

# Holistic Performance Monitoring of Hybrid Clouds: Complexities and Future Directions

**JMaitreya Natu**

Tata Research Development  
and Design Centre, Pune, India

**Ratan K. Ghosh**

Indian Institute of Technology,  
Kanpur, India

**Rudrapatna K. Shyamsundar**

Indian Institute of Technology,  
Mumbai, India

**Rajiv Ranjan**

Data61, CSIRO, Australia

In a cloud environment, computing resources and applications are provided as services over a network, typically the Internet. Cloud computing resources are largely hosted in commercial datacenters and colocation facilities and in hyperscale server farms operated by companies like Amazon, Apple, Google, Microsoft, and Facebook. These datacenters must be continuously monitored to ensure stable operation. Various monitoring tools are available to monitor different layers of an enterprise IT system—from business functions to applications to infrastructure. However, determining what, when, and where to monitor is left to the system administrators. The intuition-driven monitoring strategy is therefore ad hoc, of variable quality, and not scalable to large systems. Furthermore, this approach fails to keep up with the continuous evolution of enterprise systems, especially in hybrid cloud computing environments, which integrate multiple public and private datacenters.

## Monitoring in a Hybrid Cloud

Hybrids are found in other domains, based on assorted factors such as the economics of ownership versus pay-per-use in the presence of variable demand, performance differentials, and privacy and security-related concerns in storing sensitive information in the public cloud.<sup>1</sup> Although cloud providers assure their customers about data safety, some data owners still hesitate to give up some level of control of sensi-

tive information to service providers. Hybrid clouds aim to provide the best of public and private clouds. In addition to security, a private cloud could serve many important purposes. For example, if a private cloud can't provide SLAs, an enterprise can leverage one or more public clouds by employing cloudbursting techniques to manage SLAs.

In cloudbursting, most of the computation is localized in a private cloud. When the computa-



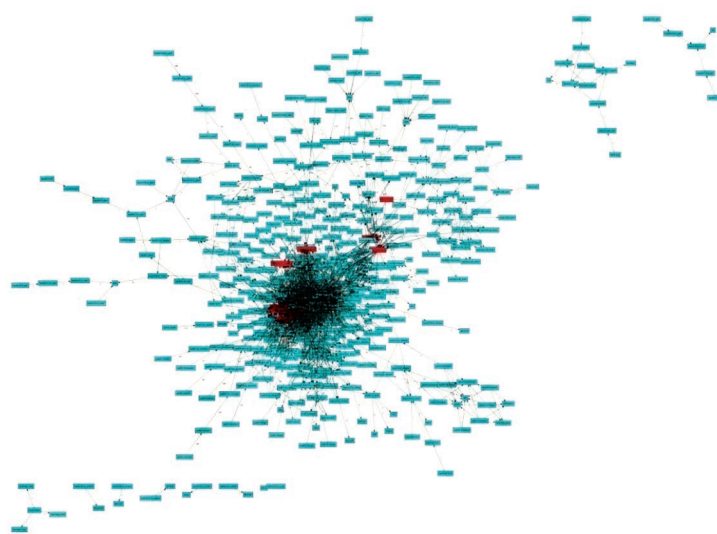
tion load increases, some computation elements are migrated to a public cloud using a suitable scheduling mechanism, where SLAs can be honored. Cloudbursting can also assist in localizing sensitive computation to a private cloud, while compute-intensive work can be relocated to a public cloud as needed. Having a public cloud could also assist in data recovery, checkpointing, and so on if the private cloud fails. By cleverly coordinating the migration of information and computation among several public clouds, along with controlling the flow of sensitive data, it is possible to shield information from unauthorized entities, and thus, effectively create a framework of secure computation on clouds.

Given the benefits of hybrid clouds, many medium-sized enterprises have started to weigh the opportunities of a hybrid cloud against a pure public cloud system. In this context, a datacenter's monitoring mechanism is even more significant. Because faults can be independently domain localized, datacenters, network services, and resources belonging to public clouds and those belonging to private clouds can be monitored independently. Problems that then arise include segregating two possible fault locations, diagnosing unforeseen interactions emerging from the interplay of two complex systems, and seamlessly integrating public and private clouds in scenarios such as business continuity or cloudbursting to relocate computation while maintaining SLAs. These all drive a need for a unified management console and APIs for monitoring faults and system tuning in the event of a fault.<sup>2,3</sup>

Converged infrastructure mapping for hybrid clouds is one of the main problems developers face in building an automated monitoring tool. Such a tool will also depend on the bursting technology the enterprise uses. However, when a mapping solution exists, conventional monitoring tools can be ported to monitor the hybrid clouds' health.

## Challenges

A complex interplay of factors present innumerable challenges to the development of a performance monitoring system for hybrid clouds. We brief some of the most important issues in the following sections.



**FIGURE 1.** A trading application stack that includes 469 software components, 2,072 communication links, and 39,567 unique paths through which incoming stock trades flow.

### Scale and Complexity

Much of the difficulty originates from the inherent scale and complexity of enterprise datacenter applications. Consider, for instance, an equity trading application run by a top-tier investment bank in the US. As Figure 1 illustrates, this single application comprises 469 nodes (software components) for processing incoming stock trades, 2,072 communication links between components, and 39,567 unique paths through which incoming stock trades flow in the system. Some of the application's critical software components (such as a credit card transaction processing app) will be hosted in a private datacenter, whereas noncritical components (such as front-end webservers) will be hosted in public datacenters. Detecting problems (for example, in end-to-end request processing latency) and tracing the problem to one or more culprit software components is difficult in such hybrid deployments.

### Leakage of Monitoring Data

Typically, public cloud service providers store monitoring data in a shared cloud storage service. For example, Amazon CloudWatch and Azure Fabric Controller maintain monitoring logs on their internal

servers. Notably, CloudWatch can be configured to store monitoring logs on Amazon S3 or SimpleDB services. However, these approaches might be unacceptable to enterprise users who don't want to leave footprints of the workload distribution (such as a customer's geolocation) related to specific computations and/or data transfers performed by a public cloud-hosted software component. Thus, a probe-based mechanism initiated from a secure private cloud would be ideal for making an informed guess on the possible problems encountered in a software component deployed on a public cloud instead of relying on the monitoring logs produced by public monitoring services.

#### Monetary Cost of Monitoring

Another way to avoid leakage of monitoring data is to transmit it to a secure location in a private cloud. The major problem here is transferring a large volume of data from a public to a private cloud. Considering the data leakage issue, optimizing monitoring costs becomes the most significant challenge in a hybrid cloud setup. A combination of probing and monitoring would, therefore, be a practical approach for a hybrid cloud environment. Aside from monitoring-related issues, isolating a problem's origin is challenging in a hybrid cloud. Whether the impact on a public cloud is symptomatic of certain issues in a private cloud or it originated from complex interactions among public cloud components should be important in analyzing the root cause. In fact, it's quite possible that a chatty application involving a significant amount of communication between clients and servers might impact performance on a public cloud. So, although the problem seems traceable to a public cloud, it actually originated in components located in a private cloud.

#### Networking of Monitoring Agents

Monitoring information in hybrid clouds can be gathered through deployment of monitoring agents across different cloud service types (such as a virtual machine, webserver, or database server) deployed across private and public datacenters. As continuous collection and reporting of monitoring data can overload a cloud node (such as a virtual machine) hosting the master monitoring agent, there is a need for an intelligent network of distributed monitoring agents

that proactively communicate among themselves to gather monitoring information, and filtering out unnecessary data. To ensure scalability, the monitoring agents also need to implement multiple network topologies for improved agent integration and monitoring information indexing in the form of self-balanced trees and distributed hash tables.<sup>4</sup> Using decentralized networking for agents allows for distributed processing and storage of information, less search time for failed nodes, and improved robustness. However, we need more research aimed at understanding the scalability of these network topologies for different configurations (such as number of software components, monitored performance metrics, and monitoring frequency) in hybrid environments.<sup>4</sup>

#### Instrumentation Versus Intrusiveness

Typically, the effectiveness of monitoring and event management depends on the number and type of data collection monitors (probes) available in the system. Retrofitting an operational system with the instrumentation required to facilitate event management, however, is always a challenge. System operators and administrators are reluctant to introduce probes into the production environment, especially if the probes are intrusive (and can potentially modify the system behavior). Further, probes originating from private datacenters targeting software components hosted in public datacenters could get blocked because of strict intrusion detection policies enforced by cloud providers. Thus, a practical fault localization strategy should enable event detection with minimum instrumentation of the hybrid application system. Much of the prior research in the area of fault localization has ignored this basic practical requirement. This prior work has yielded sophisticated alerting algorithms for fault detection and localization,<sup>5</sup> while assuming that all of the data required by the algorithm for its decision making can be easily gathered and probed. Unfortunately, in most cases, collecting such data requires significant instrumentation across public and private datacenter environments, which is a challenging problem.<sup>6</sup>

#### Incomplete and Inaccurate Knowledge of the System

IT systems in today's enterprises are distributed across geographies. Different segments are monitored



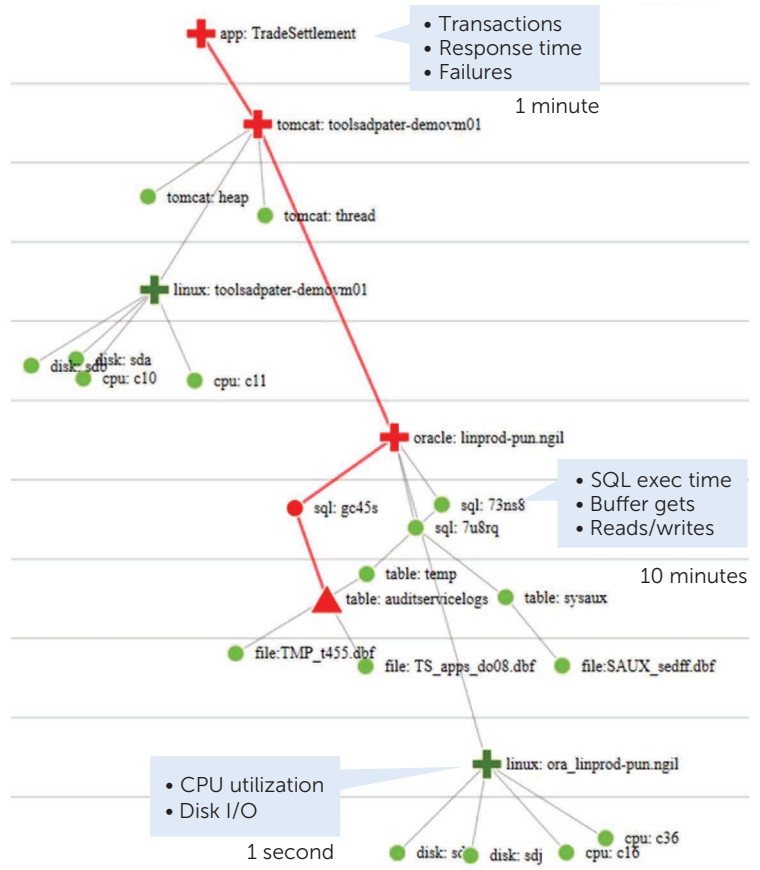
by different teams using different tools, making it difficult to maintain a unified view of the enterprise. Furthermore, IT systems are continuously adding and decommissioning business functions, software components, and infrastructure components. This continuous evolution presents a challenge in keeping the system knowledge up to date in hybrid clouds.

### Need for a Paradigm Shift

Given the increasing scale and complexity of today's IT systems, there's a compelling need for a paradigm shift from a manual intuition-led approach to an automated analytics-driven approach to monitoring these systems. Analytics-led solutions can construct a unified, trustworthy, and up-to-date view of the entire enterprise, accurately model system behavior, and answer the what-when-how of monitoring and event management. When designing a monitoring strategy for hybrid clouds, an enterprise must consider a number of questions, including which components and metrics should be monitored, and how often; what's the best way to ensure minimal monitoring overhead and minimal anomaly detection time; and how can we minimize intrusiveness arising due to active probing

### Insight 1: Need for a Comprehensive System-Wide Solution

Traditional solutions monitor components in isolation. The strategy for monitoring a layer or a component is thus configured while remaining completely agnostic to the policies across other dependent components. This leads to incomplete and inconsistent monitoring solutions. For instance, consider a Web application that serves a critical business function, such as the example in Figure 2. The application's correct functioning depends on the health of the underlying databases, operating systems, and storage and network devices distributed across public and private datacenters. Exhaustive monitoring of an operating system at a fine granularity (say, 1 ms) and minimal monitoring of a database at a very coarse granularity (say, 10 mins) makes the monitoring data unusable for various operations demanding cross-layer analytics. System administrators often face this problem while debugging performance outages. System administrators often can't debug a performance outage at an application because of the



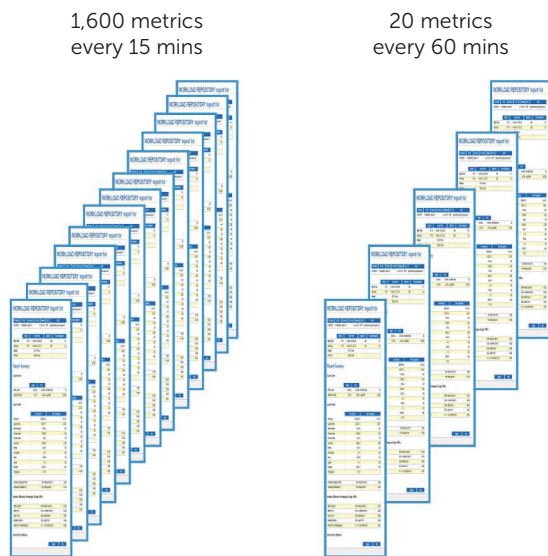
**FIGURE 2.** Different monitoring policies are used across layers of a business application and are agnostic across layers, making monitoring solutions incomplete and inconsistent.

lack of monitoring data at sufficient levels of details. This leads to delayed resolution times and often incomplete and inaccurate resolutions.

To make best use of monitoring information, it's important to cut across the silos of applications, software, and infrastructure components and generate a comprehensive and consistent monitoring plan.

### Insight 2: Need for Solutions that Adapt to System Changes

Active probing and passive monitoring have historically been two important approaches for measurement, monitoring, and managing complex systems.<sup>7</sup> However, these two approaches have traditionally been used in isolation. Passive monitoring techniques



**FIGURE 3.** Both aggressive and conservative monitoring are impractical. Volume and frequency of data are very high in aggressive monitoring, but very low in conservative monitoring.

compute at-a-point (that is compute-component-specific performance metric) metrics and can provide fine-grained metrics, but are agnostic to the end-to-end system performance. On the other hand, probing-based techniques compute end-to-end metrics but lack an in-depth view of components. These two techniques can complement each other to develop effective solutions.

**Adaptive monitoring.** Monitoring metrics need to be collected at various layers—from the hardware layer for metrics such as CPU and memory utilization, to the operating system and virtual machine layer, to the database and application layer. Fortunately, there are many tools for monitoring a wide variety of system components (see, for example, [www-03.ibm.com/software/products/en/ibmtivolinetcoolomnibus](http://www-03.ibm.com/software/products/en/ibmtivolinetcoolomnibus)). Most of these tools can monitor a large range of metrics and can be configured according to needs. However, the quality of the monitored data as well as these tools' ability to monitor across multiple data-centers are often suspect. Effective use of monitoring tools demands an understanding of monitoring requirements, which system administrators often

lack. Instead of a well-defined process for defining monitoring strategies, system administrators adopt an ad hoc, manual, and intuition-based approach. This leads to inconsistent and inadequate data collection and retention policies.

Consider an example of a logical partition (LPAR) of a mainframe box. Many commercial and open source tools monitor the performance of LPARs, capturing a wide variety of metrics. At a given instant, 1,640 metrics can be captured by the standard type 70 log produced for CPU activity at an LPAR. However, storage experts typically capture only a few representative performance metrics, collecting and analyzing a larger set of metrics only when a component becomes a performance bottleneck. Another example is an Oracle instance that can be configured to periodically generate automatic workload repository (AWR) reports (Figure 3). These reports capture the details of workloads, wait events, SQLs, tablespaces, and so on. For lack of a proper monitoring strategy, these reports are usually produced every 24 hours to capture daily statistics.

An aggressive or conservative monitoring approach can lead to either very large or very small volumes of monitoring data. Both types of approaches suffer from drawbacks that make them impractical. A very large amount of monitoring data is difficult to store, maintain, and analyze. For instance, logging 10 metrics at a rate of one sample per second will consume about 720,000 Kbytes per hour, one sample every five seconds will consume about 144,000 Kbytes per hour, and one sample every 10 minutes will consume 1,200 Kbytes per hour. Very large volumes also carry the risk of burying interesting insights. On the other hand, a very small amount of monitoring data carries the risk of losing events of interest.

The key idea of adaptive monitoring is to use probes to understand the end-to-end performance and to infer the criticality of the probed components from the insights gained from these probes.<sup>8</sup> The inferred criticality is then used to provide dynamic monitoring guidelines for these components. A poorly performing component needs monitoring at a finer level such that many metrics are collected at a high sampling rate. However, for a healthy component, a very low sampling rate might be adequate. Thus, end-to-end probing-based solutions can be used to adapt at-a-point monitoring tools.

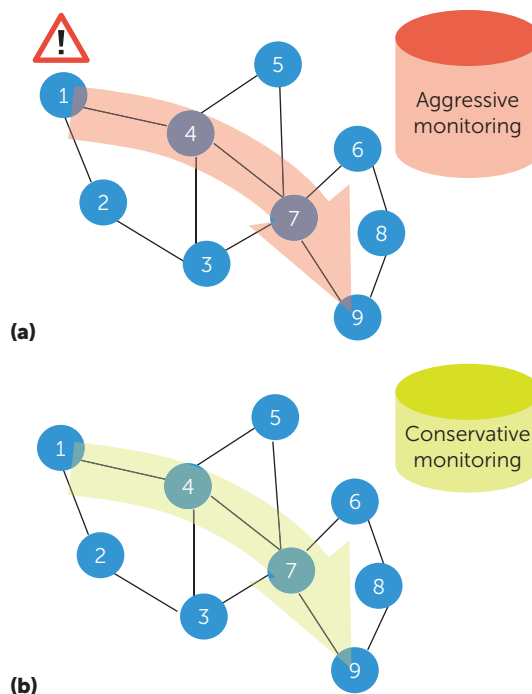


**Adaptive probing.** The ability to measure and analyze end-to-end metrics has encouraged the development of various probing-based solutions for fault localization in networks. However, previous probing-based solutions lack a node-level view.<sup>9,10</sup> All probing decisions are based on the result of probes sent in the past and don't use the information captured by passive monitors. The probes previously selected fail to balance the inherent tradeoff between probe traffic and localization time. Effective solutions can be designed using monitoring agents to adapt the probing policies. The adaptive probing-based solutions built for fault localization can provide accurate fault localization, minimize fault localization time, and minimize the additional probe traffic created in the network.<sup>11</sup>

Previously proposed adaptive-probing solutions use the following two-step approach<sup>11</sup>:

1. A small set of probes are sent periodically to detect failures in the network. The probes detect failure but don't localize the failed nodes. The selection of probes and their frequency should aim to minimize probe traffic and detection time.
2. On detection of a failure, additional probes are sent to localize the exact failure. The selection of probes should be such that fault localization accuracy is high and localization time is minimized.

The information collected by monitors can help improve these detection and localization steps (Figure 5). The metrics collected by the monitoring agent at each node can be used to estimate a potential failure or change in a node's steady state. It can then use the monitoring information to adapt the probing policies for detection and localization of failure to improve probe traffic, localization time, and localization accuracy. Initially, the monitoring data can be used to gain insights into the performance properties of nodes and paths. The probes and probe frequencies are then determined on the basis of these monitoring insights. Selecting probes in this manner helps to keep both probe traffic and detection time low. Probe selection is further refined and optimized on the basis of additional insights gained from the results of past probes and monitoring agents. With this approach, localization



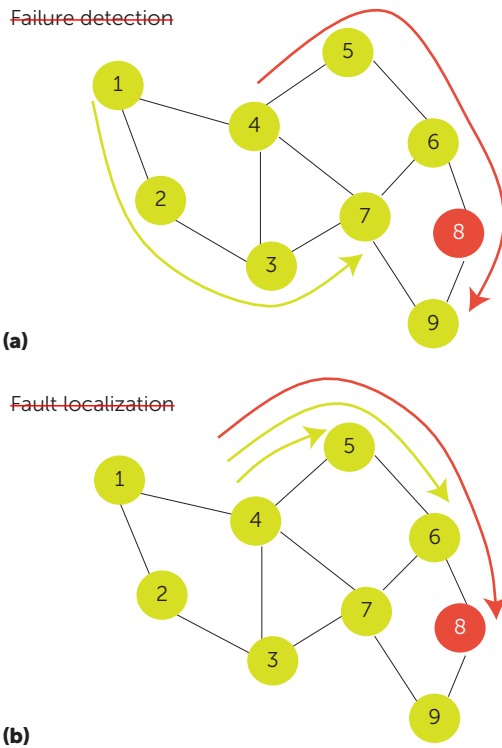
**FIGURE 4.** Adaptive monitoring in an (a) aggressive and (b) conservative monitoring approach. Probes gather information about end-to-end performance, which is used to infer component criticality.

time is kept low and the accuracy of fault localization becomes high.

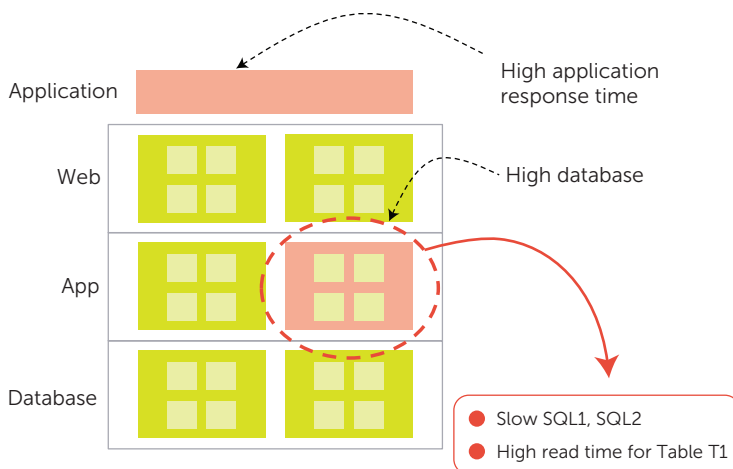
### Insight 3: Need for Efficient Ways to Manage Scale

Given the large scale of enterprise IT systems, the volume of monitoring data generated by the monitoring agents is huge. Very large volumes of data are difficult to store, maintain, and analyze. Adaptive monitoring and probing can fine-tune data collection frequencies, while various approaches can be used to efficiently manage and analyze large volumes of data.

**Invariant detection.** Many metrics are interdependent and, as a result, the value of one can be inferred from the value of another. Invariant detection refers to the approach of identifying the minimal set of metrics that can best represent all metrics of interest.



**FIGURE 5.** Two-step approach in adaptive-probing using monitoring data: (a) failure detection, and (b) failure localization.



**FIGURE 6.** Multiresolution analysis begins at a larger scope of analysis using coarse-grained filters, and iteratively narrows the scope to analyze data at a finer granularity.

Various researchers have worked on detecting invariants.<sup>12</sup> A common approach is based on principle component analysis (PCA).<sup>12</sup> The basic idea is to compute principle components and score them by computing the orthogonal projections of data points onto these principle components. Principle components can be derived using singular value decomposition (SVD).<sup>12</sup> SVD removes redundant components and selects the principle components that are important and representative of all data points.

**Multiresolution analysis.** Enterprise applications use a number of compute, communication, and storage components. Instead of analyzing all the data of all the components at the finest time granularity, multiresolution analysis allows the application to analyze the data in stages, starting with a larger scope of analysis with coarse-grained filters of spatial and temporal dimensions, and iteratively narrowing the scope and using data at a finer granularity (Figure 6).

This approach is applicable in performance debugging.<sup>13</sup> Consider a root-cause analysis of a poorly performing application. A typical application is hosted on several application and database servers. To perform root-cause analysis, various aspects of these components need to be analyzed, such as CPU, memory, disk, heap, threadpool, SQLs, and tables. Analysis at this scope can involve a very large volume of historical data. Multiresolution analysis applies an iterative approach, narrowing the scope and increasing the resolution. The first round of analysis uses computationally lightweight algorithms to collect a few representative metrics at each layer at coarse time granularity. After localizing the problem to one or more layers, an exhaustive drill-down analysis is performed in the localized domain using complex algorithms to collect additional metrics at a finer time granularity.

**Noise reduction.** A single enterprise often uses a wide variety of monitoring tools, each offering different scope and interfaces, and each configured with different policies. A silo view segregates events into layers of applications and infrastructure, limiting insights into how events correlate to service degradation or disruptions. Alert solutions often result in a large volume of false or missed alerts.



With the increased noise, identifying and resolving problems before they impact users is a significant challenge.

To avoid being overwhelmed with noise, contextual information, such as relationships and dependencies between resources, can be used to accurately identify which events are true threats to business functions and which can be safely ignored. Enterprises can use analytics to model normal behavior and system interdependencies to configure legitimate alerts, suppress duplicate alerts, and group correlated alerts (Figure 7).

**Insight 4: Need to Be Proactive over Reactive**

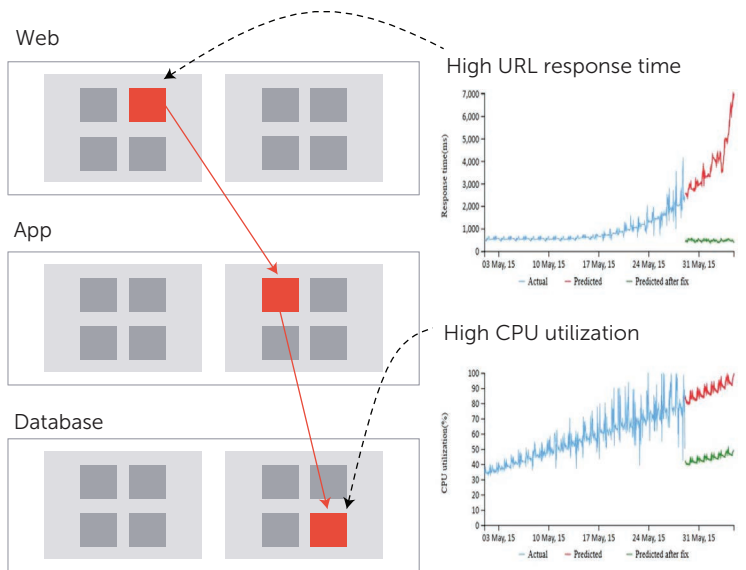
Most alerts are reactive in nature and fail to give sufficient time to take corrective actions—allowing only for quick work-arounds and fixes. Instead of the traditional post facto alert approach, we need proactive alerting solutions. The ability to predict and generate preventive alerts would allow for preventive measures to avoid the problem in the first place.

Various approaches can be applied to generate predictive alerts. For example, an enterprise could use traditional forecasting algorithms such as autoregressive integrated moving average (ARIMA)<sup>14</sup> and Holt-Winters to project independent metrics such as business workload. Another approach, which requires more complex solutions, involves identifying system dependencies and then constructing correlation and regression models to derive intermetric relationships (Figure 8). Analysts can then use various simulation-based solutions to forecast system variables and infer their impact on the overall system.

Performance monitoring of Internet of Things (IoT) applications will require clear understanding and specification of performance metrics across a range of hardware (CPU, storage, and network), software (Apache Hadoop, Apache Storm, Apache Kafka, and so on), and IoT devices (such as sensors). Some IoT applications, such as those in the smart cities and environmental risk management (flooding, earthquake, and tsunamis) domains, need to perform risk modelling using environmental datasets stored across multiple private datacenters. At



**FIGURE 7.** Enterprises can use analytics to identify types of alerts, allowing them to choose which alerts to act on.



**FIGURE 8.** Enterprises can use various forecasting and predictive approaches to achieve proactive alerting solutions, allowing them to choose which alerts to act on.

the same time, they can exploit public clouds to perform analytics over public social media data.

Monitoring in an IoT application ecosystem (a system of systems) is difficult since the performance metrics to be monitored across hardware, software,



and IoT devices might be different but interdependent. For example, key performance metrics include

- event detection and decision-making delay at the application level,
- throughput and latency in distributed messaging queuing systems (Apache Kafka),
- response time in batch processing systems (Apache Hadoop),
- response time for processing top-k queries in transactional systems (Apache Hive),
- read/write latency and throughput for distributed file systems,
- utilization and throughput for CPU resources, and
- throughput for storage and network resources.

Still to be determined are how these performance metrics across hardware, software, and IoT device layers can be defined and formulated as well as how they should be combined to give a holistic view of dataflows. To ensure application-level performance, we must also monitor dataflow metrics (data volume, velocity, variety, and sources; and types and mix of search queries) across layers to develop appropriate workload characterization models. ●●

## References

1. J. Weinman, “Hybrid Cloud Economics,” *IEEE Cloud Computing*, Vol. 3, No. 1, 2016, pp. 74–78.
2. K. Alhamazani et al., “An Overview of the Commercial Cloud Monitoring Tools: Research Dimensions, Design Issues, and State-of-the-Art,” *Springer J. Computing*, vol. 97, no. 4, 2015, pp. 357–377.
3. K. Alhamazani et al., “Cross-Layer Multi-Cloud Real-Time Application QoS Monitoring and Benchmarking As-a-Service Framework,” *IEEE Trans. Cloud Computing*, forthcoming.
4. R. Ranjan, A. Harwood, and R. Buyya, “Peer-to-Peer Based Discovery of Grid Resource Information: A Tutorial,” *IEEE Comm. Surveys and Tutorials (COMST)*, vol. 10, no. 2, 2008, pp. 6–33.
5. I. Cohen et al., “Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control,” *Proc. 6th Conf. Symp. Operating Systems Design and Implementation (OSDI)*, 2004, pp. 231–244.
6. V. Sadaphal et al., “Varanus: More-with-Less Fault Localization in Data Centers,” *Proc. Comm. Systems and Networks (COMSNETS)*, 2012, pp. 1–10.
7. M. Brodie, I. Rish, and S. Ma, “Optimizing Probe Selection for Fault Localization,” *Proc. Distributed Systems Operations Management (DSOM)*, 2001, pp. 200–204.
8. D. Jeswani, M. Natu, and R.K. Ghosh, “Adaptive Monitoring: A Framework to Adapt Passive Monitoring Using Probing,” *Proc. 8th Int’l Conf. Network and Service Management (CNSM 12)*, 2012, pp. 350–356.
9. R. Carter and M. Crovella, “Server Selection Using Dynamic Path Characterization in Wide-Area Networks,” *Proc. IEEE INFOCOM*, 1999, pp. 1014–1021.
10. B. Huffaker et al., “Topology Discovery by Active Probing,” *Proc. Symp. Applications and the Internet*, 2002, pp. 90–96.
11. M. Natu and A.S. Sethi, “Application of Adaptive Probing for Fault Diagnosis in Computer Networks,” *Proc. Network Operations and Management Symp. (NOMS 08)*, 2008, pp. 1055–1060.
12. G. Golub and W. Kahan, “Calculating the Singular Values and Pseudo-Inverse of a Matrix,” *J. Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, vol. 2, no. 2, 1965, pp. 205–224.
13. P. Bahl et al., “Towards Highly Reliable Enterprise Network Services via Inference of Multi-Level Dependencies,” *Proc. SIGCOMM*, 2007, pp. 13–24.
14. R. Calheiros et al., “Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications’ QoS,” *IEEE Trans. Cloud Computing*, vol. 3, no. 4, 2015, pp. 449–458.

---

**MAITREYA NATU** is a scientist at the Tata Research Development and Design Centre of TCS in Pune, India. His research interests include network management, network security, and enterprise management. Natu has a PhD in computer science from the University of Delaware, Newark. Contact him at [maitreya.natu@tcs.com](mailto:maitreya.natu@tcs.com).

---


**RATAN K. Ghosh** is a professor in the Department of Computer Science and Engineering at the Indian



Institute of Technology in Kanpur, India. His research interests include distributed systems, mobile computing, and wireless sensor networks. Ghosh has a PhD in science from Indian Institute of Technology, Kharagpur. Contact him at [rkg@cse.iitk.ac.in](mailto:rkg@cse.iitk.ac.in).

**RUDRAPATNA K. SHYAMSUNDAR** is a JC Bose National Fellow and Distinguished Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Bombay, in Mumbai, India. His research interests include specification and design of distributed real-time systems, logics of programs, parallel programming languages, and cybsersecurity. Shyamsundar has a PhD in computer science from the Indian Institute of Science, Bangalore, India. Contact him at [rkss@cse.iitb.ac.in](mailto:rkss@cse.iitb.ac.in) or [shyamasundar@gmail.com](mailto:shyamasundar@gmail.com).

**RAJIV RANJAN** is an associate professor (reader) in the School of Computing Science at Newcastle University, UK, and a visiting scientist at Data61, Australia. His research interests include cloud computing, content delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne. Contact him at [raj.ranjan@ncl.ac.uk](mailto:raj.ranjan@ncl.ac.uk) or <http://rajivranjan.net>.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

