High Performance Enterprise Data Processing with Apache Spark

Sandeep Varma & Vickye Jain, ZS Associates

#EUde3

Who are we?

ZS is a world leader in delivering high-performance solutions for Life Sciences companies to achieve impact

4,500+ Professionals with deep industry and domain expertise help clients significantly improve their performance

22 OFFICES WORLDWIDE

BARCELONA · BOSTON · CHICAGO · EVANSTON · FRANKFURT · LONDON · LOS ANGELES · MILAN
NEW DELHI · NEW YORK · PARIS · PHILADELPHIA · PRINCETON · PUNE · SAN DIEGO · SAN FRANCISCO
SÃO PAULO · SHANGHAI · SINGAPORE · TOKYO · TORONTO · ZÜRICH

OUR OFFERINGS RANGE FROM

Business problems posed to us

- Revamp large, complex system that is **difficult to maintain and understand**
- **SLAs** – fix adherence, speed up 4-5x, enable visual tracking
- Build **transparency in business** rules and eliminate hidden rules
- **Speed up enhancement cycles** by order of magnitude
- **Reduce costs** - infrastructure and operations

Use case highlights

Enterprise scale

50 + data sources
(S3, FTP, Internal DB, Public Cloud)

500+ business rules / KPIs

100+ analytics data packs

Full blown DevOps pipelines

2000+ end users

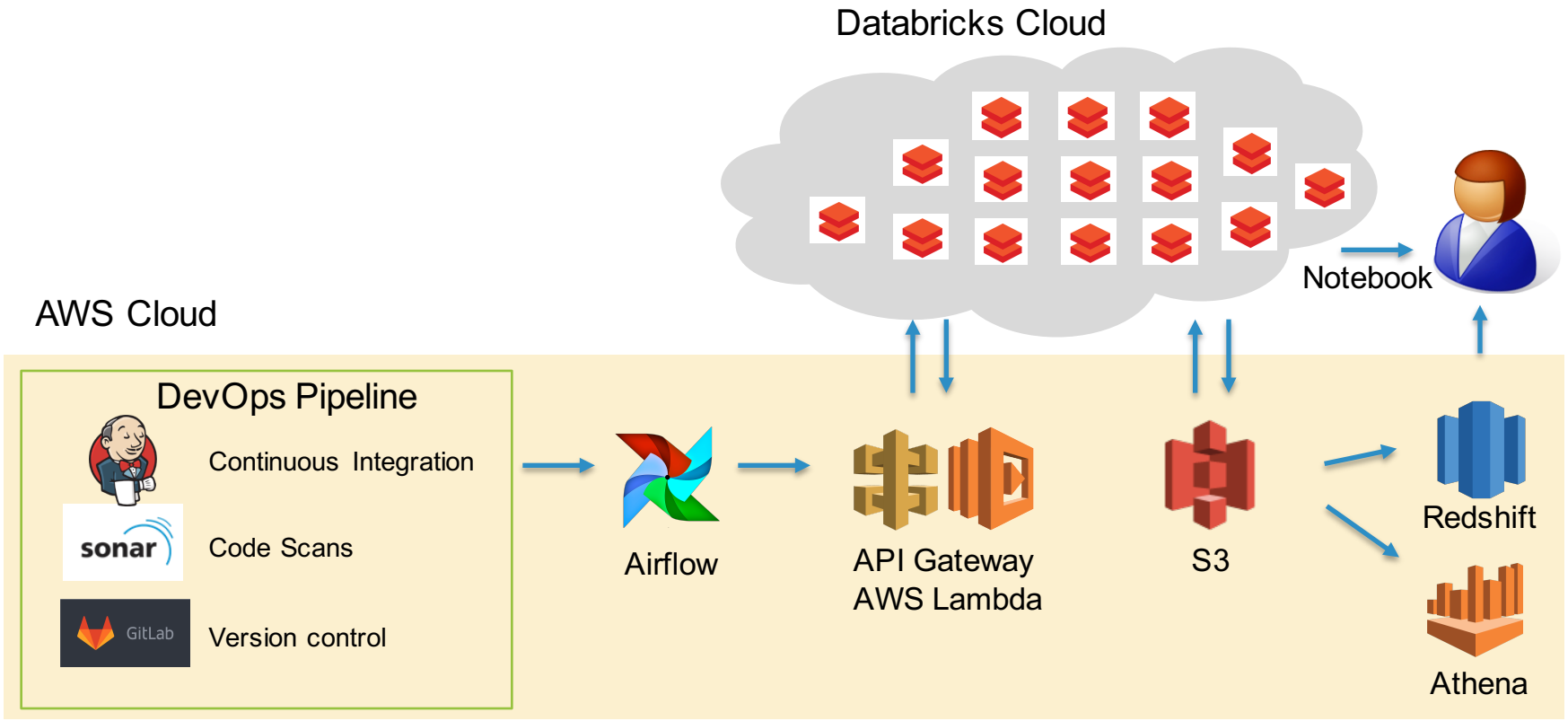
Weekly: ~0.5 TB i/p, ~3 TB o/p

Data to reports in <24 hours

Service oriented architecture

Fully elastic cloud infrastructure

High level solution architecture



Learning 1: Shortfall of combined domain and technical expertise

- **SQL** is a universal language for domain experts and natively supported by Spark making it ideal for domain experts
- Build **configurable frameworks** to abstract technical details and let domain experts focus on the business
- We chose **PySpark** given easy availability of talent and Airflow + helped some domain experts to switch to more powerful PySpark APIs for complex pieces
- From enterprises, SQL and PySpark codes are **easier to manage and maintain**, needing limited specialized skill sets at least for reading

Learning 2: Extreme performance tips in Spark ecosystem

- Stick to **S3 → Memory → S3**, avoid sinking to HDFS; ensures there are no roadblocks for segregating jobs in clusters and memory is used appropriately
- Design for **multiple-contexts** – attractive to use one context to build up long series of jobs but not resilient; **Job servers** don't help much either
- **Cost based optimizer** (in 2.2) is awesome! Analyze tables and turn it on
- Keep **UDFs in Scala**, PySpark UDFs are terrible in terms of performance
- **Deep nesting** of transformations is attractive but can fail, force actions periodically (we hit SPARK-18492 bug)

Learning 3 – DevOps for Data Management Platforms

- **Configs and input data** change more frequently than code in data management catering to large number of data sources

- **DevOps pipelines** designed for two different test groups:
  - one to test code with static data and static configs
  - one to test integrated code, configs, and data with live data and live configs

- **Game changer** for data management platforms, especially ones that change frequently

Learning 4: Architecting for adaptability

- **Service oriented architecture with API wrappers** works very well in data management and analytics enablement context
- We created a number of services with **AWS Lambda + API Gateway** on top that were called upon by orchestration code in Airflow
- Be mindful of **5 min limit with Lambda**, work with ECS or Lambda-SQS if appropriate
- Even if you are not hosting APIs, **design modules to support API type inputs** for future extensibility and to force you to think about APIs from day 1

Learning 5: Visual orchestration

- **Business desired to visualize** operational flows, “where is my pizza?”
- Developer desire to “code” instead of configure visual tools
- **Airflow strikes balance;** still in incubation state so expect bugs / issues but still very promising