Abstract—Cloud service brokerage is an emerging technology that attempts to simplify the consumption and operation of hybrid clouds. Today’s cloud brokers attempt to insulate consumers from the vagaries of multiple clouds. To achieve the insulation, the modern cloud broker needs to disguise itself as the end-provider to consumers by creating and operating a virtual data center construct that we call a meta-cloud, which is assembled on top of a set of participating supplier clouds.

It is crucial for such a cloud broker to be considered a trusted partner both by cloud consumers and by the underpinning cloud suppliers. A fundamental tenet of brokerage trust is vendor neutrality. On the one hand, cloud consumers will be comfortable if a cloud broker guarantees that they will not be led through a preferred path. And on the other hand, cloud suppliers would be more interested in partnering with a cloud broker who promises a fair apportioning of client provisioning requests. Because consumer and supplier trust on a meta-cloud broker stems from the assumption of being agnostic to supplier clouds, there is a need for a test strategy that verifies the fairness of cloud brokerage.

In this paper, we propose a calculus of fairness that defines the rules to determine the operational behavior of a cloud broker. The calculus uses temporal logic to model the fact that fairness is a trait that has to be ascertained over time; it is not a characteristic that can be judged at a per-request fulfillment level. Using our temporal calculus of fairness as the basis, we propose an algorithm to determine the fairness of a broker probabilistically, based on its observed request apportioning policies. Our model for the fairness of cloud broker behavior also factors in inter-provider variables such as cost divergence and capacity variance. We empirically validate our approach by constructing a meta-cloud from AWS, Azure and IBM, in addition to leveraging a cloud simulator. Our industrial engagements with large enterprises also validate the need for such cloud brokerage with verifiable fairness.

I. INTRODUCTION

Cloud service brokerage [6] is an emerging technology that attempts to simplify the consumption and operation of hybrid clouds. Cloud brokerage is analogous to mass-market retailers or online travel fare companies who are aggregators of services provided by end-suppliers. Cloud brokers gather multi-cloud services into a single catalog and also enable comparison across providers and a single point of purchase.

Modern cloud brokers are also seeking to shield consumers from the heterogeneity of hybrid clouds. A hybrid cloud environment provides a set of on-demand infrastructure units with consumptive billing [5] over which software systems can be deployed: it can, for example, comprise of public clouds, private clouds, as well as high-performing bare metal infrastructure. To achieve ease of hybrid consumption, the modern cloud broker can portray itself as the end-provider to consumers by composing a virtual data center abstraction that we call a meta-cloud over participating provider clouds. Consumers will see the meta-cloud as a single cloud that offers integrated and deterministic service levels and programming interfaces, while internally leveraging the power of offerings from multiple suppliers.

It is essential for such a cloud broker to be considered a trusted partner both by cloud consumers and by underpinning cloud suppliers. While working with large global enterprise customers on designing and deploying complex brokered hybrid cloud solutions, we are increasingly discerning CIO-level apprehension and discomfort on the power and influence that a cloud broker can wield, given its vantage point on top of the entire IT (Information Technology) environment. Thus a central principle of modern brokerage is trust that stems from verifiable vendor neutrality. Because consumer and supplier trust on a meta-cloud broker spring from this assumption of vendor impartiality, there is a need for a test framework to verify the fairness of cloud brokerage, i.e., a technique to assess whether a cloud broker favors specific cloud suppliers either by design or due to inadvertent flaws.

In this paper, we first take a high-level look at how a hybrid cloud broker can compose the meta-cloud abstraction. We then propose a system that can be used by a fairness auditor to test whether a meta-cloud broker who advertises operational fairness is indeed functioning impartially. Our work recognizes that fairness is not a characteristic that will be evident during each provisioning transaction, given that the primary priority of a broker would be to optimize the heterogeneity of a client’s IT environment. Our model also realizes that the semantic of fairness is complex, given inter-cloud-provider variables such as supported infrastructure t-shirt sizes, cost, and capacity. Within these constraints, fairness needs to be a trait that will get manifest over a period across many customer interactions. We develop a calculus built on principles of temporal logic to design a framework to verify the fairness of cloud brokerage. We also back up the validity of our proposed fairness calculus with experimental results.

A. Evolution of Cloud Brokerage

Cloud Brokerage is a technology that has recently emerged in response to expectation changes on IT consumption [1].
which is seeing a fundamental shift today. Retail IT consumers and enterprise business lines now want the flexibility of choice between IT providers; they also expect simplicity and ease of usage along with the flexibility. Cloud service brokers address this by providing a uniform IT experience on top of a diverse IaaS and SaaS (Software as a Service) market place. Papers such as [9] have portrayed cloud brokerage as an IT supply chain construct that efficiently moves services from suppliers to consumers through the cloud network.

Cloud brokerage is an evolving technology. Cloud brokers thus far, have mainly been service aggregators; they gather multi-cloud services into a single catalog and provide a single point of purchase via a single shopping cart in a single consumption portal. The early state of cloud broker technology has thus been analogous to aggregation services offered by Expedia in the travel industry; if an airline delivers poor service, there is no onus or liability on Expedia.

However, cloud brokerage is rapidly maturing to the form that we depict in Figure 1, where the broker constructs and operates a virtual data center that is built on top of a heterogeneous and diverse cloud supplier marketplace. Such a meta-cloud offered by a cloud broker will be retail-friendly and easy to consume since it will hide complexity and deliver an integrated IT experience. A meta-cloud broker is not merely an aggregator, but a provider of cloud services over which it has control. Unlike aggregator brokers such as Expedia, the meta-cloud broker is answerable for any violations of service levels by end-suppliers. This is analogous to the services offered by Uber in the transportation industry.

B. Motivation for Verifying Fairness of Brokerage

In today’s acquisitive market dynamics, we see an increasing number of cloud providers entering the cloud brokerage business, thereby introducing a conflict of interest. Due to this, from the perspective of commercial interest, a broker operating a meta-cloud might stand to profit by leading a customer via a preferred path. A broker could also potentially get into a secret agreement with a specific set of suppliers that give those suppliers a disproportionately high share of business as advantaged constituents of the meta-cloud. In this scenario, cloud suppliers will have a stake in ascertaining a cloud broker’s trustworthiness since they would be more interested in partnering with a broker who promises a fair apportioning of client provisioning requests.

Cloud consumers will also have a stake in going with a cloud broker that guarantees that clients will not be led through a handpicked route. The brokered meta-cloud will offer value-added platform-level services in addition to being a source of infrastructure components, for example, it could provide and profit from a service that chooses a virtual machine or container from a supplier that is the best-available-fit for requested requirements. The credibility of such services will be impacted if the broker is biased to specific suppliers.

Hence, next-generation meta-cloud brokers will need to publish and promise fair behavior. It is important to note that equitable transactional behavior cannot be the unit of measurement of fairness; rather fairness is a trait expected to manifest over a period across many similar customer interactions.

This impending scenario motivated us to design and implement a fairness tester for meta-cloud brokerage.

C. Related Work

To the best of our knowledge, no existing algorithms propose to verify the fairness of cloud brokers, even though there are papers such as [8] that seek to provide a mathematical basis for general trust relationships. [11] assumes that the cloud-brokers under test are ready to co-operate with the fairness verifier in an intrusive fashion by allowing installation of auditing agents on provisioned client VMs. In many cases, this requirement will not be reasonable. Our technique does not make this simplifying assumption. The premise in [11] is that customers do not trust cloud providers since there could be lack of fairness in pricing and SLAs, so they want a guarantee that cloud services are refundable. The paper seeks to reduce this perceived dissatisfaction by proposing that all business procedures between the consumer and cloud providers are handled by cloud brokers; it does not deal with the question of whether the broker itself is fair.

D. Organization of the Paper

The rest of the paper is organized as follows. Section [11] explains the semantic of what constitutes fair cloud brokerage and a model to describe a fair cloud broker. Section [111] offers a logical basis for establishing fairness by proposing a calculus for fair brokerage. It starts by specifying the nature of brokered fairness using temporal logic. The inference rules that are developed in this subsection are then carried further to quantify fairness and unfairness in the form of calculus, by leveraging probability theory. Section [IV] proposes an algorithm to determine a fairness quotient that can be

---

1http://www.expedia.com/
2http://www.uber.com/
implemented by an auditor who offers fairness-testing-of-brokers-as-a-service. It also ponders on the cost complexity of implementing such a service and evaluates trade-offs between cost and accuracy. Section V evaluates our proposed algorithm with experimental results that measure the effectiveness of our design, while Section VI describes a practical real-world validation of our work. Section VII lists threats to validity and associated mitigation. Section VIII highlights future research directions and concludes this paper.

II. MODELING FAIR BROKERAGE

In this section, we model fair composition of a brokered meta-cloud. Let \( P_{x,r} \) be a provisioning request for \( x \) VMs with color \( r \) (where color is a combination of functional requirements such as size and non-functional requirements such as availability) received by the meta-cloud operated by cloud service broker \( b \) as shown in Figure 2. Let \( X_i \) be the number of VMs that the brokered meta-cloud directs to participating cloud provider \( CP_i \). Then

\[
X = \sum_{i=1}^{n} X_i \tag{1}
\]

where \( n \) is the number of participating cloud providers. The problem statement that we seek to address is, for a given broker, is the apportioning of \( X_i \) flows fair?

However, fairness is not in competition with variables such as capacity and cost; rather it is complementary. To obtain such a complementary semantic of fairness, we apply the max-min fairness principle used in network traffic management [10] in the context of brokered provisioning. With this, we introduce efficient fairness and interpret it as follows: Any purchase from participating cloud providers should be equitable in the long term, but consumption from a provider can be more than equitable entitlement if it does not disadvantage other providers. A cloud supplier \( CP_i \) will not be disadvantaged by a distribution of \( X_i \) as long as the apportionment is equitable until \( CP_i \) possesses the ability to fulfill arriving provisioning requests in conformance to the required color, within cost targets. Beyond these limits of fulfillment, a provider will be deemed to have hit a flow bottleneck in the context of request \( P_{x,r} \), and the concept of equitable fairness gets replaced by those mentioned above semantic of efficient fairness.

To explain the concept of efficient fairness, let us define flow bottleneck at an edge \( i \) traversed by a provisioning flow \( X_i \), as

\[
BN(i, t, P_{x,r}, b) = CT(CP_i, r, c) \tag{2}
\]

where

- \( BN \) is the bottleneck at edge \( i \) for the provisioning flow \( P_{x,r} \) that is directed to broker \( b \) at time \( t \),
- \( CT \) flags the capacity of cloud supplier \( CP_i \) to provision a VM having color \( r \) that does not violate the cost target

\[
\sum_{r=1}^{c} \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} X_i \leq \text{available capacity of } CP_i \tag{3}
\]

III. A CALCULUS FOR FAIR BROKERAGE

We leverage the general concept of a calculus of trust proposed in [8], and develop it for our specific case of fair brokerage. We first define a fairness relationship \( CurrFair(a, b, T) \), where actor \( a \) evaluates \( b \) and determines that it fairly performs task \( T \) at the time of evaluation. We also define a complimentary relationship \( CurrUnfair(a, b, T) \), where actor \( a \) audits \( b \) and determines that it unfairly performs task \( T \) at the time of the verification. Finally, \( CurrUnsafe(a, b, T) \) is a relationship that depicts that, after evaluation, \( a \) is unsure about the fairness or unfairness of \( b \) while performing task \( T \).

The outcome of each brokered provisioning request (the identity of the cloud provider to which it is directed to) resembles the roll of an unfair dice where the result of each spin can depend on the consequence of the previous one; hence the bias of the dice may change with each toss. We choose to develop a calculus based on temporal logic to model this complex scenario and predict fairness of behavior.

A. Logical Basis for Broker Fairness

Let us start by defining the terminology necessary to define a calculus for fair brokerage. Let

1) \( f \) be the fairness tester, who is the vendor who uses our proposed algorithm to offer fairness testing of cloud brokers as a service.
is portrayed in Figure 3 and can be represented as:

**Using temporal modalities, this property of eventual fairness**

algorithm is likely to eventually sense that fact over a period. If the broker is indeed operationally fair, the verification that brokered fairness is unlikely to be transactional. However, operations, its state typically moves to Unfair or Unsure, given

**As the verification algorithm starts observing the broker’s unsure broker’s fairness is initially in an arbitrary (or Unsure) state. As the verification algorithm starts observing the broker’s operations, its state typically moves to Unfair or Unsure, given that brokered fairness is unlikely to be transactional. However, if the broker is indeed operationally fair, the verification algorithm is likely to eventually sense that fact over a period. Using temporal modalities, this property of eventual fairness is portrayed in Figure**

\[
\text{Fig. 3. Expected Automata of Eventual Fairness}
\]

2) \(b\) be the cloud broker whose fairness is sought to be estimated in the form of a fairness quotient

3) \(c\) be the client who consumes broker-fairness-as-a-service from fairness tester \(f\). This can be anyone who holds some stake in knowing whether broker \(b\) is fair, for example, a prospective subscriber of meta-cloud services or a potential cloud supplier partner.

4) \(X_b\) be the task of performing brokerage with broker \(b\)

5) \(Y_b\) be the task of performing fairness testing of cloud broker \(b\) as a service

**A client \(c\) can deem broker \(b\) to be fair at time \(t\) if**

\[
\text{CurrFair}(f, b, X_b) \land \text{CurrFair}(c, f, Y_b) = \text{True}
\] \hspace{1cm} (3)

This means that for a client \(c\) to judge that broker \(b\) operates fairly, \(c\) must rate \(f\) as operating fairly; and in turn, \(f\) has to gauge that \(b\) is performing fair brokerage. Even though fairness is a boolean decision (either "true" or "false"), the probability of a broker being fair is a number between 0 and 1. We now see how to obtain a probabilistic fairness quotient, and how that can be converted into a logical fairness decision.

**B. Eventual Fairness**

A broker need not apportion every request equitably since that will fragment the client deployment architecture across clouds. An efficient broker may, quite correctly, try to minimize the degree of hybrid fragmentation for customers to provide a uniform experience and to be cost sensitive. Fairness of brokerage has to thus necessarily be calculated over a period. It is possible that a broker, perceived to be unfair for a particular provisioning request or a sequence of requests, is indeed fair over a large number of independent provisioning requests.

This underlying nature of fairness of brokerage can be specified using Linear Temporal Logic (LTL) [2], a logic that allows operational representation on a linear-time perspective. Let unfair be the opposite of fair and let unsure be the state of uncertainty, where the degree of confidence is insufficient either to determine fairness or unfairness. Figure 3 depicts the intuitive meaning of fairness states. An estimate of the overall fairness can be expressed as:

\[
\Box \Diamond \text{CurrFair}(f, b, X_b)
\] \hspace{1cm} (4)

Where we refer to evaluated overall broker fairness in the rest of the paper, it implies \(\text{Fair}(f, b, X)\) as defined in temporal terms above.

An unfair broker \(Unfair(f, b, X)\), on the other hand, might accidentally encounter a fair state eventually but is unlikely to repeatedly re-enter a fair state after moving out of one. Thus, an unfair broker may sometimes encounter eventual fairness (in temporal terms, \(\neg \Box \Diamond \text{CurrFair}(f, b, X)\)), but not infinitely often.

\[
\neg \text{Fair}(f, b, X) \implies Unfair(f, b, X)
\] \hspace{1cm} (5)

In terms of the atomic predicates we defined earlier:

\[
\neg \Box \Diamond (\neg (\text{CurrUnfair}(f, b, X) \lor \text{CurrUnsure}(f, b, X))) \implies Unfair(f, b, X)
\] \hspace{1cm} (6)

In other words, Equation 6 (and Figure 5) represents an automata for Forever Eventual Fairness where eventual fairness is visited infinitely often.

A broker whose operation resembles the temporal pattern expressed by Equation 6 will be considered a fair broker, \(\text{Fair}(f, b, X)\):

\[
\Box \Diamond \text{CurrFair}(f, b, X) \implies \text{Fair}(f, b, X)
\] \hspace{1cm} (7)

**Eventual Forever Fairness**, depicted in Figure 4 is the scenario where the operational state remains perpetually fair, once a fair state is entered. In temporal logic terms, such overall fairness can be expressed as:

\[
\Box \Diamond \text{CurrFair}(f, b, X)
\] \hspace{1cm} (8)

In terms of the atomic predicates we defined earlier:

\[
\neg \Box \Diamond (\neg (\text{CurrUnfair}(f, b, X) \lor \text{CurrUnsure}(f, b, X))) \implies Unfair(f, b, X)
\] \hspace{1cm} (9)
C. Assessing the Broker’s Fairness

Now that we have leveraged temporal logic to develop a system of rules to represent and symbolize broker fairness, we will probabilistically quantify the above-mentioned fairness semantic in the form of calculus.

Let ϕ, µ and τ be the probability of fair, unfair and unsure brokerage respectively. When the performer is broker b, we denote these symbols as φ_b, µ_b and τ_b. Then:

\[ \phi_b = P(\text{Fair}(f, b, X_b) = 1) \]
\[ \mu_b = P(\text{Fair}(f, b, X_b) = 0) \]
\[ \tau_b = 1 - P(\text{Fair}(f, b, X_b) = 1) - P(\text{Fair}(f, b, X_b) = 0) \]

The odds that f judges b to be fair at a given point of time (one of the fair states in the automata of Figures 5 to 6) is a probability distribution between φ_b and φ_b + τ_b, and the possibility that f deems b to be unfair at any point in time (one of the unfair states in the automata of Figures 5 to 6) is a distribution between µ_b and µ_b + τ_b.

Overall trust on broker fairness then translates to:

\[ \text{Fair}(f, b, X) = 1, \text{ if } \phi_b \geq \text{MinFair} \& \mu_b \leq \text{MaxUnfair} \]
\[ \text{or } (\phi_b + \tau_b) \geq (1 - \text{MaxUnfair}) \]
\[ \text{Fair}(f, b, X) = 0 \text{ otherwise} \]

(10)

where MinFair is the minimum acceptable probability of fair brokerage and MaxUnfair is the maximum acceptable probability of unfair brokerage.

Our algorithm sets an initial user-defined target (of 0.6 for MinFair and 0.2 for MaxUnfair), but fine-tunes it over time based on an embedded feedback loop. This dynamic movement of MinFair and MaxUnfair is described in Section IV-C and Section IV-B. We explain how our proposed algorithm calculates φ_b, µ_b and τ_b in Section IV-B.

D. Fairness Quotient

The fairness quotient at time t (FQ_t), is a metric that estimates the quantum of brokerage fairness, and as alluded to in the previous section, will be a probability distribution over φ_b to (φ_b + τ_b). The Unfairness Quotient at time t (UQ_t), on the other hand, is a measure of estimated unfairness, which is a probability distribution over µ_b to (µ_b + τ_b). For the purpose of this work, we assume that τ_b is uniformly distributed across fairness and unfairness:

\[ FQ_t = \phi_b + (\tau_b/2) \]
\[ UQ_t = \mu_b + (\tau_b/2) = 1 - FQ_t \]

(11)

boolean Simple_Fairness_Algorithm (int X, int r, int * xi_ratio) {
  1. Send a provision request P_r for X VMs having color r to the meta-cloud of cloud broker "b"
  2. Log on to each VM provisioned on the meta-cloud and determine the identity of the cloud provider. Determine the provisioned ratio X_l across participating providers and store in the xi_ratio[] array
  3. If XI is equitable across supplier clouds, broker b is fair (‘strong’ fairness), return (1)
  4. If XI is not equitable across providers, then broker b is unfair

(12)

If we apply Equation (13) to Equation (11), the latter reduces to Equation (11), which becomes the logical crux of designing a solution to estimate fairness of cloud brokerage.

IV. Solution Design

A. Simple Fairness Test Algorithm & Inadequacies

We first describe a simple algorithm to discern broker fairness. The algorithm directs provisioning requests for X VMs (X = 1 to MAX_X) to the brokered meta-cloud and accesses each provisioned VM to decipher the identity of the associated cloud provider. If the provisioning flow is apportioned unequally across underpinning supplier clouds, the algorithm tests if any of the providers have encountered a fulfillment bottleneck for the requested color r. If the apportioning is not equitable even when there are no detected bottlenecks, the broker is deemed to be unfair. The pseudo code is listed in Figure 7.

The following are the main inadequacies of the simple fairness test algorithm:

1. It discounts the possibility of eventual fairness.
2. The broker could be unfair depending on the size of the provisioning request. For example, a broker could exhibit fair behavior if the number of VMs ordered from its meta-cloud in a single provisioning request is less than k, and skew favorably to specific cloud providers if the provisioning request requires more than k VMs. In general, a broker b could be fair while fulfilling certain request sizes and unfair with others. So the simple algorithm has to be enhanced to assess fairness for each possible provisioning size.
3. The simple algorithm is expensive (and slow) to implement and sustain since it captures the broker’s behavior for all provisioning sizes as well as VM characteristics.
Each provisioning request fulfilled on the brokered meta-cloud translates to cost to the provider who offers brokerage fairness verification as a service.

This brings us to a modified and improved algorithm for verifying brokerage fairness, which we describe next.

**B. Modified Approach: Probabilistic Fairness Test Algorithm**

In this section, we develop an improved algorithm that addresses the shortfalls of the simple algorithm using the temporal fairness calculus described in Section III, and returns a predicted fairness quotient of a meta-cloud broker in addition to an opinion on whether it is operating fairly.

Our modified algorithm is probabilistic and is described in Figure 10 and Figure 11. We set an epoch of freshness, pictorially showed in Figure 8, which is the currently running time window before which all data is considered stale. This is to recognize the fact that broker behavior, as well as supplier bottlenecks, can change over time. In each epoch, we randomly generate both the size of a provisioning request and a VM color. The size of a provisioning request falls between $1$ and $MAX_X$, which is the maximum size of a provisioning request that the broker’s meta-cloud can receive.

We first invoke the simple fairness algorithm to sense fairness. If deemed fair, we add the deciphered $X_i$ ratio to a table called `fair_table`, else we add it to a table named `unsure_table`. As shown in Figure 9, these tables are three-dimensional: the $X$-axis marks the cloud providers that constitute the brokered meta-cloud, so a row in the table corresponds to an $X_i$ ratio. The $Y$-axis is the size of provisioning requests received by the meta-cloud. The $Z$-axis tracks the VM color.

As alluded to in Section III-B, it must be deemed acceptable from the perspective of fairness if a broker exhibits equitable apportioning only over a period across customers. In other words, our algorithm needs the ability to detect the property of forever eventual brokered fairness, which we depicted in Figure 5 and expressed as $Fair(f,b,X)$. To enable this, our probabilistic algorithm periodically calculates the cumulative $X_i$ ratio across `fair_table` and `unsure_table`. If the accumulated $X_i$ is approximately efficiently fair, it transfers all existing entries in the `unsure_table` to the `fair_table`. This is equivalent to entering an eventually brokered fair state after navigating a series of unsure states. In effect, on a linear temporal scale, the verification algorithm periodically checks whether the broker has operationally manifested fair behavior. And in course of time, it determines whether the operational automata of the broker resembles Equation 6 (and Figure 5) that echoes brokered fairness, or whether it mirrors an automata such as the one portrayed in Equation 9 (and Figure 6) that translates to unfairness.

**Determining $\phi$, $\mu$ and $\tau$ to Reach Fairness Decisions:**

The number of samples in the `fair_table` (called `fair_units`) corresponds to the broker’s fairness, whereas the number of samples in the `unsure_table` (called `unsure_units`) corresponds to unsure fairness. As samples move from the `unsure_table` to the `fair_table`, the corresponding degree of uncertainty transmutes into degree of fairness.

```
Probabilistic_Fairness_Algorithm()
{
    If (epoch_of_freshness does not contain sufficient samples) {  
        let MAX_X be the maximum size of a VM provisioning request  
        Generate random # 'ran' between 1 and MAX_X and VM color 'v'  
        If (Simple_Fairness_Algorithm (ran, v, xi_ratio)) {
            /* broker is fair for provisioning gulp of size 'ran' and  
            color 'v' and xi_ratio holds the fair xi ratio */
            /* Plot xi_ratio across supplier clouds (CP) for request  
            of size ran in fair_table[][] */
            fair_table[ran] := xi_ratio;
        } else {
            /* Plot xi_ratio across supplier clouds (CP) for  
            request of size ran in unsure_table[][] */
            unsure_table[ran] := xi_ratio;
        }
    }
    Delete stale Samples from fair_table[][] and unsure_table[][]
}
```

Fig. 10. Probabilistic Fairness Algorithm
A more generic way to calculate $\tau_f$ would be to trend the movement ratio across epochs with logarithmically decreasing weights for past values.

Intuitively, $\tau_f$ will be more for fair brokers than for unfair brokers.

To reach a fairness decision, we apply the results from Equation 14 to Equation 11. If the application of Equation 11 determines the broker to be unfair, a further analysis of the fair_table and unsure_table can help determine if the unfairness is limited only to certain provisioning request sizes.

After systematically eliminating the $X_i$ data corresponding to values in the $X$-axis, first in isolation and next in different combinations, if the algorithm senses fairness of operation, those provisioning request sizes that correspond to the unfair bias can be isolated.

To determine the current fairness quotient ($FQ_t$), we apply the results from Equation 14 to Equation 12.

C. Adaptive Fine-tuning of the Model’s Behavior

There are four parameters in our model that are either set by the fairness verification service provider or chosen by the service consumer:

1) Value for $MinFair$, the minimum acceptable probability of fair brokerage
2) Value for $MaxUnfair$, the maximum acceptable probability of unfair brokerage
3) The time length of the epoch of freshness, the currently running time window before which all data is considered stale, and
4) The sample space size, which is the number of provisioning samples that the fairness service is allowed to gather in an epoch

Instead of statically hard-coding these values, it is possible to dynamically re-adjust them during the operation of the fairness service. To achieve this, the fairness verification service provider will need a way to at least occasionally detect generated errors. The errors could be of two types:

1) False positive, when the fairness audit marks an unfair broker as fair, and
2) False negative, when the audit tags a fair broker as unfair.

When a false positive is detected, the verification service could decrease $MaxUnfair$ in small steps, as well as gradually expand the sample space size within operational cost constraints. Conversely, when a false negative is sensed, the verification service could increase $MinFair$, as well as slightly expand the sample size. If no false positives or false negatives are encountered for a duration, the broker could dynamically increase the epoch of freshness, as well as gradually decrease the sample size within an epoch.

D. Operational Cost of the Fairness Verification Service

The fairness tester will incur a cost to operate. This operational cost is proportional to the number of VMs that the fairness algorithm provisions, in other words, the size of the sample space.

The accuracy of our algorithm is a function of the operational budget, i.e., the permissible cost to operate the fairness verification service. If cost is not a factor, there are no limits on the number of samples that the service can gather, so it can maintain granular data. However, in real life, target operational costs are hard constraints, so the number of permissible samples to sustain operations is limited. For our algorithm we set a target for the desired number of samples per epoch as shown in Figure 12. The accuracy of the fairness verification service is a consequence of the targeted cost, which in turn depends on the permissible number of samples in each epoch. This aspect is reflected in the results we present in Section V.C.

V. EXPERIMENTAL RESULTS

We modeled our algorithm for fairness verification by implementing 30 brokered meta-clouds, each composed of five underpinning supplier clouds - Amazon Web Services, Microsoft Azure, IBM Cloud and two simulated clouds. We used the CloudSim toolkit to realize the two simulated clouds. Our brokers behave as follows:

1) For all 30 brokers, we set the total constituent capacity of all 5 supplier clouds to 1000 VMs each, for the provisioning requests of the sole color $r$ (a combination of requirements as defined in Section II) that we supported.
2) We set the duration of an epoch as 1 hour.
3) The maximum size of a provisioning request accepted by all brokered meta-clouds ($MAX_X$) was set as 50 VMs.

3 http://aws.amazon.com/ec2/
4 http://azure.microsoft.com/
5 https://www.ibm.com/cloud/
4) Half the brokers (15 out of the 30) were implemented such that they manifested fair behavior over the duration of an epoch.

5) The unfair brokers (the remaining 15) were realized such that they favored a specific supplier at a rate of between 25% and 45%, chosen randomly. Thus, an unfair broker that has a 25% bias would favor a particular supplier by offering it a fourth of all received provisioning requests over an epoch. The remaining requests would be equitably distributed across all supplier clouds using the semantics of efficient fairness. This bias was distributed across sizes of provisioning requests.

A. Predicting Broker Fairness

We implemented the algorithm described in earlier sections to realize a broker fairness verification service. Depending on the target operational cost to implement verification, the fairness service chooses different number of samples across participating supplier clouds in each epoch. The outcome of our experiment is depicted in Table I. The data-set shows results when the fairness service was requested half-way into the running epoch. It can be seen that while the fairness verification service indeed produced dependable results in most cases, the algorithm did not provide an accurate picture of the broker’s behavior in 5 instances (Broker-5, Broker-7, Broker-10, Broker-12 and Broker-16). In general, errors can be expected when the operational cost target (and hence the sample space) is low and the unfair skew is relatively subtle.

B. Adaptive Improvements

In Section IV-C we described how a feedback loop that controls our model’s key parameters can improve accuracy of the fairness analysis by reducing false positives and false negative conclusions. In this section, we back that up with results from our experiment.

For the purpose of our investigation, we chose our model’s two key parameters - $MinFair$ and $MaxUnfair$ - as the variables that would be dynamically fine-tuned. We kept the the other two parameters - time length of the epoch of freshness and the sample size within an epoch - as constants. Our adaptive improvement added the following feedback bias to the fairness evaluation algorithm when operating on the above set of 30 brokers (whose behaviors are known):

1) When a false positive or false negative is encountered, the algorithm turns more conservative; it narrows down the fairness window by increasing $MinFair$ and decreasing $MaxUnfair$ by one notch.

2) When a correct evaluation is reached, the algorithm turns more liberal; it broadens the fairness window by decreasing $MinFair$ and increasing $MaxUnfair$ by one notch.

Table II is a rerun of Table I but with the above-mentioned feedback logic added to the original fairness verification algorithm. We now hit 3 erroneous predictions (Broker-1, Broker-4 and Broker-16) compared to 5 errors with the original algorithm realized in Table I. Thus, the feedback loop improved our accuracy rate; the false positive rate was cut in half.

We also computed a confusion matrix with and without the adaptive improvements as shown in Figure 13 and Figure 14 respectively. Figure 13 corresponds to the results contained in Table I. Figure 14 corresponds to the data-set in Table II. The accuracy differential between the two confusion matrices establish the benefits that accrue from the refinement that we implemented.
TABLE II
OPERATIONAL FAIRNESS DETECTION USING OUR ADAPTIVE ALGORITHM ON A SET OF 30 EMULATED BROKERS (SERVICE REQUESTED HALF-WAY INTO THE RUNNING EPOCH)

| Broker Behavior | Samples per epoch | # of VMs Provisioned | $F_Q_t$ | Fairness Decision |
|-----------------|-------------------|----------------------|---------|------------------|
| Broker-1 (Fair) | 50                | 1109                 | 0.396   | Unfair           |
| Broker-2 (Fair) | 50                | 1290                 | 0.951   | Fair             |
| Broker-3 (Fair) | 50                | 1323                 | 0.965   | Fair             |
| Broker-4 (Fair) | 50                | 1282                 | 0.509   | Unfair           |
| Broker-5 (Fair) | 50                | 1296                 | 0.774   | Fair             |
| Broker-6 (Fair) | 50                | 1634                 | 0.981   | Fair             |
| Broker-7 (Fair) | 50                | 1263                 | 0.991   | Fair             |
| Broker-8 (Fair) | 50                | 1317                 | 0.981   | Fair             |
| Broker-9 (Fair) | 50                | 1295                 | 0.830   | Fair             |
| Broker-10 (Fair)| 50                | 1519                 | 0.981   | Fair             |
| Broker-11 (Fair)| 50                | 1241                 | 0.981   | Fair             |
| Broker-12 (Fair)| 50                | 1364                 | 0.995   | Fair             |
| Broker-13 (Fair)| 50                | 1111                 | 0.990   | Fair             |
| Broker-14 (Fair)| 50                | 1217                 | 0.981   | Fair             |
| Broker-15 (Fair)| 50                | 1260                 | 0.985   | Fair             |
| Broker-16 (Unfair)| 50       | 1450                 | 0.962   | Fair             |
| Broker-17 (Unfair)| 50         | 1285                 | 0.264   | Unfair           |
| Broker-18 (Unfair)| 50         | 1483                 | 0.254   | Unfair           |
| Broker-19 (Unfair)| 50         | 1403                 | 0.169   | Unfair           |
| Broker-20 (Unfair)| 50         | 1252                 | 0.320   | Unfair           |
| Broker-21 (Unfair)| 50         | 1395                 | 0.208   | Unfair           |
| Broker-22 (Unfair)| 50         | 1336                 | 0.283   | Unfair           |
| Broker-23 (Unfair)| 50         | 1323                 | 0.122   | Fair             |
| Broker-24 (Unfair)| 50         | 1426                 | 0.226   | Fair             |
| Broker-25 (Unfair)| 50         | 1328                 | 0.169   | Unfair           |
| Broker-26 (Unfair)| 50         | 1351                 | 0.132   | Unfair           |
| Broker-27 (Unfair)| 50         | 1397                 | 0.339   | Unfair           |
| Broker-28 (Unfair)| 50         | 1226                 | 0.321   | Unfair           |
| Broker-29 (Unfair)| 50         | 1162                 | 0.328   | Unfair           |
| Broker-30 (Unfair)| 50         | 1267                 | 0.245   | Unfair           |

C. Operational Cost Versus Accuracy

As alluded to earlier, our fairness verification algorithm has a cost complexity that is proportional to the number of samples permitted per epoch. This, in turn, depends on the number of underpinning cloud providers who make up a broker’s meta-cloud construct.

The accuracy of the fairness assessment service should naturally be proportional to the number of permissible samples. To observe how the fairness verification algorithm’s accuracy varies with operational cost, we operated our set of 30 brokers (15 fair and 15 unfair as alluded to earlier) with varying sample sizes. The results are depicted in Figure 15. It can be observed that fairness verification accuracy plateaus at about 95% beyond a particular sample size, but can fall to unusable levels below a certain threshold.

VI. PRACTICAL VALIDATION

As part of our engagement experience in designing multi-vendor hybrid clouds for several enterprise clients, we have started encountering scenarios that highlight the importance of fairness auditing as a service. In a recent interaction with a large global bank, the CTO’s team attached great importance to an overall IT Guardian role to simplify and manage the risk of complexity inherent in their hybrid environment managed by multiple vendors. This digitally mature customer desired to move to a model where they did not own data centers; rather they would ask the IT Guardian to host their in-house business applications with the best-fit cloud provider and offer back an on-demand consumption experience that conformed to agreed Service Level Agreements (SLAs). Generic cloud brokerage would be a key enabler in the IT Guardian operational model; see Table III for the brokerage facets relevant to this role.

While we have not implemented and tested our solution in the context of establishing trustworthiness of an IT Guardian.

We performed the following validation of our fairness verification proposal at the IaaS pane (row #1 of Table III) with the bank’s technical team:

1) We verified both the relevance of the problem we are addressing in this paper and the practical acceptability of the proposed solution in the context of establishing trustworthiness of an IT Guardian.

2) We assessed and concluded that savings in total cost of ownership (TCO) will accrue if our fairness service is implemented. In the absence of a scientific and methodical fairness verification process, the selection of a guardian (that excludes all existing providers) and the learning curve for the chosen IT Guardian to be effective (given unfamiliarity of a new player with the bank’s vast and complex IT environment) could consume months or even years of wasted effort.

VII. Threats to Validity

In this section, we comment on the threats to validity while placing confidence in our results under 3 different axes: construct validity, internal validity, and external validity [4]. Construct validity pertains to whether a metric can measure a construct adequately. Under this head, we reiterate the
element of uncertain fairness that is inherent in our model and a corresponding divergence it can effect while predicting actual behavior. Given the innumerable permutations of deliberate and inadvertent unfairness with which a broker can potentially behave, and the practical limitations on the cost of testing, we cannot do away with the aspect of uncertain fairness; we can merely model and consciously handle it. Our algorithm specifically addresses this issue by treating fairness verification as a calculus, while also tracking the movement of accuracy vis-a-vis the permissible cost of ownership to operate the fairness verification service.

Internal validity highlights the impacts of the experimental process, conditions, history, or relationships among inputs on the observed outcome. While our methodology is free from overt biases, one imperfection is the challenge to choose the optimal values for $\text{MinFair}$, $\text{MaxUndair}$, the duration of the testing epoch, and the size of the sample space. To circumvent this problem, we introduced a feedback mechanism into our system that automatically and dynamically adjusts the optimal values of these variables that maximize accuracy while remaining within budgeted cost.

We have attempted to establish external validity by implementing our fairness verification algorithm, implementing a set of cloud brokers that exhibit different degrees of fairness and unfairness, and subjecting the latter to the former. We also modeled certain constraints and characteristics on top of underpinning supplier clouds. However, we have not taken into account, constructs such as dynamically priced spot instances where capacity and pricing can change even within an epoch.

**VIII. Conclusion and Future Work**

In this paper, we first detailed the functionality of modern cloud service brokerage and the imminent next stage of their evolution. We described how cloud brokers could monetize virtual data centers or meta-clouds constructed out of underpinning provider clouds.

We then suggested that to engage in partnerships with cloud providers and to attract subscribers for value-added cross-cloud services on the meta-cloud, brokers will have to advertise fairness principles that guarantee impartial allocation of provisioning requests across participating provider clouds over a period. It follows that such fairness needs to be verified, so we proposed an algorithm to estimate a broker’s fairness. We provided a logical basis for our method by treating fairness verification of cloud brokerage as a calculus. We showed that the nature of brokered fairness in the real world could be best represented regarding temporal logic, and hence used principles of temporal logic as the underpinning of our calculus for broker fairness. We also recognized that cost complexity of a fairness testing service must be bounded, so we designed a probabilistic algorithm that can receive operational cost as an upper bound.

The larger topic of verifying the fairness of an IT guardian that we alluded to in Section VII will be part of our future work. With such a more generic fairness framework, any provider, even if possessing a stake in offering or managing one or more underpinning functions, can be considered to play the role of an enterprise IT guardian. More generally, the fact that an IT guardian can be verified to be fair eases the risk that enterprise IT is led through a predetermined path that directly or indirectly benefits the provider.

A logical extension of the work that we presented in this paper is to solve problems that render the meta-cloud construct rugged and commercially viable. This will include providing determinism on top of federated capacity planning and service availability. Such a verifiable deterministic meta-cloud can induce a dramatic upswing in hybrid cloud adoption due to its ease of consumption.

**ACKNOWLEDGMENT**

We would like to thank Adarsh Srivastava, at the Birla Institute of Technology, Pilani (Goa Campus), for helping with the implementation of our fairness verification algorithm.

**REFERENCES**

[1] Cloud services brokerage market by service type, platform, deployment model, organization size, vertical, and region - global forecast to 2021. *MarketsandMarkets Report*, 2017.

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[3] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. D. Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41:23–50, 2011.

[4] T. D. Cook, W. Shadish, and D. Campbell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.

[5] M. D. de Assuncao, A. de Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proc. of the 18th International Symposium on High Performance Parallel and Distributed Computing (HPDC)*, pages 141–150. ACM, New York, NY, 2009.

[6] M. Greer. The new cloud on the horizon - cloud brokers. In *IEEE Ninth World Congress on Services*.

[7] A. A. Hossain, Y.-R. Shin, S.-J. Lee, C.-S. Lim, and E. Huh. Cloud broker-based refundable service on multitenant environment. In *International Conference on ICT for Smart Society (ICITSS)*, 2013.

[8] J. Huang and D. Nichol. A formal-semantics-based calculus of trust. *IEEE Internet Computing*, 14:38–46, 2010.

[9] I. Iyub, E. Zarifogluz, and A. B. Dierker. Cloud computing operations research. *Journal of Service Science*, 5:88–101, 2013.

[10] S. Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley, Reading, MA, 1997.

[11] J. Sianipar and C. Meinel. A verification mechanism for cloud brokerage system. In *Second International Conference on Computer Science, Computer Engineering, and Social Media (CSCESM)*, 2015.