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. One of the tenants of micro service design states that each component should ‘share-nothing’ including data stores. This reduces the blast radius and improves the resiliency of the over-all system.

They need a mechanism to efficiently transition their product lines from private data centers into the public cloud. These heterogenous private stores can introduce challenges for that migration as they need to become hydrated. While significant literature exists for trivial ‘lift and shift’ paradigms of an individual store there is less research on rehydrating the data network.

# 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 environments. There are two supersets of issues (1) getting the infrastructure to the cloud; and (2) migrating the customer data into the new cloud stores.

Business stakeholders often take a conservative position during cloud migrations and be hesitant to filling the switch across all customers. This leads to scenarios such as creating a new deployment instance in the cloud and then hydrate specific customers history. Transitioning small handfuls of customers per data migration iteration reduces the blast radius and provides better assurances of business continuity in exchange for more time to complete the project (Ferdiana & Putra, 2018).

Due to the longevity of the migration project the system state can end up in an arbitrary flux. This can occur from new features being are added, and internal implementation details changing. The data migration process has two choices to mitigate this issue (a) create and maintain bulk load interface on each data store; or (b) replay the traffic through the steady state interface; or (c) a combination of both choices.

For systems with large number of heterogenous private data store and a high degree of data connectivity between them; it could be prohibitively expensive to (a) create bulk load interfaces. Consider the scenario where events cascade such as ‘service A’ emits N events by ‘service B,’ which in turn emits M events to ‘service C.’ To correctly model this would essentially duplicate the large amounts of business logic and be error prone.

Another argument for businesses to desire choice (b) is that improvements to the data migration scenario are improvements to the general customer. This can be seen in terms of general correctness, backward compatibility assurances, performance and scalability scenarios.

There are potentially specific micro services where scenario (a) is desired in combination with (b). For instance, a business might have archives of physics models each requiring days of compute time. They might choose to maintain an out of band process for moving the models while replaying the metadata through steady state ingestion pipeline.

Within the context of this proposal it is assumed that the business must choose option (b) or (c) due to option (a) being prohibitively expensive to build and maintain.

# Problem Statement

Cloud Migration Strategies tend to center around (1) getting the infrastructure; and (2) getting the data into the new environment. For enterprise customers with large numbers of micro services there are challenges to successfully accomplishing both.

Patterns around moving the infrastructure is a well-studied problem, thanks in part to virtualization and containerization technologies (Cefaratti & Lin, 2018) (Ahmad, Naveed, & Noda, 2018). A lesser studied scenario is the data migration and methods for overcoming their inherent challenges.

Consider the scenario where a bulk migration of 10 million records traverses a component with a 0.01% failure rate. This will reproduce 1,000 instances of that defect which can cause inaccurate results for the customer. If all issues within the product code can be resolved, there will still be issues due to the underlying system being built on commoditized hardware. An expectation of failure needs to exist within the data migration scenario.

Another challenge is optimizing the throughput that these migrations should be performed. If the ingestion rate is too slow it will never complete, yet too fast and it cause scalability issues within the infrastructure. Those scalability issues will increase the failure rate and result invalid results.

If the system was set to scale infinitely then it could become to costly to perform the data migration. This would lessen the competitive gain of moving to the cloud and place an unjust financial burden on the business. While that might be acceptable for a short period it has already been stated that the entire enterprise migration could last years.

# Goals

## Max Flow Model

These challenges could be restated as there is a need to find the ‘min-cost max-flow’ through the distributed architecture. There is also a need to model which junctions lead to bottlenecks versus are over provisioned. If ‘service X’ is bound by the scale of ‘service Y’ then either X needs to be reduced for cost savings or Y increased to provide additional load.

The primary goal of this study would be to identify the features which are most critical to the maximum flow model. Consider a component which can only process N records per second. In cloud environments the ability to add or remove virtually unlimited resources might rephrase this scenario into ‘N records per V virtual cores per M gigabytes of memory.’

## Min Error Rate

Another requirement is maximum flow of records while minimizing the number of failures. Writing error free distributed source code is an extremely complex problem. Mechanisms are required to identify and promote self-healing of the data ingestion. A domain specific language (DSL) could be created to describe the distributed compensation policy. This would ensure consistency of the rules and remediation behaviors.

## Define New Patterns

Multi-casting data to several unreliable receivers is not a unique problem and is shared by several sciences. These challenges are well-studied in certain electrical and customer broadcast communication systems. A deeper investigation needs to exist to determine how those learnings can be applied to this use case.

# 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