بسمه تعالی

تحلیل پروژه‌های پردازش جریان آپاچی

در حوزه پردازش داده، دو نوع اصلی پردازش داریم :

- پردازش بلادرنگ (Real Time) یا همان پردازش جریان (Stream Processing) و
- پردازش انبوه (Batch Processing)

که فناوری‌های اصلی حوزه کلان داده ، مانند روش توزیع و تجمیع(MapReduce) ، و جدیدا اسپارک برای پردازش انبوه داده‌ها طراحی شده‌اند و برای پردازش بلادرنگ داده‌ها هم برای سالها، استفاده از صفهای توزیع شده و پروژه‌های محدودی مانند Storm راه حل اصلی مهندسین داده بود.

در سال‌های اخیر ، فناوری‌های پردازش بلادرنگ و داده های جریانی مانند داده‌های دریافتی از حسگرها و تصاویر ترافیک و ماهواره، داده‌های شبکه‌های اجتماعی و مانند آن که یکسره در حال تولید هستند و جریان آنها به صورت پیوسته درحال تزریق به برنامه‌های پردازشی است، پیشرفت زیادی کرده‌اند و فقط در اکوسیستم آپاچی (مجموعه پروژه های بنیاد آپاچی) امروزه بیش از ده پروژه مختلف متن باز مختلف در این حوزه داریم بعضی از آن‌ها، تفاوت بسیار کمی با یکدیگر دارند که این امر، انتخاب درست ابزار و کتابخانه‌های مورد نیاز برای پردازش جریان را امری زمان بر و تخصصی نموده است.

با هدف سهولت تصمیم‌گیری مهندسین داده، وبلاگ DataBaseLine در اقدامی تحسین برانگیز، این فناوری‌ها را در یک جدول با هم مقایسه کرده است که آنرا در زیر می‌توانید مشاهده کنید.

	Flume	NiFi	Apex	Kafka Streams	Spark Streaming	Storm	Storm + Trident	Samza	Flink	Ignite Streaming	Beam
Current version	1.6.0	0.5.1	incubating	0.9.0.1* (available in 0.10)	1.6.1	0.10.0	0.10.0	0.10.0	1.0	1.5.0	incubating
Category	DC/SEP	DC/SEP	DC/ESP	ESP	ESP	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	ESP/CEP
Event size	single	single	single	single	micro-batch	single	mini-batch	single	single	single	single
Available since (incubator since)	June 2012 (June 2011)	July 2015 (Nov 2014)	(Aug 2015)	Apr 2016 (July 2011)	Feb 2014 (2013)	Sep 2014 (Sep 2013)	Sep 2014 (Sep 2013)	Jan 2014 (July 2013)	Dec 2014 (Mar 2014)	Sep 2015 (Oct 2014)	(Feb 2016)
Contributors	11	67	53	160	838	206	206	48	159	56	74
Main backers	Apple Cloudera	Hortonworks	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GridGain	Google
Delivery guarantees	at least once	at least once	exactly once	at least once	exactly once at least once (with non-fault-tolerant sources)	at least once	exactly once	at least once	exactly once	at least once	exactly once
State management	ZooKeeper	local and distributed snapshots	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots (fault-tolerant)	distributed snapshots	checkpoints	transactional updates
Fault tolerance	yes (with file channel only)	yes	yes	yes	yes	yes	yes	yes	yes	yes	N/A
Out-of-order processing	no	no	no	yes	no	yes	yes	yes (but not within a single partition)	yes	yes	yes
Event prioritization	no	yes	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
Windowing	no	no	time-based	time-based	time-based	time-based count-based	time-based count-based	time-based	time-based count-based	time-based count-based	time-based
Back-pressure	no	yes	yes	N/A	yes	no	no	yes	yes	yes	yes
Primary abstraction	Event	FlowFile	Tuple	KafkaStream	DStream	Tuple	TridentTuple	Message	DataStream	IgniteDataStreamer	PCollection
Data flow	agent	flow (process group)	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
Latency	low	configurable	very low	very low	medium	very low	medium	low	low (configurable)	very low	low
Resource management	YARN	native	YARN	Any process manager (e.g. YARN, Mesos, Chef, Puppet, Salt, Kubernetes, ...)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN Mesos	N/A
Auto-scaling	no	no	yes	yes	yes	no	no	no	no	no	no
In-flight modifications	no	yes	yes	yes	no	no	no	no	no	no	no
API		compositional	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative	declarative
Primarily written in	Java	Java	Java	Java	Scala	Clojure	Java	Scala	Java	Java	Java
API languages	text files	GUI	Java	Java	Scala Java Python	Scala Java Clojure Python Ruby	Java Python Scala	Java	Java Scala	Java .NET C++	Java

اگر بخواهیم سیستم‌های پردازش جریان را گروه بندی کنیم، می توانیم طبقه بندی زیر برای آنها در نظر بگیریم:

جمع آوری داده مانند NiFi و Flume که به آنها اختصاراً DC می‌گوییم .

پردازش تک رخداد (SEP)

پردازش جریان رخدادی (ESP)

سامانه های حرفه ای پردازش رخداد(CEP – Coplex Event Processing)

البته، سامانه‌های حرفه‌ای پردازش داده که علاوه بر پردازش رخداد، پردازش انبوه را هم در بر می‌گیرند و یک راه‌حل جامع برای پردازش انواع داده‌ها پیشنهاد می‌کنند مانند Apache Spark,Apache Ignite,Apache Apex هم می‌توانند به لیست فوق اضافه شوند. **Kafka Streams**هم هر چند هنوز قانونی عرضه نشده است اما چون در ماه جاری وارد بازار خواهد شد در جدول فوق ، آمده است.

برخی ستون ها یا مواردی که در جدول بالا آمده است از قرار زیر است:

Back Pressure:
منظور میزان فشار و باریست که برای پردازش جریان ، بر سیستم وارد می شود و مجموعه میزان مصرف رم و سی پی یو و ... را تشکیل می دهد.

Auto-Scaling:
یا مقیاس پذیری افقی ناظر به گسترش افقی سامانه است یعنی بسته به نیاز ما، با افزودن یک سیستم به سامانه ، به طور خودکار و بدون دردسر ، مجموعه گره های پردازشی یک عدد اضافه شود و پردازش، این سیستم را هم سریعاً در بر گیرد.

In Flight Modification:
قابلیت تغییر داده ها قبل از شروع پردازش بدون نیاز به ارسال مجدد آنها یا توقف سیستم Spooker . از طلایه داران این قابلیت است.

Event Size:
منظور این است که هر رخداد جداگانه بررسی می‌شود (Single) یا هر چند تا رخداد با هم دیگر یک دسته را تشکیل می دهند و بعد برای پردازش آماده می‌شوند (MicroBatch)

Delivery Gurantees
که نحوه تضمین پردازش هر رخداد را بیان می‌کند که بسته به نیاز و نوع کاربرد، باید حتما این پارامتر مد نظر باشد. مثلا برای پردازش سنسورهای غیر حیاتی، حداکثر یک بار هم کفایت می‌کند اما برای داده‌های حساس، حداقل یک بار باید هر رخداد، بررسی و پردازش شود.