unravel

Managing costs on Databricks for AWS

Unexpected costs are eating into cloud budgets and lack of visibility to root cause and general inefficiency is costing organizations thousands, if not millions in operating their Databricks on AWS environment.

Controlling IT staff productivity costs (quick troubleshooting, meeting SLAs) are just as important as controlling cloud infrastructure costs (compute, storage, networking) in Databricks on AWS.

- **Poorly performing or failed jobs**
- **Getting to root cause analysis**
- **Missed service level agreements**
- **DevOps/ITOps blame game**
- **Chargeback/showback**

True cost go beyond data infra cost

The issues in DataOps contribute to direct organizational costs attributed to developing and operating an Databricks on AWS environment. As a result, the true costs of missed SLAs could have vast financial impacts on your business:

CONSEQUENCES OF MISSED SLAS

Banking: Fraudulent transactions in banking not discovered in time, leading to millions in theft and fines.

Healthcare: Patient profiles not accurately described, leading to lapses in care and millions in additional healthcare costs.

Retail: Customer demand not analyzed for a particular product, overestimating inventory leading to millions in waste.

Manufacturing: Equipment failure not detected with accuracy, leading to costly maintenance calls.

Tuning Databricks on AWS is largely a manual effort

Choose proper AWS EC2 topology / Properly assigning the correct AWC EC2 instance types and number of nodes is essential to control spending.

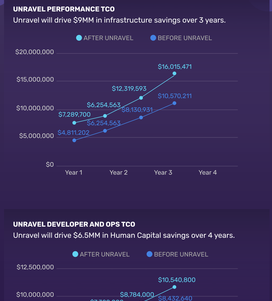
Choose proper storage and networking / Decide on the networking configuration (is public IP necessary?) as well as anticipating optimal storage (S3).

Code optimization / Inefficient Spark code in Databricks on AWS can kill your performance or cause failures - tight collaboration between DevOps and ITOps is essential to provision right level of resources.

Workload identification / Not all Databricks on AWS jobs are created equal. Data warehousing will have more persistent resource requirements (and cost) than ephemera data engineering jobs, for example. Anticipate these needs and adjust accordingly.

Unused compute elimination / Unchecked rogue resources can be a large contributor to waste. Users must understand when auto termination is warranted.

Unravel's solution



WITH UNRAVEL

Time to resolution:
Hours to minutes

Required infrastructure:
S3- $100K/year
EC2 - $200K/year
DBU - $100K/year

Required team:
Big data engineers - 10 hours/week
DevOps - 5 hours/week

WITHOUT UNRAVEL

Time to resolution:
Days to weeks

Required infrastructure:
S3- $150K/year
EC2 - $500K/year
DBU - $200K/year

Required team:
Big data engineers - 40 hrs/week
DevOps - 10 hours/week

Unravel techniques for Databricks on AWS tuning

Poorly performing or failed jobs / Visualize jobs and job runs: track individual jobs to assess performance improvments.

Getting to root cause analysis / Root cause analysis: if we see a resource having issues we can use point of time KPIs to identify the state of the applications at the point of failure.

Missed service level agreements / Single pane of glass view: ensure maximum SLAs by pinpointing possible failures before they can happen.

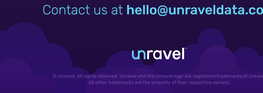
DevOps/ITOps blame game / AI-Driven recommendations: pinpoint whether the issue resides in the code or the cluster configuration via automatic recommendations.

Chargeback/showback / Per application costs: identify resource wasters by application or by user to drive accountability for responsible Databricks on AWS resource use.

Example: Unravel banking customer TCO

UNRAVEL PERFORMANCE TCO

Unravel will drive $9MM in infrastructure savings over 3 years.



UNRAVEL DEVELOPER AND OPS TCO

Unravel will drive $6.5MM in Human Capital savings over 4 years.

Interested in learning more?

Contact us at **hello@unraveldata.com**

unravel

© Unravel. All rights reserved. Unravel and the Unravel logo are registered trademarks of Unravel.
All other trademarks are the property of their respective owners.