GENERAL SYSTEM

Unleash the power of machine data to **understand the world in real time**

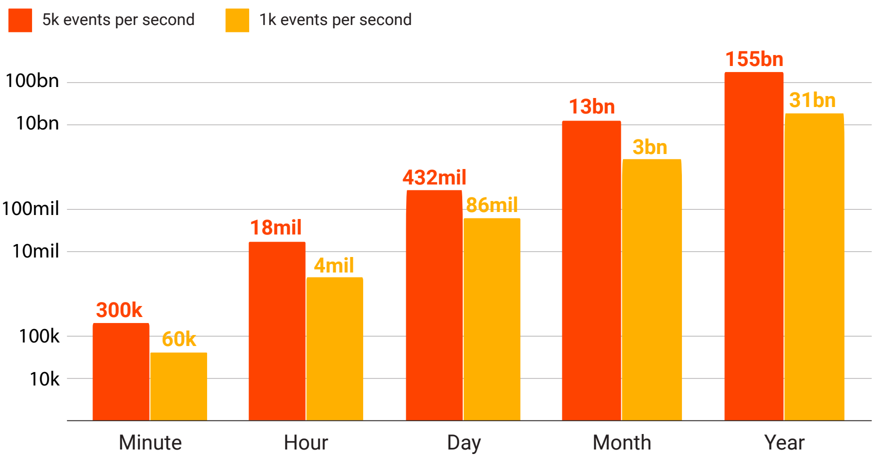
generalsystem.com

Contents

The challenge of spatiotemporal data

Vehicles, GPS trackers, IoT, robotics and sensors generate a constant flow of spatiotemporal (location and time) data. Even a moderate number of events per second soon accumulates into billions of records which becomes increasingly complex and costly to process and analyse. The chart below illustrates a system that generates 1k events per second. This will accumulate 31bn records over a year. This grows to 155bn per year if 5k events are generated per second.

Records created (log scale)



Organizations working with spatiotemporal data (location and time) often find that:

- It's complex to organize and analyse streaming data in real time
- Queries become costly and uneconomical at scale
- More complex queries such as multi entity, data layering and comparisons between current state and history are unwieldy and very unproductive
- Queries are too slow so that when an insight is found, it may no longer be relevant
- Limiting the scope of analysis to speed up processing often creates data bias and inaccurate assumptions

These restrictions translate to business inefficiencies such as manual processes and sub-optimal customer communications.

Introduction to the General System Platform

The General System Platform (GSP) is a cloud-based technology designed for fast and effective processing and analysis of real-time and/or large scale spatiotemporal data.

The technology combines two technical capabilities in a single system: billions of records can be ingested at extremely high rates, with immediately queryable indexes across attributes in the ingest stream on a single commodity server. Within milliseconds of ingestion, ad hoc low-latency queries can be executed incorporating all data streamed into the system up to that point, limited only by available storage.

Spatiotemporal data is common across domains and industries. Examples include data generated by sensors on physical moving objects, such as personal devices (e.g. smartphones or watches), fleets of vehicles, robots or drones. It also covers data generated by sensors that track moving objects, for example, a series of street cameras with automated number plate recognition.

The GSP is proving valuable in many data science and operational analysis workflows with similar requirements and characteristics:

- Up to trillions of records in a single index
- Index insertion rates of millions of records per second
- Indexed search across multiple record attributes

The GSP enables corporates and governments to unlock the potential of their data compared to any other solution available. It provides insights in a timely manner, as it does not suffer from ingestion delays typical in other systems. And it is cost effective as datasets in excess of 100bn records can be stored and queried on a single server, whereas distributed analytics solutions require a fleet of compute nodes.

How does the GSP achieve this massive performance? The GSP does not suffer from the limitations of current solutions, namely:

Index Bloat

Indices commonly require more storage than the data they index. Succinct indexing algorithms are compact but do not support complex spatiotemporal data models.

Hotspotting

Geospatial data models have a highly unpredictable distribution of both data and workload that defeat static sharding schemes.

Storage Density

Write performance degrades as storage density increases but density is required for economy. At high densities, storage caches fail for multi-dimensional data models.

Let’s find out what is unique about the GSP technology.

Unique technology

Underlying the General System Platform is a highly optimized database kernel written in C++ that incorporates all of the elements one would expect in a state-of-the-art design: thread-per-core software architecture, vectorized storage model, user space scheduling of I/O and execution, and other features that ensure maximum performance and hardware efficiency. This provides a strong foundation for scalable data processing but by itself, does not address any of the technical challenges of indexing high-velocity data flows.

The GSP kernel contains multiple technologies and architectural features, described below, not found in any similar system. Collectively, these technologies work together to enable unparalleled indexing performance for high-velocity streaming and geospatial data models.

Succinct High-Dimensionality Indexing

All data models in the implementation of the GSP are organized using novel high-dimensionality succinct indexes. Importantly, diverse and unrelated data types can be concurrently represented within a single indexing structure. The properties of these indexing structures differ from conventional indexes, and some other succinct indexes, in fundamental ways.

The storage location of a record is content addressable, similar to a hash table, based on multiple, independent attributes of the record while preserving the relationships between attributes across records in the index. Indexing is fully adaptive to the distribution of the underlying data across the indexed attributes.

As with all succinct data structures, the in-memory representation is extremely compact relative to the data being indexed. The index for 100TB of data may be small enough to fit entirely within the CPU cache. When a record is inserted, a simple lookup operation against the index identifies the disk page to which it will be appended. This direct path through storage incurs almost no write amplification, with throughput being limited primarily by I/O bandwidth.

The GSP index implementation uses a design of sufficient dimensionality to index spatiotemporal data models and additional key attributes. Most data types, such as entity IDs, consume a single dimension in the index. A few data types, notably geospatial, may consume two or three dimensions. The ability to mix and match data types in the index, up to the dimensionality budget, enables an unusual degree of flexibility for supporting different data models. Most succinct indexes are one-dimensional or require that all dimensions be the same type. GSP do not have this limitation.

Indexing of geospatial data types is not limited to points. The index is purpose-built to succinctly index polygon relationships as well, allowing query operations to directly look up geospatial polygon intersection relationships on the index with the same kind of performance and scalability one can expect with other scalar data types.

The primary limitation of this type of indexing is that it is poorly suited to storage engines not purpose-built to support them.

I/O Cache and Scheduler Designed for High-Dimensionality Indexes

High-dimensionality indexing, which is critical for GSP performance and scalability, has long been known to present fundamental problems for caching algorithms, including the storage cache in the operating system and all popular storage engines. Caching algorithms work by trying to predict future data access requirements based on previous data accesses. The problem of optimal data caching is equivalent to universal sequence prediction. Universal sequence prediction is infamously intractable even in narrowly restricted cases. All practical algorithms for effectively predicting what to cache are limited to simple access patterns in a single dimension, such as traversing an array or ordered tree.

Because the GSP relies on a high-dimensionality index as its primary data access method, the data access patterns are unable to be successfully predicted by a storage cache. Without an effective storage cache in front of the workload, almost every storage access becomes stalled by frequent page faults. Using high-dimensionality data access methods on a conventional storage engine architecture is a recipe for poor storage performance.

The GSP's storage architecture takes heed of Alan Kay's aphorism, "the best way to predict the future is to invent it." Concurrent high-throughput database kernels often have thousands of operations scheduled at any point in time, even for single queries, since they may be decomposed into myriad sub-operations internally. In thread-per-core software architectures like GSP, the I/O scheduler has perfect visibility into and control over storage access patterns for thousands of operations into the future as well as the contents of the storage cache.

Instead of relying solely on prediction to ensure efficient eviction and pre-fetching patterns, the scheduler uses its ability to 'see into the future' to dynamically reorder and optimize the sequence of operations far beyond what a storage cache can predict on its own, to maximize locality of storage access and minimize the number of storage operations.

Adaptive Re-sharding To Eliminate Hotspots

Data models with a geospatial component are prone to unpredictable 'hotspotting', the phenomenon where individual data shards become temporarily overloaded due to a sudden influx of data or queries focused on a specific geographic region. When a hotspot occurs, the throughput of the entire system becomes bottlenecked by the throughput of the most overloaded shard. Because there is no way to know when and where a hotspot will occur, minimizing their effects requires both proactive and reactive strategies.

The GSP employs a continuous adaptive re-sharding strategy to mitigate hotspots. If a shard becomes overloaded or is at risk of becoming overloaded, then it is automatically re-sharded, distributing the records of that shard over many new shards that effectively subdivide the index. This has the effect of distributing data and load over a much wider set of shards while reducing the computational cost of operations over the individual shards. Individual shards are kept small to ensure that re-sharding operations have minimal impact on tail latencies.

Re-sharding requires no user configuration or interaction and is fast, occurring automatically in the background.

In high-velocity ingest environments, individual re-sharding operations need to be sufficiently fast to ensure that they don't slow down or disrupt concurrent ingestion. A single server with fast, high-density storage may have extremely large numbers of active shards and, for the purposes of re-sharding, may need to create and destroy a significant subset of those shards each second.

High-Density Storage Architecture For Extreme Scalability

Today, servers often have upwards of a petabyte of attached storage, and for high-velocity data models, this type of storage density is often important for cost effectiveness. Storage engine architectures that map shards to files in the filesystem frequently run into significant practical performance and scalability limitations when operating at this scale. The GSP makes these limitations particularly acute due to its continuous adaptive re-sharding behaviour.

Typical Linux filesystems do not perform well when processes can have millions of open files that are continuously being created or destroyed. Like many sophisticated database kernels, GSP is able to create a specialized non-POSIX filesystem that coexists with standard Linux environments while bypassing most scalability and performance limitations of conventional filesystems. This allows the GSP to manage upwards of a petabyte of storage with consistent performance across vast numbers of logical files, while implementing additional features that conventional filesystems do not support.

This filesystem can be installed as an overlay on an existing Linux filesystem, where it looks like a set of large files, or directly installed on sets of raw block devices with no other filesystem.

Features

At the core of the General System Platform, there is a highly specialized engine. Around it, there is a set of tools that enables customers to easily extract valuable insights. For our customers, this is typically a four-step process:

Ingest

The GSP can ingest and index high volumes of data. Customers can ingest data from existing data stores and data lakes (e.g. from S3), for instance to analyze historical datasets. They can also integrate with streaming data sources such as Kafka or AWS Kinesis to process real-time data.

Integrate

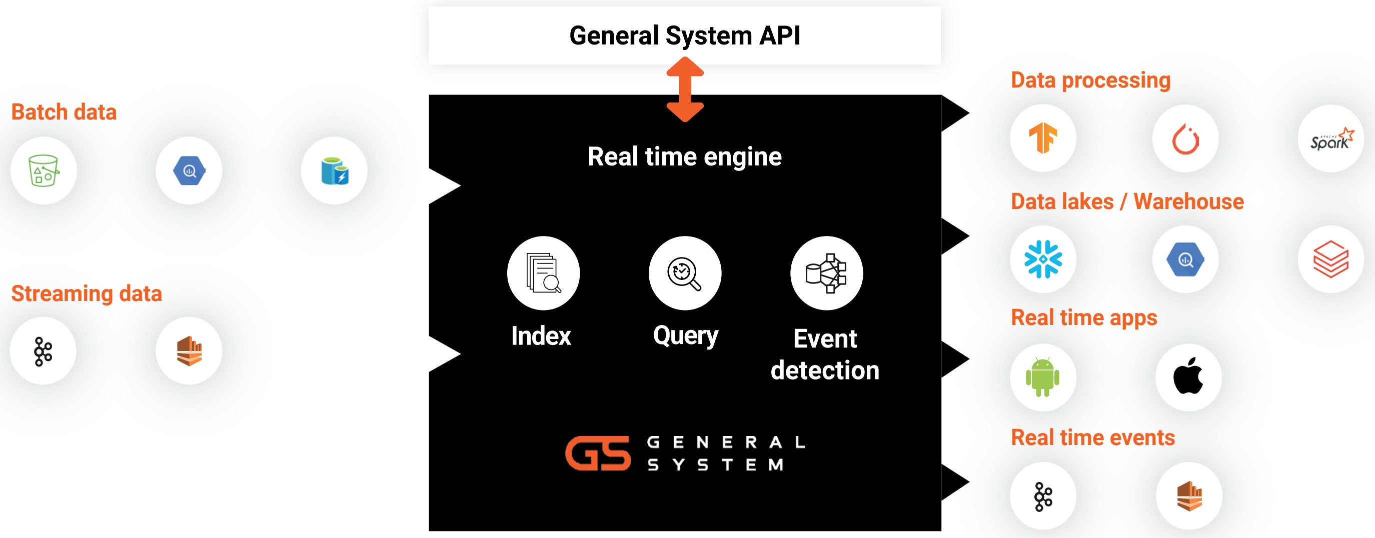
The GSP integrates easily with existing apps (mobile or website) via a Web API.
The GSP API augments processes with new insights and can generate customized events.

Analyse

With the API, some or all of your historical data can be queried to uncover new correlations, insights and business opportunities. The GSP supports Data Science interactive workflow to address ad-hoc analyses or to build models. It also supports automation of queries.

Act

Data is available for analysis immediately as it is ingested. It's possible to decide how to respond to new events and act in real time.



General System Platform schema

The schema of the GSP consists of a set of mandatory spatiotemporal fields, and a set of optional attributes. The mandatory fields are: entity id, latitude, longitude and timestamp of the record. The table below shows an example of mandatory fields:

Id	Latitude	Longitude	Timestamp	Altitude
81696	41°24’12.2”N	2°10’26.5”E	1/1/23 13:01:44	55m
81452	41°24’12.7”N	2°10’26.4”E	2/1/23 14:12:01	2m
81696	41°24’12.3”N	2°10’26.2”E	2/1/23 08:07:58	230m

Optional attributes, up to a maximum of eight, can also be stored for each record. Attributes can be of different types: integers, lists or IP addresses. When executing a spatiotemporal query, the results can be filtered based on the optional attributes. To do so, common query predicates are available, such as: equality, inequality, range, and greater or lesser than a value.

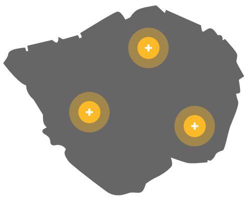
The platform can also ingest any unstructured data associated with a record for customers that require it. Unstructured data is indexed by its record identifier. The identifier is stored in the GSP and can be retrieved when querying data.

Spatiotemporal queries

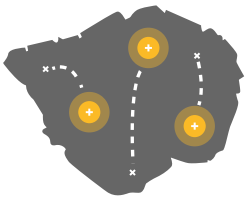
While not a database per se, GSP supports many database-like operations on the underlying data stream. Queries like “find the unique set of entities within this geographic polygon between the hours of 2pm and 3pm on Tuesday” are executed via the API as a single operation. Spatiotemporal queries can have any of the following criteria:

- A polygon, including complex shapes with hundreds of vertices, or a bounding box
- A time range
- A list of entities

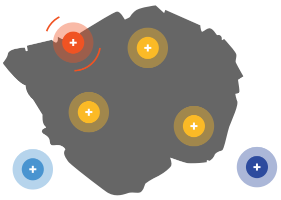
With the GSP, queries can be run across the full dataset, including historical data. The following questions can be answered by querying the GSP:



- What unique entities were in the area at a certain time?
- What entities visited a location of interest, and for how long?



- What is the historic movement of those entities?
- Which entities are deviating from their historical behavior?



- Alert me when certain entities enter, dwell or exit a location (geofences)

- What other entities were co-located or in proximity at the same time?
- What are their patterns of behavior?

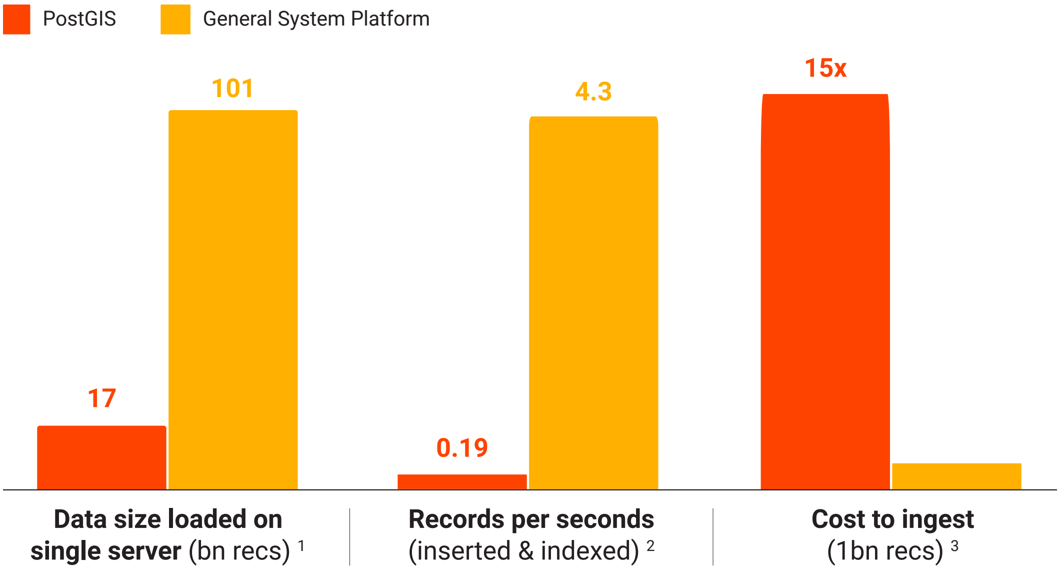
The GSP queries and support common aggregation predicates:

- Returning records
- Counting records
- Selecting the earliest or oldest record in a set
- Counting records by unique identifier

More predicates are in development and will be available in the future.

Case study

A customer had an existing solution to analyze real-time mobility data, using PostGIS as their datastore. Unfortunately, the solution could not scale to 100bn records which is what they needed. They ran a project to replace PostGIS with General System Platform and we observed the following comparative metrics:



[1] The existing solution could not work with more than 17bn records. Once migrated to the GSP, they could easily ingest over 100bn records. The PostGIS Server used was an AWS db.r6g.16x large instance. The GSP server is an AWS i3en.12x large instance.

[2] GSP ingested on average 4.3m records per second vs. 190k records per second on PostGIS. In both cases the source data was uncompressed, pre-processed files stored on AWS S3.

[3] The GSP ingestion, including ETL pipeline costs, was 15 times cheaper than PostGIS.

Use cases

Logistics

- Detecting unexpected events (e.g. stops other than pick ups or drop offs)
- Drivers' wellbeing, e.g. checking they take a break every X hours

Home Delivery

- Help balance supply of delivery vehicles and orders in real time by detecting hotspots
- Real-time, localized alerts to drivers from multiple datasets (e.g. traffic, weather, events)

Retail

- Recommending what consumers near you are buying right now
- Footfall analysis on historical mobility datasets

Media / Advertising

- Measuring exposure of Out of Home advertising
- Computing optimal day / time to buy media

Healthcare

Co-locate persons exposed (co-located) to a positive patient to recommend isolation

Fraud detection

Correlate current translation location & delivery address and past transactions locations

Data sources

Even with existing data pipelines, storage solutions and applications, the General System Platform sits alongside existing investment and delivers further value. The GSP is designed to ingest data from streaming data sources such as Kafka or Kinesis, and from data at rest, such as databases or data lakes. Some of our customers ingest both types of data, as historical data is often relevant to understand their real-time feeds.

The GSP API enables the creation and management of ingestion jobs, via a dedicated GSP Import API. You can flexibly specify the schema of your source data, map the relevant fields that the GSP will ingest and monitor the status of the ingestion. Out-of-the box, the GSP API supports AWS S3 data lakes; the API can be used to integrate with other data sources or legacy applications you may have.

To achieve optimal query performance, the GSP requires data to be ingested in rough time order. Data is typically available in this form from streaming sources and, more generally, from sensor-generated data. Ingestion speed is unaffected by time-ordering.

Integration and deployment models

The GSP has a standard web API that is used to query it. Using the API, our customers can build applications, such as:

Mobile or web apps

Add real-time updates / more precise ETAs / etc.

Data science and analytics

Integrate with BI tools: root-cause analysis, what-if scenarios

Business events processing

Enhance logic with spatiotemporal context: e.g. Is this entity deviating from its historical behaviour?

ML workloads

Retrieve data faster during training or inference

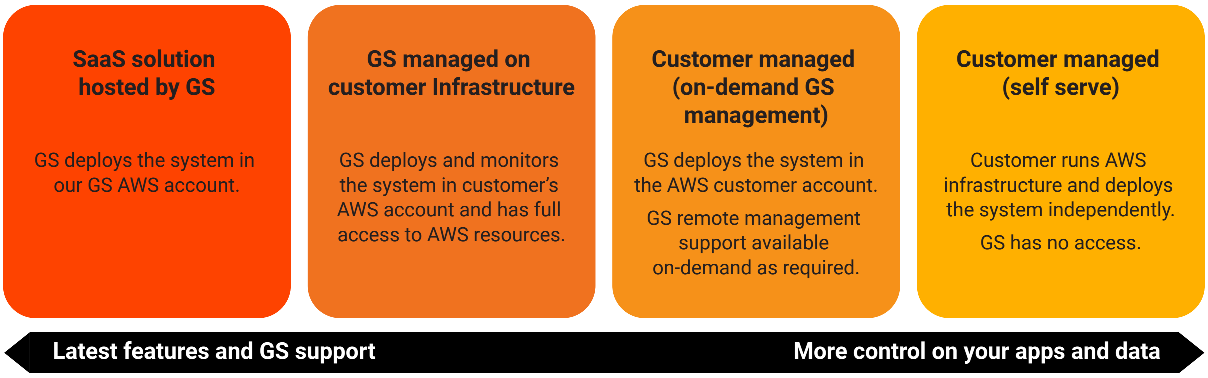
The GSP web API can be integrated with a language of your choice. We also provide an easy-to-use client-side Python SDK.

The Python SDK is used in particular to support data science use cases. It integrates out-of-the-box with popular BI tools such as Jupyter Notebooks and it can be used for interactive data discovery, visualization, model fine-tuning and rapid prototyping.

Deployment model

Many of our corporate customers prefer to consume the GSP API using the GS-hosted SaaS, as this is convenient and cost-effective. In this model, GS manages all the infrastructure on behalf of the customer, who retains full control of their data. Users can be added or removed for an organization as needed, and granular permissions can be granted, for instance, to implement a RBAC[1] model.

In the table below, we present the options available:



SaaS solution hosted by GS	GS managed on customer Infrastructure	Customer managed (on-demand GS management)	Customer managed (self serve)
GS deploys the system in our GS AWS account.	GS deploys and monitors the system in customer's AWS account and has full access to AWS resources.	GS deploys the system in the AWS customer account. GS remote management support available on-demand as required.	Customer runs AWS infrastructure and deploys the system independently. GS has no access.

Latest features and GS support
More control on your apps and data

At present, the GSP platform is available on the AWS Cloud.
We are considering adding support for other cloud services based on customer demand. Please get in touch with any specific requirements, and we will be happy to discuss options.

[1] Role Based Access Control

GENERAL SYSTEM

About General System

General System is an international company staffed by world-class data scientists, data engineers, software developers, technical innovators and sector-specific professionals.

Contact us today to arrange a live demonstration, or visit us here.