Doppler: Automated SKU Recommendation in Migrating SQL Workloads to the Cloud

Joyce Cahoon* Microsoft Redmond, WA jcahoon@microsoft.com

Wenjing Wang* Microsoft Redmond, WA wenjwang@microsoft.com

Yiwen Zhu* Microsoft Redmond, WA yiwzh@microsoft.com

Katherine Lin Microsoft Redmond, WA katlin@microsoft.com

Sean Liu Microsoft Redmond, WA seliu@microsoft.com

Raymond Truong Microsoft Redmond, WA ratruong@microsoft.com

Neetu Singh Microsoft Redmond, WA nesin@microsoft.com

Chengcheng Wan† University of Chicago Chicago, IL cwan@uchicago.edu

Alexandra Ciortea Microsoft Redmond, WA aciortea@microsoft.com

Sreraman Narasimhan Microsoft Redmond, WA sreraman@microsoft.com

Subru Krishnan Microsoft Mountain View, CA subru@microsoft.com

ABSTRACT

Selecting the optimal cloud target to migrate SQL estates from on-premises to the cloud remains a challenge. Current solutions are not only time-consuming and error-prone, requiring significant user input, but also fail to provide appropriate recommendations. We present Doppler, a scalable recommendation engine that provides right-sized Azure SQL Platform-as-a-Service (PaaS) recommendations without requiring access to sensitive customer data and queries. Doppler introduces a novel price-performance methodology that allows customers to get a personalized rank of relevant cloud targets solely based on low-level resource statistics, such as latency and memory usage. Doppler supplements this rank with internal knowledge of Azure customer behavior to help guide new migration customers towards one optimal target. Experimental results over a 9-month period from prospective and existing customers indicate that Doppler can identify optimal targets and adapt to changes in customer workloads. It has also found cost-saving opportunities among over-provisioned cloud customers, without compromising on capacity or other requirements. Doppler has been integrated and released in the Azure Data Migration Assistant v5.5, which receives hundreds of assessment requests daily.

PVLDB Reference Format:
Joyce Cahoon, Wenjing Wang, Yiwen Zhu, Katherine Lin, Sean Liu, Raymond Truong, Neetu Singh, Chengcheng Wan, Alexandra Ciortea, Sreraman Narasimhan, and Subru Krishnan. Doppler: Automated SKU Recommendation in Migrating SQL Workloads to the Cloud. PVLDB, 15(12): 3509 - 3521, 2022.
doi:10.14778/3554821.3554840

*Authors contributed equally.
†Work done while interning at Microsoft.

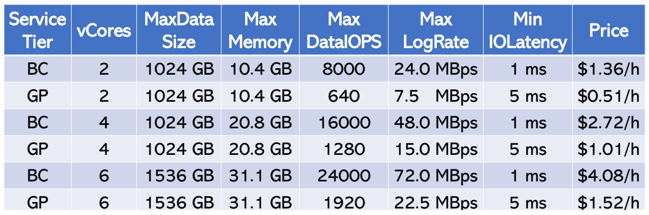
Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554840

1 INTRODUCTION

The complexities of migrating SQL estates from an on-premise data platform to the cloud cannot be understated. An extensive assessment process first takes place to evaluate what cloud targets can accommodate existing workloads. Identifying the optimal targets remains a challenge, as it not only involves understanding the compute resources required to handle customer workloads, but also involves analyzing the legacy systems as a whole—its source code, binaries, configuration files, and execution traces—to ensure feature parity and check for compatibility issues that may arise in the migration process. If (or when) optimal targets are found, multiple stakeholders must then be mobilized to execute the migration to ensure data and their applications can be ported and remains intact and secure during the process. Without proper planning, migration can lead to degraded workload performance and higher costs.

Since the DBMS market is estimated at $64.8 billion [11], and it is predicted that 75% of all databases will be migrated (or deployed) from a cloud platform [10], the total addressable market (TAM) for migrating on-premise data platforms to PaaS offerings is large. Providers have funneled resources to provide an ecosystem of solutions to ease the migration process. Current solutions range from increasing the diversity of cloud targets, also known as Stock Keeping Units (SKU), to automating various steps of the migration process to meet customer preferences in terms of budget and performance. Figure 1 illustrates a few examples of different Azure SQL Database SKU offerings, but this only accounts for about 2% of all the possible SKUs. They are architected to cater to a variety of customer workload requirements in terms of transaction rates, latency, throughput, CPU, memory, and storage.

Many tools exist to assist the process of selecting the right cloud target [19, 22]. Despite the utility of these strategies, they require significant input from the customer, and the final recommendations may still be inappropriate. As a result, cloud providers default to manual SKU selection as the de facto standard as the decision support systems proposed (e.g., [5, 19, 22]) are too hard to use and are difficult to scale. Our field experiences have shown that proper



Service Tier	vCores	MaxData Size	Max Memory	Max DataIOPS	Max LogRate	Min IOLatency	Price
BC	2	1024 GB	10.4 GB	8000	24.0 MBps	1 ms	$1.36/h
GP	2	1024 GB	10.4 GB	640	7.5 MBps	5 ms	$0.51/h
BC	4	1024 GB	20.8 GB	16000	48.0 MBps	1 ms	$2.72/h
GP	4	1024 GB	20.8 GB	1280	15.0 MBps	5 ms	$1.01/h
BC	6	1536 GB	31.1 GB	24000	72.0 MBps	1 ms	$4.08/h
GP	6	1536 GB	31.1 GB	1920	22.5 MBps	5 ms	$1.52/h

Figure 1: Examples of 6 Azure SQL SKU offerings [30, 32, 37].

SKU selection remains a problem that can take up to 60% of the total migration journey. Hundreds of thousands of databases are assessed per year, but only a fraction proceeds to migrate. There does not yet exist a straightforward way for customers to exit their existing data centers and land on an optimal cloud SKU [1, 12, 13]. As a result, a large segment of customers with SQL estates on-premise abstain from moving their workloads to the cloud because of the risks that come with the migration process [1, 3, 4, 12, 13, 21].

In this paper, we develop a data-driven SKU recommendation tool that automates the PaaS cloud target selection process and addresses the following challenges:

Privacy concerns. The most accurate approach for cloud SKU selection is workload replay. It requires accessing user data and query history and replaying the exact same operations on a new cloud SKU as the customer had executed in an on-premise data platform. However, this approach is not practical as there are few customers that are comfortable with providing such unrestricted access. Recent work has demonstrated attempts to replay anonymized data, but this methodology still requires access to the underlying raw user files to do so [8].

Too many SKU options. Microsoft Azure alone has over 200 different PaaS cloud SKUs. These SKUs are primarily segmented by deployment type, Azure SQL Database (DB) or Managed Instance (MI), and service tiers, i.e., General Purpose (GP) or Business Critical (BC)—the latter of which offers higher resilience to failure. The sheer volume of options often leads to decision paralysis, increasing the inertia among customers to modernize and migrate to the cloud.

Lack of transparency in SKU choice. While enough data now exists to leverage black-box machine learning (ML) methods for SKU selection, these approaches lack interpretability. This includes a previous internal attempt to do automated SKU recommendation by training neural nets on workload characteristics, but the cloud targets selected from these models cannot be explained. As the risk associated with migration is high, customers need to understand why a specific SKU choice is made.

Lack of customized solutions. Field engineers depend on intuition and migration experience to gauge a customers' workload requirements. For example, some applications can function under memory constraints, while others have no tolerance for hitting specific resource boundaries. Since migration assessment still depends on expert opinion and ad-hoc analyses, and customers vary greatly in their tolerance for these risks, e.g., some are willing to sacrifice on certain aspects of performance for increased cost savings [4], it is difficult to develop a fully automated system that quantifies these differences in preference and captures this complex decision-making process by experienced engineers.

Introduction to Doppler. Doppler relies solely on customers' workload performance history, or low-level resource statistics, such as latency and memory usage, as input to make a data-driven SKU selection. We address the aforementioned challenges as follows:

Doppler relies only on resource consumption patterns to evaluate the suitability of SKUs. We circumvent the problem of accessing customer data and query history by characterizing workload behavior solely through resource consumption patterns. Exploratory analyses of various workloads (and their subsequent performance history) suggest that such low-level resource statistics are sufficient to capture differences in workload, and thus differences in resource demands. Our novel methodology addresses current challenges in privacy and security and avoids time-consuming processes, like replaying customer workloads [22] and building complicated workload simulation platforms [43]. It can be easily extended to accommodate additional performance dimensions outside of the current feature set.

Doppler provides an automated and interpretable process for SKU recommendation. Existing migration tools concentrate on optimizing cost. Doppler presents a suite of optimal SKUs in the context of, not only cost, but also performance. Doppler summarizes the trade-offs between cost and performance for each SKU in the form of a price-performance curve, an example of which is shown in Figure 4b. Our use of our customer workload's performance history allows us to judge how well each workload will perform on a new cloud SKU based on concrete metrics like compute, memory, latency and throughput. Since customers' appetite for risk varies, Doppler outputs a price-performance curve to rank how various SKUs fulfill performance needs, along with its subsequent impact on cost, to provide the customer with the agency needed to make an informed decision.

Doppler "learns" from previous optimal[1] *SKU choices vetted by migration experts.* Based on Azure cloud customers (who have fixed their chosen SKUs for at least 40 days), we segment the customers based on their performance footprint as a means of categorizing their workload behavior. This involves studying how their price-performance curves change and observing what SKU they fix their workloads to based on the resulting price-performance curves. There are a large number of customers willing to negotiate on certain performance dimensions, like latency and throughput, to realize cost savings. By generating price-performance curves and measuring what performance dimensions may be negotiable, we can quantify customer preferences in a way that aligns with how experts vet optimal cloud targets. In other words, we introduce a data-driven strategy that automatically characterizes customers' workload needs as shown in Figure 3 in order to provide personalized SKU recommendations.

Doppler is an integral part of the Data Migration Assistant (DMA) tool [26] and has been used by thousands of customers since its release. We executed an experimental evaluation of Doppler over 9 months of customer data to verify that we can identify optimal cloud SKUs with our framework. Given that Doppler is able to generate the

[1]Since there is no benchmark (method) that definitively marks a SKU as "optimal," we developed our own from performance histories associated with successfully migrated customers that have fixed their SKU for at least 40 days. In this paper, our claims of "optimal" SKUs are established relative to successfully migrated Azure customers, excluding customers that are over-provisioned.

Table 1: DMA tool adoption since its release.

Month	Unique instances assessed	Unique databases assessed	Total recommendations generated
Oct-21	185	3,905	6,503
Nov-21	215	3,389	4,802
Dec-21	57	4,185	5,364
Jan-22	231	9,090	10,674

exact same SKU choice as 89.4% and 96.7% of successfully migrated SQL DB and MI customers, whose SKUs were vetted by migration experts, Doppler was subsequently implemented in Azure's DMA tool. Every new Azure migration customer uses DMA to begin an assessment of their on-premise data servers. Given the ease of use and utility of this migration assistant, it has been utilized by hundreds of customers since its release in October 2021 (see Table 1).

Doppler provides a consistent framework for making SKU recommendations not only for new migration customers but also existing cloud customers. In leveraging Azure customers' historical performance profiles, we found that we were not just able to increase the accuracy of our SKU recommendation for migrating new customers' SQL estates from an on-premise data platform to the cloud, but also right-size SKUs for our existing customer base. In one recent customer survey, over a 30-day observation window, we found that 30% of SQL databases consume 43% or less of provisioned CPU resources, and that only 5% of SQL databases reach the maximum provisioned CPU usage for more than 10% of this study's duration; a key indicator that, we have a segment of customers whose resources are over-provisioned. There is a need to not only provision new customers moving from on-premise to the cloud correctly, but also right-size existing customers to benefit from cost-saving opportunities.

Contribution. Our contributions are summarized as follows:

- Introduce a fully-automated SKU recommendation framework based on price-performance curves.
- Profile existing Azure SQL customers to understand negotiability around various performance dimensions and preferences with respect to cost/performance trade-offs.
- Deploy a recommendation engine in the Azure Data Migration Assistant v5.5 [26, 34] that has been used by 600+ unique customers.

The remainder of this paper is organized as follows. Section 6 reviews related work. Section 2 details the offerings in Azure SQL PaaS, and Section 3 presents the end-to-end architecture of Doppler. Section 4 discusses production integration efforts. Section 5 presents experimental results from back-testing on migrated customers and compares the recommendations from Doppler with an alternative baseline solution. Remaining challenges and promising directions for future research are stated in Section 7.

2 BACKGROUND

As there is a wide range of cloud offerings, e.g., SaaS (software as a service), PaaS (platform as a service), and IaaS (infrastructure as a service), we narrow the scope of this study to focus on migrating on-premise SQL workloads to Azure SQL PaaS solutions, which includes *Azure SQL Database (DB)* and *Azure SQL Managed Instance (MI)* [27, 31]. Work is ongoing to generalize the Doppler framework to support other migration scenarios, across other database systems like Oracle [40] and PostgreSQL [15]. *Azure SQL DB* offers single database deployment options that create fully managed, isolated databases; while *Azure SQL MI* offers fully managed SQL servers that host a large number of databases. Within the *virtual cores (vCore)* purchasing model [36], two service tiers are offered: *General Purpose (GP)* and *Business Critical (BC)*. The *BC* tier offers higher transaction rates and lower-latency I/O compared to that of the SKUs in the *GP* tier. Azure currently offers over 200 different PaaS cloud SKUs.

As no tool currently exists that provides personalized, accurate and secure SKU recommendations among this set of 200 cloud targets, the Azure Data Migration Assistant (DMA) [26] was conceived to ease and reduce the risks associated with migration. DMA plays a key role in the general Azure Migration Service that provides a unified migration experience for all Azure products. As shown

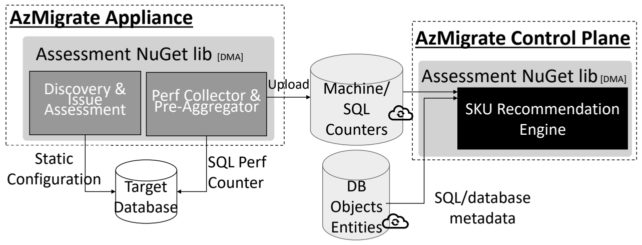


Figure 2: How our SKU recommendation engine fits in the current architecture of the Azure Migration Service.

in Figure 2, the AzMigrate Appliance consists of 2 modules: (1) Discovery & Issue Assessment to detect compatibility issues that would either block or hinder the customer's ability to migrate (e.g., unsupported, or partially supported, features that are currently in use on-premise) and (2) Performance Collector & Pre-Aggregator to gather SQL performance (perf) counters on CPU, storage, memory, IOPs, and latency. The static configuration and the perf counters are first stored locally on the target database. The perf counters are subsequently uploaded, along with SQL DB metadata, as input to the recommendation module in the Control Plane, where the following two approaches are implemented in the recommendation engine:

- **Baseline strategy.** As Doppler was not available when the DMA tool was first conceived, a naive, baseline algorithm was introduced. This SKU selection procedure involves taking the entire time-series vector collected on each available perf counter (e.g., CPU, memory) and collapsing it into one scalar value. Field engineers often chose either the max of each perf vector, or some large (95%) quantile. From these values, the cheapest Azure PaaS offering that satisfies all the requirements is suggested. Since max (or large percentiles) determine the customer's resource needs, this approach generally results in over-provisioning.
- **Doppler (elastic) strategy.** The motivation for Doppler was to improve upon the existing baseline approach. It is

the first model-driven recommendation system that has been integrated into the DMA tool in production. Since its release in DMA v5.5 [26] in October 2021, our recommendation engine has been widely used by a large number of both external and internal customers (see Table 1). In the following sections, we detail this approach and its integration with DMA.

3 DOPPLER ENGINE

In this section, we present an overview of the Doppler system architecture and the design of each module and its assumptions.

3.1 Design Principles and Architecture

The following core design principles capture the motivation behind the architectural design of Doppler.

Avoid using customer data/queries. With the need to comply with General Data Protection Regulation (GDPR) [39, 49], and the increasing concerns about data privacy and security, we developed a solution that is solely based on low-level resource consumption statistics. From our exploratory efforts, we discovered that we can generally infer various levels of *resource throttling* when we compare customers' performance history against the resource capacities of different SKUs, which is an important input for the recommendation.

Consider cost and marginal benefits. As a more expensive SKU results in better performance, we want to match migration customers to the most cost-efficient SKU and avoid over-provisioning, or instances where compute resources sit idle. We introduce our novel price-performance methodology as a starting framework for customers to understand the trade-offs between cost and performance (as measured by resource throttling probabilities).

Leverage what we know from existing customers. Since we already have a large base of (successfully) migrated customers, we want to learn from their SKU choices. In particular, we gain some understanding of their level of satisfaction with their migrated SKU by studying their behavior with regards to how often they switched SKUs. Given this telemetry, and their associated performance footprint, we are able to generate their price-performance curves and observe where on their respective price-performance curves do their SKU choices land. In other words, we have insight into customer preferences for cost over performance, and we leverage this knowledge to cluster customers into various different groups. With these profiles, we can guide new migration customers towards choosing their optimal SKU based on similarities in resource utilization to our existing migrated base.

Make sure the solution can scale. Since the de facto standard of manual SKU selection takes too long, we need an effective way for new migration customers to quickly get personalized and accurate SKU recommendations. Azure Migration Service was designed as highlighted in Figure 2 in order to make such automated SKU recommendations possible and support a higher volume of migration.

Doppler achieves the aforementioned design ideas with the system as outlined in Figure 3. Our SKU recommendation engine consists of two main modules: the *Price-Performance Modeler (PPM)* and the *Customer Profiler*. The *PPM* improves upon the current baseline

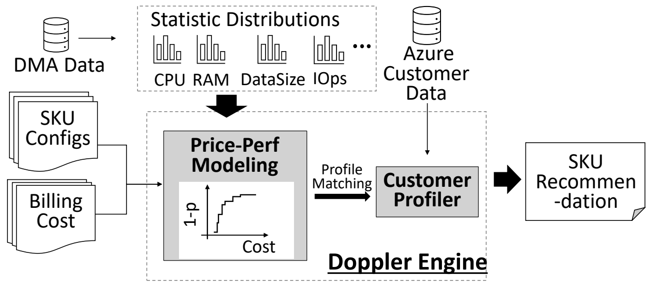


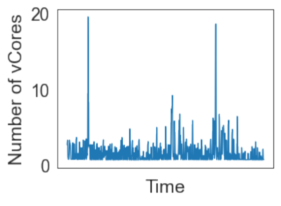
Figure 3: The Doppler engine.

strategy as it is capable of providing more flexible (elastic) SKU recommendations. It does not require that the SKU meet the customers' (max) resource use, and thus avoids over-provisioning. It takes three inputs: (i) performance counters collected from the DMA tool; (ii) all the possible cloud target PaaS SKUs; and (iii) the real-time pricing associated with each SKU. Since (ii) and (iii) are primarily fixed, the key input to the PPM is the customer performance history, which currently consists of four perf dimensions: CPU, memory, IOPs and latency. By relying on these low-level resource statistics, we circumvent the need to access customer data/queries, addressing privacy concerns. Moreover, unlike the baseline approach that reduces each time-series vector into one scalar value, PPM utilizes the full distribution of each perf dimensions' data. From these inputs, we generate a price-performance curve, which effectively provides a personalized rank of all the relevant SKUs a customer can migrate towards. With this price-performance result and internal insights from customer SKU selections in the cloud, the *Customer Profiler* can match new migration customers to our existing base to help guide them towards one optimal SKU choice. By leveraging the historical decisions from our (successfully) migrated customers, Doppler allows new customers to make a more informed decision.

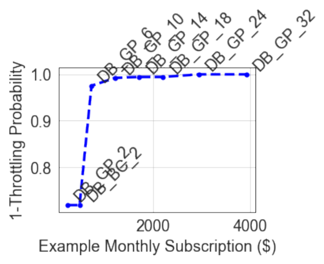
While the current version of Doppler targets workloads moving to Azure SQL DB and MI [33], Doppler can be easily extended to accommodate additional performance features and adapted to support migration scenarios for different database systems.

3.2 Price-Performance Methodology

The price-performance curve relates the price, in terms of monthly billing cost, of relevant cloud SKUs to their respective performance, as measured by each SKUs ability to fulfill the customers' resource needs for a pre-specified assessment period. One of the key challenges to Doppler is estimating performance appropriately, especially when it involves different hardware that the workload has not yet been executed upon. Current solutions have averted this challenge by focusing instead on replaying the customer workload on recommended SKUs [22], or if access to customer data is impractical, simulating workloads to replay [43]. By taking these more direct routes, these solutions are able to strictly demonstrate what the performance outcomes will be when migrating workloads to a new SKU. But these strategies also rely on strong assumptions (e.g., [24]) and extensive data access (e.g., [44, 52]). We circumvent these problems, as well as, the fact that replay is often not feasible given the large number of possible SKUs, by developing a proxy for estimating performance, which we call the probability of resource throttling.



(a) CPU usage by time.



(b) Price-performance curve.

Figure 4: Example of price-performance curve generation from performance history.

In Section 5.4, we demonstrate how this new metric provides a good approximation of the true throttling that a customer might experience, and thus is a sufficient means to approximate how well each SKU meets customer workload performance needs. The *throttling probability* of SKU_i, where $i = 1, \ldots, m$ represents some SKU among the m relevant Azure SKUs, is symbolized by $P_n(\text{SKU}_i)$. It is defined as the probability of running into resource throttling in *any* resource dimensions for customer n:

$$P_n(\text{SKU}_i) = P(r_{\text{CPU}_n} > R_{\text{CPU}_i} \cup r_{\text{RAM}_n} > R_{\text{RAM}_i} \cup \ldots \cup r_{\text{IOPS}_n} > R_{\text{IOPS}_i}). \quad (1)$$

$r_{\{\text{CPU}_n, \text{RAM}_n, \ldots, \text{IOPS}_n\}}$ denotes the vector of random variables corresponding to the resource usage for customer n as collected by the DMA tool, and $R_{\{\text{CPU}_i, \text{RAM}_i, \ldots, \text{IOPS}_i\}}$ denotes the maximum capacity for each perf dimension as fixed by a specific SKU_i. As data becomes available for additional resource dimensions (e.g., wait stats), the throttling probability definition can be extended accordingly. Based on our mathematical formulation above, there are certain performance dimensions that require small adjustments; for example, IO latency is taken as the *inverse* of the actual IO latency in order to calculate the effect of this performance dimension relative to an upper bound (R_{IOPS_i}) for each possible SKU. Figure 4 illustrates an example of a customer workload's CPU usage by time and the corresponding throttling probability when only the perf dimension of CPU is factored into this new metric.

For our SKU recommendation engine, we focus primarily on the four performance dimensions of CPU, memory, IOPs and latency. For customers that are specifically interested in migrating towards Azure SQL DB, we include two additional dimensions of log rate and storage. Since estimating the probability of throttling requires modeling all these resource dimensions *jointly*, we initially considered various statistical methods, such as multivariate kernel density estimation based on vine copulas [38] and Gaussian smoothing [45]. While these existing approaches can do a sufficient job generating the price-performance curve, the time it takes to do so is impractical. Given the complexity of these traditional estimation methods, we default to a *non-parametric multi-variate* approach. This estimation technique simply requires calculating the frequency with which all performance dimensions $r_{\{\text{CPU}_n, \text{RAM}_n, \ldots, \text{IOPS}_n\}}$ are satisfied by each SKU, at each time point. Section 5 highlights how this simple approach can achieve accurate SKU recommendations. For both the parametric and non-parametric approaches considered, we enforce monotonicity on the price-performance output so that customers cannot select SKUs that are more expensive and less performant.

Table 2: File IO characteristics associated with various Azure SQL MI General Purpose (GP) SKUs. [32]

Storage Tier	P10	P20	...	P50	P60
File size	$[0, 128]$ GiB	$(128, 512]$ GiB	...	$(2, 4]$ TiB	$(4, 8]$ TiB
IOPS	500	2300	...	7500	12500
Throughput	100 MiB/s	150 MiB/s	...	250 MiB/s	480 MiB/s

Determining file storage tier for MI. In order to make appropriate Azure SQL MI SKU recommendations, we introduce slight adjustments to our resource throttling probability calculation. This is required because, unlike SQL DB SKUs, SQL MI General Purpose (GP) IOPs resource limits are dynamically scaled to accommodate the allocated file size. This happens since the data layer for SQL MI is implemented using Azure Premium Disk storage, and every database file is placed on a separate disk. Each disk has a fixed size, and bigger disks are associated with better throughput and IOPs. From a migration standpoint, since the SKU choice for MI customers begins with fixing the file layout (i.e., a customer can choose an MI SKU that creates 3 files that can each fit within a 128GB disk), we adhere to a similar procedure prior to generating the price-performance curve as the IOPs limit $R_{\{\text{IOPs}_i\}}$ is not a fixed variable.

- **Step 1:** ***Find the correct data storage tier based on data size and workload IOPS and throughput needs***. We surveyed a number of field engineers to develop this first filtration step to identify appropriate SQL MI SKUs. We filter down the set of possible SQL MI SKUs to only those who can satisfy the *storage* requirement of the data file at a minimum of 100%. We also verify that this subset of SKUs satisfies at least 95%[2] of the *IOPS* and *file throughput* requirements. In the event that this 95% cannot be reached, we further restrict our search of relevant SKUs to Business Critical (BC) ones in **Step 2**. Table 2 highlights some of the resource limits for different storage tiers.
- **Step 2:** ***Create instance-level price-performance curves***. As **Step 1** parses down the set of SQL MI SKUs only to those that are relevant for the workload at hand, we can then generate the price-performance curve as previously described. The only change is that the IOPs limit $R_{\{\text{IOPs}_i\}}$ is calculated based on the file layout chosen—as the summation of IOPs limit on all the data files.

Limitation. While the price-performance curve is informative, in that it provides a personalized rank for all the relevant SKUs; its final recommendation is not always clear. The price-performance output does not yet take into consideration at which point the customer is willing to negotiate in terms of performance and cost (e.g., if the customer has zero-tolerance for any throttling, are they comfortable with the SKU choice that places them where the price-performance curve first hits 100%?). We initially explored various heuristics to guide the customer towards their "optimal" SKU choice, this includes:

- *Largest Performance Increase*: Selecting the SKU that sits after the point in the price-performance curve where the

[2]The rate 95% is chosen based on file layout analysis of current on-cloud Azure SQL MI resources.

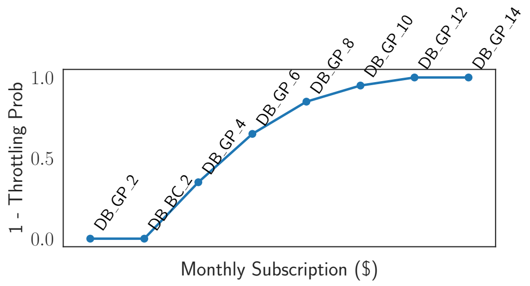


Figure 5: Example of a complex price-performance curve. Customer chosen SKU is SQL DB General Purpose 14 cores.

difference in the throttling probability is no longer significant. For example, where $P_n(\text{SKU}_i) - P_n(\text{SKU}_{i-1}) \leq \epsilon$, where ϵ is set as .001.

- *Largest Slope*: Selecting the SKU that sits after the point in the price-performance curve that has the largest slope (or rate of change) in the throttling probability. In other words, SKU_i that maximize $\frac{P_n(\text{SKU}_i)-P_n(\text{SKU}_{i-1})}{\text{Price}(\text{SKU}_i)-\text{Price}(\text{SKU}_{i-1})}$.
- *Performance Threshold*: We pick the first SKU whose throttling probability is greater than some predefined threshold, where $P_n(\text{SKU}_i) \geq \gamma$.

The problem, however, with these heuristics is that they do not always result in the optimal SKU choice. We verify this based on an assessment of existing migrated Azure customers. Assuming that the SKU choice that these customers have fixed (for at least 40 days) is the optimal SKU, we generate the price-performance curves for each customer and identify where their fixed SKU choice lands on their respective price-performance curves. Neither of these heuristics is able to accurately capture the optimal SKU. Moreover, in the scenarios where the price-performance curve is slightly more complex, as shown in Figure 5, these heuristics are not robust. Namely, following the largest performance increase strategy, this price-performance curve would recommend the optimal SKU as SQL DB General Purpose 6 cores; following the largest slope strategy, the optimal SKU would be SQL DB General Purpose 4 cores; and following the performance threshold strategy, the optimal SKU would be General Purpose 12 cores (if γ was set to 95%).

We also need to consider the limitation that customer workload requirements are highly variable. For example, some applications have high memory requirements; if it runs out of memory, the job will fail, which is worse than running into IO throttling. Other applications might require operating with very low latency, hence delays as mentioned prior, may not be acceptable. Much of the manual SKU recommendation process involves capturing these nuances in workload requirements and understanding what the customer is comfortable with. As this process of measuring workload perf requirements, however, remains much more of an art, we introduced Doppler as one means to better quantify these workload needs and measure what particular performance dimension may be negotiable or not. More specifically, as discussed in the section to follow, we want to capture customers' tolerance for throttling across various types of workloads.

We address these limitations and develop a robust means of SKU selection: one that is not as easily influenced by the shape of the price-performance curve. Our final strategy, as outlined in the section to follow, leverages our experience with thousands of successfully migrated customers and allows new customers to learn from their choices in a more systematic way. Since price-performance curves are able to inherently quantify nuances in each customer's preference, and we are able to study price-performance curves generated across Azure SQL PaaS customers at scale, we augment the Doppler framework with a profiling module that clusters customers based on their workload performance behavior and how much they are willing to negotiate on various perf dimensions. By doing so, we provide an explicit means for new customers to see how their own workloads compare, and thus make an informed SKU decision.

3.3 Customer Profiling

Since the price-performance curve alone only provides a personalized rank of relevant SKUs for handling new workloads, multiple strategies were tested to develop a principled approach towards selecting one optimal SKU. Unfortunately, the approaches discussed prior that focus purely on the shape of the price-performance curve are insufficient for identifying the best SKU. Moreover, these heuristics fail to capture differences in customer preferences for price versus performance as well as sensitivity to resource throttling in different resource dimensions. We thus study resource utilization patterns in each resource dimension, and the respective price-performance curves, among customers that have successfully migrated to further distinguish between different types of workloads that might have different tolerance levels for resource throttling. Given customers that exhibit similar resource utilization patterns in the cloud, new customers gain insight into what their optimal SKU might be.

We execute standard ML clustering techniques to profile successfully migrated customers into distinct groups. Extensive interviews with a few of our solution architects provided the domain expertise needed to understand customer preferences in price and performance, and how it manifests in their resource utilization patterns. We characterized customers based on what performance dimension (e.g., CPU, memory, IOPs, latency, etc.) they are willing to "negotiate" on. While we did not have the resources to survey each customer, we employed the same rule our customer-facing engineers used: if the *spikiness* of customers' performance counters is rare and short-lived, consider that performance dimension negotiable (for a short period of time). The customer profiling module of Doppler thus characterizes each performance dimension as "negotiable" or "non-negotiable" by assessing the duration of spikes in the raw time-series (perf counter) data. If the duration is short, the performance dimension is considered less significant, and something that can be negotiated (via cost savings). Based on what performance dimension each customer is willing to negotiate on, they were then clustered into their own respective customer group.

The calculation for spike duration is straightforward. Doppler first identifies the max peak value(s) within the time-series data of each performance dimension. The variances of the counters are also captured, and a window is formed (one standard deviation) below the max value. The total duration in which resource utilization is within this window is then assessed. If the total duration lasts for greater than a threshold percentage (ρ) of the total assessment period, the performance dimension is cast as non-negotiable. Sensitivity analyses were conducted to better tune the ρ threshold. This is referred to as the *threshold algorithm*. Several other techniques were attempted to compare against this thresholding approach; and while some proved useful in summarizing the resource utilization patterns as our field engineers would, we elected this simple thresholding procedure for its transparent interpretation and high performance. Section 5.2 compares these competing procedures.

- *MinMax Scaler AUC*: The area under the curve (AUC) is calculated on the empirical cumulative distribution function (ECDF) for each performance dimension. AUC is used as a means to approximate whether the resource is used steadily. As highlighted in Figure 6, higher AUC values tend to describe workloads that had transient spiky usage. For this strategy, the AUC is derived after the performance values are normalized to take values between 0 and 1 (e.g., min-max scaled).
- *Max Scaler AUC*: Similar to the approach listed above, the AUC is calculated after the resource values are only max scaled (e.g., $\frac{r_i}{max(r_i)}$). This calculation better identifies large spikes in resource use.
- *Outlier percentage*: The portion of (performance) counters that exist at least three standard deviations away from the average were calculated as a means to capture spiky usage.
- *STL variance decomposition*: Using time-series technique, Seasonal & Trend decomposition using Loess (STL) [6], the observed time-series data (R_i) for each perf dimension is decomposed into a trend (T_i), seasonality (S_i) and residual (I_i) component. The residual component is then used for clustering, and transformed as follows: $max(0, 1 - \frac{var(I_i)}{var(R_i)})$, where $var(\cdot)$ denotes variance. This conversion is meant to capture the variance explained by the trend and seasonality; the closer this value is to 1, the more the observed performance is explained by trend and seasonality.
- *MinMax Scaler AUC result combined with thresholding*: The profiling vector produced from MinMax AUC strategy is concatenated with vectors from the thresholding algorithm to characterize each customer.

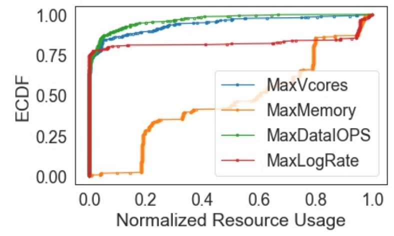
To better capture the heterogeneity in tolerance levels of resource throttling in different dimensions, the group membership g_n for a customer n is designed as a function of the negotiability of each resource dimension. In other words,

$$g_n = f(w_{\text{CPU}_n}, w_{\text{RAM}_n} \dots w_{\text{IOPS}_n}) \tag{2}$$

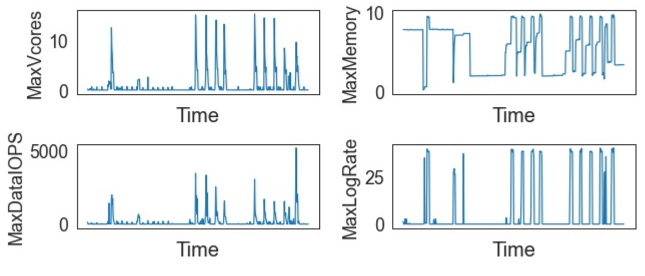
where $w_{\{\text{CPU}_n, \text{RAM}_n, \dots, \text{IOPS}_n\}}$ denotes the negotiability of different resource dimensions as defined in the strategies listed above. A range of standard ML clustering algorithms such as k-means [16] and hierarchical clustering [18] can then be executed on the resulting g_n in order to profile customers into different groups.

Based on the grouping of on-cloud customers, we study their preference in price versus performance by examining where their SKU choice lands on their respective price-performance curves and examine the corresponding throttling probabilities. This is formalized as follows:

$$\overline{P}_g = \underset{\forall n,\, g_n = g}{\mathbb{E}} \left[P_n(SKU_{i_n^*}) \right] \tag{3}$$



(a) Empirical CDF (ECDF) plots.



(b) Raw time series.

Figure 6: The ECDF and time series associated with various performance dimensions.

where $SKU_{i_n^*}$ denotes the chosen SKU for customer n. For a new customer n' who is identified to be in the same group, we assume that a SKU with similar (or slightly lower) throttling probability will be most likely chosen. This SKU can be found by:

$$\min_i |P_{n'}(\text{SKU}_i) - \overline{P}_{g_{n'}}| \tag{4}$$

$$\text{s.t. } g_{n'} = f(w_{\text{CPU}_{n'}}, w_{\text{RAM}_{n'}} \dots w_{\text{IOPS}_{n'}}) \tag{5}$$

$$P_{n'}(SKU_i) \le \overline{P}_{g_{n'}} \tag{6}$$

From the competing procedures listed, the thresholding algorithm provides optimal input for clustering as it is able to segment customers into groups whose price-performance curves are distinct. One concrete example is shown in Figure 4a, in which there is a segment of customers whose workloads exhibit very short (and uncommon) periods of high CPU utilization. Following the baseline strategy, these customers would be over-provisioned and allocated SKUs that meet this high CPU need. However, following the Doppler framework, these customers would be matched to a cloud user group that avoids selecting the SKU that results in over-provisioning; in fact, for the particular customer whose workload is highlighted in Figure 4b, while the cheapest SKU on their price-performance curve (that fulfills 100% of their resource needs) would push them to select an expensive GP machine with 24 cores, with Doppler, they would be able to leverage historical decisions made by similar customers to learn that it is more optimal to select a cheaper SKU. In short, customers that exhibit similar behavior in regard to spiky CPU usage will tend "negotiate" in terms of their CPU resources, and select SKUs with fewer cores, knowing that this choice may result in some level of throttling but larger cost savings over time. By profiling current Azure customers, Doppler provides insight for new migration customers on how similar customers have made their SKU decision and guides them towards a more cost-effective SKU choice.

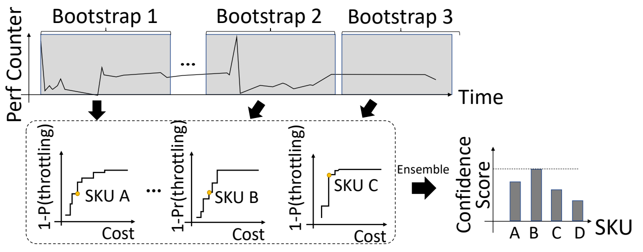


Figure 7: Confidence score based on bootstrapping samples.

3.4 Confidence Score

Since the optimal SKU recommended by Doppler is sensitive to the time window in which performance counters are collected, a secondary metric—the *confidence score*—is developed in order to provide the customer additional certainty towards the final result. This confidence score is derived by bootstrapping the raw customer performance data, generating the respective price-performance curve, profiling the workload based on the bootstrapped data, and obtaining the optimal SKU from this process multiple times. By using a random subset of the data, and repeating the process to generate a Doppler recommendation, we get a better idea of how robust the original SKU recommendation may be. The confidence score is the proportion of bootstrapped runs that have the same recommendation as the original.

As discussed in the section to follow, we typically find that resource utilization patterns that are stable (e.g., no spiky usage) generally have SKU choices made with high confidence scores. In cases where utilization is not consistent, we typically get SKU choices that are associated with low confidence scores. In these particular scenarios, the confidence score can act as a guardrail, suggesting that the customer run the DMA tool for a longer period of time and collect more data (e.g., 1-day's data is often not sufficient to capture standard workload behavior).

4 DMA INTEGRATION

Given the accuracy at which Doppler is able to pinpoint optimal SKU choices from price-performance curves, it has been integrated into the DMA tool [26] and publicly released in October 2021. Three modules were specifically developed for the DMA tool to support the (Doppler) elastic approach:

Data Preprocessing Module transforms the raw time-series data from perf counters into a format that can be ingested by the Doppler recommendation engine. Given that the existing baseline strategy compresses the original data into one scalar value, this separate module is needed to avoid such high dimension reduction. Since the DMA tool is designed to run for several days, preferably weeks, perf counters are collected every 10 minutes, then aggregated at the file, database and instance levels. Additional inputs of relevant SKU resource limits and customer profiles as illustrated in Figure 3 are calculated offline and saved in the application as static input. A billing interface exists to compute the prices for each SKU.

SKU Recommendation Pipeline runs the Doppler Engine to build customized price-performance curves and recommend the optimal SKU based on customer usage profiling. This pipeline depends on the performance counter input, the customer profiling results and relevant SKUs from the data preprocessing module.

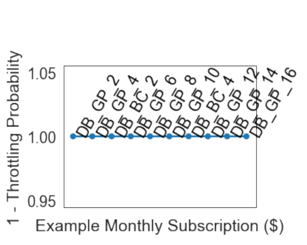
Resource Use Module provides a visualization dashboard for customers to better understand their workload resource needs. It outputs time series and distribution plots of customer usage across various perf dimensions, as well as, the price-performance curve, so that customers can understand why they received a specific SKU recommendation.

The runtime for all the above modules is installed on customers' local machines to protect user privacy. This makes it difficult to automatically track recommendations produced by Doppler as they are currently stored locally. There are surveys that track how content new migration customers are with their Doppler SKU; but efforts are underway to integrate the DMA tool with Azure Migrate Service, which will provide an online means to track every step of a customers' migration journey. We will ultimately be able to keep a record of all the recommended SKUs from Doppler and whether these SKUs were selected for migration, and we will be able to examine the retention of each customer. This feedback loop will be integrated in the Doppler framework, to improve our customer profiling module and help us better understand the preferences of successfully migrated customers.

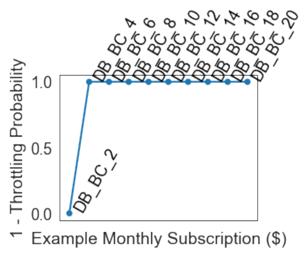
5 EXPERIMENTS

We evaluate the (Doppler) elastic strategy approach using performance history from customers that ran workloads on Azure SQL PaaS and customers that ran workloads exclusively in on-premise data platforms. For cloud customers, we examined the data from customers that ran workloads on Azure SQL DB and SQL MI between June 2020 and March 2021. We filtered down this set to perf counters collected on 9,295 SQL MI and 7,041 SQL DB as this subset includes customers that have fixed their SKU choice for at least 40 days. In order to test how well Doppler works as a SKU recommendation engine, we assume that these migrated customers that have fixed the SKU choice for this duration are satisfied with their SKU, and thus, their fixed SKU is the optimal choice. This assumption of SKU retention as the "optimal" SKU is thus utilized to back-test Doppler. If our framework can match this SKU choice, we assume that we are "correct" and that Doppler is an effective engine for mapping on-premise (and cloud) workloads to the optimal cloud target. For customers that have workloads running exclusively on-premises, we study the performance footprint from 257 SQL servers with 1,974 databases collected from Azure Migrate. For such customers, to evaluate the accuracy of the recommended SKUs, we generate synthetic workloads that mimic the performance history from that of the customer, and we execute these workloads on a subset of machines to test the appropriateness of our SKU choice.

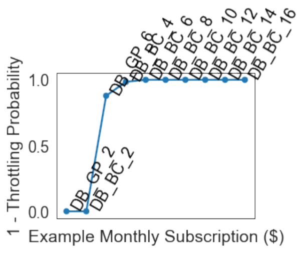
We start our experimental efforts by first comparing the price-performance curves for cloud customers versus on-prem in Section 5.1. We then backtest the Doppler SKU recommendation engine over cloud migration data to understand how well the price-performance curves and subsequent customer profiling work in identifying the optimal cloud target in Section 5.2. Since there is no ground truth label for the optimal SKU for new migration customers, running workloads in on-premise data platforms, we benchmark our SKU recommendations against the baseline algorithm in Section 5.3. We also leverage synthetic workloads to validate the performance of our recommended SKUs in Section 5.4.



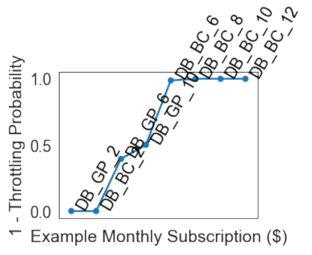
(a) Flat.



(b) Simple.



(c) Complex I



(d) Complex II.

s

Figure 8: Major types of price-performance curves.

5.1 Typical Price-Performance Curves

There are three typical price-performance curve shapes (see Figure 8):

- **Flat**: All relevant SKUs satisfy 100% of the customers' workload resource needs.
- **Simple**: There is a bifurcation between SKUs that either satisfy 100% or 0% of resource needs. Given this division, the cheapest SKU that results in 0% throttling probability is the clear choice.
- **Complex**: The price-performance curve results in a rank of a wide range of SKUs with a variety of throttling probabilities.

Based on the performance histories of workloads running on Azure SQL MI and DB, 73.3% of this subset of SQL DB customers and 74.9% of SQL MI customers exhibit a flat price-performance curve (see Figure 9). In these scenarios, Doppler recommends the cheapest SKU as it is the most cost-efficient option. Among this set, we are able to identify approximately 10% of customers that were over-provisioned, as their fixed SKU choice places them much farther along their price-performance curve. There are a few customers that were paying for SKUs that satisfied 4× their max resource needs. These particular cases are indicative of customers that are highly over-provisioned; Doppler thus is also useful as a means to identify and right-size existing customers.

On the other hand, 26.2% of this subset of SQL DB and 21.7% of SQL MI customers fall in the complex price-performance category. While this segment does not appear significant, compared to the proportion of customers with flat price-performance curves, they account for a large fraction of Azure SQL PaaS revenue. In fact, this subset of customers accounts for more than *all* Azure SQL PaaS revenue from flat price-performance customers combined, hence the need for the Doppler framework.

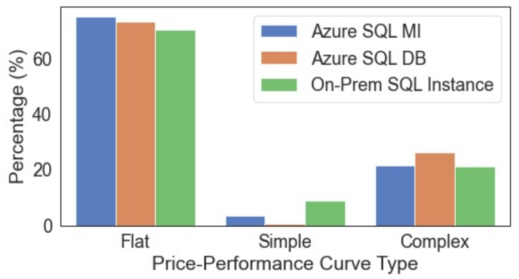


Figure 9: Breakdown of different price-performance curve types within our training data set.

Table 3: Scores associated with each Azure SQL MI customer group (differentiated by the performance dimension negotiability in which 0 denotes negotiable).

Group	vCores	Memory	IOPS	Average (Std) Score
1	0	0	0	0.8500 (0.057)
2	0	0	1	0.9739 (0.054)
3	0	1	0	0.9351 (0.017)
4	0	1	1	0.9692 (0.051)
5	1	0	0	0.9869 (0.026)
6	1	0	1	0.9974 (0.045)
7	1	1	0	0.9668 (0.015)
8	1	1	1	0.9974 (0.056)

Table 4: Accuracy of Doppler in identifying the optimal SKU based on standard k-means clustering.

Negotiability Definition	DB	MI
MinMax Scaler AUC	77.3%	74.3%
Max Scaler AUC	78.5%	73.9%
Thresholding Algorithm	77.6%	75.1%
Outlier percentage	78.1%	74.1%
STL Variance Decomposition	78.1%	74.6%
MinMax Scaler AUC adjusted with timeseries	77.8%	75.5%

The same breakdown of the three typical price-performance curves also exists when applied to performance histories from workloads that are run exclusively in on-premise data platforms (see Figure 9).

5.2 Backtesting with Cloud Data

As there does not yet exist a scalable SKU recommendation engine for selecting optimal SKUs, that map on-premise workloads to the cloud, we need a means to test the accuracy of Doppler. We accomplish this by leveraging internal data we have on successfully migrated customers in Azure and assume that customers that have fixed their cloud SKU for at least 40 days have selected the optimal SKU for their workload needs. We also exclude over-provisioned customers as identified in the previous section. The frequency at which Doppler can match the same (fixed) SKU as these customers is taken as one proxy to measure the utility (accuracy) of Doppler.

5.2.1 Customer Profiler. As outlined in Section 3.3, various strategies were tested to summarize the raw time-series performance

Table 5: Elastic strategy performance excluding over-provisioned customers.

Customer Type	Accuracy	Micro Accuracy
DB	89.4%	GP: 89.0% / BC: 95.6%
MI	96.7%	GP: 97.6% / BC: 86.9%

data for current cloud customers. Each strategy involves compressing the time-series vector for each performance dimension into one scalar value, such that each customers' workload could be described by a simple vector that represents the negotiability of each resource dimension. For example, for a customer in which we can access the workload performance traces for the dimensions of CPU, memory, IOPs and latency, under the thresholding strategy, Doppler can reduce this complex time series matrix into one vector. An example of what this output vector might look like is $\langle 0, 0, 1, 1\rangle$, where 1 denotes the performance dimension this specific customer is willing to negotiate on (e.g., IOPs and latency). By attempting various summarization strategies (e.g., MinMax AUC, thresholding, etc.) for the time series performance traces, and then trying various standard ML clustering strategies on the summarized output, we are able to arrive at a means of clustering existing customers into distinct groups.

For SQL DB recommendations, the following perf dimensions are summarized: CPU, memory, IOPs and log rate. Since four dimensions are considered, there are thus $2^4 = 16$ possible customer groups for workloads that suit SQL DB SKUs. For SQL MI recommendations, only CPU, memory and IOPs are summarized; thus, there are only $2^3 = 8$ possible customer groups for these such workloads. We tested more complex clustering strategies, but found that straightforward enumeration is sufficient in separating customers into distinct groups.

The price-performance curves of customers within each group given the thresholding algorithm have similar shapes, and the chosen SKUs land around the same throttling probability values as suggested by the small standard deviation values in Table 3. As expected, SQL MI customers in group 1 are willing to negotiate on any of the three perf dimensions considered. As a result, they tend to fix their SKU choice on the price-performance curves that have a lower score; in other words, they are willing to experience some level of throttling in order to realize cost savings over time. Unlike this particular group, other customer groups (e.g., group 8) are not willing to negotiate at all, hence the choice of SKUs that have a high score and are associated with lower throttling.

Table 4 compares the accuracy of SQL DB and SQL MI SKU recommendations using Doppler across the various summarization strategies discussed in Section 3.3. The thresholding algorithm reaches comparable performance as the others; while the Max Scaler AUC performs the best as it better captures nuances of the performance distribution. Since calculating the AUC is more time-consuming, and the accuracy of our (simple) thresholding approach is not too far off from the optimal, the thresholding approach was selected for implementation in DMA. The final strategy deployed in production utilizes the thresholding algorithm, then employs straightforward enumeration to profile customers into various groups.

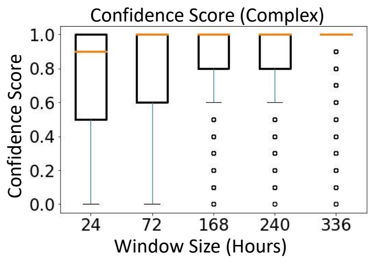


Figure 10: Confidence score distribution for the SKU recommended based on 30-day data.

The majority of cases when Doppler recommends a SKU that does not match the chosen customer SKU involves a flat price-performance curve. In these scenarios, Doppler recommends the cheapest SKU that fulfills the customers' utilization needs at 100%. However, since there is a large proportion (>10%) of customers that are over-provisioned, the most cost-effective option from Doppler does not match the fixed cloud SKU. Figure 8a illustrates one example where the GP 2 cores machine can easily meet the customers' workload resource needs at 100%, but this customer instead allocated themselves with an 80 core machine. By right-sizing, the customer realized over \$100k in annual savings. As efforts are underway to right-size highly over-provisioned customers, we remove this particular subset of customers from our original backtesting data set. Table 5 highlights how the accuracy of Doppler drastically improves when over-provisioned customers are excluded from the ground truth labels.

5.2.2 *Confidence Score.* Since the SKU recommendation generated with Doppler is highly sensitive to the time period in which the performance counters are collected, a confidence score is surfaced along with the optimal SKU choice to provide the customer with some additional guarantees. We use the confidence score to encourage greater data collection, as Figure 10 shows higher confidence associated with SKU recommendations that are made on performance input collected over a longer time span. We examined confidence scores for successfully migrated customers in which we have at least 30 days' worth of performance data. As shown in Figure 10 by the various bootstrap window sizes tested, the confidence score values shift up as the time window of data collection increases past the 1-week interval. Preliminary results suggest that 1-week is the minimum duration needed to capture the variability in perf data required for a reasonable SKU recommendation. We thus encourage new migration customers to run the DMA tool for at least seven days.

5.2.3 *Customers with Changing SKUs.* Given the ease with which customers can change their allocated Azure SQL PaaS, we study successfully migrated SQL DB customers that made one SKU change between June 2020 and March 2021. This results in a subset of 77 customers that either upgrade or downgraded their initial SKU choice. For this set of performance histories, we study the price-performance curves generated by Doppler *before* the change and *after* the change to understand whether the price-performance curve is able to pick up on the changing customer resource utilization needs. As shown in Figure 11, price-performance curves adapt to changes in resource usage and can detect the need to change

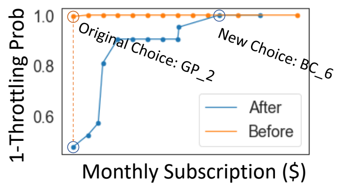


Figure 11: Example of a set of price-performance curves before (dotted line) and after (solid line) a SKU change.

SKUs. For the particular customer workload highlighted here, the customer initially was using SQL DB GP 2 cores, but switched to SQL DB BC 6 cores. Doppler is able to pick up the need for this change as shown by the price-performance curves generated before (dotted line) and after (solid line) the transition. If the customer had stuck to the original SKU choice of GP 2 cores, they would experience significant throttling (>40%). The new SKU of BC 6 cores meets the customers' resource needs at 100%. Since changes in resource utilization patterns trigger changes in the price-performance curves, Doppler can automatically detect the need to change SKUs to accommodate changing workload requirements.

5.3 Data from On-Prem Workloads: Comparison with Baseline Strategy

Since there is no objective ground truth label for the "optimal" SKU for new migration customers that have not yet migrated their workloads to the cloud, we compare the SKU recommendations generated from Doppler against the recommendations generated from the baseline strategy. As the baseline strategy simply reduces each perf dimension into one scalar (max) quantile value, we expect the SKU recommendations from this naive approach to be less optimal. For comparative purposes, the baseline strategy is set to reduce the time series vector to the value that corresponds to the 95% percentile. Since the majority of performance histories were extracted from relatively idle workloads, we focus on three real customers whose perf history would allow for a robust SKU recommendation. For this limited set, we identified 10 instances in which Doppler is able to provide a more appropriate SKU recommendation. More specifically, 80% of the time Doppler recommends a SKU that can actually meet customers' workload latency requirements, while the baseline incorrectly specifies a lower-end SKU. For the rest of the cases, the baseline strategy actually fails to provide any SKU recommendation as it assumes that no SKU can meet the requirement of all resource dimensions at 100%. It is in these particular scenarios that Doppler is especially useful, as it allows customers to negotiate on various perf dimensions to select relevant SKUs. Further investigations are ongoing to compare how the baseline SKU recommendations against that of the Doppler strategy.

5.4 Synthesized Workload

As workload replay is still considered the best practice when it comes to validating whether a new SKU can handle a specific workloads' resource needs, we verify Doppler with this strategy. Given that we want to operate within the confines of certain data privacy restrictions, in that we want to be able to make SKU recommendations without accessing customer data/queries, we leverage a tool that synthesizes new workloads solely based on the customers'

Table 6: SKUs used to execute synthetic workloads

ID	vCPU	Memory	Cache	Throughput	Disk
SKU1	4 cores	16 GB	100 GB	6000 IOPs	2TB SSD
SKU2	8 cores	32 GB	200 GB	12000 IOPs	
SKU3	16 cores	64 GB	400 GB	154000 IOPs	
SKU4	32 cores	128 GB	800 GB	308000 IOPs	

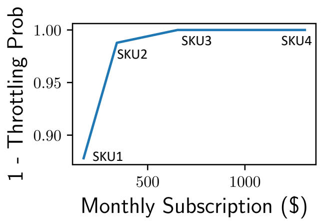


Figure 12: The price-performance curve for the synthesized workload generated on relevant SKUs.

performance history. While still a work in progress, this tool has demonstrated success at reconstructing new workloads that resemble real customer workloads simply by taking as input historical performance traces. The synthesized workload is generated by combining pieces of standardized benchmarks (e.g., TPC-C [46], TPC-DS [47], TPC-H [48], and YCSB [7]) with different database sizes (i.e., scaling factors), query frequency, and concurrency (number of concurrent clients). When executed on the same machine as that of the original workload, the performance traces of these synthesized workloads mimic that of the original. In this section, we validate our SKU recommendation approach by executing synthesized workloads on a selected number of relevant SKUs.

Figure 12 shows one example of a price-performance curve generated from the performance footprint from a synthesized workload. Using Doppler, the optimal SKU is identified as SKU2, details of which are outlined in Table 6. To validate the accuracy of this SKU recommendation, we replay the synthesized workload on the four relevant SKUs ranked in this price-performance curve. Since there is no perf counter that directly measures "throttling", we examine the suitability of the four SKUs via the CPU and latency performance traces. As shown in Figure 13, we can see that the lower-end SKUs results in an increase in IO latency. This is expected as this specific SKU has fewer cores, and with heavy workloads, latency should increase as the resource hits performance bottlenecks (e.g., the customer might experience this as throttling). SKU2, however, is appropriate as latency is within the range that the customer is comfortable with. Additional discussion with our solution engineers verifies that for this synthesized workload, SKU2 is the most cost-efficient option without sacrificing too much on performance. When executing this workload on the cheaper SKU option, SKU1, the workload is severely throttled, which further supports the SKU recommended by Doppler. While our work here is anecdotal, the execution of several synthesized workloads results in the same outcome as illustrated in Figures 12 and 13.

5.5 Discussion

Evaluation of our elastic strategy with data from existing cloud customers and from new migration customers, that have exclusively run their workloads in on-premise data platforms, demonstrates

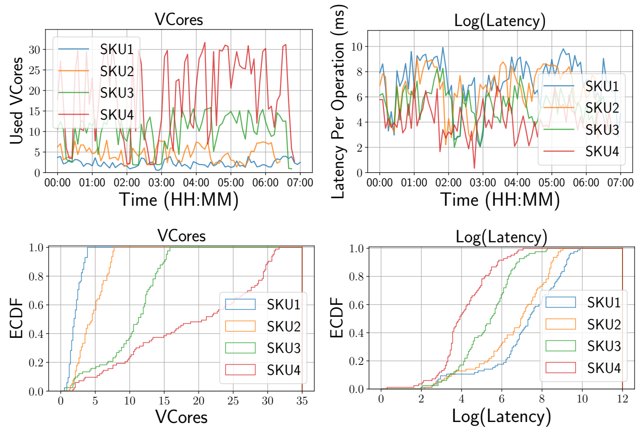


Figure 13: Performance counters when the synthesized workload is executed on four different Azure SQL SKUs.

the ability of Doppler to provide quick, personalized SKU recommendations with high accuracy. More specifically, Doppler is able to generate the exact same SKU choice as successfully migrated customers have chosen (and fixed for at least 40 days) for 89.4% and 96.7% of SQL DB and SQL MI customers, respectively. Since this result excludes customers that are highly over-provisioned, we believe the accuracy we achieve with Doppler is a good proxy for the true accuracy of the best SKU. Future work includes adding a feedback loop in which we can re-train our clustering step based on customer satisfaction with their allocated SKU. From the perspective of our solution architects, few of our customers are under-provisioned.

Our experimentation also revealed that approximately 10% of existing SQL PaaS customers are over-provisioned, initiating a key program in which over-provisioned customers are right-sized to help them realize significant cost savings over time. Doppler has also been successfully deployed and used in the field as its recommendations can be clearly explained by the price-performance curves. This framework can also be easily extended to include additional perf dimensions.

Doppler is also designed to be plugged into any migration assessment infrastructure to enable other migration scenarios. Efforts are underway to integrate Doppler into a broader total cost of ownership (TCO) project, in which customers moving to Azure would be able to systematically compare the differences between keeping their workloads on-prem, moving to a hybrid cloud, or transferring workloads to Google Cloud Platform (GCP) [14], Amazon Web Service (AWS) [2], and/or Azure. Doppler plays an important role in the TCO application in ensuring customers get an accurate picture of what the optimal SKUs and what their costs will be.

6 RELATED WORK

We exclude discussion around the efforts related to workload replay (e.g., [9, 22, 51]) or simulation (e.g., [43]) due to the practical constraints imposed by lack of direct access to user data and cloud billing costs limits.

Several surveys [12, 17, 20, 23, 41, 50] highlight the complexity of this decision-making point and cite tools like the Cloud Target Selection (CTS) [22] and Cloudle [19] that support migrating legacy applications to the cloud. CTS is a large question catalogue that aids new migration customers in choosing the right provider and cloud SKU; Cloudle is similar, but also allows customers to specify additional functional, technical and cost requirements. The REMICS process [42] introduces an array of model-driven solutions to support migration of legacy systems, which is more extensive than survey-like approaches (e.g., CTS) and search engine-type strategies (e.g., Cloudle). It starts with an analysis of the legacy system including its source code, binaries, configuration files, and execution traces, and extends to other knowledge discovery methods that can be translated to cover aspects of the business process, rules, components and implementation. Thus, the migration process is much more hands-on and customized in adapting existing cloud services to that of the customer. However, those tools still require significant user input therefore are difficult to scale and hard to use.

7 CONCLUSION

Given the TAM for migrating on-premise workloads to the cloud is almost $50B and growing continuously, there is a need for tools that can make quick, accurate and personalized SKU recommendations. The current de facto standard of manual investigations in order to map workloads to appropriate cloud targets is too time-consuming and difficult to scale. We introduce Doppler as a means to ease one key bottleneck of migration, automating the SKU decision point. This automation is possible because of our novel end-to-end SKU recommendation framework, based on traditional price-performance methodology that makes recommendations solely on input from customers' performance footprint. This strategy has been back-tested on customer data that have successfully migrated to Azure SQL PaaS, and has subsequently been integrated into production with the release of DMA v5.5 in October 2021.

Since Doppler has demonstrated significant improvement over the baseline algorithm for making Azure SQL DB and MI recommendations, work is currently underway to extend this approach to assess other offerings like Azure SQL serverless [29], hyperscale [28], IaaS (e.g., Azure SQL VM [35]) and other database systems (e.g., Oracle [40]). One concrete example is our engagement with Azure Data Factory (ADF) [25], in which Doppler has been adapted to recommend appropriate compute infrastructure optimized by cost and performance. Given the simplicity of the Doppler framework, we believe it can be applied to address many product challenges in which understanding the trade-offs between cost and performance is needed.

ACKNOWLEDGMENTS

We thank Jack Li, Zeljko Arsic, Aleksandar Ivanovic, Kirby Repko, Venkata Raj Pochiraju, Ajay Jagannathan, Drazen Sumic, Debbi Lyons, Randy Thurman for their workload insights. We thank Yuanyuan Tian, Anja Gruenheid, Rathijit Sen and VLDB reviewers for their feedback.

REFERENCES

[1] Adel Alkhalil, Reza Sahandi, and David John. 2017. A decision process model to support migration to cloud computing. *International Journal of Business Information Systems* 24, 1 (2017), 102–126.

[2] Amazon.com, Inc. 2022. *Amazon Web Service.* Retrieved Jan 4, 2022 from https://aws.amazon.com/

[3] Vasilios Andrikopoulos, Anja Reuter, Mingzhu Xiu, and Frank Leymann. 2014. Design support for cost-efficient application distribution in the cloud. In *2014 IEEE 7th International Conference on Cloud Computing*. IEEE, 697–704.

[4] Vasilios Andrikopoulos, Zhe Song, and Frank Leymann. 2013. Supporting the migration of applications to the cloud through a decision support system. In *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, 565–572.

[5] Patricia V Beserra, Alessandro Camara, Rafael Ximenes, Adriano B Albuquerque, and Nabor C Mendonca. 2012. Cloudstep: A step-by-step decision process to support legacy application migration to the cloud. In *2012 IEEE 6th international workshop on the maintenance and evolution of service-oriented and cloud-based systems (MESOCA)*. IEEE, 7–16.

[6] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition. *J. Off. Stat* 6, 1 (1990), 3–73.

[7] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking cloud serving systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing*. 143–154.

[8] Shaleen Deep, Anja Gruenheid, Kruthi Nagaraj, Hiro Naito, Jeff Naughton, and Stratis Viglas. 2021. Diametrics: benchmarking query engines at scale. *ACM SIGMOD Record* 50, 1 (2021), 24–31.

[9] Leonidas Galanis, Supiti Buranawatanachoke, Romain Colle, Benoît Dageville, Karl Dias, Jonathan Klein, Stratos Papadomanolakis, Leng Leng Tan, Venkateshwaran Venkataramani, Yujun Wang, et al. 2008. Oracle database replay. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1159–1170.

[10] Gartner. 2019. Gartner Says the Future of the Database Market Is the Cloud. https://www.gartner.com/en/newsroom/press-releases/2019-07-01-gartner-says-the-future-of-the-database-market-is-the. Accessed: 2022-02-28.

[11] Gartner. 2020. Market Share Database Management Systems Worldwide 2020. https://www.gartner.com/en/documents/4001330/market-share-database-management-systems-worldwide-2020. Accessed: 2022-02-28.

[12] Mahdi Fahmideh Gholami, Farhad Daneshgar, Graham Low, and Ghassan Beydoun. 2016. Cloud migration process—A survey, evaluation framework, and open challenges. *Journal of Systems and Software* 120 (2016), 31–69.

[13] LS Girish and HS Guruprasad. 2014. Survey on service migration to cloud architecture. *International Journal of Computer Science & Engineering Technology* 5 (2014), 507–510.

[14] Google. [n.d.]. Google Cloud Platform. https://cloud.google.com. Accessed: 2022-01-25.

[15] The PostgreSQL Global Development Group. 2021. PostgreSQL. https://www.postgresql.org/. Accessed: 2021-11-03.

[16] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)* 28, 1 (1979), 100–108.

[17] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. 2013. Cloud migration research: a systematic review. *IEEE transactions on cloud computing* 1, 2 (2013), 142–157.

[18] Stephen C Johnson. 1967. Hierarchical clustering schemes. *Psychometrika* 32, 3 (1967), 241–254.

[19] Jaeyong Kang and Kwang Mong Sim. 2010. Cloudle: a multi-criteria cloud service search engine. In *2010 IEEE Asia-Pacific Services Computing Conference*. IEEE, 339–346.

[20] Ravi Khadka, Amir Saeidi, Andrei Idu, Jurriaan Hage, and Slinger Jansen. 2013. Legacy to SOA evolution: a systematic literature review. *Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments* (2013), 40–70.

[21] Markus Klems, Jens Nimis, and Stefan Tai. 2008. Do clouds compute? a framework for estimating the value of cloud computing. In *Workshop on E-Business*. Springer, 110–123.

[22] Aliki Kopaneli, George Kousiouris, Gorka Echevarria Velez, Athanasia Evangelinou, and Theodora Varvarigou. 2015. A model driven approach for supporting the Cloud target selection process. *Procedia Computer Science* 68 (2015), 89–102.

[23] Stephen Lane and Ita Richardson. 2011. Process models for service-based applications: A systematic literature review. *Information and Software Technology* 53, 5 (2011), 424–439.

[24] Viktor Leis and Maximilian Kuschewski. 2021. Towards cost-optimal query processing in the cloud. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1606–1612.

[25] Microsoft. 2021. Azure Data Factory. https://docs.microsoft.com/en-us/azure/data-factory/introduction.

[26] Microsoft. 2021. Azure Data Migration Assistant. https://docs.microsoft.com/en-us/sql/dma/dma-overview?view=sql-server-ver15.

[27] Microsoft. 2021. Azure SQL Database. https://azure.microsoft.com/en-us/products/azure-sql/database/.

[28] Microsoft. 2021. Azure SQL Database hyperscale. https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tier-hyperscale.

[29] Microsoft. 2021. Azure SQL Database Serverless. https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview.

[30] Microsoft. 2021. Azure SQL DB Resource Limits. https://docs.microsoft.com/en-us/azure/azure-sql/database/resource-limits-vcore-single-databases/.

[31] Microsoft. 2021. Azure SQL Managed Instance. https://azure.microsoft.com/en-us/products/azure-sql/managed-instance/.

[32] Microsoft. 2021. Azure SQL MI Resource Limits. https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/resource-limits/.

[33] Microsoft. 2021. Azure SQL Migration Guides. https://docs.microsoft.com/en-us/azure/azure-sql/migration-guides/database/sql-server-to-sql-database-overview.

[34] Microsoft. 2021. Azure SQL October 2021 New Updates. https://www.youtube.com/watch?v=dKgIqe0x6Bc&t=1542s.

[35] Microsoft. 2021. SQL Server on Azure VM. https://docs.microsoft.com/en-us/azure/azure-sql/virtual-machines/.

[36] Microsoft. 2021. VCore Purchasing Model Overview. https://docs.microsoft.com/en-Us/azure/azure-sql/database/service-tiers-sql-database-vcore. Accessed: 2021-10-25.

[37] Microsoft. 2022. Azure SQL Database pricing. https://azure.microsoft.com/en-us/pricing/details/azure-sql-database/single//.

[38] Thomas Nagler and Claudia Czado. 2016. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis* 151 (2016), 69–89.

[39] State of California Dept of Justice. 2022. *California Consumer Privacy Act*. Retrieved Feb 23, 2022 from https://oag.ca.gov/privacy/ccpa.

[40] Oracle. 2021. Oracle Database. https://www.oracle.com/database/. Accessed: 2021-11-03.

[41] Maryam Razavian and Patricia Lago. 2015. A systematic literature review on SOA migration. *Journal of Software: Evolution and Process* 27, 5 (2015), 337–372.

[42] Andrey Sadovykh, Christian Hein, Brice Morin, Parastoo Mohagheghi, and Arne J Berre. 2011. REMICS-REuse and Migration of legacy applications to Interoperable Cloud Services. In *European Conference on a Service-Based Internet*. Springer, 315–316.

[43] Santiago Gómez Sáez, Vasilios Andrikopoulos, Michael Hahn, Dimka Karastoyanova, Frank Leymann, Marigianna Skouradaki, and Karolina Vukojevic-Haupt. 2015. Performance and Cost Evaluation for the Migration of a Scientific Workflow Infrastructure to the Cloud.. In *CLOSER*. 352–361.

[44] Rathijit Sen, Abhishek Roy, Alekh Jindal, Rui Fang, Jeff Zheng, Xiaolei Liu, and Ruiping Li. 2021. AutoExecutor: predictive parallelism for spark SQL queries. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2855–2858.

[45] Bernard W Silverman. 2018. *Density estimation for statistics and data analysis*. Routledge.

[46] TPC Benchmarks. 2021. *TPC-C*. Retrieved Aug 19, 2021 from http://tpc.org/tpcc.

[47] TPC Benchmarks. 2021. *TPC-DS*. Retrieved Aug 19, 2021 from http://tpc.org/tpcds.

[48] TPC Benchmarks. 2021. *TPC-H*. Retrieved Aug 19, 2021 from http://tpc.org/tpch.

[49] European Union. 2022. *General Data Protection Regulation*. Retrieved Feb 23, 2022 from https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex.

[50] Yi Wei and M Brian Blake. 2010. Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing* 14, 6 (2010), 72–75.

[51] Khaled Yagoub, Peter Belknap, Benoit Dageville, Karl Dias, Shantanu Joshi, and Hailing Yu. 2008. Oracle's SQL Performance Analyzer. *IEEE Data Eng. Bull.* 31, 1 (2008), 51–58.

[52] Yiwen Zhu, Subru Krishnan, Konstantinos Karanasos, Isha Tarte, Conor Power, Abhishek Modi, Manoj Kumar, Deli Zhang, Kartheek Muthyala, Nick Jurgens, et al. 2021. KEA: Tuning an Exabyte-Scale Data Infrastructure. In *Proceedings of the 2021 International Conference on Management of Data*. 2667–2680.