IN PRODUCTION

AppsFlyer cut Spark costs 50% without rewriting a job.

AppsFlyer brings in about 1 PB of new data every day. Its platform team had already cut infrastructure costs by moving to EMR Serverless. AppsFlyer reported 50% lower costs across its compaction jobs with Flarion.



APPSFLYER

Adi Belan
VP Engineering
Cloud, AI & Data

> “What surprised me was how fast it got to production. Bringing a new vendor into a data platform usually takes months. With Flarion, it was weeks.”

INDUSTRY	PLATFORM	WORKLOAD TYPES	CUSTOM CODE
Mobile marketing attribution	AWS EMR Serverless	• Compaction • Kafka-to-S3 streaming	Java/Scala UDFs in streaming

©2026 Flarion Inc. All rights reserved.

01 / THE STARTING POINT

AppsFlyer's platform was already optimized.

AppsFlyer moved from Hadoop to Kubernetes to EMR Serverless and had already right-sized resources for Spark. Further savings had to come from somewhere else.

Start with the hardest test.

AppsFlyer came to Flarion with its hardest workload. Compaction was its largest recurring workload: about 200 jobs, each running at least once a day and some every hour. Each sorted and shuffled recent data into larger Delta Lake files.

The platform team had already optimized the cluster settings and the compaction code, and didn't think there was much left to save. Flarion still cut the costs by 50%.

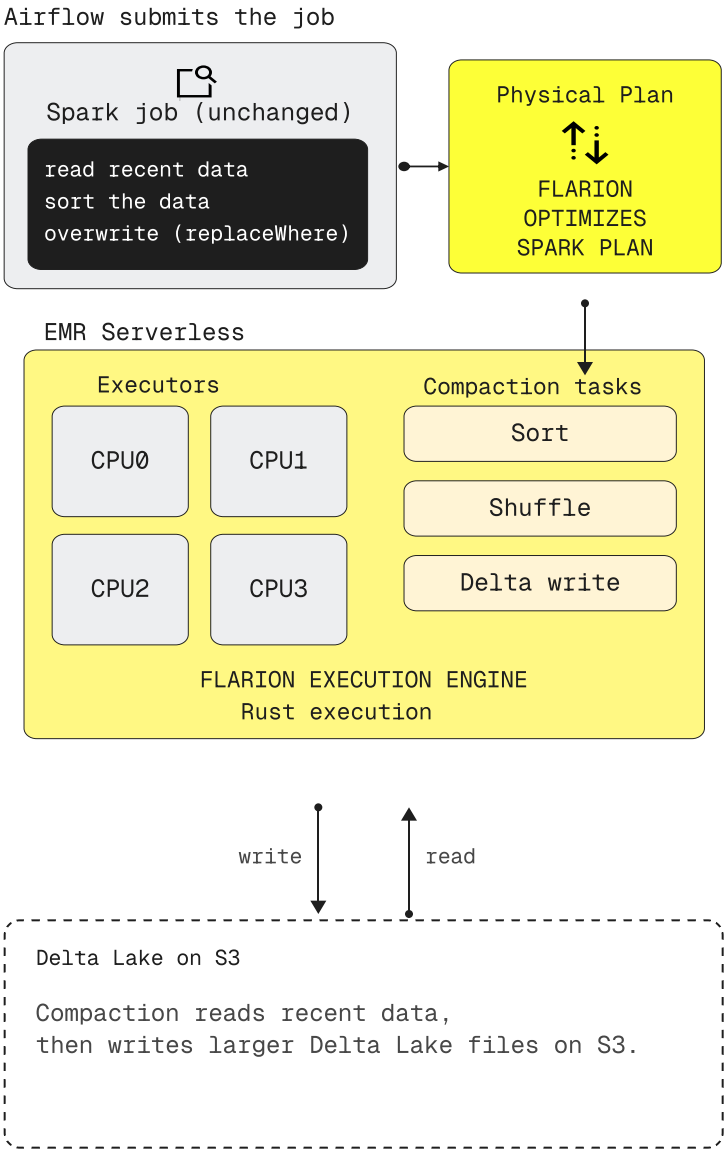
ALREADY OPTIMIZED	
COMPACTION	Custom code, not Delta OPTIMIZE
SORT	In the job, before every write
EXECUTORS	Released after one idle second
CLUSTERS	About 99% in use, by design

Lower costs, without a rewrite.

AppsFlyer did not want another integration project. Flarion offered a way to lower costs without rewriting the existing Spark jobs.

Flarion rebuilt the compaction workload in its lab from AppsFlyer's query plans. On Flarion's engine, it projected 40 to 50% lower cost. The next step was to prove that on AppsFlyer's jobs.

FLARION INSIDE A SPARK JOB

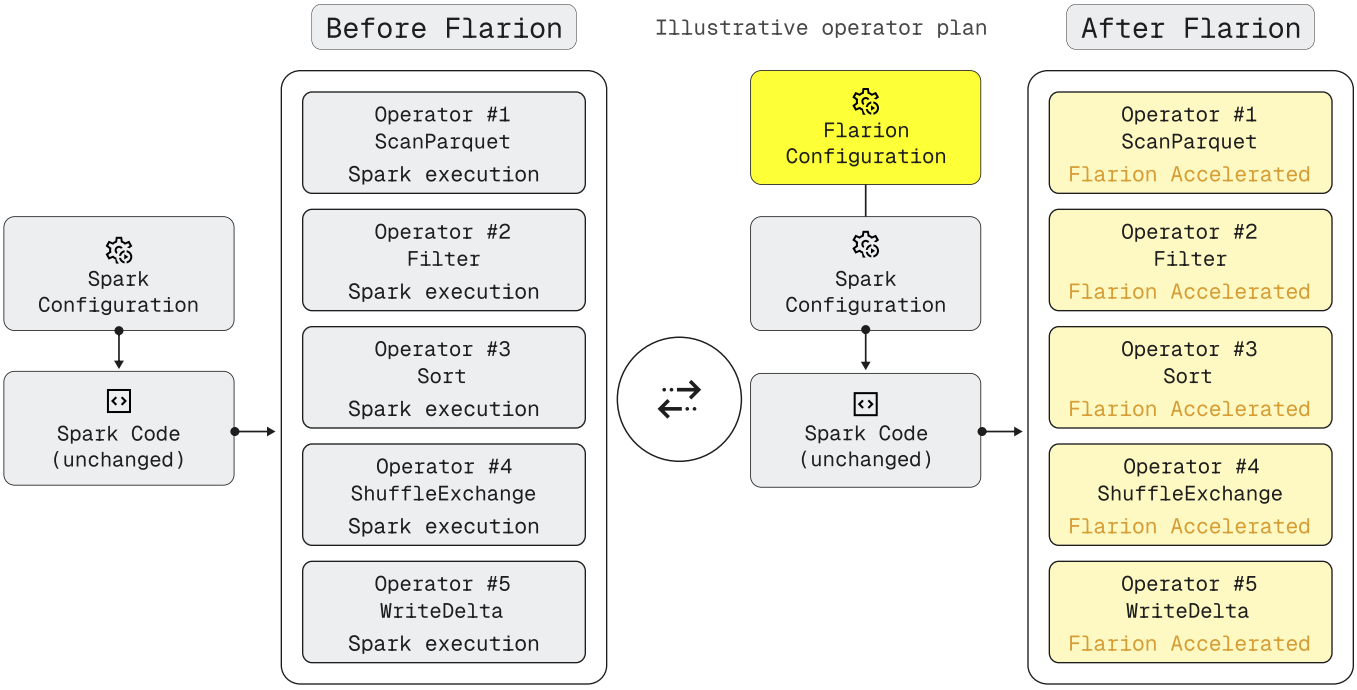


©2026 Flarion Inc. All rights reserved.

02 / THE IMPLEMENTATION

Flarion changed the engine. AppsFlyer kept its code.

One senior platform engineer worked with Flarion from the first test runs through production. The team added the engine through the container image, Spark settings and Airflow template it already used.



APPSFLYER'S CHANGES

AppsFlyer added two JARs to its EMR Serverless image, Flarion's engine and its Delta Lake integration, plus four lines of Spark configuration. The image passed the usual CI and security scan.

WHAT RAN WITH FLARION

Flarion's Rust engine ran every step of compaction: the read, the sort, the shuffle and the Delta write. AppsFlyer's jobs ran as written, with no code changes.

CHECKED BEFORE PRODUCTION

Results on two test workloads matched Spark's. Then all 200 compaction jobs moved to Flarion at once. Business data stayed in AppsFlyer's AWS account. Flarion did not access it.

Compaction went first. Streaming followed.

01

Assessment

An offline assessment of the compaction workload projected 40 to 50% lower cost.

02

1 week of telemetry

The plugin set a baseline of runtime and cost for each workload.

03

Production

All compaction jobs went into production with Flarion. Costs and run times fell from day one.

04

Streaming

Kafka-to-S3 jobs followed with no lag or incidents. Flarion's right-sizing guidance took the saving to 40 to 60%, depending on the workload.

©2026 Flarion Inc. All rights reserved.

03 / PROOF OF ROI

Switch it off, and the cost comes back.

With Flarion, AppsFlyer's compaction jobs cost 50% less. While upgrading to a newer EMR release, the team ran them without Flarion. Costs rose and failures came back.

LOWER COST · ALL COMPACTION JOBS

50% lower cost

AS MEASURED ON ACTUAL AWS BILL

COMPACTION JOBS

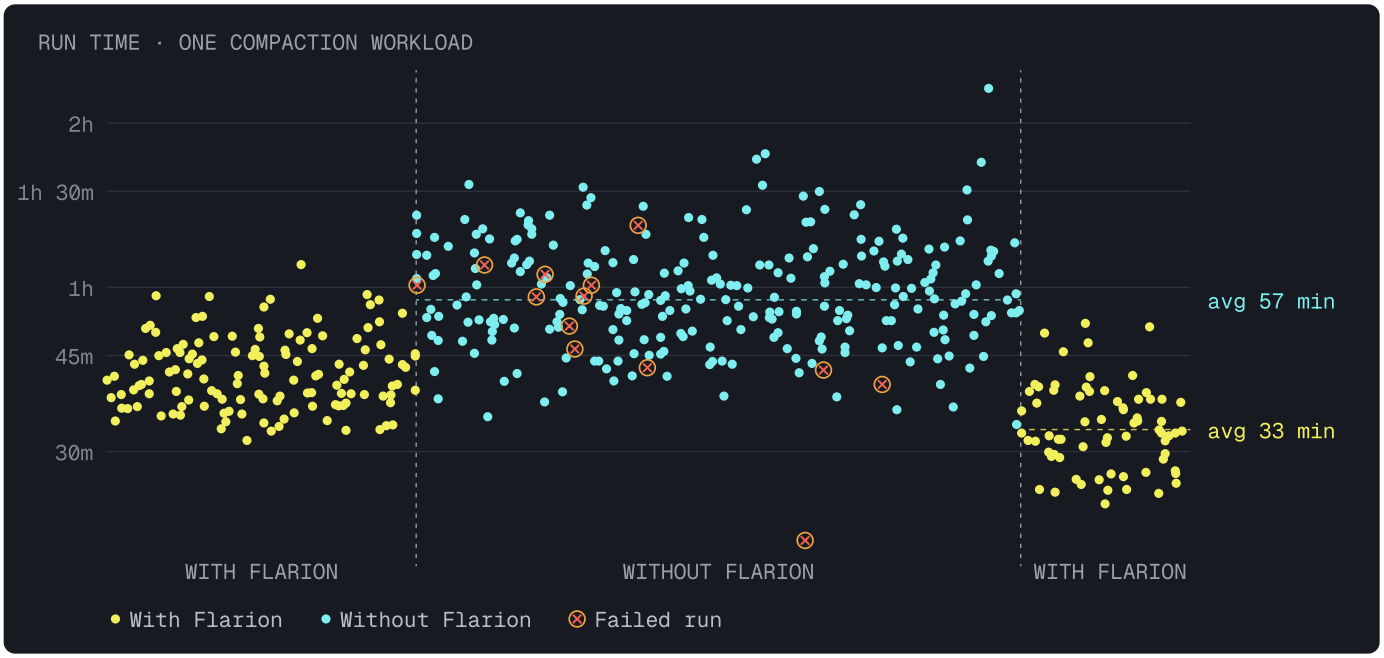
~200

ALL IN PRODUCTION WITH FLARION

FAILED RUNS · ONE COMPACTION WORKLOAD

0 failures

WITH FLARION



One compaction workload · Without Flarion during an EMR upgrade · Source: Flarion telemetry

What the upgrade showed.

During the move to a newer EMR release, the team ran its compaction jobs without Flarion. Costs rose and runs started failing. When Flarion went back in, run times dropped and the failures stopped.

Faster and more reliable with Flarion.

The chart follows one of about 200 compaction jobs. With Flarion, it ran with no failures. Without Flarion, runs averaged 57 minutes and started to fail. With Flarion back, runs averaged 33 minutes with no failures.

©2026 Flarion Inc. All rights reserved.

04 / THE RESULTS

The jobs cost less to run.

MEASURE	RESULT	CONTEXT
Compaction jobs	**50% lower**	AWS bill, reported by AppsFlyer, across all compaction jobs.
Code and Spark tuning	**0 changes**	No job rewritten, no Spark tuning. Flarion was added through the container image.
Kafka-to-S3 streaming	**40 to 60% lower**	Cloud cost on the Kafka-to-S3 streaming jobs, reported by AppsFlyer; the saving depends on the workload.

Ivan Kosianenko
Staff Engineer
EXCERPT

"No issues with Flarion. No lag or incidents during the switch..."

WHAT IT TOOK

One senior platform engineer worked with Flarion from the first test runs through production, using the team's existing image and job-submission process. Full rollout took weeks, and the savings started on day one: compaction cost less from its first production runs.

ROLLOUT SCOPE

Compaction went first, all of its jobs at once. Kafka-to-S3 streaming followed. Both run in production with Flarion.

Adi Belan
VP Engineering
Cloud, AI & Data

"More and more of our resources are going into AI, and AI depends on high-quality data. Making Spark faster and cheaper lets us put fewer resources into the data platform and more into AI."



WORKLOAD ASSESSMENT

Start with your Spark jobs.

Estimate the savings with an assessment, then validate the result on your own jobs.

Assess a workload →

flarion.io/run-assessment

©2026 Flarion Inc. All rights reserved.