Evolving from Traditional Systems to AIOps: Design, Implementation and Measurements

Shijun Shen
CNCERT/CC
Peking, China
ssj@cert.org.cn

Jiuling Zhang
CNCERT/CC
Peking, China
zhangjl@cert.org.cn

Daochao Huang
CNCERT/CC
Peking, China
huangdc@cert.org.cn

Jun Xiao
CNCERT/CC
Guangzhou, China
xiaojun@cert.org.cn

***Abstract*—AIOps (Artificial Intelligence for IT Operations) has been proved to be effective in quality improvement, cost reduction and efficiency improvement, and is considered as the ultimate solution for IT operation and maintenance. But for most enterprises, it is still challenging to evolve from traditional systems to AIOps. This paper reviews the development of IT operation and maintenance technologies in the past two decades, and introduces five abilities that a typical AIOps system requires, namely perception, detection, location, action and interaction. Focusing on these abilities, we propose a novel AIOps system called *Proton*. *Proton* adopts the layered design with interoperability services between modules, which makes it well compatible with traditional heterogeneous systems. We have implemented *Proton* with some key considerations including data categories, database cluster, service gateway and operation safety. *Proton* has been deployed in a large IT system environment with tens of thousands of devices, and the measurements reveal that the fault self-healing rate of *Proton* exceeds 80% for the scenario of server ping failure.**

Keywords—AIOps, IT operation, maintenance.

I. Introduction

In 2016, Gartner first introduced the concept of AIOps, defined as Algorithmic IT Operations, which was originally evolved from the earlier concept of ITOA (IT Operations Analytics). Then in 2017, while AI became more and more popular in many fields, Gartner redefined AIOps as Artificial Intelligence for IT Operations according to public opinion [1]. AIOps systems utilize big data, machine learning and other advanced analytic technologies to directly and indirectly enhance IT operations.

The operations landscape today is more complex than ever. IT O&M (Operation and Maintenance) teams have to fight an uphill battle managing the massive amounts of data and billions of alarms generated by modern IT systems. AIOps has been proved to be effective in quality improvement, cost reduction and efficiency improvement. For example, MTTD (Mean Time to Detect) can be reduced from 10 minutes to one minute with the help of AIOps, and MTTR (Mean Time to Repair) can be even reduced from 60 minutes to 30 seconds. The IT O&M department may cut 70% of the staff and greatly reduce the expenses [2]. AIOps is expected to be the ultimate solution in IT O&M area. With the increasing popularity of AIOps in recent years, both Internet multinational giants and small companies are exploring to build their own AIOps systems [3-6].

According to the practice of most enterprises, the development of IT O&M technologies can be divided into five eras along with the rapid progress of IT industry in the past two decades, as described in Fig. 1.

1) The age of manual: All operations were carried out by manually logging into devices without any automatic means. Generally, it does not exist in production systems, but quite common in students' experiments.

2) The age of scripts: O&M functions were achieved by writing automatic scripts, such as shell, Perl, etc. Scripts are still popular up to now because of its convenience.

3) The age of small systems: The disadvantage of scripts is that they are not user-friendly, so small systems based on C/S architecture (and soon replaced by B/S) became popular. Unfortunately, Data sharing and interoperability were poorly supported between these systems.

4) The age of platforms: When interoperability became a big problem, operation platforms with unified framework and standard began to rise. Modules running on these platforms usually shared storage and computing resources, which benefited from cloud computing and big data technology.

5) The age of AIOps: Regarded as an extension of the age of platform, AIOps is a combination of artificial intelligence technology and big data technology. Currently, it is still in the primary stage in terms of technology and application.

Manual
Scripts
Small Systems
Platforms
AIOps

Fig. 1. The development of IT O&M technologies.

The problem is that the boundaries of these eras are not always clear. Actually, most IT O&M teams have to use multiple (or even dozens) of tools, which belong to different eras, to deal with various maintenance requirements. For example, although they have already built a unified platform, they are still using some small operation systems left over from the past, and sometimes they have to write some scripts for emergency. And of course, manual operations cannot be avoided in some cases. Different tools have different user interfaces and different user permissions. How to evolve from these traditional heterogeneous systems to AIOps is of great challenge.

At present, researches on AIOps mainly focus on the algorithm field, such as anomaly detection [7-9], clustering analysis [10], failure prediction [11-13] and cost optimization [14]. In [15], the author shared the experience of applying the AIOps approach to IBM Cloud Object Storage service by data collection, data processing, result presentation and notification, in order to get actionable insights into system's behavior and health. In [16], the author introduced an AIOps system with multiple layers from data collection, data analysis, intelligent decision-making to visual presentation.

Although many enterprises, including Google, Microsoft, AWS and Baidu, have started the construction of AIOps system in previous years [3-6], there are few researches on how to build an AIOps system from scratch.

In this paper, we propose a novel AIOps system, namely *Proton*. *Proton* adopts the layered design with interoperability services between modules, which makes it well compatible with traditional systems. We have implemented *Proton* with some key considerations, and deployed it in a large IT system environment with tens of thousands of devices.

The following of this paper is structured as follows: Section II presents the design of *Proton*, including ability list, layered design and interoperability; Section III gives implementation consideration of the system, including data categories, database cluster, service gateway and operation security; Section IV evaluates our system in terms of fault self-healing rate, storage performance and service response delay; Section V concludes the paper and outlines future research plans.

II. Requirements and Design

In this section, we introduce five abilities that a typical AIOps system needs to have, namely perception, detection, location, action and interaction. According to these abilities, we present the design of *Proton*, including layered design and interoperability.

A. Five Abilities

According to our practice, a typical AIOps system should involve five abilities, as shown in Fig. 2.

1) Perception: The ability to perceive the environment, which emphasizes the capacity to collect information including CMDB (Configuration Management Data Base) and monitoring data in a typical IT O&M system. Perception is the foundation of all other abilities.

2) Detection: The ability to detect abnormalities. Through the analysis of massive perceptual data and historical data, it identifies the abnormal content in either time domain or spatial domain. A large number of statistical methods, machine learning algorithms, deep learning algorithms can be competent for this work.

3) Location: The ability to locate root causes. Once anomaly is detected, we need to find out the most likely cause(s) through correlation analysis, expert experience and statistics. However, this process is usually very challenging for both manual and automated AIOps systems. Most of the cause(s) can be found successfully by statistics or simple algorithms, while the others are hidden deeply under the surface. If there are multiple anomaly causes at the same time, the location will be even tougher.

4) Action: The ability to operate the objects in the IT system, that is, the ability to affect the environment. After the previous process of perception, detection and location, the ultimate goal is to solve the problem. Potentially actions include simply restarting a process or a device, sending a message or making a call to the operator, or switching the entire data center to the backup node. In practice, the capability set of actions reflects the level of automation. Meanwhile, actions may involve great risks, which need to be carefully audited in all aspects.

5) Interaction: The ability of rich human-computer interaction. Curiously, many reports on AIOps rarely refer to interactivity, which improves O&M efficiency in many ways. For example, front-line operators have to repeatedly carry out identity authentication, system switching and manual typing, while human-computer interaction technologies such as face recognition, gesture recognition and natural language processing, can make these jobs simple and comfortable. As efficiency improvement is a main goal of AIOps, human-computer interaction should be paid more attention.

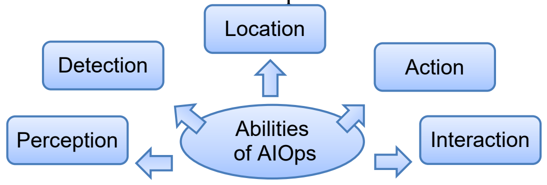

Fig. 2. Five abilities of a typical AIOps system.

B. Layered Design

According to the above abilities, we present the hierarchical architecture of *Proton* with layered design to get better interoperability between modules, as shown in Fig. 3.

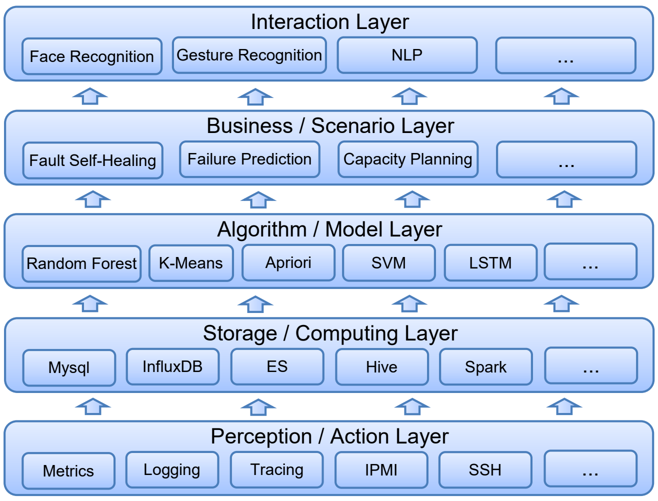

Fig. 3. The hierarchical architecture of Proton.

1) Perception/Action Layer: The abilities of perception and action are directly interacted with the environment (or IT systems), thus they can be grouped as a separate layer.

2) Storage/Computing Layer: The ability of perception collects massive data, which should be stored and calculated efficiently. Generally, big data solution is a good choice for this layer.

3) Algorithm/Model Layer: The abilities of detection and location abilities rely on various algorithms, models and experiences, which can be shared in many business scenarios, so we combine these two abilities into one layer.

4) Business/Scenario Layer: Based on Algorithm/Model Layer, a large number of IT O&M applications are supported, such as fault self-healing, hard drive failure prediction, capacity planning, Q&A systems. We put these applications into a separate layer named Business/Scenario Layer.

5) Interaction Layer: The Business/Scenario Layer needs to interact with people. In order to make this interaction more intelligent and convenient, an interaction layer is also needed.

C. Interoperability

The above hierarchical structure involves a large number of functional modules. As we have stressed before, most IT O&M teams usually have lots of traditional heterogeneous systems, which were built in different years with different programming frameworks. How to carry out efficient communication and cooperation among these modules, or namely interoperability, is of great challenge. In fact, even for the newly established departments from scratch, the completion of all modules will last for many years, during which the adopted framework and even programming language may change, thus a standard interoperability protocol is also needed.

After years of practice, we finally adopt a hybrid structure of micro-service framework and big data framework. The restful protocol used in micro-service framework can minimize the coupling between modules while still ensure the interoperability. In massive data processing scenarios, big data framework has better performance than micro-service framework.

III. Implementation Considerations

According to the design layout above, we have built several key modules and let them work together. This process will last for many years. In order to speed up the construction, we adopt the following principles: (1) Urgent things first. We select the applications urgently needed in Business/Scenario Layer, and give priority to the construction of the modules that these applications depend on. (2) Avoiding reconstruction. We prefer to modify the interface of the existing systems if their functionality meets the needs, rather than to rebuild them, which may take a lot of time.

During the implementation of *Proton*, we emphasize some key considerations including data categories, database cluster, service gateway and operation safety. These should be helpful for the construction of other AIOps systems.

A. Data Categories

For a modern IT O&M department, the data collected from Perception/Action Layer includes the following categories: (1) metrics, i.e. numerical data changing with time, such as CPU utilization, memory utilization and network traffic; (2) logging, i.e. unstructured text data generated by various hardware devices, operating systems, middle-ware and applications; (3) tracing, which describes the call relationship between modules or between functions within a module, helpful for application fault location; (4) configuration data, i.e. CMDB, such as equipment information, business configuration and other structured data; (5) workflow data, such as the data generated by the order management system; (6) multimedia data, which is usually unstructured and non-text, such as pictures, audio, video, etc.

In order to support the above data categories, we use a variety of storage types as shown in Table I: (1) Influxdb time series database cluster, which stores metrics data; (2) Mysql, to store configuration and workflow data; (3) ES (Elastic Search) cluster, to store logging and tracing data; (4) Hive, to store multimedia data. We have developed some storage standards to store these data in a unified way, so as to simplify the process of Detection/Location Layer. We also clean the data before storage, and monitor the health status of the stored data in real time.

TABLE I. Storage Types for Various Data Categories

Sequence	*Data Categories*	*Storage Types*
1	Metrics	Influxdb time series database cluster
2	Logging	ES cluster
3	Tracing	ES cluster
4	Configuration data	Mysql
5	Workflow data	Mysql
6	Multimedia data	Hive

B. Time Series Database Cluster

Open source framework can basically meet the above storage requirements. In *Proton*, because the Influxdb cluster is not open-source currently, and we need some custom requirements for data preprocessing, we have implemented a multi-node cluster based on the standalone version of Influxdb. In order to achieve this, we construct a hash function h to map the index x of the incoming data to be stored to a large integer space $\boldsymbol{N}$ (such as 65536). x can be the object name or IP (a tag in Influxdb).

$$h(x) = d,\ \ d \in \boldsymbol{N} \tag{1}$$

Suppose to build a cluster with k nodes initially, each node is responsible for a part of the hash values, and meets:

$$\bigcup_{\mathrm{i=0}}^{\mathrm{k-1}} \boldsymbol{N}_i = \boldsymbol{N} \tag{2}$$

where $\boldsymbol{N}_i$ is the set of hash values node i responsible for.

To ensure load balancing, the number of hash values that different nodes are responsible for should be basically the same, that is:

$$1 - \delta \leq \frac{|\boldsymbol{N}_i|}{|\boldsymbol{N}_0|} \leq 1 + \delta \tag{3}$$

where δ is a small value, such as 0.05.

In order to store or query data, we firstly calculate the hash value of the data, and then look up the table to get the nodes responsible for the hash value. Suppose $\boldsymbol{S}(d)$ is the node set responsible for hash value d, then:

$$\boldsymbol{S}(d) = \{\, \boldsymbol{N}_i \mid i \in [0, k-1] \land d \in \boldsymbol{N}_i \,\} \tag{4}$$

In response to node failure or expansion, we will store r copies of the same data, as in:

$$|\boldsymbol{S}(d)| = r,\ \ \ d \in \boldsymbol{N} \tag{5}$$

We set r to 2 or 3. Usually r copies of data are stored in different cabinets as many storage systems do to further improve reliability.

C. Service Gateway

Under the standard micro-service architecture, there is usually a service gateway as the entrance of all services. The service gateway can realize load balancing, failover, unified authority control, as well as service statistics and evaluation.

At first, we also adopted the gateway mode, but finally we gave up because of the following reasons: (1) the service gateway has a bottleneck in high-performance network communication; (2) the service gateway adds a fault point; (3) the service gateway cannot achieve fine-grained permission control of business, such as controlling the visibility of each data. (4) Our system uses multiple frameworks, such as Python and Java Spring Cloud, thus the compatibility of the unified gateway is not guaranteed. As an alternative, *Proton* adopt DNS and other mechanisms to achieve load balance, while service statistics is done by the services themselves.

D. Operation Safety

Perception, detection, location and interaction generally do not threaten the security of the platform, but action does. Although supporting more automated instructions is an effort direction of AIOps systems, actions should be carried out under the premise of security. Under the micro-service architecture, a large number of applications in Business/ Scenario Layer will call the instructions concurrently, thus a complete mechanism is necessary to ensure the security.

We mainly design two aspects of security mechanism: (1) Authority. Like many systems, we have strict permission control over the invocation of instructions. (2) Operation approval, which is divided into single point approval and global approval. Single point approval refers to the validity of instructions to a single object, such as continuously restarting a device (note that instructions may be issued by multiple applications that do not know each other). Global approval refers to the evaluation of all called instructions, such as whether there are too many device restart operations issued within a day. Generally speaking, global approval is harder to design than the other two.

IV. Measurements

Based on the above scheme, we have deployed *Proton* in an IT system environment with tens of thousands of devices, and some preliminary results have been achieved. In this section, we focus on three indicators of the system, namely, fault self-healing rate, storage capacity and service response delay.

A. Fault Self-Healing Rate

Fault self-healing rate refers to the proportion of faults recovered automatically without any human intervention among all. In other words, the fault is automatically detected, located and recovered. Since one of the ultimate goals of AIOps is to reduce manual operation, fault self-healing rate is a key indicator for AIOps systems. At present, the technology of fault self-healing across all scenarios is not mature, and we usually evaluate the fault self-healing rate in a certain scenario, such as server crash, process deadlock, storage overflow and so on.

We chose the scenario of server ping failure, which may be caused by server hardware or software failure, network problem, or cabinet power failure. Different fault causes need different treatment. Fig. 4 and Fig. 5 respectively show the failure rate and self-healing rate of this scenario. It can be seen that the failure rate of this scenario is about 0.05% daily and fluctuates randomly. The fault self-healing rate is about 60% at first days and 80% after optimization. With further optimization of the system, we expect to achieve more than 95% of the self-healing rate, and most of the other failures can only be recovered by replacing the hardware manually.

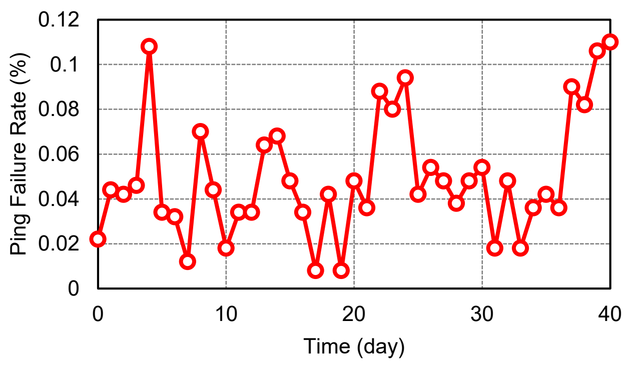

Fig. 4. Server ping failure rate every day.

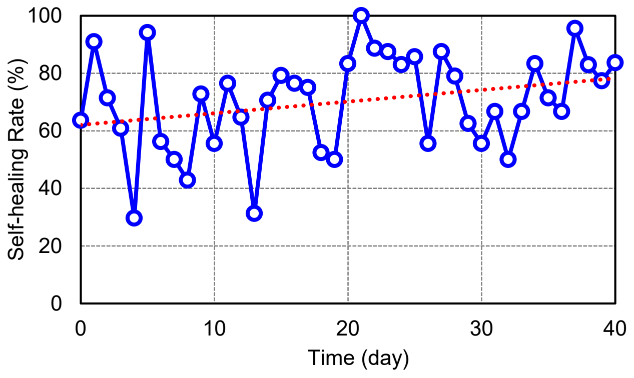

Fig. 5. Self-healing rate of server ping failure every day.

B. Storage Capacity

Based on the description in section III-B, we have built a time series database cluster. We need the system to be able to store 100 million metrics with a total storage capacity of 1PB. At present, 32 nodes are used to construct the cluster.

We evaluated the current performance of the system. The maximum write rate refers to the maximum number of metrics written per second, and the maximum query rate refers to the maximum number of queries supported per second for random metrics. As shown in Fig. 6, the maximum write rate is about 80,000 metrics/second, and the maximum query rate is about 7,500 metrics/second. Under different amount of concurrent connections, the max query rate is relatively stable, while the max write rate changes regularly. This may indicate that the processing of concurrent network connections needs to be optimized. When the amount of data increases in the cluster, the performance will gradually decrease, and it should be further optimized.

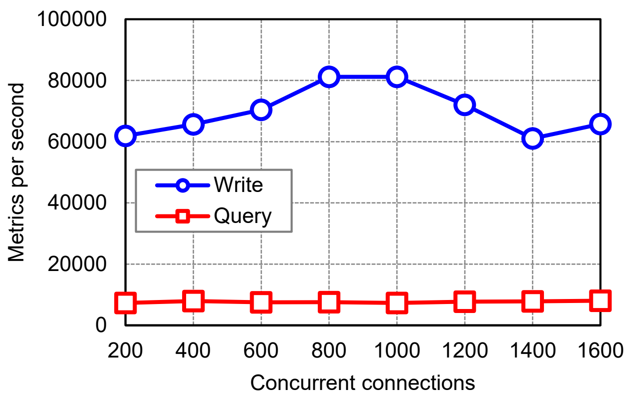

Fig. 6. Max write/query rate under different concurrent connections.

C. Service Response Delay

Service Response delay refers to the time interval between request and response packages. It is mainly related to business complexity, server performance and network delay. At present, our platform provides about 100 different services. We evaluated the response delay of these services in a typical day.

Fig. 7 represents the distribution of service response delay. About 93.2% of requests response within 1 second, and 63.1% response within 100 milliseconds. The average response time is about 292 milliseconds.

Fig. 8 shows the response delay of two typical services within one day. Interestingly, the response delay is densely distributed over several discrete values for both services. This may represent some of the most commonly invoked patterns for this type of service (such as a combination of query parameters). By analyzing the response latency of these services, we can also find some performance bottlenecks.

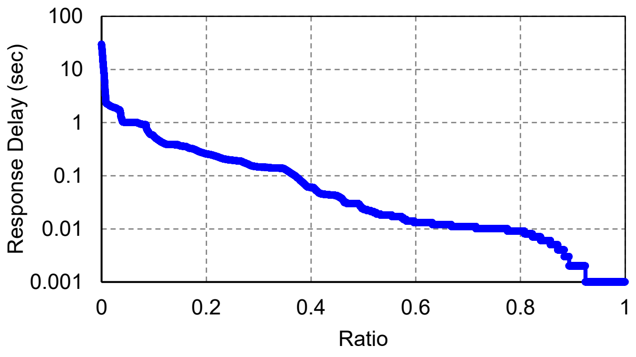

Fig. 7. Distribution of response delay for various services.

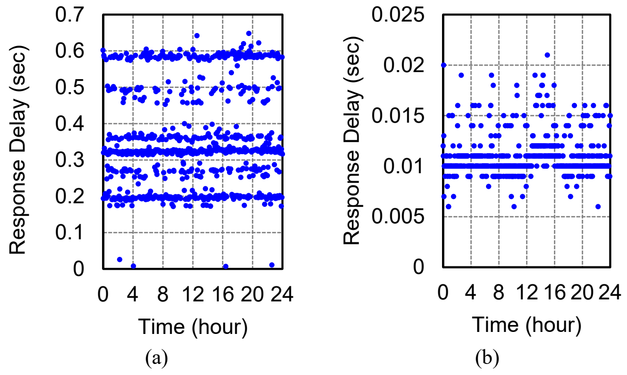

Fig. 8. Response delay of two typical services in a day.

V. Conclusions

AIOps, first proposed by Gartner in 2016, has been proved to be effective in quality improvement, cost reduction and efficiency improvement, and is considered as the ultimate solution for IT O&M. But for most enterprises, there are still many challenges to evolve from traditional systems to AIOps. This paper reviews the development of IT O&M technologies in the past two decades, and introduces five abilities that a typical AIOps system needs to have, namely perception, detection, location, action and interaction.

According to these abilities, we propose a novel AIOps system called *Proton*. *Proton* adopts the layered design with interoperability services between modules, which makes it well compatible with traditional heterogeneous systems. We have implemented *Proton* with some key considerations including data categories, database cluster, service gateway and operation safety. *Proton* has been deployed in a large IT system environment with tens of thousands of devices, and have achieved good results. In particular, the fault self-healing rate exceeds 80% for the scenario of server ping failure.

There are still many modules to be optimized or newly built in *Proton*. In the next work, we will focus on improving the detection and location success rate of the system.

Acknowledgment

This work is supported by The National Key Research and Development Program of China (2016YFB1000105).