Cost-Effective, Workload-Adaptive Migration of Big Data Applications to the Cloud

Victor Giannakouris, Alejandro Fernandez, Alkis Simitsis, Shivnath Babu
Unravel Data Systems
{victor,alejandro,alkis,shivnath}@unraveldata.com

ABSTRACT

More than 10,000 enterprises worldwide use the big data stack composed of multiple distributed systems. At Unravel, we build the next-generation APM platform for the big data stack, and we have worked with a representative sample of these enterprises that covers most industry verticals. This sample covers the spectrum of choices for deploying the big data stack across on-premises datacenters, private and public cloud deployments, and hybrid combinations of these. In this paper, we present a solution for assisting enterprises planning the migration of their big data stacks from on-premises deployments to the cloud. Our solution is goal driven and adapts to various migration scenarios. We present the system architecture we built and several cloud mapping options. We also describe a demonstration script that involves practical, real-world use-cases of the path to cloud adoption.

KEYWORDS

Cloud migration; Big data stack; Application performance management

ACM Reference Format:
Victor Giannakouris, Alejandro Fernandez, Alkis Simitsis, Shivnath Babu. 2019. Cost-Effective, Workload-Adaptive Migration of Big Data Applications to the Cloud. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3299869.3320240

1 INTRODUCTION

Many applications in fields like health care, genomics, financial services, self-driving technology, government, and media are being built on what is popularly known today as the big data stack. What is unique and challenging about the big data stack is that it comprises multiple distributed systems, including: storage like HDFS, S3, ABS; distributed processing engines used in ETL like MapReduce, Tez, and Hive (usually running on MapReduce or Tez); MPP SQL systems used in business intelligence like Vertica, Impala, Presto, BigQuery, Redshift; systems to support streaming applications like Kafka; systems to support data-science workloads that leverage machine learning and AI, like Spark; and so on.

We have worked closely with around 50 enterprises using the big data stack, and have had detailed conversations with around 350 more of these enterprises. Sizes of their compute and storage clusters vary from few tens to few thousands of nodes. These enterprises cover almost every industry vertical and run their stacks typically in on-premises datacenters, but increasingly transitioning to private and public cloud deployments or in hybrid combinations (e.g., regularly-scheduled workloads like the Big Data ETL runs in on-premises datacenters while the non-sensitive data is replicated to one or more public clouds where ad-hoc workloads like Big Data BI run). Several reasons drive enterprises to move to cloud deployments including reducing operational costs, enforcing operational resilience, increasing workforce productivity and business agility. A critical, technical and business challenge that enterprises planning to migrate their big data stack to the cloud face is how to choose a target, cloud deployment, what workloads to migrate, and how.

In this paper, we describe an effective solution to assist enterprises to prepare, plan, design, and validate their path to cloud adoption. Our solution is goal driven and adapts to various migration scenarios. Next, we present the solution architecture and several cloud mapping options. During the demonstration, the attendees will explore various options for generating a cloud migration scenario and will generate their own use-case stories using example cluster configurations.

2 ARCHITECTURE

Our system architecture is illustrated in Figure 1.

Full-Stack Data Collection. We collect monitoring data from every level of the big data stack. For example, (i) SQL queries, execution plans, data pipeline dependency graphs, and logs from the application level; (ii) resource allocation and wait-time metrics from the resource management and

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3320240

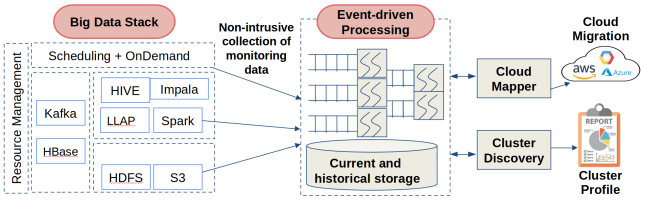

Figure 1: System architecture

scheduling level; (iii) CPU, memory, and network usage metrics from the infrastructure level; (iv) data access and storage metrics from the file-system and storage level; etc. Collecting this data in non-intrusive and low-overhead ways from production clusters is a technical challenge that has received attention in the research community [4].

Event-driven Data Processing. Some of the clusters that we work with have more than 500 nodes and run multiple hundreds of thousands of applications every day across ETL, BI, data science, and streaming. These deployments generate tens of terabytes of logs and metrics every day. We collect and process this data to populate interactive, online dashboards (not shown in the figure) presenting a near real-time, complete view of the full big data stack. Collecting and storing all the monitoring data in a single place opens up interesting opportunities to apply statistical analysis and learning algorithms for identifying and addressing for example failed, stuck, runaway, underperforming applications [1].

Cluster Discovery and Cloud Mapping. Collecting a plethora of monitoring data enables full visibility of the big data stack usage and thus, allows us to make an informed decision of how on-premises clusters can be effectively migrated to the cloud. For that, we use two modules. *Cluster discovery* collects low-level metadata of the on-premises cluster nodes (including resources like cores, memory, disk and their usage) and information like currently installed applications (e.g. Spark and Impala). *Cloud mapper* processes the generated cluster profile and recommends migration plans to a cloud provider. Next, we detail these two modules.

2.1 Cluster Discovery

Cluster discovery collects metadata like cluster node specifications and cluster usage metrics. Example specifications are resources (e.g., cores, memory, disk), stack type (e.g., HDP, CDH, MapR), cluster topology, services, workflow schedulers, etc. Example usage metrics include aggregates over time windows for multiple application types and workloads.

2.2 Cloud Mapper

Cloud mapper uses the cluster profiles to generate plans for migrating an on-premises cluster to a cloud provider; i.e., a set of Virtual Machine (VM) recommendations. Currently, we support a number of target cloud providers: Amazon Elastic Cloud (EC2), Amazon Elastic MapReduce (EMR), Google Compute Engine (GCE), and Microsoft Azure. We generate migration plans using three approaches: Lift-and-Shift (LS), Usage-Based-Cost-Reduction (UBCR) and WorkloadFit (WF).

We represent a node in the vector space as a 3-D vector using: *cores*, *memory*, and *disk*. Depending on the approach, we use one of the following two vector representations. A specification-based representation: X_{specs} =[*cores*, *memory*, *disk*]. A usage-based representation, which considers resource usage over the last N days: X_{usage} =[$cores \cdot usage(cores)$, $memory \cdot usage(memory)$, $disk \cdot usage(disk)$], where $usage(r)$ returns the usage percentage of resource r.

Our recommendation algorithm also considers an *objective* for optimizing one of either *similarity* or *cost*. In the former case, it generates recommendations based on vector similarity using the *euclidean distance*. In the latter case, it considers the cost per hour of each candidate VM instance. (Note that a VM's cost differs among different cloud providers.) Next, we describe in detail our approaches.

Lift-and-Shift. LS produces a migration plan that matches as close as possible the on-premises cluster specifications to a set of VM instances. We use the X_{specs} vector type and the *similarity* as objective. For each of X_{specs} input vectors of the on-premises cluster, we recommend the VM instance that is *closer* in the vector space with respect to the Euclidean distance. Typically, on-premises clusters are more powerful that what the typical workloads running on them needs, so LS may result into a costly, in terms of budget, solution, especially when a on-premises cluster is largely underutilized. As an optimization, we prune early all candidate VM instances whose resource specs are inferior to those of the on-premises cluster and ensure that any VM in the recommendation space is at least as powerful as the on-premises node.

Usage-Based-Cost-Reduction. UBCR considers cluster specifications and the usage metrics collected by the system. Each input node is represented by a X_{usage} vector. VM instances that cannot accommodate the workload of the on-premises nodes are ignored. Intuitively, if the on-premises cluster is underutilized UBCR has larger search space than LS. The candidate VMs are sorted with respect to a desired objective, e.g., *similarity* or *cost*. The algorithm recommends the top VM in the sorted list. Since UBCR accounts for the actual cluster usage and opts to minimize the cost, it generates cloud deployment recommendations that in general have better utilization than LS at a reduced operational cost.

WorkloadFit. This approach differs from the previous strategies since it does not aim to produce a *1-1* VM type recommendation mapping for each node, but rather analyzes the cluster's workloads to find the required resources and which assignment of VM types will yield those requirements.

This is first done by generating a 24 hour x 7 day *heatmap* for each resource r such as cpu, memory, and disk, given a

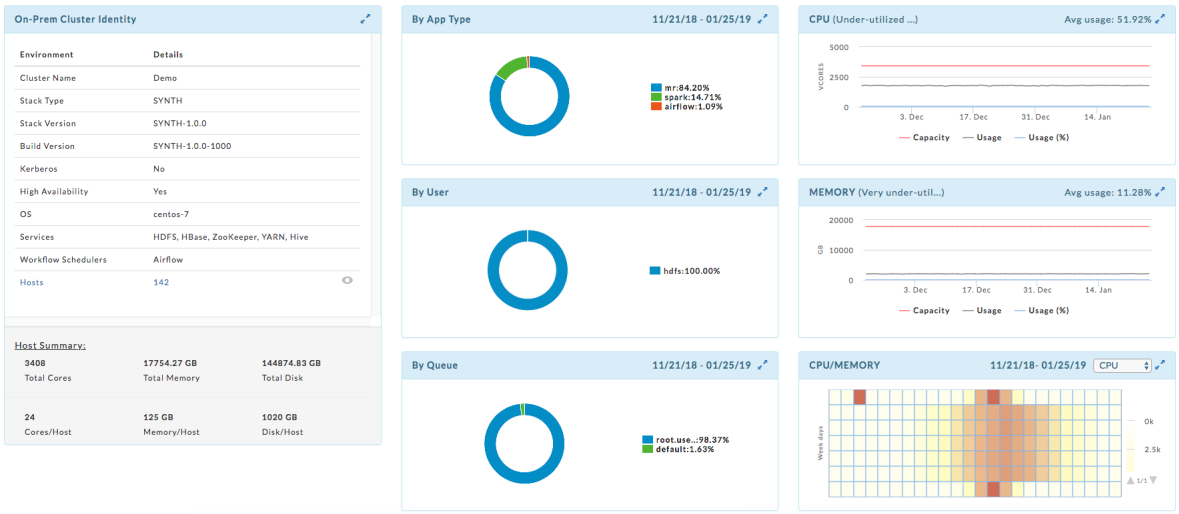

Figure 2: Cluster discovery dashboard

time interval [start, end] (see the right bottom corner of Figure 2). Since we compute metrics for the entire cluster, we compute total capacity and utilization of cpu, memory, and disk across all cluster nodes. Hence, cluster utilization is computed through *usage/capacity*, e.g., for CPU as $CPU_{util} = \frac{cores_{used}}{cores_{available}}$, and similar for memory and disk. Thus, a heatmap of CPU utilization can be created as:

$$Heatmap(cpu\_util) = \frac{\sum_{node}^{nodes} avg(timeseries(H, cores_{used}))}{\sum_{node}^{nodes} avg(timeseries(H, cores_{total}))}$$

and every time slot has a utilization ratio, where $timeseries(H, r)$ is the time series data of the resource r of node H.

Next, we rank the utilization ratio into percentiles to generate a [0-100]-ranged *heat score*, indicating how relatively hot that resource was utilized. For example, consider that on average on Monday at 9:00am the cluster had a total capacity of 1000 virtual cores, 500 cores utilized, utilization percentage of 50, and perhaps that was the highest utilization making it have a *heat score* of 100. Using the *heat score*, we compute how much cpu and memory are needed to meet the requirements of the X^{th} percentile of all workloads. So that meeting the 100^{th} cpu percentile will need the maximum amount of say cpu observed during any of the 24x7 time slots. If we combine this for multiple resources, then for example, to meet 90% of all workloads, we need to meet the 90^{th} percentile of $[cpu, memory, disk]$, so that $SLA(90\%) = [cpu_{req}(90\%), mem_{req}(90\%), disk_{req}(90\%)]$.

In order to meet $X\%$ of all workloads, we consider all assignments of a single VM type or up to 2 VM types that can produce the necessary requirements. Hence, the solution space has $n + \binom{n}{2}$ possible VM type(s) to consider in a solution. When considering solutions of a single VM type such as EC2 r4.4xlarge, we know that the VM type has 16 cores; so if the requirements are for 161 cores, we will need 11 of those VM instances. That same VM type has 122 GB of RAM, so if the requirements are for 1464 GB of RAM, then we will need 12 of those instances. Since we want to meet the requirements for both CPU and memory, we take the max, which is 12. When considering solutions of two VM types such as EC2 r4.4xlarge and r4.2xlarge, then if the requirements call for 168 cores, then we can suggest 10 instances of r4.4xlarge and 1 instance of r4.2xlarge. Each assignment of VMs that meets the requirements also has an associated hourly cost to run, so we optimize for cost by picking the solution with the lowest value. This allows our recommendation algorithm to show for each percentile how many resources are needed, what is the optimal assignment of VM types, how many instances of each VM type are needed, and what is the total cost to run. We also provide an option to "round-up" the mapping choice to favor a larger group of same VM types instead of multiple VM types.

3 DEMONSTRATION

Our presentation will demonstrate cluster discovery and cloud mapping scenarios using two multi-node clusters. Our demonstration script begins by showing how we profile on-premises clusters and then, we will investigate alternative scenarios for cloud migration.

Cluster Discovery. We will create and explain cluster profiles, investigate various usage patterns (e.g., compute or memory intensive, periods with heavy/light load). As an example, Figure 2 depicts a snapshot of a cluster discovery dashboard for a cluster of 142 nodes. On the left, it shows information about the cluster specification. In the middle, it shows an aggregated view of application types, user and queue usage on the cluster. On the right, it shows aggregated resource usage, including a heatmap that shows usage patterns spanning a period of two months.

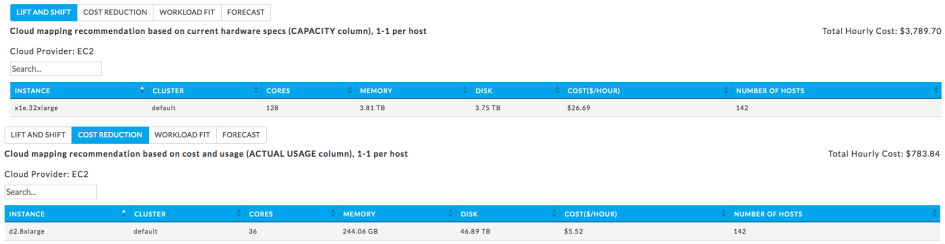

Figure 3: Cloud mappings options: LS and UBCR

Scenario 1: Moderate load. We assume a cluster with moderate load and show how our techniques lead to migration strategies with varying cost. In this scenario, LS results in an expensive solution, whereas usage and workload aware strategies are better fit. Figure 3 shows the cost of the LS plan ($3,790/hour) and UBCR plan($784/hour) in this scenario[1].

Scenario 2: Heterogeneous workloads. Running different types of applications and workloads on the same cluster is typical. We will demonstrate a scenario where 30% of the cluster is being used by Spark MLlib to run distributed machine learning jobs, which are largely memory and CPU intensive applications. 70% of the cluster is reserved for ETL and also by HDFS serving as Datanodes, and these typically have large disk I/O footprint. An example migration plan would be to use 100 *storage-dense* d2.xlarge instances and 42 *memory-optimized* x1.16xlarge instances.

Scenario 3: Multiple usage-workload patterns. We will investigate how various usage patterns affect the cloud migration cost and will explore in detail how UBCR, WF, and FR provide cost effective solutions. Figure 4 shows example options of choosing alternative workload fit strategies and how these are associated with cost and workload coverage.

Finally, for off-script presentation and discussion, we will provide interactivity, where the participants can browse example configurations and cloud mapping recommendations. Example scenarios we could explore include various heatmap patterns, cost reduction with datacenter takedown, cloud-to-cloud migration, what-if analysis including forecasting scenarios, resource usage partitioning by breaking the workload into smaller clusters, and so on.

4 CONCLUSIONS

We presented a system architecture and alternative design strategies for enabling effective cloud migration. There are efforts that we consider complementary to our work in research areas in the database and systems community which have addressed challenges in stacks composed of multiple systems, self-driving databases, workload and resource management for the cloud, and so on (e.g., [2, 3, 5, 8–10]). There are also cloud migration related works focusing on specific platforms and use cases (e.g., [6, 7]). Our key differentiator

[1]Cost obtained from endpoints like https://awstcocalculator.com/

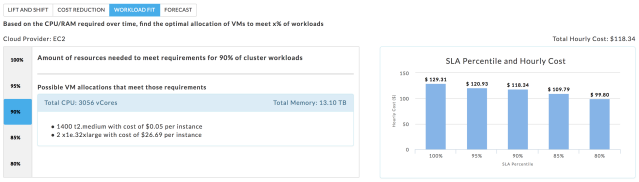

Figure 4: Workload fit options

is our focus on the entire big data stack and on large-scale, enterprise-grade applications and deployments.

At Unravel, we are building the next generation performance management system by solving real-world challenges arising from the big data stack which is a gold mine of data for applied research, including AI and ML. We are working with many enterprises that have challenging problems; and by helping them understand and address these problems, we help them scale at the right cost.

REFERENCES

[1] A. Arvanitis et al. 2019. Automated Performance Management for the Big Data Stack. In *CIDR 2019*.

[2] Martyn Ellison, Radu Calinescu, and Richard F. Paige. 2018. Evaluating cloud database migration options using workload models. *J. Cloud Computing* 7 (2018), 6.

[3] G.J.L. Paulraj et al. 2018. A combined forecast-based virtual machine migration in cloud data centers. *Computers & Electrical Engineering* 69 (2018), 287–300.

[4] Herodotos Herodotou and Shivnath Babu. 2011. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. *PVLDB* 4, 11 (2011).

[5] K. Lolos et al. 2017. Elastic management of cloud applications using adaptive reinforcement learning. In *IEEE Big Data 2017*.

[6] Ali Khajeh-Hosseini, David Greenwood, and Ian Sommerville. 2010. Cloud migration: A case study of migrating an enterprise it system to iaas. In *IEEE CLOUD, 2010*.

[7] Ali Khajeh-Hosseini, Ian Sommerville, Jurgen Bogaerts, and Pradeep Teregowda. 2011. Decision support tools for cloud migration in the enterprise. *arXiv preprint arXiv:1105.0149* (2011).

[8] Ryan Marcus and Olga Papaemmanouil. 2016. WiSeDB: A Learning-based Workload Management Advisor for Cloud Databases. *PVLDB* 9, 10 (2016).

[9] Ryan Marcus and Olga Papaemmanouil. 2017. Releasing Cloud Databases for the Chains of Performance Prediction Models. In *CIDR 2017*.

[10] Jennifer Ortiz, Brendan Lee, and Magdalena Balazinska. 2016. PerfEnforce Demonstration: Data Analytics with Performance Guarantees. In *SIGMOD 2016*.