A Bayesian System for Cloud Performance Diagnosis and Prediction

Emanuel Palm
Luleå University of Technology,
Luleå, Sweden
E-mail: emapal-4@student.ltu.se

Karan Mitra, Saguna Saguna, and Christer Åhlund
Luleå University of Technology,
Skellefteå, Sweden
E-mail: {$firstname.lastname$}@ltu.se

***Abstract*—The stochastic nature of the cloud systems makes cloud quality of service (QoS) performance diagnosis and prediction a challenging task. A plethora of factors including virtual machine types, data centre regions, CPU types, time-of-the-day, and day-of-the-week contribute to the variability of the cloud QoS. The state-of-the-art methods for cloud performance diagnosis do not capture and model complex and uncertain inter-dependencies between these factors for efficient cloud QoS diagnosis and prediction. This paper presents ALPINE, a proof-of-concept system based on Bayesian Networks. Using a real-life dataset, we demonstrate that ALPINE can be utilised for efficient cloud QoS diagnosis and prediction under stochastic cloud conditions.**

I. Introduction

The use of public cloud providers such as Amazon Web-services and Google Compute Engine for service hosting is becoming increasingly prevalent. The use of such providers offers potentially elastic, secure, scalable and cheap on-demand access to computing, network, and storage resources as-a-service [1]. The general quality and performance of such services are improving, costs are decreasing, and management is becoming simpler due to significant technological advancements in areas such as containerization and orchestration. This trend is partially fueled by the fact that the demand for cloud computing is increasing, and more cloud vendors are entering the world markets. CloudHarmony [2], a major cloud provider comparison website, today lists ninety-five such cloud providers. As the amount of companies providing such services continues to multiply, it becomes increasingly difficult to overview and compare their offerings. Potential buyers may resort to cloud vendor specifications or to third party benchmarks to evaluate their options. Unfortunately, these specifications and benchmarks may not provide a complete picture of the quality of service (QoS) offered by them. Further, they may be incomplete, biased or applicable in limited settings.

Cloud QoS benchmarking, diagnosis, and prediction are highly challenging problems [3], [4], [5], [6]. These challenges exist due to the presence of a significant number of factors that affect cloud QoS. For instance, factors including virtual machine types, data center location (regions), application workload, price/cost, network and storage types may affect each other in a sophisticated way; thereby influencing cloud QoS in a complex manner. Typically, a cloud provider combines several of these factors and build service configurations. These configurations are then offered to the end users as-a-service. For example, Amazon Web Services provides six hundred and seventy-four such combinations differentiated by price, geographical region, and QoS [7]. Each combination of these services may lead to cloud QoS variations thereby necessitating cloud QoS monitoring and diagnosis before and after the selection of a cloud provider to ensure the performance of an application or service hosted on that cloud provider. It is not hard to imagine that the challenges mentioned above are further exacerbated by the presence of the plethora of cloud vendors in the cloud market space.

In this paper, we present ALPINE, a Bayesian system for cloud performance diagnosis and prediction. ALPINE is based on Bayesian Networks (BNs) and includes CloudInfer, a web client (front-end system) to allow the stakeholders (e.g., cloud administrators, system architects, managers, and programmers) to infer cloud performance by varying several factors (virtual machine type, regions, QoS, time-of-the-day, day-of-the-week) simultaneously. The ALPINE system back-end that incorporates BNs then take the factor inputs to infer the cloud performance probabilistically. We will demonstrate how quickly the stakeholders can make sense of complex parametric relationships for efficient cloud performance diagnosis and prediction which is not possible with the state-of-the-art methods presented in [8],[2],[9].

The rest of the paper is organized as follows: Section II presents the related work. Section III describes the ALPINE prototype system. Section IV outlines architectural and design choices made in developing the application. Finally, section V presents a discussion.

II. Related Work

Recently, cloud performance monitoring, benchmarking and prediction problems has got significant interest from both industry and academia [8], [2], [4], [9], [10]. There are several commercial and academic cloud monitoring and benchmarking systems available in the cloud landscape. For example, CloudHarmony [2] provides cloud benchmarking, reporting and selection service based on several parameters such as

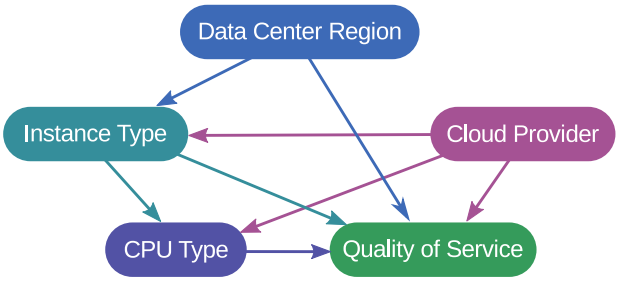


Fig. 1: Nodes of CPU BN, with edges denoting children.

regions, latency and throughput. Amazon Webservices provide a cloud monitoring service called CloudWatch [8], for monitoring virtual machine instances running on Amazon EC2 clouds. CloudWorkbench [11] provides a Web-based system for benchmarking infrastructure-as-a-service (IaaS) clouds. However, these systems only benchmark clouds or simply provide raw aggregated measurements of cloud performance. These systems are not suitable for root-cause diagnosis and prediction of cloud QoS in stochastic cloud environments. ALPINE differentiate itself from these systems by being capable of predicting cloud performance under uncertainty and with missing scarce or sparse data.

III. The ALPINE System

A. Bayesian Network Modelling

ALPINE comprises five BNs developed by us using realistic data (for brevity, we do not indicate how to model and develop BNs in this paper). These BNs capture CPU, I/O and memory related performance for two major cloud providers. We used the comprehensive data set provided by Cito and Leitner [3]. This data set captures the performance related to two major public cloud providers; namely, Amazon Elastic Compute Cloud (AWS) and Google Compute Engine (GCE). Using this dataset, we created several BNs. These BNs were tested using 10-fold cross validation. Once, the prediction results were deemed suitable by the experts; they were incorporated in ALPINE. Figure 1, shows an example BN incorporating several factors (as nodes in a BN) mentioned above. Table 1 lists these factors and their descriptions. The straightforward and intuitive CloudInfer Web client serves these BNs using a web service and hides all the complexity of understanding and modelling BNs. The stakeholders can utilise CloudInfer by varying the factor values (using probability distribution) to diagnose and predict cloud performance.

TABLE I: Description of Bayesian Network Nodes.

Node Name	Description
Cloud Provider	AWS and GCE.
Data Center Region	Location of utilized data center.
Instance Type	micro, small, large, ioopt, cpuopt
CPU Type	CPU model assigned to a virtual machine
Quality of Service Value	Based on five benchmarks

B. CloudInfer Web Service and Client

The CloudInfer Web service exposes a canonical RESTful HTTP API, using the JSON format for serving and interpreting the structured data. The service endpoints are listed in table II.

TABLE II: Endpoints exposed by CloudInfer Web service.

HTTP Path	Method	Description
/networks	GET	Lists known BNs.
/networks/{tag}	GET	Serves BN with {tag}.
/users	GET	List registered users.
/users	POST	Creates new user.
/users/{id}	GET	Serves user with {id}.
/users/{id}/permissions	POST	Grants permission to user.

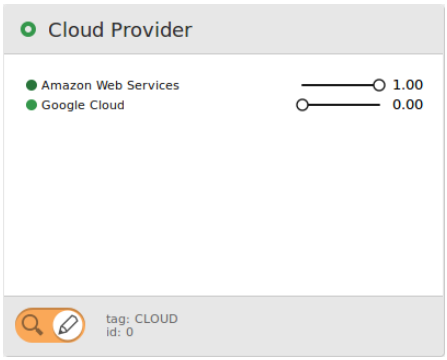
Of the endpoints listed, the most significant are `/networks` and `/networks/{tag}`, which are used to list available BNs and request BN details or results, respectively. For example, issuing a GET HTTP request with the path `/networks/cpu` would lead to the service responding with a structured description of the `cpu` BN (that captures CPU related performance). To query the same BN for performance diagnosis, evidence (for nodes) can be set in a BN using the API; as a result, probability of states belonging to node(s) will be returned as a response. This query is represented as: `[{"node_id":0,"outcomes":[0.90,0.10]}]`. This query shows that evidence for node 0 is set for state 1 as 0.90 and state 2 as 0.10.

The other endpoints are all related to users and authorization. The web service relies on Google+ [12] integration for authorizing requests and creating new users.

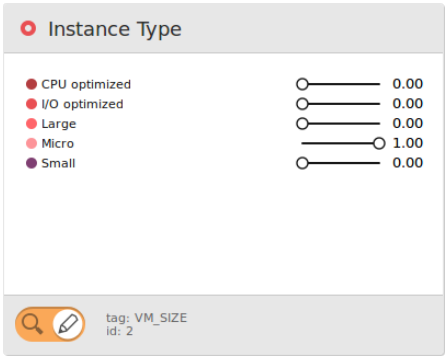
C. CloudInfer Web Client

The CloudInfer Web client allows the stakeholders to interact with BNs served by the CloudInfer service. It presents BN nodes either as reader or writer blocks, as can be seen in figures 2c and 2b. The former kind presents probability distributions in the form of circle charts, while the latter presents sliders that may be used to enter arbitrary probability distributions. After user authentication and selecting a desired BN from a list, the CloudInfer Web client presents a view containing only reader blocks. At the bottom of each such block, there is a slider that may be used to toggle block mode (see bottom left of figures 2c and 2b). By using that toggle to put one or more blocks into writer mode, and then providing relevant probability distributions to those writer blocks, the remaining reader blocks may be updated, causing their values to reflect their relations to the given probability distributions.

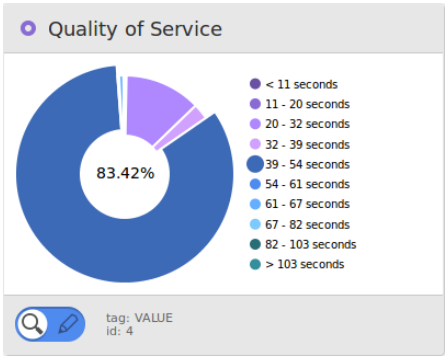
If, for example, configuring the `Cloud Provider` and `Instance Type` blocks (setting the probability of `Cloud Provider` node states and the probability of `Instance Type` node states) of the `Cloud CPU Benchmark` as shown in figures 2a and 2b, setting all remaining blocks in reader mode, and then finally clicking the *Update* button in the user client, the client will present the inferred `Quality`



(a) Cloud Provider writer block.



(b) Instance Type writer block.



(c) Quality of Service reader block.

Fig. 2: Screenshots of CloudInfer Explorer client blocks.

`of Service` probability distribution (e.g., 83.42% chance that the task will be completed between 39 and 54 seconds using "AWS" as a `Cloud Provider` and using "micro" `Instance Type`) as depicted in figure 2c.

IV. ALPINE SYSTEM IMPLEMENTATION

A. CloudInfer Web Service

The CloudInfer Web service, depicted in figure 3, is written in Java 8, and notably uses the SMILE library [13] for managing Bayesian Networks, and the JAX-RS Jersey framework [14] for managing HTTP requests.

The service HTTP layer is responsible for delegating the de-serialization of incoming request data, serialization of response messages, authentication of incoming requests, persistence of

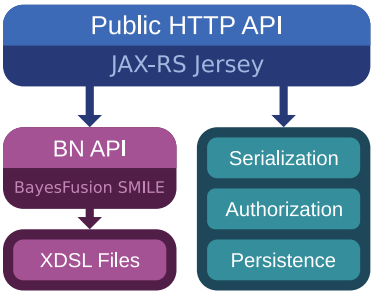


Fig. 3: General design of the CloudInfer Web service.

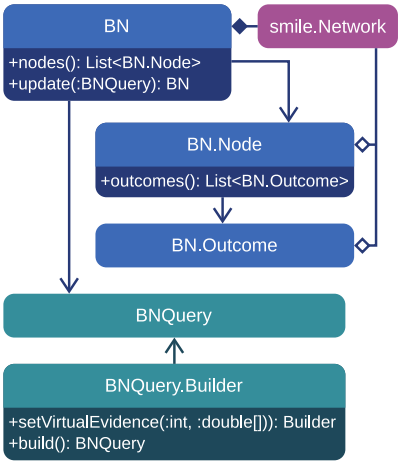


Fig. 4: BN API UML diagram.

user data, and, finally, the presentation of known BNs. The `smile.Network` class provided by the SMILE library [13] is wrapped by another class named `BayesianNetwork`, depicted in figure 4 as `BN`. This class, and its companion classes, implement a `MediaEncodable` interface, and provides their own static generic decode methods, meaning all the classes can be converted to/from JSON, or potentially any other supported encoding. When BN requests are received, these are translated, if relevant, into `BayesianNetworkQuery` objects, depicted as `BNQuery` in figure 4. These are then provided to the `update()` method of a relevant `BayesianNetwork`, which returns a new `BayesianNetwork`, with its outcomes altered in relation to the provided query. Created BN objects are subsequently serialized and returned.

The service relies on Maven [15] to manage building and build profiles, on Docker [16] to produce containerized runtime environments, and on Glassfish [17] to run the application.

B. CloudInfer Web Client

The CloudInfer Web client is built as a single-page web application, with parts written in HTML5, CSS, SVG, and ECMAScript 5. The JQuery [18] library is used to complement the ECMAScript 5 standard library, and D3.js [19] is used to render charts, toggle-buttons, sliders, etc. As the web service requires Google+ authentication, the web client also utilizes the Google+ JavaScript API [20] library.

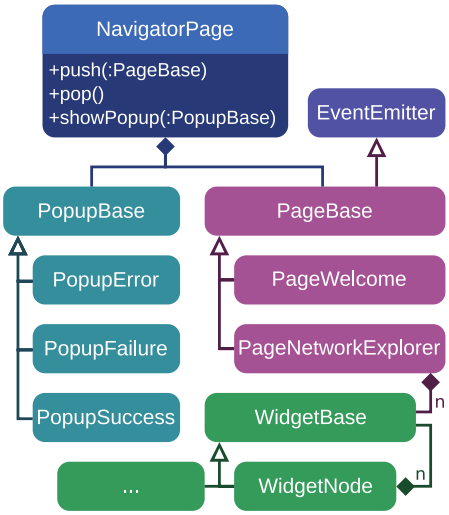


Fig. 5: CloudInfer client UML diagram.

Some significant client components are depicted in figure 5. ECMAScript 5 is a prototype oriented programming language, lacking explicit object oriented programming (OOP) constructs. Even so, an OOP approach was used when designing the application, significantly yielding the would-be classes `NavigatorPage`, `PageBase` and `WidgetBase`. `NavigatorPage` manages transitioning between implementations of `PageBase`, which in turn describes ECMAScript and CSS dependencies, provides HTML5 content and scripts for handling user interaction. `WidgetBase` serves as base class for more complicated widgets built using the D3.js [19] library. A special `include()` function was written in order to manage asynchronous loading of ECMAScript and CSS resources. The application relies significantly on publish/subscribe mechanics to handle user interaction and asynchronous behavior. No architectural pattern, such as MVC or MVVM, is employed as the web client contains no significant logic not strictly associated with data visualization, and is otherwise limited in scope. The CloudInfer Web client is deployed in the same Docker [16] container as the web service, and runs in the same Glassfish [17] environment.

V. Discussion

As more stakeholders increasingly depend on cloud providers for hosting their applications, the demand for efficient management of these services increases. Due to the stochastic nature of clouds, it is important that the cloud performance is continuously monitored. Further, based on collected data, root-cause diagnosis of cloud performance variability is performed. There are already several academic and commercial cloud monitoring systems available in the cloud landscape as mentioned previously. However these systems are limited in many ways. For example, most systems such as [10], [6] only benchmark clouds and collect monitored data. Other systems such as [8],[2],[9] consider simpler metrics to understand cloud performance. These methods they do not handle uncertainty in cloud performance diagnosis and prediction caused due to the relationships between multiple factors such as regions, cloud providers, VM type, day-of-the-week and time-of-the-day. Modelling relationships between these factors is a highly challenging task requiring extensive data collection and the development of sophisticated statistical and mathematical models. In the demo, using one-of-a-kind proof-of-concept implementation, we will show how ALPINE leverages BNs for efficient cloud performance diagnosis and prediction. We will showcase several use cases related to five different benchmarks to diagnose cloud QoS performance related to CPU, memory and disk I/O operations in stochastic cloud conditions.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[2] "CloudHarmony," accessed July 28, 2016. [Online]. Available: https://cloudharmony.com

[3] P. Leitner and J. Cito, "Patterns in the chaos — a study of performance variation and predictability in public iaas clouds," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 3, p. 15, 2016.

[4] J. S. Ward and A. Barker, "Observing the clouds: a survey and taxonomy of cloud monitoring," *Journal of Cloud Computing*, vol. 3, no. 1, p. 1, 2014.

[5] K. Alhamazani, R. Ranjan, F. Rabhi, L. Wang, and K. Mitra, "Cloud monitoring for optimizing the qos of hosted applications," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 765–770.

[6] K. Alhamazani, R. Ranjan, P. Jayaraman, K. Mitra, F. Rabhi, D. Georgakopoulos, and L. Wang, "Cross-layer multi-cloud real-time application qos monitoring and benchmarking as-a-service framework," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[7] M. Zhang, R. Ranjan, M. Menzel, S. Nepal, P. Strazdins, W. Jie, and L. Wang, "An infrastructure service recommendation system for cloud applications with real-time QoS requirement constraints," *IEEE Systems Journal*.

[8] "Amazon CloudWatch," accessed July 29, 2016. [Online]. Available: http://aws.amazon.com/cloudwatch

[9] B. Varghese, O. Akgun, I. Miguel, L. Thai, and A. Barker, "Cloud benchmarking for performance," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 535–540.

[10] K. Alhamazani, R. Ranjan, K. Mitra, F. Rabhi, P. P. Jayaraman, S. U. Khan, A. Guabtni, and V. Bhatnagar, "An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art," *Computing*, vol. 97, no. 4, pp. 357–377, 2015.

[11] J. Scheuner, J. Cito, P. Leitner, and H. Gall, "Cloud workbench: Benchmarking iaas providers based on infrastructure-as-code," in *Proceedings of the 24th International Conference on World Wide Web*. ACM, 2015, pp. 239–242.

[12] "Sign In Users," accessed July 13, 2016. [Online]. Available: https://developers.google.com/+/web/signin

[13] "SMILE Engine," accessed July 5, 2016. [Online]. Available: http://www.bayesfusion.com/#!smile-engine/ko900

[14] "Jersey - RESTful Services in Java," accessed July 5, 2016. [Online]. Available: https://jersey.java.net

[15] "What is Maven?" accessed July 11, 2016. [Online]. Available: https://maven.apache.org/what-is-maven.html

[16] "What is Docker?" accessed July 11, 2016. [Online]. Available: https://www.docker.com/what-docker

[17] "GlassFish," accessed July 13, 2016. [Online]. Available: https://glassfish.java.net

[18] "jQuery," accessed July 14, 2016. [Online]. Available: http://jquery.com

[19] "D3 - Data Driven Documents," accessed July 14, 2016. [Online]. Available: https://d3js.org

[20] "Google+ JavaSscript API," accessed July 14, 2016. [Online]. Available: https://developers.google.com/+/web/api/javascript