Section 3: Week 7: Cloud Migration via DSL

Nate Bachmeier

TIM-8110: Programming Languages and Algorithms

May 19, 2019

North Central University

# Cloud Migration via DSL Outline

Contoso is a provider of online student analytic services with a vast collection of micro services hosted in their private data center. They need a mechanism to efficiently transition their product lines from private data centers into the public cloud. This can introduce significant challenges proportional to the depth of the data storage network, which can be extensive for certain distributed service architectures.

## Background

Many businesses like Contoso are actively working to transition their proprietary systems into the public cloud. This enables them to reduce infrastructure costs and improve the agility to provide new features to their customer base. These capabilities are well documented through highly optimized ‘pay per use’ pricing model and instant access to a virtually unlimited amount of resources.

For many existing service providers, the journey to the cloud can be complex as it requires moving their proprietary systems into new deployment models. There are two supersets of issues (1) getting the infrastructure to the cloud; and (2) migrating the customer data into the new cloud stores.

These challenges are compounded for micro service architectures which often have numerous private data stores. It is also common practice for micro services to call other services causing the natural formation of data dependency graphs. Each store within the graph needs to be fully hydrated or they will produce erroneous results.

## Problem Statement

There are two distinct optimization problems that need to be addressed (1) how can the infrastructure be transitioned to cloud native platforms; and (2) what is the quickest method to hydrate the data dependency graph?

To constrain the discussion the solution is built upon a commoditized platform and does not offer reliable Quality of Service (QoS). There are also human resource constraints which limit the number of edits to the existing system. For instance, it is not possible to simple rewrite the entire product line as that would be far too complex. As the owners of the source code they can make certain modifications to a large percentage of the total services.

As the hydration process occurs it needs to be done in the shortest amount of time possible. Since the QoS between components is unreliable the hydration process needs to be verifiable and performed in such a manner as to not miss notifications. If notifications are missed, then the customer will receive incorrect results and perceive the product as poor quality. This portion of the problem could be restated as a ‘maximum flow problem.’

## Goals

To address transitioning the infrastructure into cloud native solutions, a domain specific language (DSL) will be created. The objective of this language is to encapsulate many of the differences between two environments and inject those missing aspects during compilation. This would enable the widest breath with the smallest number of edits.

Next a generic distributed validation scheme will be proposed such that accounts of all permutations through the system. This needs to consider scenarios where a service receives M events and emits zero to N outputs. It is also possible that event M will arrive multiple times to the service perhaps out of order.

Finally, a generic model will be created to describe the maximum flow of the data hydration. This will enable the development team to prioritize future optimizations which provide the most benefit.

To measure the success of the infrastructure migration, the net savings of the modified service versus the cost to implement will be used to determine a break even point. For instance, if the cloud native platform saves 10c per hour and costs 1000$ to implement; then the breakeven would be 10,000 compute hours (or 1.14 compute years). Assuming there are 25 instances of the service in production the realized breakeven is 16.7 wall days.

The data migration solution will per measured in terms of maximum supportable flow through the entire data network. It is also a design constraint that the maximum flow be reached with the minimal scale of each micro service so that it can be accomplished as cheaply as possible. Data validation errors will be recorded and penalized from the final score, scenarios where the system encounters and error and is able to self-heal within an acceptable service level are not.

## Relevance and Significance

Businesses of all sizes and shapes are actively migrating their workloads onto the public cloud. While tools and platforms exist for simple web sites they are lacking for complicated distributed applications often found in enterprise environments. Having the capability to transform existing code bases could reduce the time and resource requirements needed to make that transition. Further having the ability to model their data migrations in terms of flow control would enable the prioritization of future efforts. This would further improve the efficiency of their journey into the cloud.

The alternative would be a costlier transformation which has a higher probability of failure. Businesses which cannot successfully move to the cloud will lack its key capabilities such as efficient operational expenses and improved agility. This makes them less competitive and more likely to be superseded by a modernized competitor.

## Literature Review

## Approach

### Background Notes to cover

1. Why would a business need to transition to the cloud?
   1. Reduce infrastructure costs, improve provisioning agility
2. How would that grow their business?
   1. Instant procure resources, less time on infra more time on features
3. What challenges would this cause in terms of design patterns and data movement?
   1. Getting the data into the cloud is complex for existing micro service designs
   2. There are many numbers of data stores which are full of random id mappings to each other, requiring the data to naturally propagate same as steady state
   3. Simply uploading the data to the cloud can be complex for distributed storage with hybrid storage
4. Why do these problems exist?
   1. Hybrid storage models deter a one size fits all migration strategy
   2. Bulk import interfaces may be difficult to implement for each feature area, require significant effort
5. What common options exist today?
   1. Transition the existing systems through containerization
   2. Rewrite the applications as cloud native
   3. Use an event replay through steady state
   4. Lift and shift
6. How else could they address the problem tomorrow?
   1. Create a DSL and transform the existing apps at compile time
   2. Inject cloud native aspects into the components design, such as queues/hashtables
   3. Use metadata programming to provide hints to cloud native rehosting
   4. Use scheduling algorithms to more efficiently replay the events through steady state
   5. Reduce the number of size of their data dependency graph

### Problem statement notes

1. How does Contoso transition both their infrastructure and data in the most efficient manner possible?
2. They are limited that the solution is built on a commoditized platform and thus the reliability and availability are not guaranteed
3. The resources of Contoso are finite, so they need to make the fewest changes to their business logic
4. How do they reach the maximum flow of the migration network?

### Goal notes

1. Propose a scheduling algorithm which uploads the data the fastest accounting for its load dependencies
2. Propose a strategy for distributed validation of the migrated data
3. Propose a set of requires for a DSL language which enables cloud native migration of the services, optimized for lower overhead
4. Success will be measured in terms of increase in maximum flow across the dependency graph. The success will be penalized if validation finds error and errors in validation are penalized even greater.

### Relevance notes

1. Cloud Migration is impacting businesses of all sizes as they transition to the modern platform.
2. Many simple web sites simply backup/restore into the cloud, however larger micro service based systems have a more complex data dependency graph
3. If a reliable solution is not found these businesses will not be able to economically scale their business to new markets, and cannot be as competitively efficient