

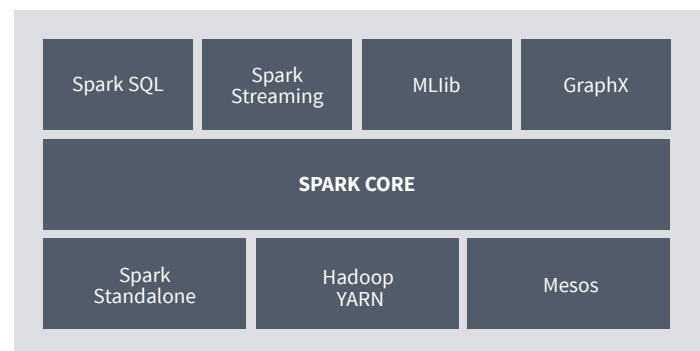
Unravel® for Spark

Apache Spark

There's a deluge of data generated around us from a multitude of online, IoT, health, financial services, and scientific computing sources, just to name a few. It is ever increasing in **volume, velocity, and variety**. Furthermore, ensuring **veracity** and the need for faster processing/analytics is critical to extracting business value from this data for a wide range of workloads and applications. From product usage reports to recommendation engines, enterprises are now running various types of **Big Data Applications** in production to provide their customers greater value than ever before.

Legacy compute paradigms and platforms are grossly inadequate in handling such stringent constraints for processing big data at scale. Hence the ongoing trend to distribute big data applications over large compute clusters.

Apache Spark™ is a powerful open-source cluster/distributed computing framework for scalable and efficient analysis of big data applications that runs on commodity computer hardware clusters. Spark provides an interface for programming entire clusters with built in data parallelism and fault tolerance while hiding the underlying complexities of utilizing distributed systems.



Spark has seen rapid adoption by enterprises on a massive scale across a wide swath of verticals, applications, and use-cases. Spark is also replacing MapReduce as the processing engine component of Hadoop™ for all practical purposes.

BENEFITS OF SPARK

The **benefits** of Spark include speed (up to 100X faster in-memory execution engine than Hadoop MapReduce) and easy access to all Spark components (write applications in R, Python, Scala, and Java quickly) via unified high-level APIs. Spark can also handle a wide range of workloads (ETL, business intelligence, analytics, machine learning, graph processing, etc.) and perform interactive SQL queries, batch processing, streaming data analytics, and data pipelines.

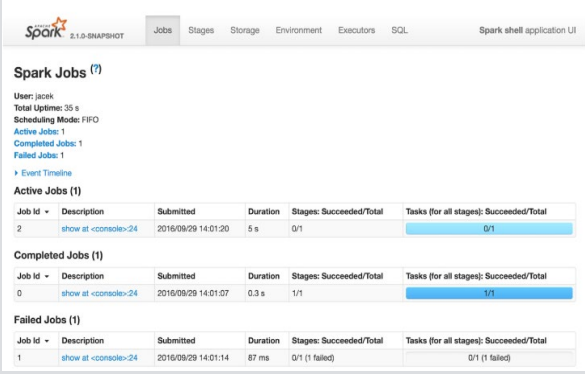
MONITORING CHALLENGES

Building big data apps on Spark that monetize and extract business value from data have become a default standard in larger enterprises. While Spark offers tremendous ease of use for developers and data scientists, deploying, monitoring, and optimizing production apps can be an altogether complex and cumbersome exercise. These create significant challenges for the operations team (and end-users) who are responsible for managing the big data apps holistically while addressing many of the business requirements around SLA Management, MTTR, DevOps productivity, etc.

Tools such as Apache Ambari™ and Cloudera® Manger™ primarily provide a systems view point to administer the cluster and measure metrics related to service health/performance and resource utilization. They only provide high-level metrics for individual jobs and point you to relevant sections in YARN or Spark Web UI for further debugging and troubleshooting. A guided path to address issues related to missed SLAs, performance, failures and resource utilization for big data apps remains a huge gap in the ecosystem.

Spark Web UI is the default web interface available in Spark to monitor and inspect jobs. While it provides details for jobs, stages, storage, environment, executors, logs etc., users have to jump across too many screens/tabs to observe, assimilate and digest the disparate pieces of data, potentially missing critical and actionable information. In many organizations, due to security and access control reasons, Spark Web UI could also be locked down for end-users lowering the visibility of app performance and hampering remedial measures to address issues.

Spark uses a master/worker architecture consisting of a “driver” and many “executors”. As program code run on these “executors”, copious amount of logs are generated that have information about the application, as well as how the application interacts with the rest of the Spark platform. Root cause analysis of an application issues and failures from these raw, verbose, messy, and distributed logs is arduous task even for Spark experts; leave alone the many new users coming to the platform



The screenshot shows the Spark Web UI interface. At the top, there are navigation tabs for Jobs, Stages, Storage, Environment, Executors, and SQL. Below the navigation, the page title is "Spark Jobs (7)". The user is identified as "User: jack". Summary statistics include "Total Uptime: 35 s", "Scheduling Mode: FIFO", "Active Jobs: 1", "Completed Jobs: 1", and "Failed Jobs: 1". There is a link to "Event Timeline".

Under "Active Jobs (1)", there is a table with one row:

Job id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	show at <<console>-24	2016/09/29 14:01:20	5 s	0/1	0/1

Under "Completed Jobs (1)", there is a table with one row:

Job id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	show at <<console>-24	2016/09/29 14:01:07	0.3 s	1/1	1/1

Under "Failed Jobs (1)", there is a table with one row:

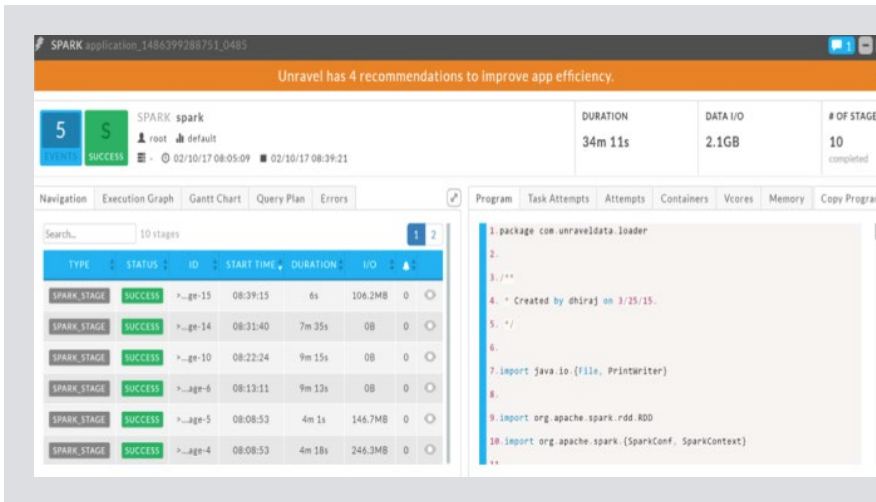
Job id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	show at <<console>-24	2016/09/29 14:01:14	87 ms	0/1 (1 failed)	0/1 (1 failed)

with limited or no knowledge of distributed systems. Stitching together a cohesive view of apps, services, and infrastructure from such a fragmented set of tools can cause **multiple blind spots** for all stakeholders. Monitoring such blind spots become critical especially when scaling production grade multi-tenant applications. A comprehensive and easy to use approach is necessary to capture the essence of big data Spark apps, analyze and provide recommendations to fix issues with performance, bugs, and inefficiencies.

Unravel® for Spark

Enterprises have increasingly adopted distributed computing systems to build and deploy cloud-native big data apps. Enterprises have to innovate such that Operations teams spend less time figuring out how well open source components work together and more on ensuring improved SLA management, MTTR, and efficiency of apps in production. To that end, a modern generation of Application Performance Management (APM) software has emerged to support analytical, machine learning, IoT, and artificial intelligence applications running on big data platforms such as Spark, Hadoop, Kafka, and NoSQL.

Unravel for Spark provides a comprehensive full-stack, intelligent, and automated approach to Operations and application performance management across the big data architecture. The Unravel platform helps you to analyze, optimize, and troubleshoot big data applications and operations in a seamless, easy to use, and frictionless manner.



The Unravel for Spark provides a detailed view into the behavior of Spark applications. It's a one-stop shop to:

- Resolve inefficiencies, bottlenecks, and reasons for failure within applications
- Optimize resource allocation for Spark executors
- Detect and fix poor partitioning
- Detect and fix inefficient and failed Spark apps
- Tune JVM settings for driver and executors

Unravel for Spark bridges the gaps that exist in current fragmented approaches for big data APM and addresses the blind spots from an applications usage perspective. For example, one of the most common pitfalls in running big data apps is to use too many tasks for a query. This increases overhead, wait time for resource allocation, and ergo the duration of the query. It also causes resource contentions at the cluster level, affecting other concurrent queries. Unravel's unique AI driven intelligence provides insights into such queries and provide recommendations for significant app speedup.

Irrespective of the type of Spark job, the **Application Manager** view is split into two. **Errors, Logs, and Configurations** can be found on the left, and every job has **Program, Task Attempts, Graphs** [sub tabs include **Attempt, Containers, Vcores, Memory**], and Resource. The Unravel Web UI **Applications** tab for Spark provides a 360° view of an app on a single glass pane with all relevant and correlated information:

Locate and analyze Spark app performance

- Supports both ad-hoc apps and oft-running workflows or data pipelines
- Display key performance indicators specific to an application, such as status, duration, data I/O, # of stages, task distribution, cross annotation on Scala/R/Python code view
- Drill down views from Spark jobs to stages to task execution; Gantt chart of stage timelines to drilling down bottlenecks, Errors, task logs of drivers and

executors, and configurations; Spark stage timeline distribution charts

- CPU and OS Memory resource view for drivers and executors

AI driven Intelligence engine provides insights into:

- Utilization of memory resources
- Utilization of Spark storage memory
- Opportunity for RDD Caching
- Contention for CPU resources
- Underutilization of container resources

AI driven recommendation engine to improve app efficiencies in simple English

- Spark executor memory new value
- Spark executor instances new value
- Spark default parallelism new value

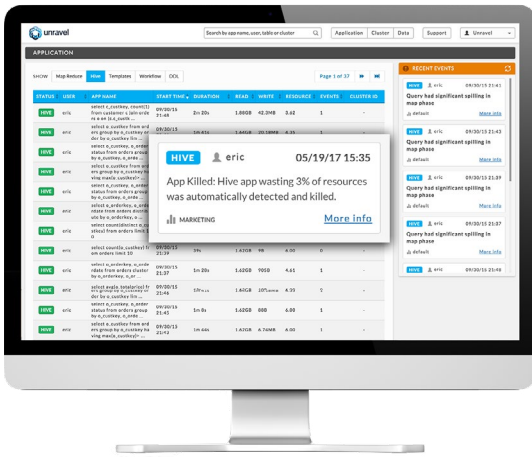
For failing and inefficient Spark Apps, Unravel provides

- Rich error view to support a root cause analysis— which metric to look at and what parameter needs to change to get the app running again
- Monitoring Spark pipelines when there's unpredictable performance
- **Auto Actions** feature to send pro-active alerts and enforce policies created to automatically detect and eliminate rogue apps, fix problems so that clusters, resource/cost optimizations, and SLA needs can be met; enforce best practices

BUSINESS BENEFITS OF USING UNRAVEL

Unravel makes sure businesses can achieve their goals with big data spark applications in a multi-tenant deployment whether on-premise, cloud, or hybrid environments. Some of the tangible benefits, include:

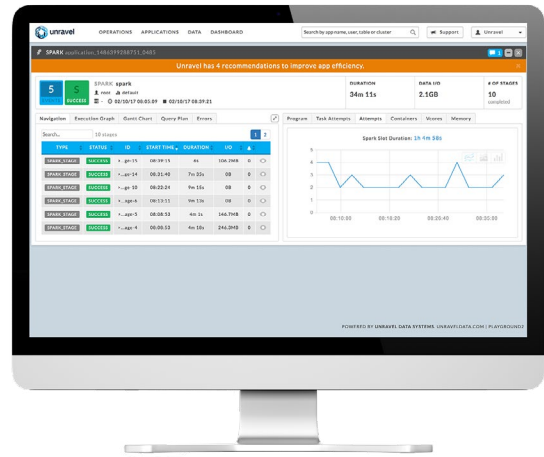
- Improved reliability as a result of meeting SLAs and ensuring no disruption in revenues—100% Apps on-time
- Optimized resource utilization ensures lower infrastructure and project costs—up to 60% reduction
- Improved productivity of all stakeholders—up to 98% reduction in MTTR



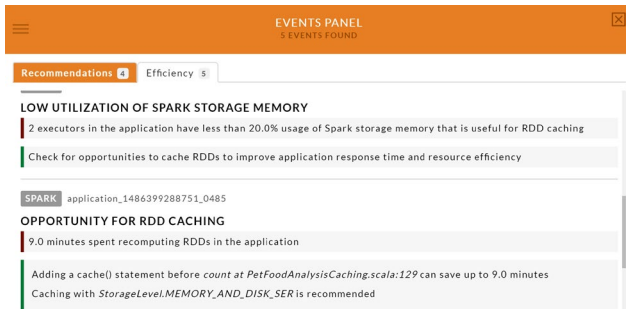
Enforcing automated policies and actions to meet SLA needs, diagnose and kill rogue apps

Sample business benefits in finance and marketing use-cases:

- Enhanced security and reliability for improved risk management in fraud detection apps
- Better response times, regulatory compliance, and meet deadlines for ETL before market close in financial apps
- Enhance customer experience and improve targeted marketing for better Rol



Built-in intelligence KPIs, tasks, jobs, stages related to an application (left) Attempts, containers, memory (right)



Events/Recommendations/Efficiency: query used too many reducers. Unravel shows that this query has one job using too many reduce tasks, and provides a recommendation for the configuration

OTHER FEATURES INCLUDE:

- Errors and warning messages
- Search for your app(s) in a variety of ways
- Graph the resources the application/job/stage consumed
- Histograms showing the distribution of map and reduce task duration, input and output size
- Donut graphs show the percentage of successful (green) and of failed (orange) tasks

