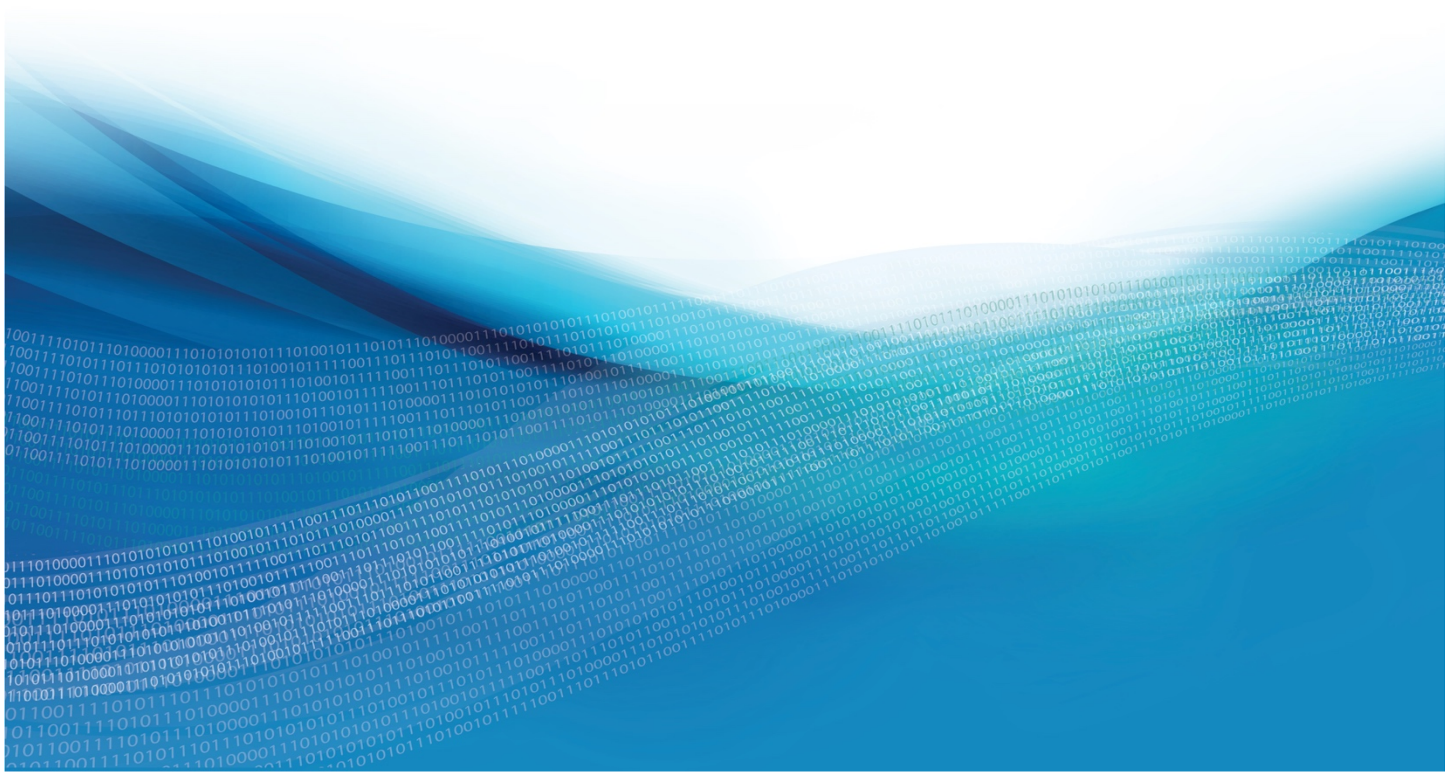
White Paper

12 Best Practices for Modern Data Ingestion



StreamSets

Introduction: Treat data movement as a continuous, ever-changing operation and actively manage its performance.

Before big data and fast data, the challenge of data movement was simple: move fields from fairly static databases to an appropriate home in a data warehouse, or move data between databases and apps in a standardized fashion. The process resembled a factory assembly line.

In contrast, the emerging world is many-to-many with streaming, batch or micro-batched data coming from numerous sources and being consumed by multiple applications. Big data processing operations are more like a city traffic grid — a network of shared resources — than the linear path taken by traditional data. In addition, the sources and applications are controlled by separate parties, perhaps even 3rd parties, so when the schema or semantics inevitably change, something known as data drift, it can wreak havoc with downstream analysis

Because modern data is so dynamic, dealing with data in motion is not just a design-time problem for developers, but also is a runtime problem requiring an operational perspective that must be managed day-to-day and evolved over time. In this new world, organizations must architect for change and continually monitor and tune the performance of their data movement system.

StreamSets, the provider of the industry's first data operations platform, offers the following 12 best practices as practical advice to help you manage the performance of data movement as a system and elicit maximum value from your data.

1 Limit Hand Coding As Much As Possible

It has been commonplace to write custom code to ingest data from sources into your data store. This practice is dangerous given the dynamic nature of big data. Custom code creates brittleness in dataflows where minor changes to the data schema can cause the pipeline to drop data or fail altogether. Also, since instrumentation has to be explicitly designed in and often isn't, dataflows can become black boxes offering no visibility to pipeline health. Lastly, low-level coding leads to tighter coupling between components, making it difficult to upgrade your infrastructure and stifling organizational agility.

Today, there are modern data ingest systems that create code-free plug-and-play connectivity between data source types, intermediate processing systems (such as Kafka and other message queues) and your data store. The benefits you get from such a system are flexibility instead of brittleness, visibility instead of opacity, and the ability to upgrade data processing components independently. If you're worried about customization or extensibility, these tools usually augment their built-in connectors with support for powerful expression languages or the ability to plug in custom code.

2 Minimize Schema Specification; Be Intent-Driven

While it is a standard requirement in the traditional data world, full schema specification of big data leads to wasted engineering time and resources. Consuming applications often make use of only a few key fields for analysis, plus big data sources often have poorly controlled schema that change over time and force ongoing maintenance.

Rather than relying on full schema specification, dataflow systems should be intent-driven, whereby you specify conditions for, and transformations on, only those fields that matter to downstream analysis. This minimalist approach reduces the work and time required to develop and implement pipelines. It also and makes dataflows more reliable as there is less to go wrong.

3 Plan for Both Streams and Batch

Despite all of the hub-bub about streaming analytics, enterprise data it is still a batch-driven world based on applications and source databases developed over the past 30 years. So while you are planning for cybersecurity, IoT and other new age applications that capitalize on streams, you must account for the fact that this data often needs to be joined with or analyzed against batch sources such as master or transactional data. Rather than setting up a streaming-only framework, practical needs demand that you incorporate streaming into the legacy batch-driven fabric while maintaining or improving performance and reliability of the overall data operation.

4 Sanitize Raw Data Upon Ingest

The original mantra for early Hadoop users was that you should store only immutable raw data in your store. As technology meets the real world, we have learned that there are some serious downsides to not sanitizing your data upon ingest. Raw data, like untreated water, can make you sick. This approach is what has spawned the “data swamp” metaphor from Gartner and others. Removing this risk by having data scientists clean the data for each consumption activity is a common approach but is clearly an inefficient use of resources. Plus, storing raw inputs invariably leads you to have personal data and otherwise sensitive information in your data lake which increases your security and compliance risk.

With modern dataflow systems you can and should sanitize your data upon ingest. Basic sanitization includes simple “row in, row out” transformations that enforce corporate data policies and normalize or standardize data formats. More advanced sanitization includes rolling average and other time-series computations the results of which can be leveraged broadly by data scientists and business analysts.

Sanitizing data as close to the data source as possible makes data scientists much more productive letting them focus on use-case specific “data wrangling” rather than reinventing generic transformations that should be centralized and automated.

5 Address Data Drift to Ensure Consumption-Ready Data

An insidious challenge of big data management is dealing with data drift; the unpredictable, unavoidable and continuous mutation of data characteristics caused by the operations, maintenance and modernization of source systems. It shows up in 3 forms: structural drift (changes to schema), semantic drift (changes to meaning) or infrastructure drift (changes to data processing software, including virtualization, datacenter and cloud migration).

Data drift erodes data fidelity, reliability of your data operations and ultimately the productivity of your data scientists and engineers. It increases your costs, delays your time to analysis and leads to poor decision making based on polluted or incomplete data.

If your end goal is to democratize data access by having as much data as possible available to as many users as possible, say through Hive or Impala queries, then you should look for data movement tools and systems that can detect and react to changes in schema and keep the Hive metastore in sync, or at the very least alert you to changes.

6 Don’t Count on Managed File Transfer

New data sets are often unbounded and continuous, such as ever-changing logs, clickstreams and IoT sensor output. Use of managed file transfers or other rudimentary mechanisms for these dynamic source types creates a fragile architecture that will require constant maintenance to remain viable.

Files, because their contents vary in size, structure and format, are challenging to introspect on the fly. This means you lose visibility into changes that should be communicated to consuming systems and applications.

If you’re intent on relying on a file transfer mechanism, consider pre-processing the files to standardize the data format to simplify inspection and profiling, or adopt an ingestion tool or framework that does this for you.

7 Instrument Everything in Your Dataflows

You can never have enough visibility in a complex dataflow system. End-to-end instrumentation of your data movement gives you a window into performance as you contend with the challenge of evolving sources and systems. This instrumentation is not just needed for time-series analysis of a single dataflow to tease out changes over time. It can - more importantly - help you correlate data across flows to identify interesting events in real time.

Organizations should endeavor to capture details of every aspect of the overall dataflow architecture while minimizing overhead or tight coupling between systems. A well-instrumented approach will asynchronously communicate the measured values to external management systems and allow you to drill down from coarse metrics used for monitoring to the fine-grained measurements ideal for diagnosis, root cause analysis and remediation of issues.

8 Don’t Just Count Packages; Inspect Contents

Would you feel secure if airport security solely counted passengers and luggage rather than actually scanned baggage for unusual contents? Of course not, yet the traditional metrics for data ingestion are throughput and latency. The reality of data drift means that you’re much better off if you profile and understand the values of the data itself as it flows through your infrastructure. Otherwise you leave yourself at risk to unannounced changes in data format or meaning. A major change in data values might indicate a true change in the real world that is interesting to the business, or might indicate undetected data drift that is polluting your downstream analysis.

An additional benefit of data introspection is that it allows you to identify personal or otherwise sensitive data transiting your infrastructure. Many industries and geographies have strict requirements around storage of personal data, such as the EU’s 2018 “right to be forgotten” GDPR requirements. Continually monitoring incoming data for patterns helps companies comply by providing real-time detection and tracking of any personal data they are collecting and storing.

9 Implement a DevOps Approach to Data Movement

The DevOps sensibility of an agile workflow with tight linkages between those who design a system and those who run it is well-suited to big data movement operations. Data pipelines will need to be adjusted frequently in a world where there is a continual evolution of data sources, consumption use cases and data-processing systems.

Traditional data integration systems date back to when the waterfall development methodology was king and tools from that era tend to focus almost exclusively on the design-time problem. This is also true of the early big data ingest developer frameworks such as Apache Sqoop and Apache Flume. Fortunately, there are now modern dataflow tools that provide an integrated development environment (IDE) for continual use through the evolving dataflow life cycle.

10 Decouple Data Movement from Your Infrastructure

Unlike monolithic solutions built for traditional data architectures, big data infrastructure requires coordination across best-of-breed - and often open source - components for specialized functions such as ingest, message queues, storage, search, analytics and machine learning. These components evolve at their own pace and need to be upgraded based on business needs. Thus, the large and expensive lockstep upgrades you're used to in the traditional world are being supplanted by an ongoing series of one-by-one changes to componentry.

To keep your data operation up to date in this brave new world, you should use a data movement system that acts as a middleware layer and keeps each system in the data movement chain loosely coupled from its neighbors. This enables you to modernize *a la carte* without having to re-implement foundational pieces of infrastructure.

11 Engineer for Complex Deployment Patterns

Not only have dataflows become complex but they now span a range of deployment alternatives. Industry surveys confirm that enterprise are expecting to deploy data across multiple clouds while still retaining on-premise data operations. And edge operations are morphing from simple collection to include simple or complex processing depending on the device constraints, urgency and robustness of connectivity. Since each deployment option has it's own advantages, it is a mistake to expect a single approach to work now and forever. Realistically, business requirements will dictate an enterprise architecture that combines many of them.

Regardless of where you are in your journey, it is best to assume a world where you have data stored in many different environments and build an architecture based on complete "workload portability" where you can move data to the point of analysis based on the best price and performance characteristics for the job, and to do so with minimal friction. Also, you should assume that the constellation that describes your multi-cloud will change over time as cloud offerings and your business needs evolve.

12 Create a Center of Excellence for Data in Motion

The movement of data is evolving from a stovepipe model to one that resembles a traffic grid. You can no longer get by with a fire-and-forget approach to building data ingestion pipelines. In such a world you must formalize the management (people, processes and systems) of the overall operation to ensure it functions reliably and meets internal SLAs on a continual basis. This means adding tools that provide real-time visibility into the state of traffic flows, with the ability to receive warnings and act on issues that may violate contracts around data delivery, completeness and integrity.

Otherwise you are trying to navigate a busy city traffic grid with ever-changing conditions using a paper map, with the risk that the data feeding your critical business processes and applications arrives late, incomplete or not at all.

Let StreamSets Help You Build a High-Performance Data Ingestion Practice

To help enterprises implement many of the best practices above, StreamSets has developed the industry's first data operations platform. It helps enterprises master the life cycle of their data movement, including efficient development, operational visibility and tight control over performance.

The StreamSets platform combines StreamSets Data Collector™ , award-winning open source software for the development of batch and streaming data pipelines, with StreamSets Dataflow Performance Manager (DPM™) , "air traffic control" for managing the real-time performance of all dataflows from a single point.

Key features of the platform include:

Smart pipelines to conquer data drift – Inspects data while it is in motion and detects and resolves unexpected changes on the fly.

A living data map to conquer data sprawl – Displays all data movement on a single canvas. Its ability to auto-update brings continuous integration and continuous deployment (CI/CD) methods to data flows.

Data SLAs to conquer data urgency - Sets and enforces rules around dataflow performance to ensure that business rules for quality and timeliness are met.

To learn more about StreamSets, its products and use cases, please visit www.streamsets.com.