Evaluation Scheme of Data-center-platform Service Based on Link Tracking Technology

Pan Zhang\textsuperscript{1,4}, Jianguang Hong\textsuperscript{2}, Yuan Wang\textsuperscript{3}, Yingwen Ren\textsuperscript{1} and Xuewei Ding\textsuperscript{1}

\textsuperscript{1}State Grid Information & Telecommunication Branch Beijing, China
\textsuperscript{2}State Grid Zhejiang Electric Power Company, LTD. Zhejiang, China
\textsuperscript{3}State Grid Electric Power Research Institute Co., Ltd., Nanjing, China
China Realtime Database Co., Ltd., Nanjing, China
\textsuperscript{4}Author to whom any correspondence should be addressed
zhangpan_23@126.com

Abstract. In the data-center-platform, all data is open to the outside world in the form of microservices. Due to lack of proper management, it is difficult to find the source of the problem from the intricate service call network when the service has a problem, thus missing the golden opportunity for stop loss. In addition, if an interface suddenly increases in time, it is no longer necessary to query the time-consuming situation by service, and the performance bottleneck of the service cannot be visually analyzed. In order to solve these problems, we propose an evaluation scheme of data-center-platform service based on link tracking technology. We use appropriate data tracking tools to collect data service operation data, analyze the operation data by designing a data service evaluation model, and find problematic data services in time.

1. Introduction

The most popular technology trend in the current digital economy is the data-center-platform. This concept was proposed by Ali and quickly spread and applied in many Internet companies. In the data-center-platform, all types of data are open to the public through data services. Users and other information systems use data by invoking data services.

Data services use a microservices architecture as is shown by figure 1. "Microservices architecture" describes a specific way of designing software applications as a set of independently deployable services. Under the traditional architecture, all functions of the application are implemented in the same process, and the expansion of the application is to replicate the entire process on multiple servers. Under the microservices architecture, application functions are split into smaller and independent services, and these services can be managed and deployed independently. In this way, application expansion and modification can be performed only for some services as needed, without affecting other running services.
Similarly, there are the following disadvantages:

1. The microservice architecture brings too many operations and maintenance operations, which may require the team to possess certain DevOps skills.

2. Distributed systems can be complex and difficult to manage, because the problem of distributed deployment tracking is difficult. As the number of services increases, the complexity of management increases.

As a hot concept in the big data industry, data center platform was first proposed by Ali [2]. Data center refers to the collection, calculation, storage and processing of massive data through data technology, and the unification of standards and caliber. Data center platform realizes data integration and data capability precipitation across business domains through data modeling. Through unified DaaS, data encapsulation and open sharing can be realized to meet the application requirements quickly and flexibly. Data development tools can meet the needs of personalized data and applications, realize the servitization of data applications, and promote data operation [3]. Data center has been widely used in all walks of life [4-7].

Therefore, relevant research on enterprise data service evaluation needs to be carried out to grasp the performance and stability of data services in real time to ensure that various complex data services meet the production and operation needs of enterprises.

2. Introduction to Link Tracking Technology

2.1. Link tracking technology

The term "link tracking" was proposed in 2010, when Google published a Dapper paper that introduced the implementation principles of Google's self-developed distributed link tracking, and also introduced how they can achieve transparency for applications at low cost.

In fact, Dapper was just an independent call link tracking system at the beginning, and gradually evolved into a monitoring platform, and many tools were born based on the monitoring platform, such as real-time early warning, overload protection, and index data query.

In addition to Google's dapper, there are some other well-known products, such as Ali's Hawkeye, Dianping's CAT, Twitter's Zipkin, Naver (the parent company of the famous social software LINE) pinpoint, and domestic open source skywalking.

2.2. Principles of link tracking technology

If we want to know where a problem occurs in an interface, you must know which services are called by the interface and the order in which they are called. If these services are connected together, it looks like a chain and is called a calling chain. If you want to implement the call chain, you need to make an identifier for each call, and then arrange the services according to the size of the identifier, you can see the calling sequence more clearly, and temporarily name the identifier spanid. In the actual scenario, you need to know the status of a certain request call, so only the spanid is not enough, you have to make...
a unique identifier for each request, so that you can find out all the services called by this request based on the identifier, and this identifier is named traceid.

Now according to spanid, you can easily know the order of the called services, but it cannot reflect the hierarchical relationship of the calls. As shown in the following figure, multiple services may be a chain of stepwise calls, or they may be called by the same service at the same time. So you should record who called each time, and use parentid as the name of this identifier.

Up to now, the order of calls and hierarchical relationship has been known, but after problems with the interface, the problematic link cannot be found. If a service has a problem, the service being called and executed must take a long time. At this time, the above three identifiers are not enough, and a time stamp needs to be added. The time stamp can be a little more precise and accurate to the microsecond level. It is not time-consuming to record only the timestamp when the call is initiated. It is necessary to record the timestamp when the service returns. The time difference can only be calculated from the beginning to the end. Can’t tell whose time stamp.
Although the total time from service invocation to service return can be calculated, this time includes the execution time and network delay of the service. Sometimes it is necessary to distinguish these two types of time to facilitate targeted optimization. How to calculate the network delay? The process of calling and returning can be divided into the following four events.

1. Client Sent is referred to as cs, and the client initiates a call request to the server.
2. Server Received is abbreviated as sr, which means that the server has received the call request from the client.
3. Server Sent is referred to as ss, which means that the server has completed the processing and is ready to return the information to the client.
4. Client Received is abbreviated as cr, which means that the client receives the return information from the server.

If the timestamp is recorded when these four events occur, the time consuming can be easily calculated, for example, sr minus cs is the network delay at the time of call, ss minus sr is the service execution time, cr minus ss is the service response Delay, cr minus cs is the execution time of the entire service call.

In fact, in addition to recording these parameters, you can also record some other information in the span block, such as the name of the service that was invoked, the name of the service being called, the return result, the IP, the name of the service that was called, and finally, the same spanid Combining a large span block completes a complete call chain.

![Figure 5. Link tracking data processing.](image)

2.3. Common link tracking tools
At present, tools based on link tracking technology include cat, zipkin, pinpoint, skywalking, etc. The following will analyze and study the application of technology.

Cat, open source by Dianping.com, is a real-time application monitoring platform developed based on Java, including real-time application monitoring and business monitoring. The integrated solution is to implement monitoring through code embedding, such as: interceptors, annotations, filters, etc. Cat is very intrusive to the code, the integration cost is higher, and the risk is greater.

Zipkin, open sourced by the Twitter team, Zipkin is a distributed tracking system. It helps to collect data when it is necessary to solve potential problems in the timing of the city’s microservice architecture. It manages data collection and search. This product is relatively simple to use in combination with spring-cloud-sleuth, and it is easy to integrate. But the function is relatively simple.

Pinpoint, open sourced by the South Korean team naver team, is used for link monitoring for large-scale distributed systems, using tools written in java. Inspiration comes from shortness and ingenuity, which helps to analyze the overall structure of the system and how internal components are called to
provide a good solution in distributed applications. Use java probe bytecode increase technology to realize monitoring of the entire application. Zero intrusion on application.

Skywalking, open sourced by individual Wu Sheng (Huawei developer) in 2015, joined the Apache incubator in 2017. The application performance monitoring system for distributed systems, especially for microservices, cloud native and containerized (Docker, Kubernetes, Mesos) architecture, the core of which is a distributed tracking system. It uses the java probe bytecode addition technology to monitor the entire application and zero intrusion into the application.

3. Data Service Evaluation Scheme Design

3.1. Data Service Evaluation Index Model
Combined with the needs of operation evaluation, the key indicators of design evaluation are as follows:

- Throughput, the real-time throughput of corresponding components, platforms, and physical devices can be calculated according to the topology.
- Response time, including the response time of the overall call and the response time of each service.
- Error record, according to the service returns the statistical unit time abnormal times.

Considering system performance and realizability, other requirements for the evaluation method of data service operations are as follows:

**The performance consumption of the probe is small.** The impact of APM component services should be small enough. The service call buried point itself will cause performance loss, which requires low loss of call tracking. In practice, a part of the request will be selected to analyze the request path by configuring the sampling rate. In some highly optimized services, even a little loss can be easily detected, and it may force the deployment team of online services to have to shut down the tracking system.

**Low code intrusion.** As a business component, there should be as few intrusions as possible or no intrusion into other business systems, which is transparent to users and reduces the burden on developers. For application programmers, there is no need to know that there is a tracking system. If a tracking system wants to take effect, it must rely on the active cooperation of the developers who rely on the application, then the tracking system is also too fragile, often because of bugs or negligence of the tracking system implanted code in the application, the application has problems, so that it cannot be meet the need for "ubiquitous deployment" of tracking systems.

**Strong scalability.** An excellent call tracking system must support distributed deployment and have good scalability. The more components that can be supported, the better. Or provide a convenient plug-in development API, for some components that are not monitored, application developers can also extend themselves.

**Capable of analyzing data.** Data analysis should be fast, with as many dimensions as possible. The tracking system can provide fast enough information feedback to respond quickly to abnormal conditions in the production environment. Comprehensive analysis can avoid secondary development.

3.2. Data Service Evaluation Scheme Design
According to the data service operation evaluation model, the data service evaluation scheme is designed. The main functional modules include: burying and generating logs, collecting and storing logs, analyzing and statistically calling link data, displaying and decision support.

**Burying points and generating logs.** The buried point is the context information of the system at the current node, and can be divided into the buried point of the client, the buried point of the server, and the two-way buried point of the client and the server. The buried point log usually contains the following content: traceId, spanId, call start time, protocol type, caller ip and port, requested service name, call time, call result, exception information, etc. At the same time, expandable fields are reserved for Prepare for the next expansion.

**Collecting and storing logs.** Mainly support the distributed log collection scheme, and add MQ as a buffer at the same time. Functions include:
There is a daemon on each machine for log collection, the business process sends its own trace to the daemon, and the daemon sends the collected trace to the next level.

- Multi-level collector, similar to pub / sub architecture, can be load balanced.
- Real-time analysis and offline storage of aggregated data.
- Offline analysis: The logs of the same call chain need to be aggregated together.

**Analysis and statistics of call link data.** The call chain tracking analysis function is to collect Spans with the same TraceID, and the timeline is the timeline. Stringing ParentID is the call stack. Throw exception or timeout, print TraceID in the log. Use TraceID to query the calling chain and locate the problem.

The dependency measurement includes three scenarios:
- Strong dependence: the failure of the call will directly interrupt the main process
- High dependency: the probability of calling a dependency in a link is high
- Frequent dependencies: The same dependency is called many times in one link

Analysis functions include offline analysis and real-time analysis, including:
- Offline analysis: Summarize by TraceID, restore the calling relationship through Span ID and ParentID, and analyze the link form.
- Real-time analysis: Direct analysis of a single log without summary and reorganization. Get current QPS, delay.

**Evaluation and analysis.** The evaluation and analysis function can provide managers with fully detailed decision-making basis, so that managers can more clearly understand the actual situation at the grass-roots level in their work.

### 3.3. Overall architecture design

The main process of carrying out data service evaluation includes:

1. Compile a data service indicator model and formulate a data collection plan according to the model.
2. When the access subject requests the data service, a unique identification is generated for the access subject, all operations and data related to the identification are stored in a log file and various monitoring indicators are collected along the business link.
3. Real-time monitoring and flow pressure evaluation of the system network through the flow pressure evaluation technology. Once the stability of the data service is affected, the link tracking technology is called to collect, store, and analyze the call event data in the distributed system, and related Log information to locate performance bottlenecks to assist development and operation personnel in fault diagnosis and ensure the stability of services.

![Figure 6. Overall architecture diagram.](image-url)
4. Conclusion
This paper mainly proposes a data service evaluation scheme based on link tracking technology. By designing the data service evaluation index model, the data service evaluation standard is clarified. By analyzing the existing link tracking tools, the existing link tracking technology capabilities are clarified, and the appropriate link tracking tool is selected to obtain data service operation data. Finally, the operation data is analyzed according to the model, and evaluates data services according to indicators.

Acknowledgment
Fund Project: This paper was supported by the science and technology project "research and application of key technologies in enterprise-level data-center-platform based on full-service unified data center" of State Grid Corporation of China (5211XT190033).

References
[1] Benjamin H. Sigelman, Luiz Andr´e Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, Chandan Shanbhag. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Google Technical Report dapper-2010-1, April 2010.
[2] Tassi, Andrea, et al. "Agile Data Offloading over Novel Fog Computing Infrastructure for CAVs." 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring) IEEE, 2019.
[3] Sungryong, Hong, et al. "Constraining cosmology with big data statistics of cosmological graphs." Monthly Notices of the Royal Astronomical Society 4:4.
[4] Mirjana Pejić Bach, et al. "Text Mining for Big Data Analysis in Financial Sector: A Literature Review." Sustainability 11(2019).
[5] Coleman, Mark R. "ERP Grants First Action Status to Candidate Tissue Residue Methods for High-Priority Animal Drugs." Journal of AOAC International 4:4.
[6] B.S. Li, Q.G. Hu, X.F. Chen, B.Q. Gao. “Research and Design of Data Platform for Power Grid Enterprise,” Ele. Pow. Inf. Com. Tec., vol. 7, 2019.
[7] L.N. Feng, X. Zhu. “Integration of Resources to Improve Efficiency to Build Operators in Data Center Platform System,” C-Enterprise Management. vol. 1, pp. 29-33, 2019.
[8] Pahl, Claus, and P. Jamshidi. "Microservices: A Systematic Mapping Study." 6th International Conference on Cloud Computing and Services Science SCITEPRESS - Science and Technology Publications, Lda, 2016.
[9] Payam, Emami Khoonsari, et al. "Interoperable and scalable data analysis with microservices: applications in metabolomics." Bioinformatics 19:19.
[10] Apel, Sebastian, F. Hertrampf, and Steffen Späthe. "Toward a knowledge model focusing on microservices and cloud computing." Concurrency & Computation Practice & Experience 1(2019).
[11] Mazzara, Manuel, et al. "Microservices: Migration of a Mission Critical System." IEEE Transactions on Services Computing (2019).
[12] Cruz, Waldemar, F. Liu, and L. Michel. "A Counting-Based Approach to Scalable Micro-service Deployment." (2019).
[13] Munge, Kenneth, et al. "A critical analysis of purchasing arrangements in Kenya: the case of micro health insurance." Bmc Health Services Research 19.1(2019):45.
[14] Z.H. Li. “Analysis of the Development and Impact of Microservice Architecture,” China CIO News, vol. 1, pp.154-155, 2017.
[15] D. Guo, W. Wang, G.S. Zeng. “A New Cloudware PaaS Platform Based on Microservices Architecture,” Netinfo Security, vol. 11, pp.15-20, 2015.
[16] Hall, Mary, et al. "Smoothing the Path to Computing; Pondering Uses for Big Data.” Communications of the Acm 62.3(2019):8-9.
[17] Mohammadi, M, and A. Al-Fuqaha. "Enabling Cognitive Smart Cities Using Big Data and Machine Learning: Approaches and Challenges." IEEE COMMUNICATIONS MAGAZINE (2018).
[18] Jiang, Yuxuan, Z. Huang, and D. H. K. Tsang. "Towards Max-Min Fair Resource Allocation for Stream Big Data Analytics in Shared Clouds." IEEE Transactions on Big Data 4.1(2018):130-137.