Parallel Cloud Computing: Making Massive Actuarial Risk Analysis Possible

By Joe Long and Dan McCurley

This article will walk through a cloud use case where we were able to cut a three-month machine learning exploration project[1] down to just under four days using a mixture of open source tools and the Microsoft Azure cloud. This translates to an approximate 25-fold reduction in serial compute time for such a task. We will give a short introduction to the cloud while sharing our experience of managing the pool of data-crunching machines that ran our analysis. In closing, we will discuss lessons learned and ways to improve the plan of attack, as well as touch on the importance of state management to aid in efficiency and the reproducibility of results when using the cloud.

SETTING THE STAGE FOR THE CLOUD

Machine learning is spreading quickly across many industries and is showing promising results for making better predictions and automating manual tasks. However, with increases in data size and the greater power of more complex algorithms, the computing resources it takes to crunch the numbers increase as well. Nowadays, it may take days or months to conduct an analysis on a single machine. There is a solution, though: Thanks to advances in cloud computing, the phrase "the sky's the limit" has a whole new meaning as we now have the ability to speed up time if the reward outweighs the cost of doing so.

In order to utilize the time-saving efficiencies of the cloud, a large computational process must be able to be broken down into independent tasks that can be run in parallel. Not every process fits this mold. Some processes rely on a series of sequential calculations, where each calculation is dependent on the ones that precede it. An example of such a process would be calculating a single sequence of time-dependent events, which would not be a good use case for the parallel compute capabilities of the cloud.

Machine learning, however, is full of many processes that can be broken down into independent tasks calculated in parallel, which can then be merged together after all independent calculations have been completed. A good example of this would be an ensemble method such as the random forest algorithm, which is used to develop a predictive model comprised of hundreds to thousands of independent decision trees that are averaged together to produce a single prediction. Another easily parallelizable example is the Monte Carlo simulation. These algorithms are prime candidates for the massive parallel computing abilities of the cloud. Almost all supervised learning algorithms use some kind of resampling technique (e.g., bootstrapping, cross-validation) to optimize the bias-variance trade-off for generalization. Most resampling techniques are embarrassingly parallel and can benefit greatly from cloud computing.

In our case, we used the cloud to help with a large machine learning exploration project, which was comprised of many calculations done in open source R. Our initial exploration started with a single heavy-duty, bare-metal machine that could handle traditional memory and compute intensive tasks. We quickly discovered that in order to run the full exploration analysis we mapped out, we would miss our deadline. Our initial estimate was that the full analysis—when run sequentially on our in-house machine—was expected to take 90 days of continuous computer run time. However, with some manual effort to break the analysis into semi-equal chunks, we estimated we could run it in Microsoft's Azure cloud and complete all of our calculations in less than a week. This approximately 25-fold reduction in serial compute time to run our analysis gave us more time to digest the results, giving us the ability to run further variants of our initial exploration plan. More variants can equal better value to the client.

THE MAGIC BEHIND THE CLOUD

"There is no cloud—it's just someone else's computer" is a common meme used to explain cloud services. While this phrase helps one understand the basic idea of the cloud, it does not fully recognize the great capabilities and flexibilities of the modern cloud infrastructure. The concept of the cloud dates back to the 1960s and is commonly attributed to J.C.R. Licklider and John McCarthy.[2] Joseph Licklider is credited for his core concept of a Galactic Network or "Network of Networks" and John McCarthy for theorizing utility computing. These ideas reached commercial viability in 2002 when Amazon Web Services (AWS) started providing web-based, pay-as-you-go services to companies to store data and run applications. Current major competitors to AWS include Microsoft Azure and Google Cloud.

All of these providers offer similar ways to access their resources. It is helpful to think of these resources in three main categories:

1. *Infrastructure as a service (IaaS)* creates a virtual data center in the cloud similar to what your company would have in an information technology (IT) climate-controlled room. It's easy to adopt but expensive to run.

2. The second way to access cloud resources is through *platform as a service (PaaS)*. In this method, the cloud provider takes care of storage and computation and provides a platform to do a focused type of work. If you want a database that is always available, but don't want to deal with any maintenance or tuning, this is an excellent solution.

3. Thirdly, *software as a service (SaaS)* allows a company to build a custom solution that can only exist in a cloud environment. Salesforce, Office 365 and G Suite are examples of SaaS.

Viewed in this context, our computing project was an example of an IaaS. But by the end of our exploration we had migrated much closer to a PaaS solution. The actual difference can get quite fuzzy.

THE LEARNING CURVE

Once we realized on-premise calculations would take too long, we turned to the task of determining how many (and what capacity) computers would be needed for a cloud solution. After a period of research on best approaches for parallelizing our process in the cloud, we estimated that 63 virtual machines (VMs) should be able to handle the work in a reasonable time frame. Each machine had eight cores and 56 gigabytes of RAM, giving us a total of over 500 cores and 3,500 gigabytes of RAM at our disposal. For this project, we chose to provision the machines with Windows as the operating system due to familiarity, but we note this costs about 50 percent more in license fees than an equivalent Linux VM. We wrote PowerShell scripts to automate cloning and administration of the machines. Later in this article we will describe a new tool that makes things much easier (and transitions this solution from pure IaaS to something closer to PaaS). At the time of our project, this setup had a sticker price of less than $2 per hour to run each virtual machine of this size in Azure.

Our first step was creating the initial VM and then installing R and all the R packages we would need to run our analysis. Once we had our initial VM configured, we created 62 clones of it using the Invoke-Parallel PowerShell script Warren Frame discussed in his "Invoke PowerShell on Azure VMs" article,[3] which had some other helpful pointers we used along the way.

Now we had 63 VMs available to process data but hit a roadblock. How do we launch our R scripts on the VMs in a coordinated way? For this, we ended up using another script by Warren (Invoke-AzureRmVmScript) to invoke commands remotely on the VMs. We wrapped these commands in the Invoke-Parallel script to kick off the R scripts simultaneously across the VMs. An additional script served the purpose of deallocating VMs after the R scripts finished running to measure progress and limit costs. Allocated VMs charge per minute and deallocated VMs carry no compute charges.

Once all the VMs completed their tasks we collected our data and analyzed our results. In the end we ran a total of 90 days' worth of parallel compute time across the VMs, with the longest VM running for a total of three-and-a-half days at a total cost of around $3,000. The equivalent cost of buying and setting up similar machines would have required weeks of setup and tens of thousands of dollars of hardware purchase for the same result. Of course, the cloud approach also required a fair amount of time spent crafting and debugging the PowerShell scripts, which adds significant soft costs in addition to the hard costs. Additionally, when using an IaaS solution over time there would also be the ongoing costs associated with keeping the VM image up-to-date with the latest security updates.

THINGS KEEP ON EVOLVING

After completing our first large run in the cloud, we found that Microsoft was working on an R package simultaneously that automated many of the tasks we had done in PowerShell. This R package is called doAzureParallel, leveraging an Azure service called Batch. The package allows a user to create a pool of VMs in the Azure Batch service with a few lines of R code and then register it as the parallel back end for the R *foreach* package. If you are already familiar with the R *foreach* package then making the transition to using doAzureParallel is done simply by running some code that creates the pool in Azure Batch. Any existing *foreach* code using the *%dopar%* function can then be used as is.

Azure Batch allows you to easily launch a pool of Linux VMs, which as we mentioned earlier is much more cost-effective than using a pool of Windows-based VMs. The auto scaling features of Azure Batch allow dynamically scaling up or down the number of VMs in a pool based on the demand of the tasks you are running. Another option is to use a mix of dedicated or low-priority VMs in a pool. Cloud providers make excess compute capacity available at steeply discounted rates with the caveat that these machines can be interrupted by those willing to pay at the higher rate. If this happens, the current task you are running gets canceled and reassigned on another low-priority machine. Therefore, it is recommended to only use the low-priority machines if you have short-running tasks or your calculation can progress despite multiple restart attempts.

One recently added feature of doAzureParallel worth noting is its ability to seamlessly run R inside a Docker container on the VMs within your pool. This is similar to how we cloned a

custom VM image in our initial IaaS approach. It allows use of a prespecified environment that keeps R versions and packages in sync, which ensures reproducibility of results. The added benefit with the doAzureParallel Docker container approach is that now you can rely on Azure Batch to create up-to-date VMs each time you run an analysis, ensuring that you have the latest security updates. By default, doAzureParallel uses the "rocker/tidyverse:latest" image that is developed and maintained as part of the rocker project.[4] However, you can also specify a custom Docker image, which allows you to lock in a version of R if you are concerned about duplicating results long term.

In our case, doAzureParallel has helped us move our initial IaaS approach to more of a PaaS approach. Now we can rely on doAzureParallel to maintain the administration work of creating pools of VMs with up-to-date security updates, which are running our prespecified environments. Using such solutions allows users to focus more on the analysis they are trying to conduct rather than spending the time managing the infrastructure it runs on.

LESSONS LEARNED AND RECOMMENDATIONS

Taking a look back at our journey in the cloud, we have some final recommendations for those looking to get the most out of these exciting new tools.

- If you plan on using the cloud for an analysis in R, check out the well-documented doAzureParallel package. Even if you don't plan on using R for analysis you might find some workflows that help with other languages as well.
- The tools cloud providers have are constantly evolving and iterating, and it is essential to be aware of what new tools are made available. For example, moving from the highly manual cloning of machines to Azure Batch for automated compute pool creation was revolutionary and much easier to use.
- We highly recommend the use of Docker containers or some other state management when conducting work in R or any other language if you need repeatable results over a long span of time.
- Finally, we recommend using Linux-based VMs over Windows if your task allows you to, as it can provide a welcome cost savings. Also investigate the use of low-priority VMs (or spot pricing in the AWS world) if your workflow supports short-running tasks.

Table 1 gives an estimate of potential cost reductions we could have achieved if we were to rerun our analysis applying these recommendations using the doAzureParallel package. For comparison, we have also estimated the cost of using AWS as the cloud provider. Note that these are estimated costs as of Jan. 23, 2018; pricing may vary in your region or the contract you have in place with Microsoft Azure or AWS.

Table 1
Potential Cost Reductions

VM Option	Total Compute Hours	Price Per Hour[1]		Total Cost	
		Azure[2]	AWS[3]	Azure	AWS
Windows OS	2,151	$1.17	$1.05	$2,516.67	$2,258.55
Linux OS	2,151	$0.78	$0.67	$1,677.78	$1,441.17
Linux OS with low priority[4]	2,151	$0.14	$0.07	$301.14	$150.57

1. Estimated prices from Microsoft Azure and AWS online pricing for VM compute charges only. Does not Include storage or data transfer prices, which can become meaningful if not managed efficiently.
2. Azure A10 VM with eight cores and 56 gigabytes of RAM in the North Central U.S. region.
3. AWS r.3.2xlarge VM with eight cores and 61 gigabytes of RAM in the U.S. East (Ohio) region.
4. Assumes tasks were run without the VMs being preempted.

As you can see, the cloud is more than just someone else's computer. It's an ecosystem of resources that can be leveraged to explore ideas and complete tasks that were once unfeasible to achieve with the local computing resources of the past. ■

Joe Long is an assistant actuary and data scientist at Milliman. He can be reached at *joe.long@milliman.com*.

Dan McCurley is the Cloud Solutions Architect at Milliman. He can be reached at *dan.mccurley@milliman.com*.

ENDNOTES

1 A research and development project conducted by the Milliman Advanced Risk Adjusters™ (MARA™) product group. See *http://www.millimanriskadjustment.com* for more information about MARA.

2 Mohamed, Arif. A History of Cloud Computing. *Computer Weekly.com*, March 2009, *http://www.computerweekly.com/feature/A-history-of-cloud-computing* (accessed Feb. 1, 2018).

3 F., Warren. Invoke PowerShell on Azure VMs. *Rambling Cookie Monster*, *http://ramblingcookiemonster.github.io/Invoke-AzureRmVmScript/* (accessed Feb. 1, 2018).

4 Tan, J.S. Scale Up Your Parallel R Workloads with Containers and doAzureParallel. *Revolutions*, Nov. 21, 2017, *http://blog.revolutionanalytics.com/2017/11/doazureparallel-containers.html* (accessed Feb. 1, 2018).