Sage: Practical & Scalable ML-Driven Performance Debugging in Microservices
Extended Abstract

Yu Gan[1], Mingyu Liang[1], Sundar Dev[2], David Lo[2], and Christina Delimitrou[1]
[1]Cornell University, [2]Google

1. Motivation

Cloud applications are progressively shifting from *monolithic* services to graphs with hundreds of single-purpose and loosely-coupled *microservices* [2, 7, 8, 22, 48, 49]. Several large cloud providers, such as Amazon, Twitter, Netflix, and eBay have already adopted this application model [2, 7, 8].

Microservices offer several advantages, such as rapid integration and deployment, but they also introduce new challenges with respect to resource management, as dependencies between tiers introduce backpressure effects, causing unpredictable performance to propagate through the system [22, 23]. Diagnosing such performance issues empirically is both cumbersome and prone to errors, especially as typical microservices deployments include hundreds or thousands of unique tiers. Similarly, current cluster managers [17, 18, 19, 27, 29, 30, 31, 32, 35, 36, 38, 42, 45, 53, 56] are not expressive enough to account for the impact of microservice dependencies, thus putting more pressure on the need for automated performance debugging systems.

2. Limitations of the State of the Art

With the increased pervasiveness of the cloud, monitoring and performance debugging systems that track and investigate system and application behavior over time have gained attention. Several tools, such as [9, 14, 20], construct causal paths and diagnose performance issues in distributed systems. There are also several production-level distributed tracing systems, including Dapper [46], Zipkin [6], Jaeger [4], and Google-Wide Profiling (GWP) [39]. Dapper, Zipkin and Jaeger record RPC-level traces for sampled requests across the calling stack, while GWP monitors low-level hardware metrics. These systems aim to facilitate locating performance issues, but are not geared towards taking action to resolve them.

On the performance debugging front, there has been increased attention on trace-based methods to analyze [12, 20, 37], diagnose [9, 10, 13, 16, 24, 25, 26, 34, 40, 51, 54, 55], and in some cases anticipate [21, 23, 50] performance issues in cloud services. Autopilot [41], for example, adjusts the number of tasks and CPU/memory limits automatically to reduce resource slack while guaranteeing performance.

While most of these systems target cloud applications, they almost always focus on single-tier services, and even the ones targeting microservices [23] rely on supervised learning and invasive instrumentation to correctly diagnose the root causes of unpredictable performance. This is problematic in real cloud deployments, as labeling training data with the root causes of QoS violations requires manually diagnosing past performance issues, or injecting new ones whose cause is known. This is impractical, as introducing performance issues hurts the availability and user experience of live applications. Similarly, instrumenting the application and kernel is non-trivial, especially in cases of third-party microservices, whose source code may not be available. Therefore, it is important to explore performance debugging techniques that can uncover the impact of dependencies between microservices without the need for data labeling or expensive instrumentation.

3. Key Insights

We present Sage, a root cause analysis system that leverages unsupervised learning to identify the culprit of unpredictable performance in complex graphs of microservices. Specifically, Sage uses Causal Bayesian Networks to capture the dependencies between microservices, and counterfactuals to examine the impact of microservices on end-to-end performance. Sage does not rely on data labeling, hence it can be entirely transparent to both cloud users and application developers, scales well with the number of microservices and machines, and only relies on lightweight tracing that does not require application changes or kernel instrumentation.

The main design principles in Sage are the following:

- **Unsupervised learning**: Sage focuses on unsupervised learning to circumvent the overhead of labeling training data, which is not practical and/or scalable in real deployments. Instead, it shows that low-frequency live traces collected using infrastructure readily available in cloud providers today, coupled with a set of analytical and ML methods are sufficient to correctly identify the culprits of QoS violations in a complex system.
- **Robustness to sampling frequency**: Sage does not require tracking individual requests to detect temporal latency patterns, making it robust to tracing frequency. This is important, as production tracing systems like Dapper [46] employ aggressive request sampling to reduce overheads [15, 43]. In comparison, previous studies [23, 44, 52] collect traces at the granularity of 10s-100s of milliseconds, which can introduce significant monitoring overheads.
- **User-level metrics**: Sage only uses user-level metrics that can be easily obtained through cloud monitoring APIs and service-level traces from distributed tracing frameworks, such as Jaeger [4] or Zipkin [6]. It does not require any kernel-level information, which is expensive, or even inaccessible in many cloud platforms.
- **Partial retraining**: A major design premise of microservices is enabling frequent updates. Retraining the entire debugging system every time the code or deployment of a mi-

croservice changes is prohibitively expensive. Instead Sage implements partial and incremental retraining, whereby only the microservice that changed and its immediate neighbors are retrained, greatly reducing overheads.
- **Fast resolution**: Empirically examining different sources of unpredictable performance is costly in both time and resources, especially due to the ingest delay cloud systems have in consuming monitoring data, causing a change to experience inertia before propagating on recorded traces. Sage models the impact of the different probable root causes concurrently, enabling faster QoS recovery.

4. Main Artifacts

Sage is an ML-driven performance debugging system for interactive cloud microservices. An overview of the ML pipeline and system architecture of Sage can be found in Figures 1 and 7 of the original paper. The main artifacts we present are:

ML pipeline: Sage contributes an unsupervised ML pipeline consisting of a *causal Bayesian network* (CBN) and a *graphical variational auto-encoder* (GVAE). The CBN is trained on RPC-level distributed traces [6, 46] to capture the dependencies between microservices, as well as causal relationships between individual microservices and the end-to-end performance. The CBN also captures the latency propagation from the backend to the frontend. Second, Sage uses a graphical variational auto-encoder (GVAE), a deep generative model, to generate hypothetical scenarios (counterfactuals [33]), which tweak the performance and/or usage of individual microservices to values known to meet QoS, and infers whether the change restores QoS. Using these two techniques, Sage determines which set of microservices initiated a QoS violation, and adjusts their deployment or resource allocation accordingly.

System design and implementation: We have designed and implemented the end-to-end debugging system, including the tracing infrastructure, ML pipeline, and performance debugging system. The system uses Jaeger [4], a distributed RPC tracing system for end-to-end execution traces, and the Prometheus Node Exporter [5] to collect hardware/OS metrics, container-level performance metrics, and network latencies. Sage uses a centralized master for trace processing, root cause analysis, and actuation, implemented in approximately 6KLOC of Python, and per-node agents for trace collection and container deployment. It also maintains two hot stand-by copies of the master for fault tolerance. The GVAE model is built in PyTorch, with each VAE's encoder, decoder, and prior networks using a DNN with 3-5 fully connected layers.

Validation methodology & large-scale evaluation: We validate Sage's root cause detection accuracy using both synthetic microservice topologies based on *Apache Thrift* [1, 47], a widely-used RPC framework, and an end-to-end application from the DeathStarBench suite implementing a Social Network [22]. We use wrk2 [3], an open-loop HTTP workload generator, to send requests to the front-ends of all applications. We first validate Sage's accuracy in a controlled local cluster, and then we demonstrate Sage's scalability on a large-scale deployment on Google Compute Engine.

5. Key Results

We compare Sage with autoscaling techniques, which are widely used in industry, as well as recent work on performance debugging (CauseInfer [11], Microscope [28], and Seer [23]), targeting both monolithic and microservice applications.

In the dedicated local cluster, we show that Sage achieves 91%-95% root cause detection accuracy, and can quickly take action and restore QoS. It significantly outperforms the autoscaling techniques, by learning the impact of microservice dependencies, instead of memorizing usage thresholds for a particular cluster state. We also show that Sage outperforms prior work on performance debugging, namely CauseInfer and Microscope, which often identify the wrong paths in the dependency graph when searching for root causes. Finally, we show that the unsupervised models in Sage achieve very similar accuracy to the supervised model of Seer, but are significantly more practical and scalable. Specifically, unlike Seer, Sage does not require millisecond-level tracing of queue lengths across the system stack, and it does not need labeling data for training. This makes Sage more portable in datacenter deployments, especially when the application includes libraries or tiers that cannot be instrumented.

We also evaluate Sage's ability to adjust to changes in application design, which are common in microservices. To adapt to such changes, Sage uses transfer learning and partial retraining to localize the model updates to only neurons affected by a change. We show that transfer learning allows Sage to keep its detection accuracy high, and makes retraining $3-30\times$ faster than when retraining the model from scratch.

Finally we evaluate Sage's scalability on 188 container instances on Google Compute Engine. Although we deploy $6.7\times$ more containers compared to the local cluster, the training and inference times only increase by 19.4% and 26.5% respectively. The detection accuracy is not impacted by system scale. Finally, we show that in addition to resource-related performance issues, Sage is also able to detect performance problems caused by software bugs, and isolate the microservice where the bug resides.

6. Why ASPLOS

Sage tackles performance predictability in microservices, a challenging problem that many cloud providers face, especially on an increasingly prevalent application model. Performance debugging and cloud systems are popular topics in ASPLOS and involve hardware, operating systems, and application-level innovations. Sage leverages ML for performance debugging, which also fits in the ML for systems topic.

7. Citation for Most Influential Paper Award

For introducing a data-driven approach in cloud performance debugging, and showing the potential and benefits of using unsupervised learning to address the performance challenges of interactive microservices in a scalable and practical manner.

References

[1] Apache thrift. https://thrift.apache.org.

[2] Decomposing twitter: Adventures in service-oriented architecture. https://www.slideshare.net/InfoQ/decomposing-twitter-adventures-in-serviceoriented-architecture.

[3] giltene/wrk2. https://github.com/giltene/wrk2.

[4] Jaeger: open source, end-to-end distributed tracing. https://www.jaegertracing.io/.

[5] prometheus/node_exporter. https://github.com/prometheus/node_exporter.

[6] Zipkin. http://zipkin.io.

[7] The evolution of microservices. https://www.slideshare.net/adriancockcroft/evolution-of-microservices-craft-conference, 2016.

[8] Microservices workshop: Why, what, and how to get there. http://www.slideshare.net/adriancockcroft/microservices-workshop-craft-conference.

[9] Marcos K. Aguilera, Jeffrey C. Mogul, Janet L. Wiener, Patrick Reynolds, and Athicha Muthitacharoen. Performance debugging for distributed systems of black boxes. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, page 74–89, New York, NY, USA, 2003. Association for Computing Machinery.

[10] Mona Attariyan, MIchael Chow, and Jason Flinn. X-ray: Automating root-cause diagnosis of performance anomalies in production software. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 307–320, Hollywood, CA, 2012. USENIX.

[11] P. Chen, Y. Qi, P. Zheng, and D. Hou. Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1887–1895, 2014.

[12] Xu Chen, Ming Zhang, Morley Mao, and Paramvir Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *Proc. of OSDI*. 2008.

[13] L. Cherkasova, K. Ozonat, Ningfang Mi, J. Symons, and E. Smirni. Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pages 452–461, 2008.

[14] Michael Chow, David Meisner, Jason Flinn, Daniel Peek, and Thomas F. Wenisch. The mystery machine: End-to-end performance analysis of large-scale internet services. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 217–231, Berkeley, CA, USA, 2014. USENIX Association.

[15] Google Cloud. *Cloud Monitoring documentation*, 2020.

[16] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. Correlating instrumentation data to system states:a building block for automated diagnosis and control. In *HP Laboratories Palo Alto, HPL-2004-183, October 19, 2004*.

[17] Christina Delimitrou and Christos Kozyrakis. Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Houston, TX, USA, 2013.

[18] Christina Delimitrou and Christos Kozyrakis. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *Proc. of ASPLOS*. Salt Lake City, 2014.

[19] Christina Delimitrou, Daniel Sanchez, and Christos Kozyrakis. Tarcil: Reconciling Scheduling Speed and Quality in Large Shared Clusters. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SOCC)*, August 2015.

[20] Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica. X-trace: A pervasive network tracing framework. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, pages 20–20, Berkeley, CA, USA, 2007. USENIX Association.

[21] Yu Gan, Meghna Pancholi, Dailun Cheng, Siyuan Hu, Yuan He, and Christina Delimitrou. Seer: Leveraging Big Data to Navigate the Complexity of Cloud Debugging. In *Proceedings of the Tenth USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, July 2018.

[22] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayantara Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud and Edge Systems. In *Proceedings of the Twenty Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2019.

[23] Yu Gan, Yanqi Zhang, Kelvin Hu, Yuan He, Meghna Pancholi, Dailun Cheng, and Christina Delimitrou. Seer: Leveraging Big Data to Navigate the Complexity of Performance Debugging in Cloud Microservices. In *Proceedings of the Twenty Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, April 2019.

[24] M. Grechanik, C. Fu, and Q. Xie. Automatically finding performance problems with feedback-directed learning software testing. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 156–166, 2012.

[25] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. Performance anomaly detection and bottleneck identification. *ACM Comput. Surv.*, 48(1), July 2015.

[26] Guoliang Jin, Linhai Song, Xiaoming Shi, Joel Scherpelz, and Shan Lu. Understanding and detecting real-world performance bugs. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, page 77–88, New York, NY, USA, 2012. Association for Computing Machinery.

[27] Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu. Energy-aware virtual machine dynamic provision and scheduling for cloud computing. In *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD)*. Washington, DC, USA, 2011.

[28] JinJin Lin, Pengfei Chen, and Zibin Zheng. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In *International Conference on Service-Oriented Computing*, pages 3–20. Springer, 2018.

[29] David Lo, Liqun Cheng, Rama Govindaraju, Luiz André Barroso, and Christos Kozyrakis. Towards energy proportionality for large-scale latency-critical workloads. In *Proceedings of the 41st Annual International Symposium on Computer Architecture (ISCA)*. Minneapolis, MN, 2014.

[30] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving resource efficiency at scale. In *Proc. of the 42Nd Annual International Symposium on Computer Architecture (ISCA)*. Portland, OR, 2015.

[31] Jason Mars and Lingjia Tang. Whare-map: heterogeneity in "homogeneous" warehouse-scale computers. In *Proceedings of ISCA*. Tel-Aviv, Israel, 2013.

[32] David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. In *Proceedings of the 38th annual international symposium on Computer architecture*, pages 319–330, 2011.

[33] Michael Moore. Causation in the law. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2019 edition, 2019.

[34] Karthik Nagaraj, Charles Killian, and Jennifer Neville. Structured comparative analysis of systems logs to diagnose performance problems. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 353–366, San Jose, CA, 2012. USENIX.

[35] Ripal Nathuji, Canturk Isci, and Eugene Gorbatov. Exploiting platform heterogeneity for power efficient data centers. In *Proceedings of ICAC*. Jacksonville, FL, 2007.

[36] Ripal Nathuji, Aman Kansal, and Alireza Ghaffarkhah. Q-clouds: Managing performance interference effects for qos-aware clouds. In *Proceedings of EuroSys*. Paris,France, 2010.

[37] Kay Ousterhout, Ryan Rasti, Sylvia Ratnasamy, Scott Shenker, and Byung-Gon Chun. Making sense of performance in data analytics frameworks. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 293–307, Oakland, CA, May 2015. USENIX Association.

[38] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. Sparrow: Distributed, low latency scheduling. In *Proceedings of SOSP*. Farminton, PA, 2013.

[39] Gang Ren, Eric Tune, Tipp Moseley, Yixin Shi, Silvius Rus, and Robert Hundt. Google-wide profiling: A continuous profiling infrastructure for data centers. *IEEE Micro*, pages 65–79, 2010.

[40] Patrick Reynolds, Janet L. Wiener, Jeffrey C. Mogul, Marcos K. Aguilera, and Amin Vahdat. Wap5: Black-box performance debugging for wide-area systems. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, page 347–356, New York, NY, USA, 2006. Association for Computing Machinery.

[41] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, and John Wilkes. Autopilot: Workload autoscaling at google. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery.

[42] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. Omega: flexible, scalable schedulers for large compute clusters. In *Proceedings of EuroSys*. Prague, 2013.

[43] Amazon Web Services. *Amazon CloudWatch User Guide Document History*, 2020.

[44] Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. ??-diagnosis: Unsupervised and real-time diagnosis of small- window long-tail latency in large-scale microservice platforms. In *The World Wide Web Conference*, WWW '19, page 3215–3222, New York, NY, USA, 2019. Association for Computing Machinery.

[45] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of SOCC*. Cascais, Portugal, 2011.

[46] Benjamin H. Sigelman, Luiz André Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, and Chandan Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. Technical report, Google, Inc., 2010.

[47] Mark Slee, Aditya Agarwal, and Marc Kwiatkowski. Thrift: Scalable cross-language services implementation. *Facebook White Paper*, 5(8), 2007.

[48] A. Sriraman and T. F. Wenisch. μ suite: A benchmark suite for microservices. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–12, 2018.

[49] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International Conference on Machine Learning*, pages 3329–3337, 2017.

[50] Yongmin Tan, Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Chitra Venkatramani, and Deepak Rajan. Prepare: Predictive performance anomaly prevention for virtualized cloud systems. In *Proc. of the 32nd IEEE International Conference on Distributed Computing Systems*. 2012.

[51] Jason Teoh, Muhammad Ali Gulzar, Guoqing Harry Xu, and Miryung Kim. Perfdebug: Performance debugging of computation skew in dataflow systems. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '19, page 465–476, New York, NY, USA, 2019. Association for Computing Machinery.

[52] Jörg Thalheim, Antonio Rodrigues, Istemi Ekin Akkus, Pramod Bhatotia, Ruichuan Chen, Bimal Viswanath, Lei Jiao, and Christof Fetzer. Sieve: Actionable insights from monitored metrics in distributed systems. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, Middleware '17, page 14–27, New York, NY, USA, 2017. Association for Computing Machinery.

[53] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.

[54] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical techniques for online anomaly detection in data centers. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 385–392, 2011.

[55] Helen J. Wang, John C. Platt, Yu Chen, Ruyun Zhang, and Yi-Min Wang. Automatic misconfiguration troubleshooting with peerpressure. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6*, OSDI'04, page 17, USA, 2004. USENIX Association.

[56] Hailong Yang, Alex Breslow, Jason Mars, and Lingjia Tang. Bubble-flux: precise online qos management for increased utilization in warehouse scale computers. In *Proceedings of ISCA*. 2013.