Backtesting in the Cloud

A Scalable Market Data Optimization Model for
Amazon's AWS Environment

A *Tick Data Custom Data Solutions Group* Case Study

Bob Fenster, Software Engineer and AWS Certified Solutions Architect
Bill Bryson, Director of Engineering
Neal Falkenberry, President

June 2015

Introduction

In 2013 Tick Data, Inc. left a traditional data center in favor of Amazon's AWS cloud environment. The daily tasks performed there include the daily receipt and processing of very large quantities of financial market data as well as distributing that data to clients around the world. The company has gained considerable expertise in the process, particularly in the area of connecting hosted market data to analytics and scaling processing power on an as-needed basis for data processing and analytics. From that expertise grew Tick Data's *Custom Data Solutions Group*.

This paper describes a specific client case whereby Tick Data's *Custom Data Solutions Group* was asked to assist a hedge fund in the creation of an AWS virtual private cloud ("VPC"), connecting that VPC to market data through various Amazon APIs, implementing web page and programmatic means of scaling processing power as needed, and finally, turning control and access of that VPC over to the client. The client's end goal was to bring analytics to hosted data, enable scalable computing power as needed for backtesting and optimizations, and effective cost management by turning the environment on/off as needed. In short, Tick Data built a scalable, secure trading strategy optimization platform for a client and handed over the keys.

The Problem

Cloud computing provides firms that backtest and optimize trading strategies or other data intensive optimizations access to processing power that revolutionizes important aspects of performing analysis on large datasets ("Big Data"). Analysts can provision a variable number of CPUs to perform their tasks. Coupled with the availability of on-demand large market data sets, a cloud strategy is an extremely attractive may to access variable computing and data resources in new ways.

Pointus Trading Partners, a high frequency, quantitative hedge fund, was one of the first customers of the new Tick Data API service available on the Amazon Web Services cloud that provides on demand access to research-quality market data. Pointus had previously performed their backtesting in a more traditional way, by purchasing data, transporting the data back to their internally hosted server farm, maintaining that data (e.g. applying corporate actions) and performing analysis (primarily parameter searches) on their existing grid of computers.

Pointus hired Tick Data to build a backtesting framework using Amazon's AWS. While Pointus's primary objective was a low-cost, highly scalable backtesting environment, the experiment provided a test case for moving analytics to hosted data and presented an opportunity for Pointus to evaluate eliminating the need for internal market data management altogether. A hosted data approach eliminates the costly process of downloading all the data to local infrastructure, and provides direct access to terabytes of global, multi-asset class data that Tick Data has stored and available through its API in AWS and available for license on a per symbol basis.

There were four primary objectives for the assignment:

(1) Design an elegant AWS cloud-based optimization environment that was scalable and reusable.

(2) Achieve the results as inexpensively as possible without sacrificing scalability, access, or ease of use.

(3) Train, configure and launch a reference trading model implementation as a proof of concept.

(4) Access data on demand and avoid the need to permanently store large quantities of data in the cloud for access by optimizers.

(5) Train Pointus staff to assume control, operate, and access to the AWS environment.

First Steps

Pointus provided a simple reference trading model for use in the assignment. The model required trade and quote data from a small number of related securities. With relatively few parameters, the backtest iterated through parameter values, selected market neutral portfolio weights, and generated statistics that include the calculation of total return and a risk measure.

The objective was to iterate the trading model over a large parameter space and a large universe of symbols by distributing the optimization across many EC2 instances (processing power). Pointus had a Matlab™ script that was able to download the required data from the Tick Data API, run the strategy on a parameter set, return the performance statistics, and advance to the nest parameter set. **The challenge was to automate the distribution of the parameter search across many EC2 instances with strategy results aggregated in a single location.**

The first step was to create a single machine instance within AWS and run one iteration of the strategy. This step did not require any special knowledge. Tick Data assisted Pointus in creating an AWS account. Tick Data then used the AWS console to launch one EC2 instance running Amazon Linux to be used as the core processing machine ("Amazon Machine Instance, or AMI") that will be cloned as the optimization client. The instance was provided an internet routable IP address. Pointus was immediately able to connect via ssh and install the remaining components required to support running the backtest.

While Matlab™ itself has an optional distributed computing app that is available on the Amazon cloud, the project decided against using it for two reasons. First, that module is designed to be used in a traditional "always on" computer grid or cluster rather than a scale up / scale down cloud configuration. Furthermore, part of the experiment was to build a low-cost optimization cluster. The Matlab™ distributed computing model, although very feature rich and impressive, is quite expensive.

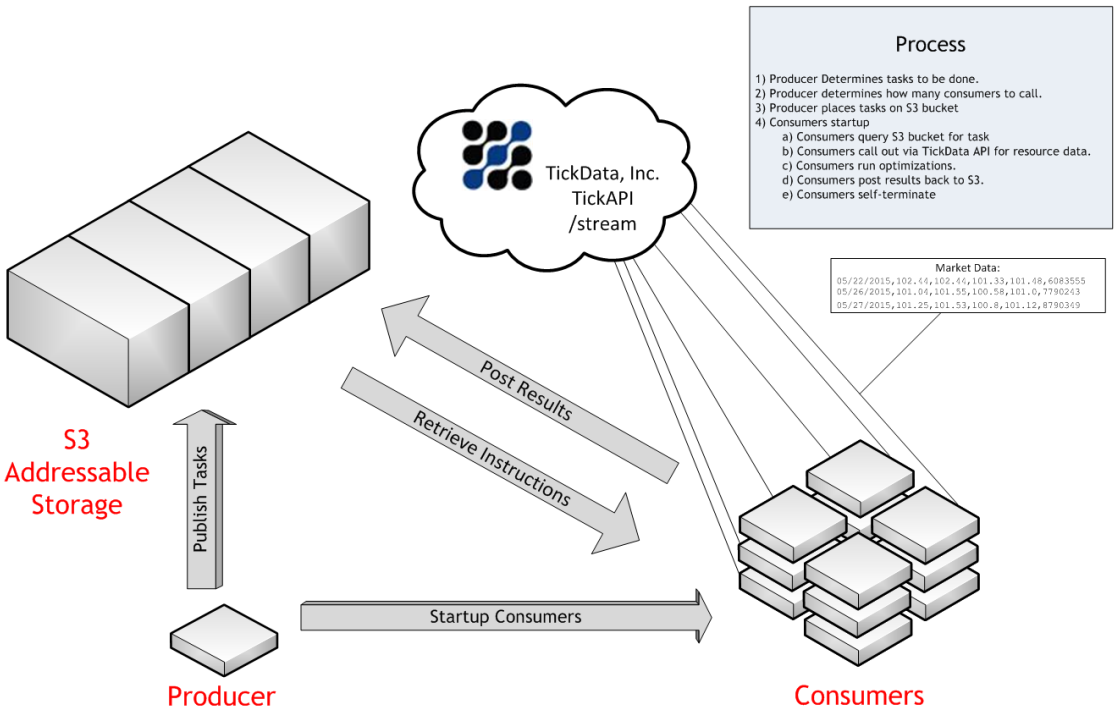
Pointus decided to install Octave™, an open source analysis tool that uses a similar syntax as Matlab™. Octave was installed very easily with a single call to yum. Pointus converted the model's Matlab™ code to Octave™, downloaded market data from the Tick Data API, ran the strategy over the data, and deposited the results in a designated directory. A snapshot (or "AMI") of the successfully installed server was stored for use as a template when creating individual instances of servers that would each handle a portion of the optimization. Such instances are often referred to as "children" or "generations" in optimization terminology.

Building an Infrastructure

"Spinning up" is Amazon vernacular for starting a new EC2 instance or server. While "spinning up" one instance and running that instance is relatively easy, replicating it across an army of instances for distributed processing is far more challenging. To run a material number of instances and communicate with all of them, it would be impractical to have static IP addresses for each instance. Amazon only provides 5 static IPs per account so spinning up more than five instances required Tick Data to build an infrastructure to support communication between multiple machines that does not require static IPs. In addition, security requires that we support this communication through a VPN.

Amazon has a full array of enterprise quality tools to create a secure infrastructure called a VPC (virtual private cloud). Tick Data runs its entire enterprise market data processing and distribution system in an AWS VPC. Tick Data's network experts recommended that Pointus deploy an infrastructure template from CloudFormation™, Amazon's library of templates to quickly create AWS components. The chosen template created a fully configured public and private subnet. A NAT server was also created to allow secure communication between the private subnet and the internet, both in and out.

Furthermore, Tick Data created a security infrastructure to restrict access to the VPC resources. It is not initially obvious how to create the security roles and groups necessary to ensure good external security, but the AWS environment does allow for a rich security design. For this assignment, we restricted access to the VPC by source IP address. We then created two supplemental user IDs, one analyst ID and another to be used by instances to access AWS resources in behalf of the model. The AMI snapshot created in the previous step was updated to use the credentials created for the model.



Producer-Consumer Model in the Cloud

Once the infrastructure was created to support spinning up multiple instances Tick Data worked on the design for spinning up, monitoring and managing those instances. It is very tempting to employ traditional design techniques. Traditionally, in a local implementation, a producer server would create tasks for child consumer servers that would execute those tasks. The machines would communicate through a shared drive, central database, or messaging through a direct connection (such as the Matlab Distributed Computing™ service). For highest reliability, all the servers would be running, connected to the shared drive, and the database would be highly available.

In the cloud, where the requirement is to spin up an arbitrary number of instances, have them run their assigned tasks and then terminate (too keep costs at a minimum), we need to use a "cloud oriented" design. Since each new instance will go through a startup initialization to configure that instance, there are reliability concerns with designing the interactions in the same way as a standard infrastructure. Ideally, each instance spun up should be relatively independent and have the least number of dependencies on other instances.

Fortunately, Amazon has productized a number of designs to solve common problems in a way that maximizes the benefits of the cloud. For example, Amazon's Elastic Beanstalk™ service dynamically spins up instances in response to increases in web requests for data. In fact, Tick Data's API uses this service as part of its load-balancing practice. Amazon's Kenesis™ service manages streams of big data from "producers" and spins up "consumer" instances to consume the real time data stream. While Tick Data experts have extensive experience implementing these services, for the simple implementation considered in this paper, the design only uses the services included in the free tier from Amazon. This will change as Pointus moves from proof of concept to core production use.

Using these AWS components as examples, Tick Data's design leveraged a basic consumer/ producer model to task out work clients. The primary producer runs Matlab/Octave™ code that does the preliminary analysis and determines the tasks to be performed by the consumer instances. The producer uses the AWS API to spin up consumer instances, and utilizes command line options to seed these instances with a startup script specifying a task. The startup script has the instance self-tag to, identify where the instance's work-load, and the location to deposit the completed results. The producer, utilizing Amazon's Cloudwatch™, can monitor for the completion of all work items and ultimately assign a next set of tasks or correlate the all the responses into the solution to the posed problem.

Data Transfer

In Tick Data's design the startup script is the only direct information transferred between the producer and the consumer. The producer provides the code and parameters of the task by placing it in a shared repository. Again, in a typical non-cloud implementation, a shared network drive or a database may be the ideal way to achieve the sharing of this data. This would be possible in Amazon with their EBS storage which can be accessed like a shared drive. However, in an environment with an indeterminate number of clients, mounting a shared file system will overtax the file server and adversely impact performance.

Tick Data's design for Pointus uses Amazon's S3 data storage facility. This data storage is also used by companies such as DropBox, Pinterest, Novartis, and Expedia. S3 has a number of characteristics that are useful for our application. Files are sent to a bucket in S3 from anywhere, not solely from computers linked to it in our private subnet of our VPC. Files can be shared with any computer that has credentials for that bucket, again whether in our VPC or externally. Finally, S3 storage is less than half the price of more traditional EBS storage and S3 has a feature to expire content which will be useful to ensure that the instructions for our consumers will be automatically deleted after a short period of time. The intent is not to use S3 for permanent storage, but rather, temporary storage to be used only during optimizations.

Conclusion

Within a few days Tick Data and Pointus had designed and deployed a backtesting environment and successfully run 20 EC2 instances simultaneously and aggregated the relevant trading results to a single file. The deployment and trial run exposed numerous advantages to the design and use of AWS as a scalable, yet low cost optimization platform.

The producer is entirely independent of the AWS infrastructure, meaning that the producer can run inside or outside of AWS. Traditional setups where a scheduled optimization or coordination between job instances is required, producers can run on a server within AWS. However, in single run jobs, the producer can run on a local machine or even a laptop to setup the instances to run remotely at AWS. The producer is no longer needed until the results are to be collected from the S3 data store, which can also be accessed from anywhere with the proper credentials. It is possible to receive the results of a 100+ instance optimization on your mobile device.

Consumers are each independent, made possible because of Tick Data's on-demand API for data. Because each consumer can request the data it needs inexpensively and then terminate, there is no need to setup a means of sharing or centrally accessing data. The Tick Data service handles all the storage requirements, and our consumers only have to request the data on an as-needed basis. This leverages the ability to license market data on a per-symbol basis versus licensing data for an entire exchange.

There are a number of enhancements that were discussed during the post mortem of the experiment. Future design might include the use of Amazon's SQS Message Queue Service™ to communicate between the producer and consumers to enable a real time or delayed exchange of information. Using this "cloud oriented" service has many of the same benefits of using S3 as the storage solution.

Further optimization on a use-case by use-case basis could also significantly lower cost in a real world implementation. For example, today Amazon charges for instances in 60 minute blocks. Therefore, for jobs that take significantly less than 60 minutes, grouping consumer jobs on a single server would have significant savings over spinning up many servers to each perform a short task. Tick Data has also performed significant analysis of the cost benefit trade-off between creating more instances versus a fewer number of higher power instances with the resulting shorter processing time. Additional cost savings are available by making use of AWS's spot instance market over reserved instances if the client has flexibility in time to completion of the optimization.

Pointus believes the time savings associated with utilizing Tick Data's *Custom Data Solutions Group* to setup its Amazon VPC were substantial. The most significant cost savings will be the need to no longer maintain an in-house server farm and acquire, maintain, and depreciate servers for use in optimizations. Further, the expertise in specific design elements related to distributing processing, the ability to scale up /scale down to manage costs, and the ability to license data "on the fly" on a per symbol basis just that needed for the optimization are potentially just as significant.

Please contact Neal Falkenberry, CFA, at (703) 757-1370 for additional information on this project or to inquire about Tick Data assisting you in the design of your own cloud-based optimization environment.