MODERN BATCH: MANAGING THE MADNESS?

My previous installment in this series explored what batch,(or application workflow orchestration) is, and how it still remains an essential discipline in computing. This time, I want to focus on the value of a batch/workflow approach for "modern" technologies like cloud, containers, microservices and serverless computing.

It may be ironic that the management approaches and best practices that have evolved over the years in managing enterprise batch computing may be exactly what's needed in today's modern, elastic environments.

Managing Business Applications

As soon as you have more than a single business function or even a single function consisting of multiple components, you have a management requirement. You need to be able to:

- Determine whether things that are running should be running
- See whether there are any things that should be running but are not
- Have visibility into whether what's running is running on time or correctly (as opposed to just looping or waiting)
- See confirmation that notification of a failure occurred and tools and facilities to analyze and correct the problem are mobilized

The challenge of meeting the above basic requirements increases with the number of components

and technologies in use to implement a business function.

Modern applications may contain components that use traditional relational databases or newer ones like noSQL, traditional file systems or Hadoop Distributed File Systems, ERPs, SaaS solutions, etc.

Architectures may consist of traditional hierarchical flows, be message-based with a "publish/subscribe" relationship among application components or take a microservices approach.

For all these varieties of application structure, operational instrumentation remains a requirement that development, DevOps and IT Ops teams must address to meet the service delivery demands of the business.

The Value of Batch Management in a Modern Environment

In these evolved environments, applications still defer processing until either some data is aggregated or some other collection of events occurs before subsequent processing is performed. Furthermore, whether waiting for events, which is just the modern equivalent of date and time, processes have to connect to databases and other applications, logs must be captured, visibility of these relationships is required, etc. All these actions are part of what "automation" means, and in the absence of any other way to accomplish it, developers frequently fall back to using basic tools which lead to custom development of automation that is either embedded in application code or in scripts. If one wants to argue the need for such management, just examine EVERY modern environment, whether it's the leading ERP, database, Big Data, or cloud providers. ALL these environments keep reinventing a batch management solution. If this was not a fundamental requirement, we would not continue to see SM36/SM37, SQL Agent scheduler, Oozie, Airflow, Azkaban, Luigi, Chronos, Azure Batch and most recently AWS Batch, AWS Step Functions and AWS Blox to join AWS SWF and AWS Data Pipeline.

The problems with all these tools are:

- Their solutions assume that their application or ecosystem is the only one you will ever use so they don't even acknowledge existence of other environments or technologies
- Building a comprehensive, sophisticated solution is hard and takes a long time, and so far, precious few solutions have been able to meet the diverse requirements I've been discussing here

Recently, yet another challenge has emerged that makes all the above even more challenging: digital transformation. The pressure to accelerate delivery of new business capabilities that run reliably and meet the stringent risk and governance requirements is immense. Having to do that while still supporting and frequently integrating "traditional" technologies makes the challenge that much greater.

Many organizations have turned to DevOps as one of the techniques to enable them to meet the delivery expectations of the business. Although DevOps enables faster delivery, it does little to improve operational tools that are deficient in the level of visibility and manageability they offer in the production environment. Requirements for a DevOps toolset to help address those issues include:

- Support highly heterogeneous collections of platforms and applications
- Understand business service levels

- Provide a way to visualize connections among components across that heterogeneity
- Support quick access to debugging and problem analysis data or offer business users insights into their workloads

The answer may lie in working backwards. Find a solution that provides the functionality you need to run your production environment, not your development environment. After all, that's where the rubber meets the road so to speak.

You won't find many toolsets that can give you the broad functionality you require but there are some. And then find which of those solutions has been DevOps-enabled. That may be an even smaller subset. However, don't compromise by choosing a tool that's comfortable to use and deploy for developers but not for the diverse users in production. Remember that applications spend the vast majority of their lives in production and that is the target environment you should aim for.

This is the third of a 4-part blog series on "modern batch." You can read the other blogs here:

- Part 1: Modern Batch Processing: A Thing of the Past or Essential Discipline?
- Part 2: Modern Batch: To Batch or Not to Batch? There's no Question!
- Part 4: Modern Batch: The Ops in DevOps fully evolved – "Jobs-as-Code!"