Scaling Integrations For Modern Workloads

tray.io

CONTENTS

Modern integration and automation workloads are subject to more significant demand spikes, unpredictability, and volume than ever

Technologists often face a choice: Plan ahead by paying for and provisioning Containers or Workers such as v Cores that should be ready to crunch peak volume or risk processing failures and hours hunting through logs to diagnose scalability bottlenecks.

Integration scalability demands have changed.

Event-driven integration is overtaking traditional polling or daily scheduled workflows. The ubiquity of Web-hook support or native triggers included in most modern cloud-based applications means collectively, a single Sales, Marketing, or Services applications stack can easily consist of 40+ apps . These, in turn, can throw tens or hundreds of millions of events per second that can place massive demand on an integration platform to queue, route, transform, and aggregate data and trigger multiple downstream workflows.

A large enterprise can run over 1,300+ apps across the organization[2] with the number of apps increasing by 5-10% annually

The sheer number and growth of business apps and the need for increased business process connectedness can place a massive strain on integration and automation tools, and the ops teams that support them.

Data is increasing in volume and unpredictability.

Predictably, data volumes continue to increase, growing 23% annually, with enterprise data growing at 2X the rate of consumer data . However, what's changed more is Peak Volume. It can now reach 100X of average volume for some organizations, especially when dealing with seasonal E-Commerce transactions, social spikes, IoT surges, or usage crunches from mobile internet applications.

Legacy integration platforms haven't kept pace

Event-driven integration is overtaking traditional polling or daily scheduled workflows. The ubiquity of Web-hook support or native triggers included in most modern cloud-based applications means collectively, a single Sales, Marketing, or Services applications stack can easily consist of 40+ apps . These, in turn, can throw tens or hundreds of millions of events per second that can place massive demand on an integration platform to queue, route, transform, and aggregate data and trigger multiple downstream workflows.

The expansion in apps that must be connected, increasingly real-time demands, and the growing gulf between average and peak processing means that for technologists and integration specialists, sizing and provisioning traditional integration and platforms is not sustainable. They are too complex to size, too costly to configure, requiring over-purchasing Atoms, v Cores, or Workers Nodes from the platform vendor. Nor to mention considerations around load balancing, parallel processing, and other areas.

All of this to meet anticipated demand which typically ends up being hard to maintain, requiring constant monitoring for failed executions, API retries, or errors related to under-sizing.

It may have been viable in an era for dozens of integrations in the enterprise, but not for the connected enterprise[3] with hundreds or thousands of integrations at play.

Beyond IT, business teams and citizen integrators looking to connect their stack using departmental and point-to-point tools can quickly get overwhelmed as low-end tools. They can rapidly get stretched beyond what they were designed for, triggering Flood Protection and other Timeouts, creating roadblocks.

Time to rethink integration platform architecture

Surprisingly, some integration platforms' codebases date back nearly twenty years. As a result, they were never designed for contemporary integration demands—that require event-driven integration at scale and flexible processing to meet unpredictable volumes and demand spikes.

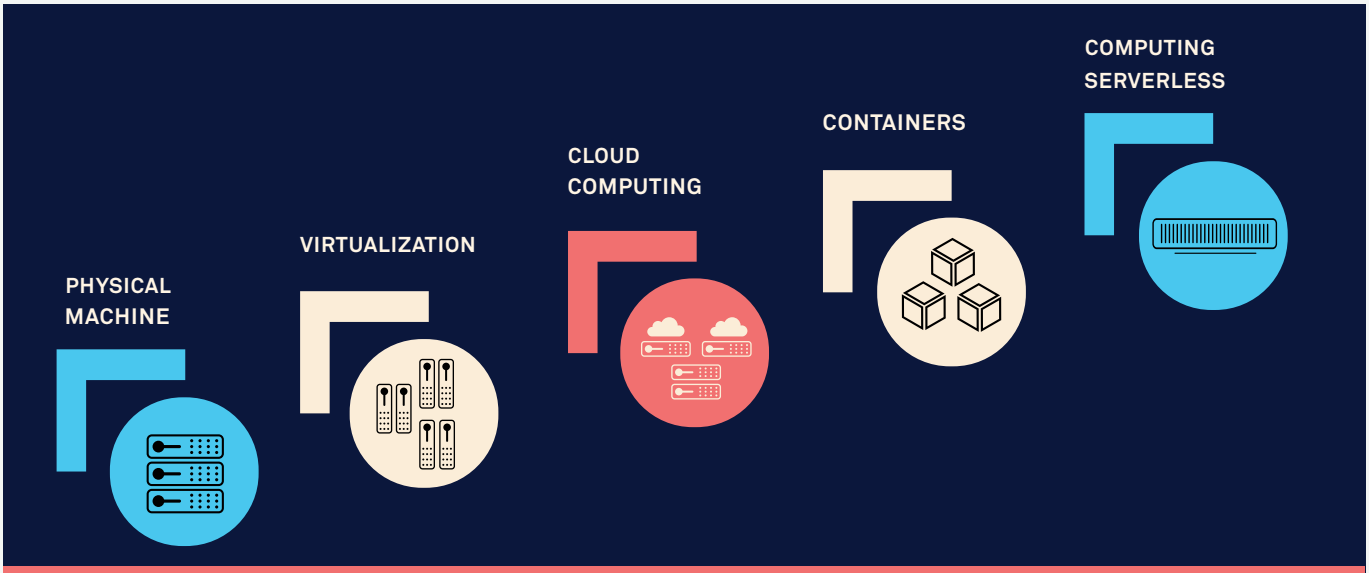
Instead, modern integration platforms take advantage of the latest innovations in cloud-native computing and elastic serverless processing and achieve massively parallel scale, on-demand.

Predicts 2022: Modernizing Software Development is Key to Digital Transformation, 3 December 2021, Gartner

The Future of Integration is serverless

> “By 2025, 60% of new event-driven applications will use serverless computing due to its rapid elasticity, cost agility, and low operational overhead”
>
> —Gartner

Serverless computing has experienced a dramatic rise in the last five years—providing fine-grained elasticity through the decomposition of execution into highly atomic serverless functions that run on a cloud Platform-as-Service such as AWS, Microsoft Azure, or Google Cloud Platform.

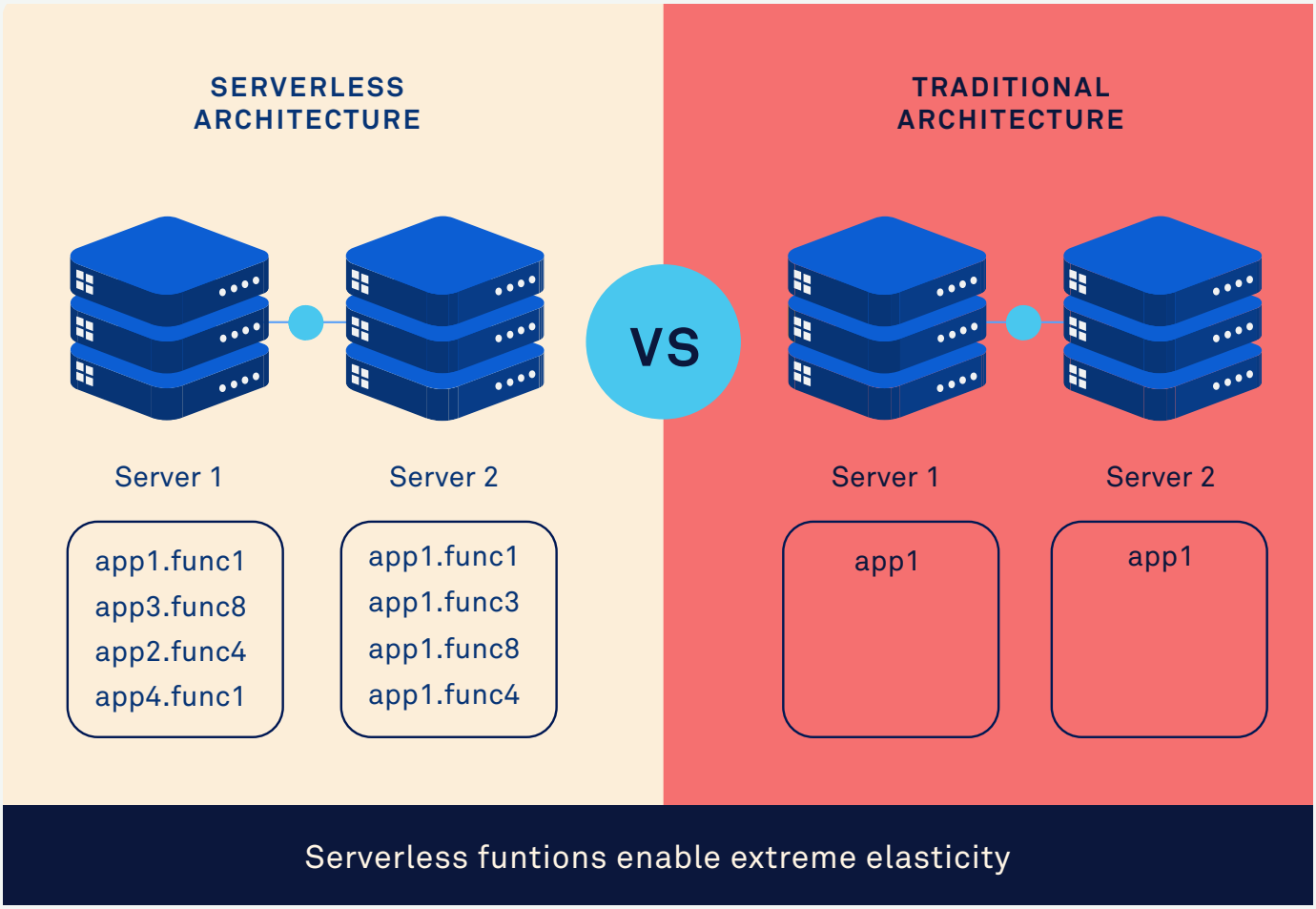


Serverless computing is the latest evolution in computing to provide more elasticity.

Serverless architectures provide the following four traits

1. The granular use of computing resources
2. Resources that do not need to be pre-allocated
3. Highly scalable and flexible
4. Users only pay for the resources they use, not purchase nodes/servers.

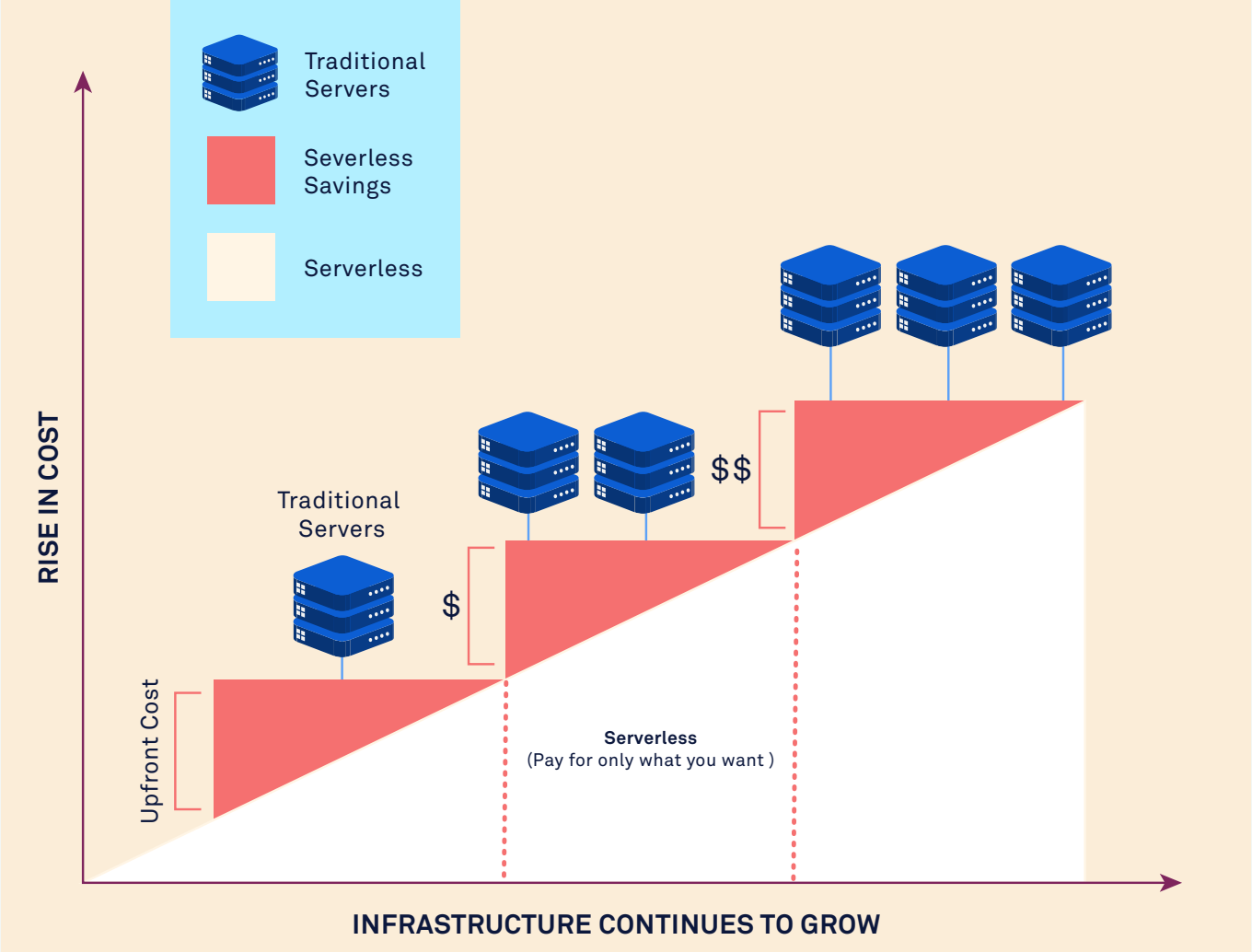
Serverless architectures automatically provision computing resources required to meet a workload on-demand or respond to a specific event. They automatically scale those resources up or down in response to increased or decreased demand. And then automatically scale resources to zero when the application stops running. It means the most efficient use of computing resources for customers, with the least operational overhead.



Serverless computing has been adopted for various use cases, from microservices, mobile back-end processes, handling sensor data for IoT, E-Commerce apps. However, most integration platforms are pre-serverless architectures, as they were built before the invention of serverless computing.

In the context of integration, pre-serverless applications often require a higher degree of sizing expertise and pre-allocation and purchase of integration worker nodes to handle anticipated workloads. Worker nodes typically carry a per-worker cost and run a certain degree of transactional or API volume. Therefore, more must be purchased and deployed as workloads increase.

Pre-serverless architectures mean living within constraints and devoting ops resources to managing them. For example, an integration process may run smoothly for a week before it slows or crashes due to high memory usage, too much transactional volume that cannot be processed within a timeframe, or too many concurrent requests. The ops team will then examine the logs, determine if it's a resource issue, make the necessary changes, restart the worker due to a memory leak, or purchase more nodes from the integration vendor.



Traditional architectures eliminate the need to provision a priority

While in a true serverless architecture for an integration platform, each workflow step or task in an integration workflow flow (such as a trigger, transformation, or insert) is executed as an individual Lambda serverless function and run on the underlying Platform-as-a-Service on-demand, elastically scaling in milliseconds.

There are no persistent worker nodes, and no resources are used if there is no activity. There is no need to size or pay for anticipated demand. It provides extreme elasticity.

In addition, because each workflow step is a granular serverless function executed on-demand, it provides opportunities for a high degree of concurrency with serverless workflow steps being run across a cloud platform.

	Serverless integration	Pre-serverless integration
Provisioning required	NO	YES
Cost for “Worker nodes” to scale	NO	YES
Sizing for “Worker nodes”	NO	YES
Lights out elastic processing	YES	NO

Serverless Scalability Case Study: Eventbrite

Eventbrite (NYSE: EB) is the global self-service ticketing platform for live experiences. As a result, the company needed to deliver integrations to its customers at a massive scale. It has more than 650,000 creators on its platform, managing more than 4.6M events in nearly 180 countries. As a result, any integration Eventbrite rolls out can easily create a surge in demand on its integration platform. Eventbrite (NYSE: EB) is the global self-service ticketing platform for live experiences. As a result, the company needed to deliver integrations to its customers at a massive scale. It has more than 650,000 creators on its platform, managing more than 4.6M events in nearly 180 countries. As a result, any integration Eventbrite rolls out can easily create a surge in demand on its integration platform.

9.5B
Tasks in 10 months

111K
Active integrations

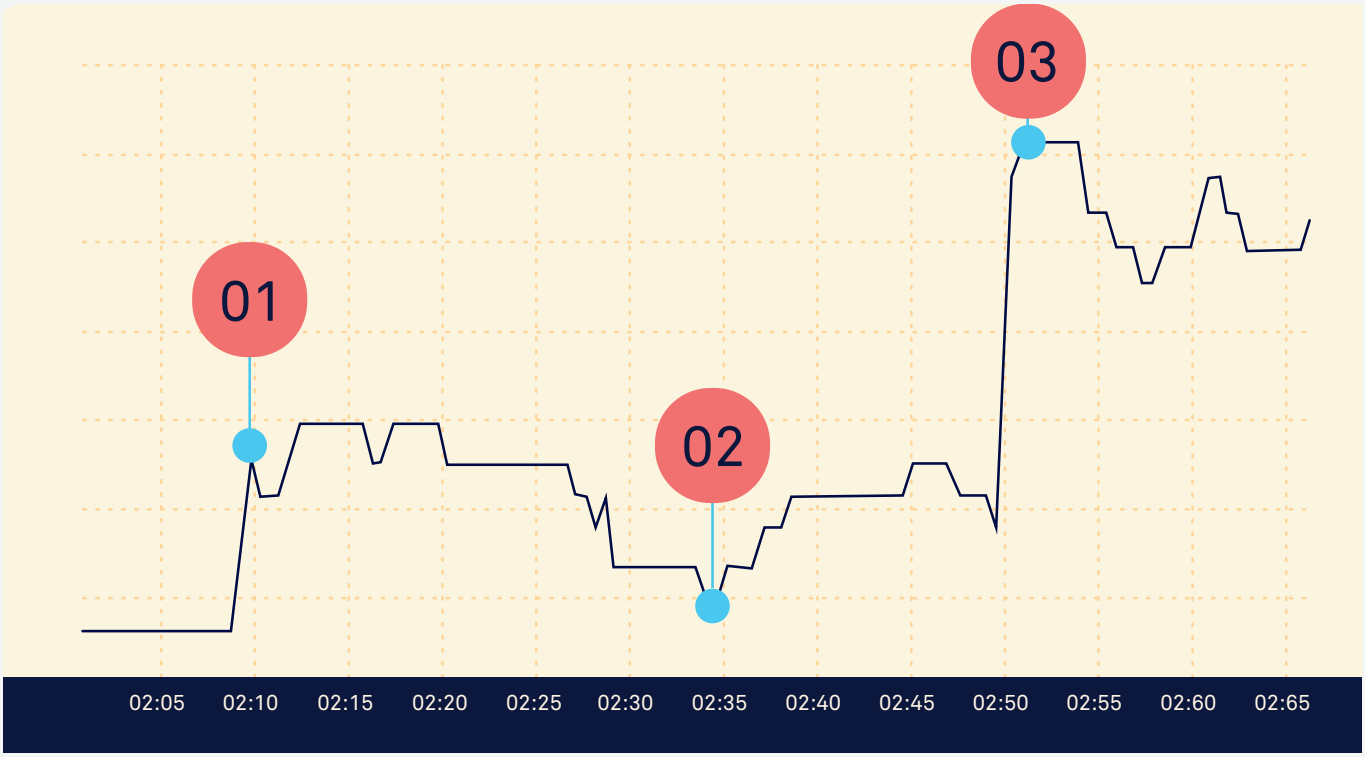
1.38M
End-customer workflows

75%
Reduction in operational overhead

By choosing the serverless Tray Platform, Eventbrite enabled more than 111,000 active customer integrations in less than 12 months without adding any staff to its operations team. So, while it was able to reduce the number of engineers for each integration from six to one—more importantly, Eventbrite avoided the massive operational overhead from integrations at scale that comes with older, non-serverless integration platforms.

Serverless Elastic Processing in Action with the Tray Platform

The ideal elastic integration architecture scales dynamically and instantly without intervention or pre-allocation of resources. The real-life example below shows how this can play out on Tray Platform’s serverless architecture, lights out, over one hour.



Tray Platform instantly scales with workloads without IT/Op's planning/intervention

At (1) customer integration transactional load unexpectedly triples before 2:10 AM. The serverless Tray Platform automatically triggers workflows that consist of serverless Lambda functions on AWS that elastically accommodate demand.

At (2) 2:35 AM, processing volume demand halves, scaling down compute, however shortly after, it triples again at (3) 2:50 AM. Again, no pre-planning or operational intervention is required.

A pre-serverless architecture would have required pre-allocating enough resources and purchasing enough workers to handle the burst of peak volume at (3). If the ops team had only planned for first peak at 2:10AM, and bought/provisioned a small bonus over that, then integrations may have failed temporarily.

The Tray Platform scales so elastically because the unit of processing for the Tray Platform is a Task, a coarse-grained or fine-grained workflow step, a serverless function that consumes resources when needed, and zero resources when the work is complete.

Tasks can include an integration, multiple API calls, multiple transactions, a message/event, multiple rows/pages of data, a processed document. Tray.io's large enterprise customers routinely process ~10 billion Tasks per month and have processed up to ~20 billion Tasks per Month, without requiring provisioning or sizing a priori. In some cases, their demand is highly seasonal, only spiking to billions of tasks for a fraction of a period.

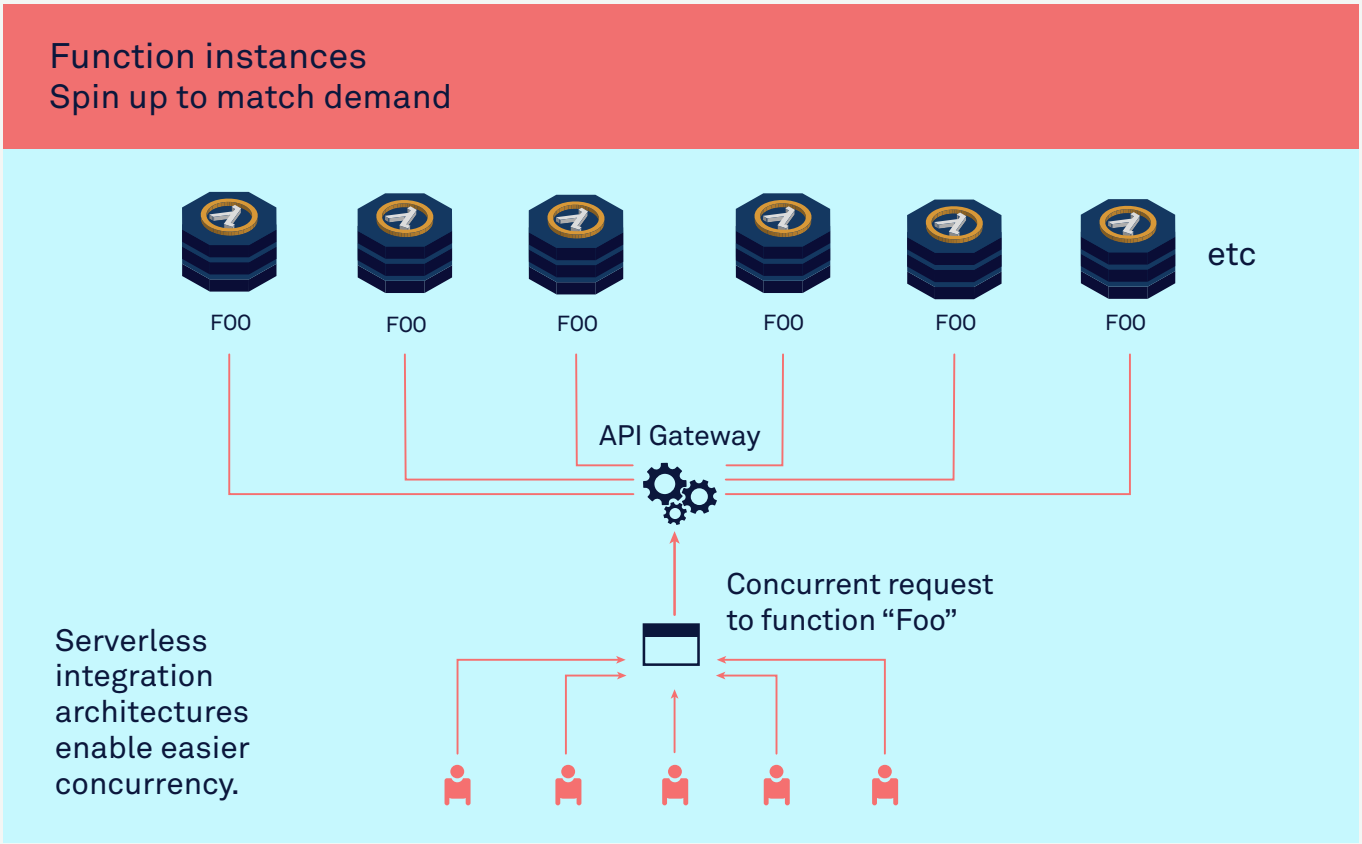
> “MuleSoft has a large overhead, and we would have required skills, a dedicated team, and a strong ongoing sustenance model. Tray Platform is the other way: more user-friendly and easy to scale up. It can be managed by one or two internal resources with Tray’s support.”
>
> **—Sunita Raja, Head of Business Technology, Udemy**

The Future of Integration is Highly Parallel

The shift to event-driven integrations, where workflows can be triggered thousands of times in a matter of seconds and where large numbers of them may run concurrently for a portion of their runtime, can be demanding on inelastic architectures.

It’s where true serverless architectures shine because each workflow instance is effectively a series of Lambda functions—so whether a five or thousand simultaneous workflow instances, the compute is all on-demand.

While a pre-serverless architecture certainly provides concurrency, it is often constrained by allocating worker nodes, threading limits, compute limits, and other factors. However, keeping all of this at the ready is cost-prohibitive. And there are often constraints in older architectures, with shared resources, such as shared databases, limiting parallelism when spreading work across nodes.



In contrast, a modern serverless integration architecture can dynamically spin up practically unlimited workflows consisting of Lambda functions that run for a fraction of a second. In many cases, each Lambda function is stateless, with limited contention for shared resources— providing practically unlimited parallelism.

In the case of an event-based integration, which might trigger workflow hundreds or thousands of times in a second, a serverless architecture can execute numerous workflows concurrently, massively parallel—automatically, and lights out.

Scalability Case Study: AdRoll shifts to real-time processing

AdRoll needed an easy, fast way to refresh hundreds of attributes for approximately 650,000 Salesforce opportunities continuously. But they were constrained to just a handful of attributes refreshing periodically using a previous custom-coded integration. This approach not only came with a high operational overhead but also led to out-of-date data in Salesforce's opportunity records.

By leveraging Tray Platform's modern architecture, AdRoll now processes over five million integrations and has moved from syncing a small number of attributes to around 240 - at no increase in operations.

The Future of Integration is auto-scaling: Real-time + Big Data

As discussed, event-driven integration is on the rise, and serverless architectures are ideal for handling this kind of use case. But analytical databases like Snowflake and Redshift require bulk data loading—less high-frequency workloads, more raw transactional volume associated with Extract-Transform-Load (ETL) / Extract-Load-Transform (ELT), or Data Integration (DI) processes.

Many integration platforms are designed for one use case, triggered fairly atomic integrations (Application Integration), or bulk data integration (ETL/DI) scenarios, but not both. Organizations often use multiple tools for application and data integration due to different scalability demands.

However, the emerging demand for Reverse-ETL use cases means one platform that can scale to both app and data integration is preferable but must be flexible enough to scale with the unique characteristics of both workloads.

Reverse-ETL is where data isn't just flowing into data warehouses like Redshift or Snowflake for analytics it's also flowing out of them to drive business processes on-demand to drive activities like personalized email campaigns or website product recommendations.

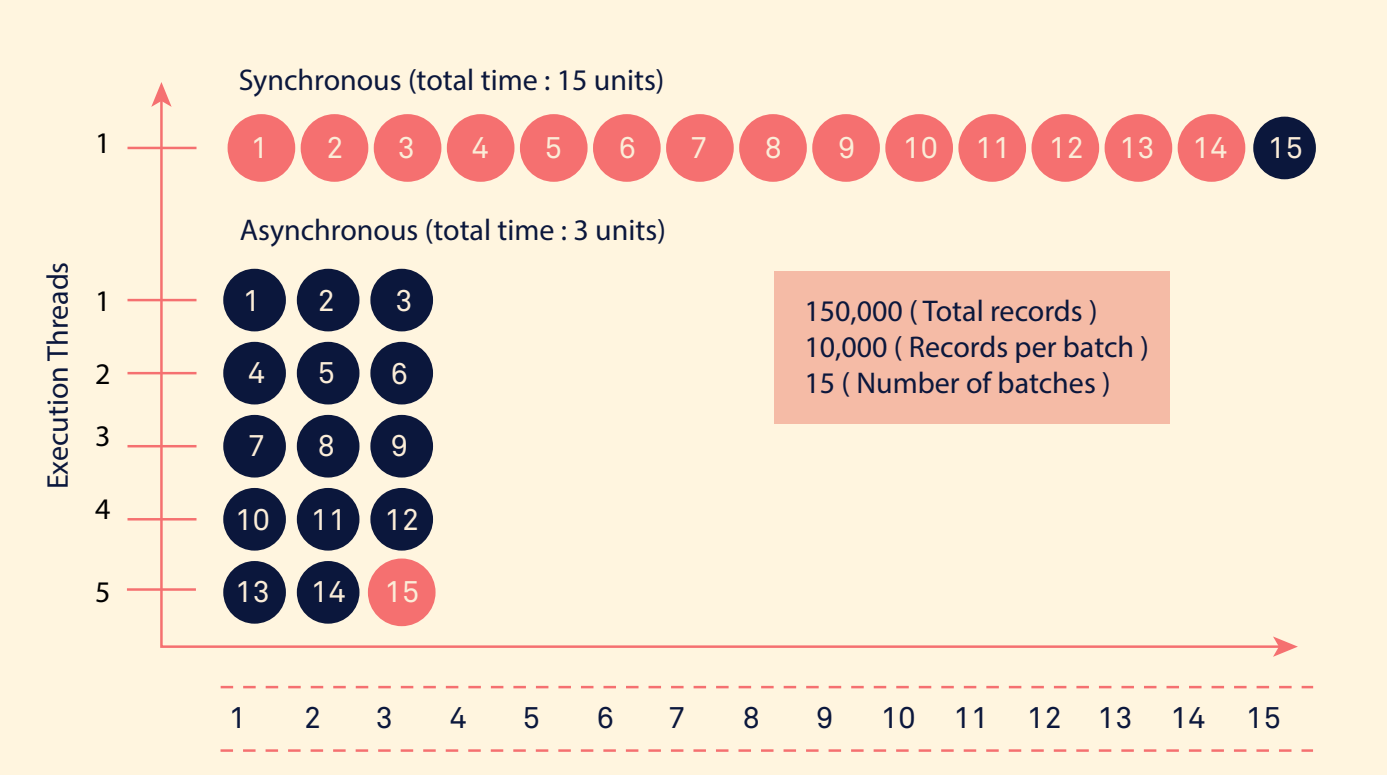
Reverse ETL: Where bulk loading meets on-demand business process

Modern serverless integration architectures enable highly parallel bulk processing

Processing large volumes of data not only means sheer volume in terms of the number of records and throughput, but it also often means complex processing (such as aggregation or enrichment) operations.

Because serverless architectures can elastically scale compute on-demand, they can efficiently parallelize the process. For example, an overarching workflow can batch off chunks of data to call separate sub-workflows that operate in parallel for processing. As a result, these data processing/enrichment workflows run concurrently, processing multiple batches simultaneously, with no need to wait until the previous batch is finished.

The following diagram is a good illustration of how this works.



Serverless integration architectures enable practically unlimited concurrent bulk processing.

The contrast with less elastic architectures is clear. Typically, there are hard concurrency limits that limit parallel processing. While in a serverless integration architecture, the level of parallelism is practically unlimited, just a set of concurrently executing Lambda functions while enabling robust control over the business logic that determines the “batches.”

The following diagram is a good illustration of how this works.

Tray Platform can execute a callable workflow to process bulk data concurrently.

Data Integration Case Study: Enterprise software leader streamlines PostgreSQL data load

One of the world's largest technology companies sought to gain analytical visibility into their tens of millions of sales opportunity transactional detail. They needed to run reporting on those logs to provide visibility into its sales funnel for budgeting, forecasting, and reporting by loading into PostgreSQL.

The goal was to shift from a reporting window that took close to a full day to complete to load and refresh every five minutes.

By moving from hand-coded integration to Tray Platform's flexible serverless architecture, they cut their processing window by 99% while also reducing the operational workload, saving 40 hours per week from a team of 8 (including six engineers) by reducing the need to monitor and optimize resources.

> "We'd previously had eight people doing 40 hours a week on this. This was a process that used to involve taking days to build things out in SQL, then spending hours in Excel. We got that down to five minutes a day, and it's fantastic."
>
> **— Business Operations Team**

SUMMARY

Today's integration and automation workloads require way more elasticity and flexible scaling without the associated increase in cost and operational overhead. Legacy integration architectures were never designed to meet this requirement. Modern integration platforms combine a serverless and highly parallel architecture, and flexible processing, without requiring IT/Ops sizing and provisioning.

ABOUT TRAY.IO

Tray.io is the leader in low-code automation and integration. The Tray Platform is built to be fast, flexible, elastically scalable, and trusted to empower organizations to evolve faster and drive efficient growth. Builders can now connect their stack using a modern, low-code user experience to innovate business processes together rapidly.

Contact Paul Turner, Tray.io

tray.io