Nondisruptive remote debugging platforms can free developers from the challenge and inefficiency of sifting through volumes of log files as well as the tasks associated with reproducing an application-related issue in another environment.

A Data-Driven Remote Debugging Platform Accelerates Bug Resolution and Enhances Understanding of Applications

March 2021

Written by: Arnal Dayaratna, Research Director, Software Development

Introduction

In recent years, the demand for digital solutions has exceeded the ability of enterprise IT to supply them. Expectations on the part of business stakeholders for enterprises to rapidly deliver digital solutions for immediate business problems have required enterprises to confront the challenge of shipping code faster without compromising quality and security. The requirement for enterprises to ship code faster means that enterprises need to empower developers to accelerate developer velocity, which is defined as the speed with which developers can complete development-related projects.

In parallel, cloud-native environments have increasingly become ubiquitous to modern software development workflows. Distributed architecture provides scalability and simplified development with the use of microservices, containers, and the like. However, the very benefit for which distributed architecture was designed also can create an inherent struggle to understand and troubleshoot issues — as code continuously spreads and shifts over multiple repositories.

Remote debugging accelerates developer velocity and provides developers with enhanced operational agility regarding the debugging of production-grade applications. Importantly, remote debugging technologies are useful for both net-new application development and the modernization of legacy applications. In the case of net-new applications, remote debugging helps developers debug cloud-native applications that have a distributed infrastructure spanning hybrid cloud infrastructures and containers. With respect to the modernization of legacy applications, remote debugging provides tools for developers to debug changes to the legacy application monolith and to the microservices that are derived from it. For example, remote debugging tools can be used to debug applications that run on both legacy and cloud environments by empowering developers to remotely access all of the environments on which an application runs.

AT A GLANCE

WHAT'S IMPORTANT

Remote debugging platforms empower developers to handle the complexity of modern applications by using data to understand their code in real time, as it is running.

KEY TAKEAWAYS

Remote debugging platforms make debugging simple and accessible in any environment, from cloud native to on premises and from development to production. In addition, remote debugging enhances the operational agility of developers by enabling them to collect data about applications on demand.

The growing adoption of remote debugging (sometimes known as production debugging) has been driven by the growing adoption of cloud computing, work-from-home development practices, and deepened use of agile development methodologies. Cloud adoption, for example, has proliferated the number of applications that are deployed on a public cloud or a private cloud. IDC forecasts that net-new production-grade cloud-native apps will increase from 10% of all apps in 2020 to 70% of all apps in by 2024, due to adoption of technologies such as microservices, containers, dynamic orchestration, and DevOps. The volume of applications that are developed and deployed on the cloud means that development teams are spending more and more time debugging increasingly complex, dynamic, remotely deployed code.

Typically, remote debugging requires sifting through voluminous amounts of log files or attempting to replicate the issue in question in another environment (preproduction, staging, etc.). The practices of reviewing log files and replicating a bug in another environment are less than ideal. Finding the answer to an application issue in log files can be akin to hunting for a needle in a haystack. Meanwhile, replicating an application issue in another environment can be not only challenging to execute but also misleading because test environments do not necessarily capture the set of conditions responsible for the issue in question. Both approaches are also very time consuming, leading to prolonged mean time to resolution (MTTR) as well as to developers' "wasting" time on nonproductive activities (determining the root cause) rather than fixing the code issue or developing new features.

All this means that the most talented software engineers are often solicited to tackle the challenge of remote debugging because of their ability to navigate the intricacies posed by a surfeit of log data, difficulties associated with replicating an application-related issue, and lack of access to a holistic picture of an application that is not limited to source code. Other challenges associated with remote debugging include ensuring that the source code that developers are viewing via a code editor or debugging platform is synchronized with the application that they are debugging. As more releases are deployed to an application, keeping source code in a debugging tool in sync with the application under review may require skillful collaboration with the team responsible for updating the application. Additionally, developers may struggle to overcome security restrictions related to the inspection of application source code and subsequently focus on reviewing log files or reproducing the bug in question in another environment over which they have more control.

Definitions

» **Understandability.** This refers to the task of understanding the design, architecture, and performance of applications by means of live application data that provides insight into the functioning of an application.

» **Remote debugging.** Remote debugging refers to the practice of debugging applications that run in an environment that is different from the local environment on which a developer works and that allows the developer to debug the code without disrupting the use of the system by its intended end users (e.g., debugging in production).

» **Nonbreaking breakpoints.** These are breakpoints within an application that empower developers to obtain code and live data about an application without stopping the application.

Benefits

There are numerous benefits to remote debugging, including the following:

- **Ad hoc debugging.** Remote debugging provides developers with the ability to debug applications that are in production without restarting, changing, or redeploying code and without disrupting the normal operation of that code. Developers can insert nonbreaking breakpoints within an application that provide them with a snapshot of data about the selected segments of code, without interfering with the functioning of the application. Instead of relying solely on pre-inserted log files in an effort to identify a bug that is responsible for an application-related issue, developers can interactively request and instantly receive real-time messages that reflect the status of their code and provide relevant insights that can be used to remediate the application issue in question.
- **Increased developer velocity.** The ability to obtain real-time data about an application in production while it's running accelerates the ability of developers to debug applications. Developers can use remote debugging to expediently locate bugs within an application. Because remote debugging provides data-driven analytics regarding bugs within an application, developers can more quickly identify the true root cause of problems within application code and more quickly begin the process of fixing the bugs in question. Debugging in production is also the most expensive form of debugging compared with debugging in staging and development environments.
- **Enhanced developer agility.** Developers can use remote debugging to debug applications by collecting live application data. Moreover, remote debugging empowers developers to debug applications without stopping applications, restarting applications, or redeploying code (and without disrupting its normal functionality to end users). This ability to debug applications without stopping or restarting them enhances the operational agility of developers because they can collect data about applications on demand, from any environment, with a few clicks.
- **Relevance for both modern and legacy applications.** In the case of legacy applications, the delivery of live application data on the part of remote debugging platforms helps developers understand legacy applications that they are likely to have not developed themselves. In addition, remote debugging enables developers to understand the impact of modernization work such as refactoring monolithic applications into microservices architectures and the use of APIs and service mesh technology by providing data-driven insight into bugs that arise during the modernization process.

  Remote debugging capabilities are particularly relevant for applications with distributed architectures, where code and data are decentralized, and therefore require a much more nuanced and disciplined methodology for data collection and analysis. The ability of remote debugging to debug distributed applications renders it highly relevant for cloud-native development and applications. The conjunction of the relevance of remote debugging for on-premises, monolithic applications as well as microservices-based and distributed applications renders it a versatile debugging platform that provides value throughout the software development life cycle (development, testing, and production) and can support a variety of applications, infrastructures, and devices.

» **Understandability.** One of the challenges faced by contemporary developers is the task of understanding applications that they may not have developed or used. Remote debugging's enablement of data-driven debugging empowers developers to understand applications that they are debugging as they run. The platform's use of nonbreaking breakpoints also enables developers to obtain insight into the live functioning of the application. Meanwhile, the platform's ability to enable the collection of application data without disrupting the functioning of an application allows developers to nonintrusively understand an application and obtain insight into how the application works in production.

Trends

IDC envisions at least three major trends in software development over the next five years. First, IDC forecasts a proliferation of cloud-native development, which is defined as development that is optimized for cloud infrastructures by means of the use of microservices, containers, container orchestration frameworks, and DevOps. According to IDC's *PaaSView and the Developer 2020*, roughly 30% of developers regularly use microservices and containers for production-grade applications. In 2020, this adoption of containers and microservices represented a 50% increase compared with 2019. IDC forecasts continued growth of cloud-native applications such that "[b]y 2024, net-new production-grade cloud-native apps will expand to 70% from 10% of all apps in 2020, due to adoption of technologies such as microservices, containers, dynamic orchestration, and DevOps," as related by an IDC technology prediction from *IDC FutureScape: Worldwide Future of Digital Innovation 2021 Predictions*. Put differently, the number of cloud-native applications is set to grow sevenfold between 2020 and 2024 as more developers adopt cloud-native technologies and development practices. This shift means that applications are likely to be more distributed than ever as a result of the meteoric adoption of cloud-native development.

Another notable trend that IDC observes is a marked shift in the reuse of code on the part of developers. Specifically, IDC forecasts that the reuse of third-party code in new apps and digital solutions will increase from 40% of code in 2020 to 80% of code in 2024, leading to a corresponding 1.5x increase in the number of apps produced each year. This growth in the reuse of code is attributable to the emphasis on contemporary prioritization of developer velocity and the associated need for developers to ship code faster without compromising quality.

The third notable trend that IDC has observed is the monetization of software solutions, not only by technology companies but also by G2000 firms more generally. "By 2023, the need for G2000 companies to access and monetize multipartner solutions will lead to 5x growth in marketplaces," as noted in *IDC FutureScape: Worldwide Future of Digital Innovation 2021 Predictions*. This monetization of digital solutions is driven by the maturation of a sharing digital economy in which enterprises are increasingly comfortable sharing and reusing digital solutions. According to IDC's *PaaSView and the Developer 2020*, 90% of developers noted that their organization is interested in participating in a hub for sharing digital solutions and becoming a digital hub for the exchange of digital solutions. As such, enterprises are seeking ways to identify, reuse, and purchase code that is relevant to them and subsequently focus their development efforts on the components of their digital solutions that are truly proprietary.

Considering Rookout

Rookout's nondisruptive remote debugging platform circumvents many of the challenges specific to traditional modalities of remote debugging by giving developers a new way to access live data from an application. This ability to request ad hoc live data from any line of code of the application can free developers from the challenge and inefficiency of sifting through volumes of log files as well as the tasks associated with reproducing an application-related issue in another environment. Moreover, by gaining visibility into the runtime code and data flows of an application, developers have the opportunity to understand an application's dependencies on other applications, data stores, and infrastructure, thereby obtaining more nuanced and comprehensive insight into the root causes of a bug or an application-related issue.

Rookout's ability to access live data regarding an application removes much of the trial and error specific to remote debugging and accelerates the time required to resolve bugs and issues substantially. Developers can insert nonbreaking breakpoints into an application to identify the segment of the application they wish to understand and subsequently export relevant data from the application that they can review or import into any external logging or monitoring platform such as New Relic, AppDynamics, and Splunk. This data-driven approach to remote debugging enhances the forensic work specific to identifying the cause of a bug within an application by providing developers with guided insight. As such, Rookout drastically reduces the time required for bug resolution and subsequently helps accelerate developer velocity by enabling organizations to develop and enhance digital solutions faster.

Rookout's acceleration of development enhances not only the developer experience but also the end-user experience, insofar as users stand to enjoy accelerated delivery of innovation as well as the minimization of application-related downtime. In addition, Rookout's enablement of data-driven debugging frees enterprises from dedicating their most talented developers to the task of remote debugging given the complexity of approaches to remote debugging that do not rely upon direct access to source code and live data from applications. The conjunction of Rookout's acceleration of developer velocity and ability to enable enterprises to delegate remote debugging to more junior software developers translates into faster time to market, an increased rate of innovation, and labor-related cost savings for software development.

Rookout's remote debugging solutions provide developers with data-driven insight about bugs within an application that reduces the trial and error associated with debugging.

Another differentiation of Rookout in the remote debugging space is its focus on helping developers understand applications, in addition to identifying and remediating bugs in source code. For example, contemporary developers face the task of debugging applications that they have not developed, worked on, or even used. Reading the lines of code is never enough to fully understand the application. This experience of debugging applications that are foreign to a developer is exemplified by the debugging of legacy applications, newer applications that have been developed by different development teams, and third-party code. Given that the first step in debugging an application involves understanding the application as a whole, Rookout's ability to provide developers with data-driven guidance about code that is relevant to a bug gives developers preliminary insight into the design, architecture, and behavior of the application they are debugging.

Rookout's data-driven platform for remote debugging differs from monitoring and observability solutions that specialize in the aggregation of metrics, logs, and traces geared to understand drivers of application performance. Given its specialization in helping developers instantly understand applications and reduce the time required for bug resolution, Rookout is more appropriately viewed as a vendor that specializes in understandability, namely the task of understanding how applications have been designed and developed and how they function. Rookout is a vendor that specializes in understandability; the company's products should be differentiated from static code analysis tools given that Rookout's delivery of live data to developers provides them with dynamic, heuristic guidance about how to understand applications. Developers can use Rookout's disruptive innovation in the remote debugging space to obtain actionable business intelligence about how to resolve bugs most effectively and understand the applications on which the developers work without disrupting those applications and their end users.

Challenges

The principal challenge faced by Rookout involves training developers regarding best practices for using its platform. As such, Rookout would do well to provide developers with guidance about how to interpret live data extracted from an application using the Rookout platform to obtain insight into the cause of a bug or an application issue. In addition, Rookout will need to train developers about where to insert nonbreaking breakpoints to optimize the operational workflow associated with gathering live data from applications.

Developers and development teams also stand to benefit from an enumeration of best practices for sharing data that has been collected by the Rookout platform. What is the optimal communication structure within a development team for sharing live data collected from the Rookout platform? What kind of developer persona is best qualified to review live application data, and who should be expected to implement the bug fix or resolution? How should a development team automate the workflow that ensues subsequent to the collection of data? These team and organizational questions regarding the workflow for bug resolution constitute other notable challenges for Rookout.

Conclusion

Remote debugging is likely to continue to experience a rise in adoption due to the intensification of enterprise needs for accelerated developer velocity, enhanced developer agility, growing usage of third-party code, increased adoption of the cloud, and the maturation of work-from-home practices. Put simply, remote debugging helps developers resolve bugs and application-related issues faster, and this improvement in developer velocity and agility enables organizations to more easily achieve goals related to the rapid digitization of business processes and application portfolios. That said, remote debugging has traditionally involved shortcomings such as requiring developers to sift through volumes of log files or replicating the issue in question in another environment. The difficulties of replicating application issues in new environments, and the onerous task of poring through large volumes of log files, can render remote debugging time consuming, costly, and complex, thereby requiring the participation of an organization's most talented engineers.

Rookout's remote debugging solutions provide developers with data-driven insight about bugs within an application that reduces the trial and error associated with debugging. Developers can use Rookout to obtain actionable business intelligence about bugs by leveraging live application data that allows them to debug applications in production without disrupting end users or the application itself. Developers who use Rookout are likely to accelerate the time required to identify, understand, and remediate bugs and related causes of application issues. Another advantage of Rookout is that developers can more easily understand applications that they have not written or worked on previously.

As such, Rookout is not only a debugging platform but also a solution designed to facilitate a deeper understanding of applications on the part of developers. If Rookout can address the challenges outlined in this paper, IDC believes that its platform for remote debugging can increase developer velocity and agility by providing data-driven tooling for debugging and understanding applications.

About the Analyst

Arnal Dayaratna, *Research Director, Software Development*

Dr. Arnal Dayaratna is Research Director, Software Development at IDC. Dr. Dayaratna focuses on software developer demographics, modalities of software development, trends in programming languages and other application development tools, and the intersection of these development environments and the many emerging technologies that are enabling and driving digital transformation. Dr. Dayaratna's research examines how the changing nature of software development relates to broader trends in the technology landscape.

IDC Custom Solutions

The content in this paper was adapted from existing IDC research published on www.idc.com.

IDC Research, Inc.
140 Kendrick Street
Building B
Needham, MA 02494
T 508.872.8200
F 508.935.4015
Twitter @IDC
idc-insights-community.com
www.idc.com