’s driving not only larger memory but also the need for faster memory.”

The data explosion is having an impact on systems. “As the amount of data increases in our world from tens of terabytes to hundreds of terabytes inside a server, we are faced with a problem in moving the data back and forth from SSDs to the CPUs. It’s going to be an energy problem and we will run into several system bottlenecks,” said Manish Muthal, vice president of the Data Center unit at Xilinx, at the panel.

The term “in-memory computing” isn’t new and can be used in various ways. Among them are:

- o The database world uses in-memory computing for caching and other apps.
- o Chipmakers are developing chip technologies to handle the processing tasks in memory for neural networks and other applications.
- o There are some newfangled approaches in the works, namely neuromorphic computing.

- o **What is near-memory?**

Besides in-memory technology, it's also possible to incorporate the memory and logic chips in an [advanced IC package](#), such as 2.5D/3D and fan-out.

- o Some refer to this as near-memory computing. Like in-memory, the idea is to bring the memory and logic closer in the system.
- o “The world is driving more data in systems. So, processors need large amounts of memory. And the memory and processor need to be very close,” said Rich Rice, senior vice president of business development at ASE. “So, you need packaging solutions that enable it, whether it's 2.5D or a fan-out with a substrate approach. This could also be PoP structures like package-on-package.”
- o In [2.5D](#), dies are stacked on top of an [interposer](#), which incorporates [through-silicon vias](#) (TSVs). The interposer acts as the bridge between the chips and a board, which provides more I/Os and bandwidth.
- o For example, a vendor could incorporate an FPGA and [high-bandwidth memory](#) (HBM). HBM stacks DRAM dies on top of each other, enabling more I/Os. For example, Samsung's latest HBM2 technology consists of eight 8Gbit DRAM dies, which are stacked and connected using 5,000 TSVs. This enables 307GBps of data bandwidth. In a traditional DDR4 DRAM, the maximum bandwidth is 85.2GBps.