

Alibaba Cloud Log Service

Best Practices

Issue: 20190920

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>switch {stand slave}</code>

Contents

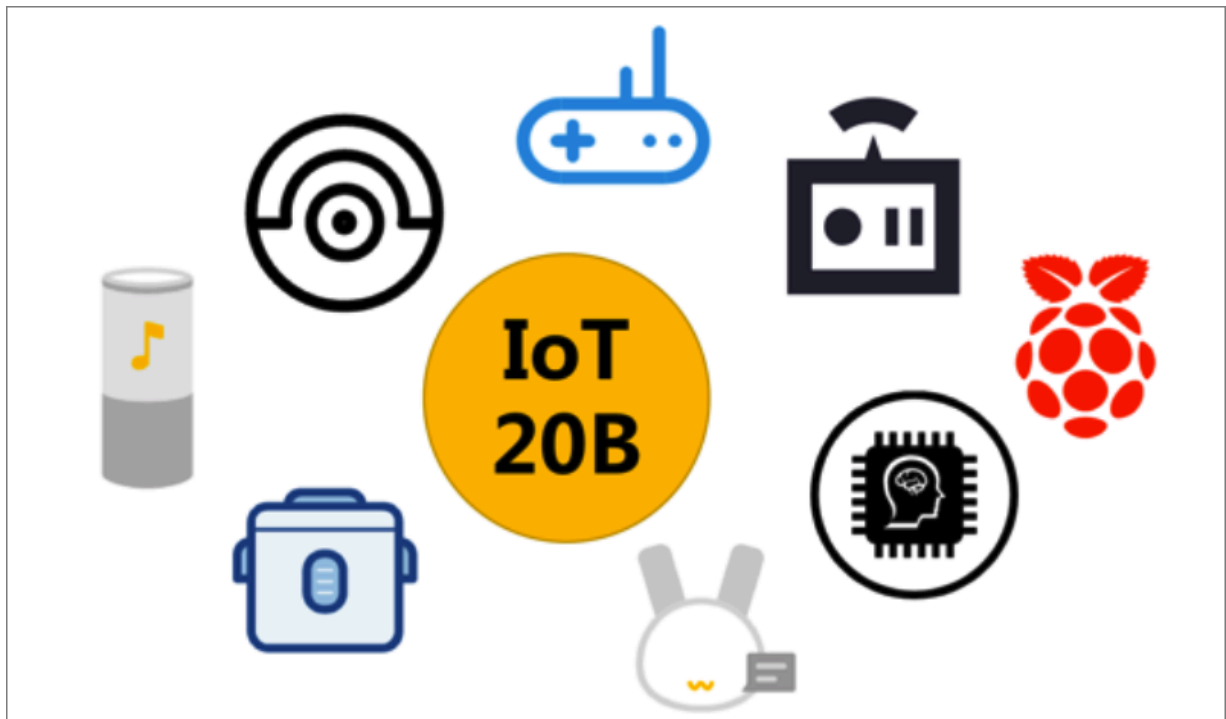
Legal disclaimer.....	I
Generic conventions.....	I
1 Collection.....	1
1.1 Collect IoT or embedded development logs.....	1
1.2 Collect logs through Web Tracking.....	11
1.3 Build a service to read logs from mobile apps directly.....	18
1.4 Collect data over the public network.....	24
1.5 Collect data from multiple channels.....	27
1.6 Manage logs.....	34
2 Query and analysis.....	39
2.1 Display query and analysis results on multiple pages.....	39
2.2 Analyze sales system logs.....	44
2.3 Analyze website logs.....	45
2.4 Analyze vehicle track logs.....	59
2.5 Analyze access logs of Layer-7 Server Load Balancer.....	64
2.6 Analyze NGINX access logs.....	72
2.7 Query MNS logs.....	81
2.8 Query and analyze AppLogs.....	89
2.9 Perform association query and analysis on logs and the data from databases.....	97
2.10 Perform association query and analysis on logs and tables from OSS.....	101
3 Consumption.....	104
3.1 Use Function Compute to cleanse log data.....	104
3.2 Build a monitoring system.....	110
3.3 Consume metering and billing logs.....	112
3.4 Use a consumer library to consume logs in high reliability mode.....	117
3.5 Cleanse data through ETL.....	127
4 Shipping.....	131
4.1 Connect to a data warehouse.....	131
5 Operations logs.....	134
5.1 Activate, monitor, and consume operations logs.....	134
6 Alerting.....	141
6.1 Configure an alert.....	141

1 Collection

1.1 Collect IoT or embedded development logs

Internet of Things (IoT) is enjoying high growth. More and more IoT devices are being applied to our daily life, such as smart routers, various TV dongles, Tmall Genie, and robot vacuum cleaners, bringing us the convenience of intelligence. According to Gartner's prediction, 20 billion smart devices will be put into service by the end of 2020, which gives a glimpse of the huge market in this field. The embedded development model in the traditional software field is faced with great challenges in the IoT field. In addition to the large number and wide distribution, IoT devices are difficult to debug and restricted in hardware. As a result, traditional device log solutions cannot meet the demands perfectly.

Based on years of Logtail development experience and the characteristics of IoT devices, the Log Service team has customized a log data collection solution for IoT devices: C Producer Library.

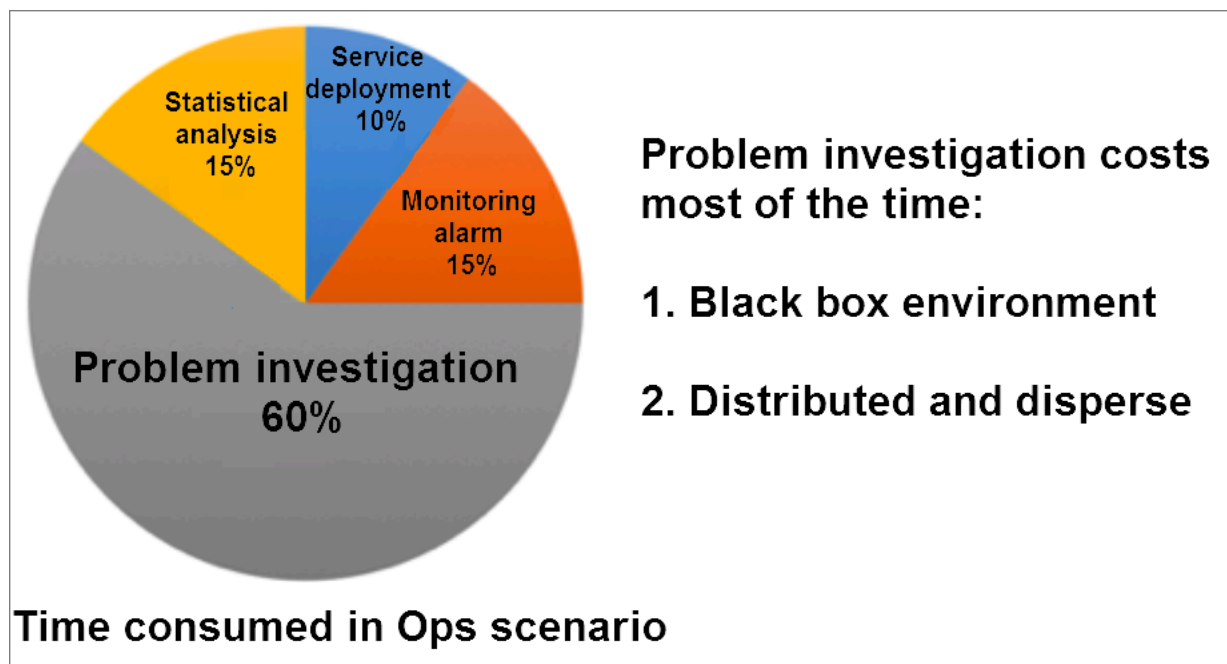


Embedded development requirements

The IoT or embedded development engineers must have profound knowledge and experience in development and the capability to manage, monitor, and diagnose

a large number of black boxes. Embedded development mainly has the following requirements:

- **Data collection:** How can the engineers collect data from millions or even tens of millions of devices distributed all around the world in real time?
- **Debugging:** How can the engineers use one solution to meet the requirements of both online data collection and real-time debugging in development?
- **Online diagnosis:** When an error occurs on an online device, how can the engineers locate the device quickly and check the error context?
- **Monitoring:** How many devices are currently online? How is the working status distribution? How is the geographic distribution? How does a device send alerts in real time when an error occurs?
- **Real-time data analysis:** How is the data generated by devices integrated with real-time computing and big data warehouses to build user profiles?



Major challenges in the IoT field

When thinking about the solutions to the preceding problems, we find that the approaches in the traditional software field are ineffective in the IoT field. The main challenges arise from the following characteristics of IoT devices:

- **Large numbers of devices:** In the traditional O&M field, a company managing 10,000 servers is qualified for a large one. However, managing 100,000 online devices is only a small threshold in the IoT field.

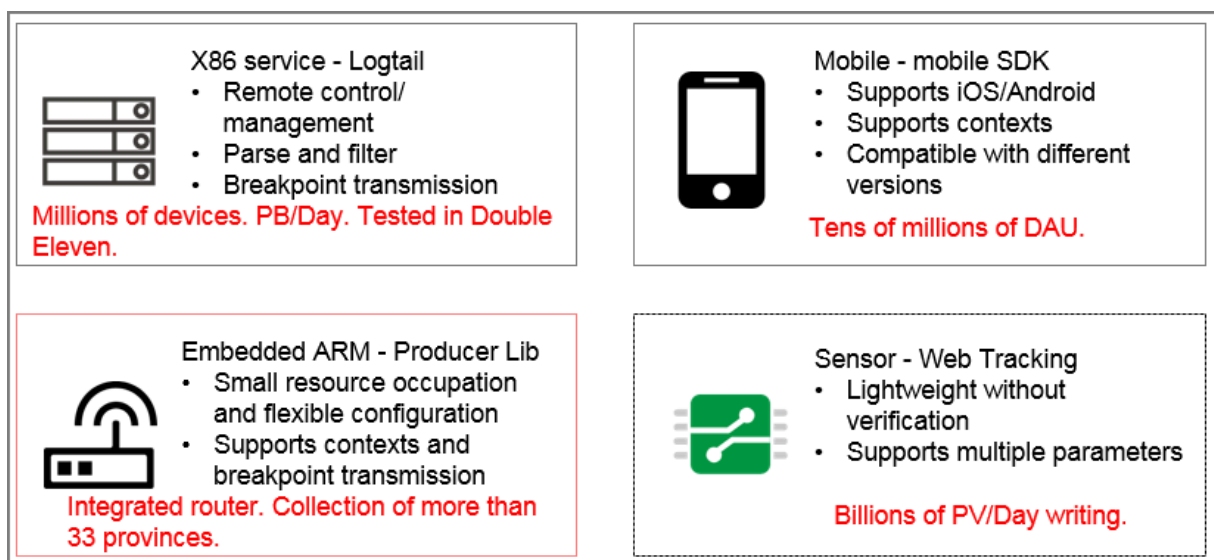
- **Wide distribution:** The deployed hardware devices are usually distributed around the country or even the world.
- **Black box:** IoT devices are mostly in unknown states. It is difficult to log on to and debug these devices.
- **Restrictions in resources:** IoT devices are relatively restricted in hardware to reduce costs, for example, the total memory size may only be 32 MB. As a result, traditional log collection approaches for PCs do not work in the IoT field.

C Producer Library: a log data collection solution customized by Log Service

Logtail, the client of [Log Service](#), is deployed on millions of x86 servers. For more information, see [Logtail technology sharing](#): and . In addition, Log Service provides a variety of collection solutions:

- **Mobile SDK:** It collects data from Android or iOS platforms with tens of millions of daily active user (DAUs).
- **Web Tracking (JS):** Similar to Baidu Tongji and Google Analytics, it uses a lightweight collection mode without signature.

In the IoT field, based on years of Logtail development experience and the characteristics of IoT devices in terms of CPU, memory, disk, network, and application mode, we have developed a log data collection solution for IoT devices: C Producer Library.



Features of C Producer Library

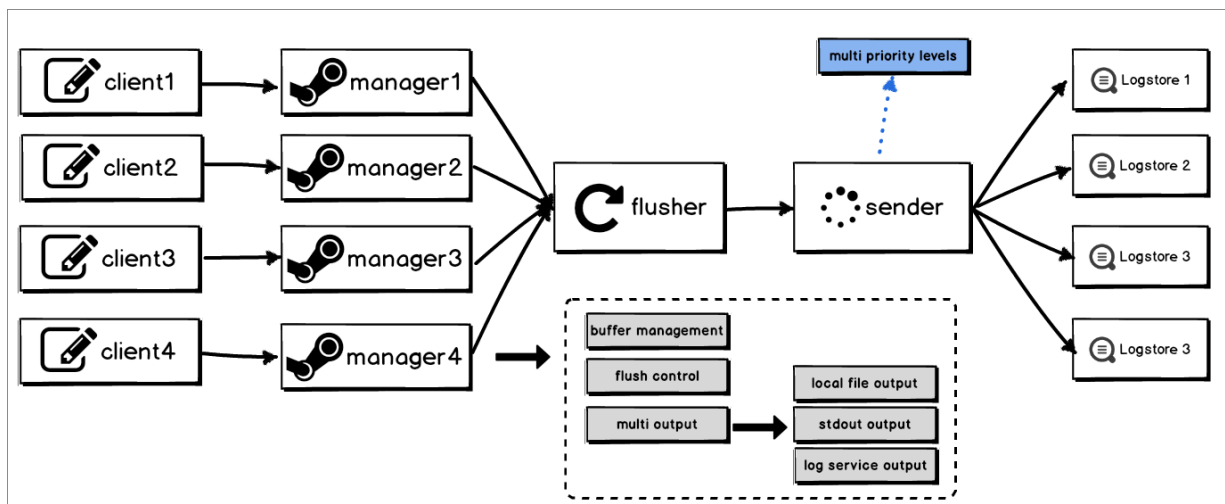
Like a lightweight Logtail, C Producer Library offers high stability, high performance, and low resource consumption. Although it does not have the feature of real-time

configuration management in Logtail, C-Producer Library has 70% of the features of Logtail, including:

- **Multiple tenants:** C Producer Library can process various types of logs (such as Metric, DebugLog, and ErrorLog) according to their priorities. Multiple clients can be configured and each client can be separately configured with the collection priority and target project or Logstore.
- **Context query:** Logs generated by the same client are in the same context, and the relevant logs before and after a log can be viewed.
- **Concurrent sending and resumable upload:** The upper limit of cache can be set. Logs fail to be written when the upper limit is exceeded.

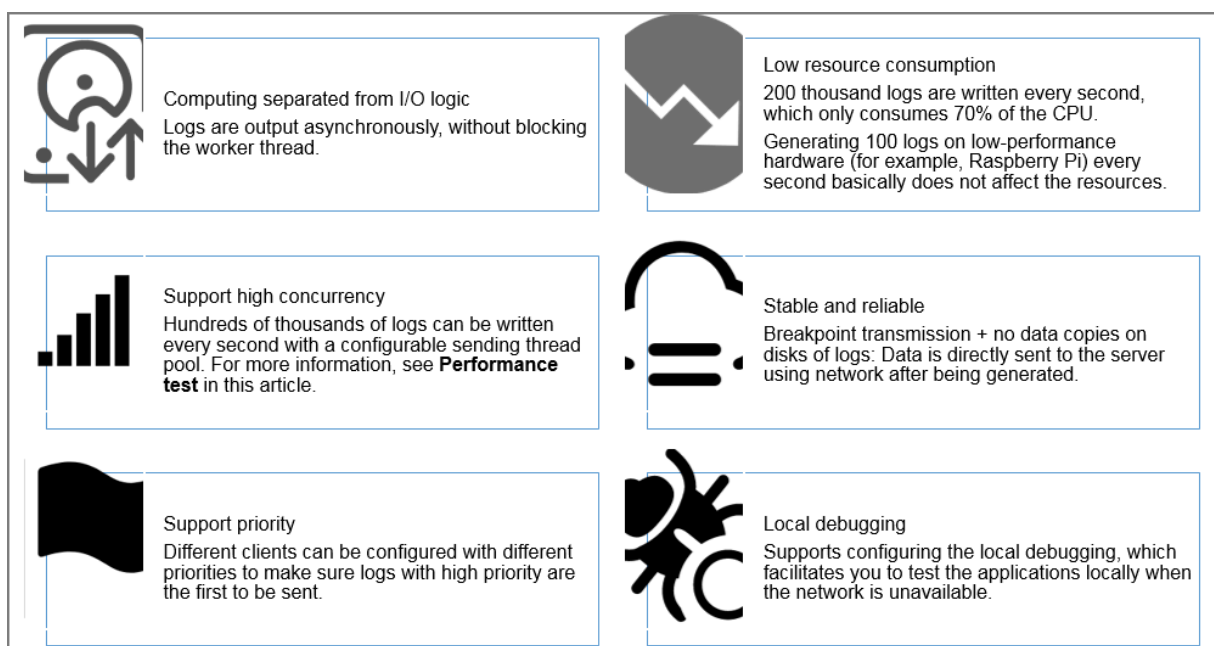
In addition, C Producer Library provides the following features specific to IoT devices , including:

- **Local debugging:** Logs can be exported to local devices. You can set the rotation, log quantity, and rotation size.
- **Fine-grained resource control:** Different cache upper limits and aggregation modes can be set for different types of data or logs.
- **Log cache compression:** The cache of the data failed to be sent can be compressed to reduce the memory usage of devices.



Advantages of C Producer Library

As a custom solution for IoT devices, C Producer Library has obvious advantages in the following aspects:



- **Highly concurrent write on the client:** Hundreds of thousands of logs can be written every second with a configurable sending thread pool. For more information, see "Performance test" in this topic.
- **Low resource consumption:** Only 70% of the CPU is occupied when 200,000 log entries are written every second. The resources are not affected when up to 100 log entries are generated on low-performance hardware (for example, Raspberry Pi) every second.
- **No data copies on disks of client logs:** Data is directly sent to the server through the network after being generated.
- **Client computing separated from I/O logic:** Logs are generated asynchronously, without blocking the working thread.
- **Multiple priorities:** Different clients can be configured with different priorities to make sure that logs with higher priorities are sent first.
- **Local debugging:** Local debugging can be configured to facilitate you to test the applications locally when the network is unavailable.

In the preceding scenarios, C Producer Library simplifies application development. You do not have to consider log collection details or worry about the impact of log collection on your business operations. This makes data collection significantly easier.

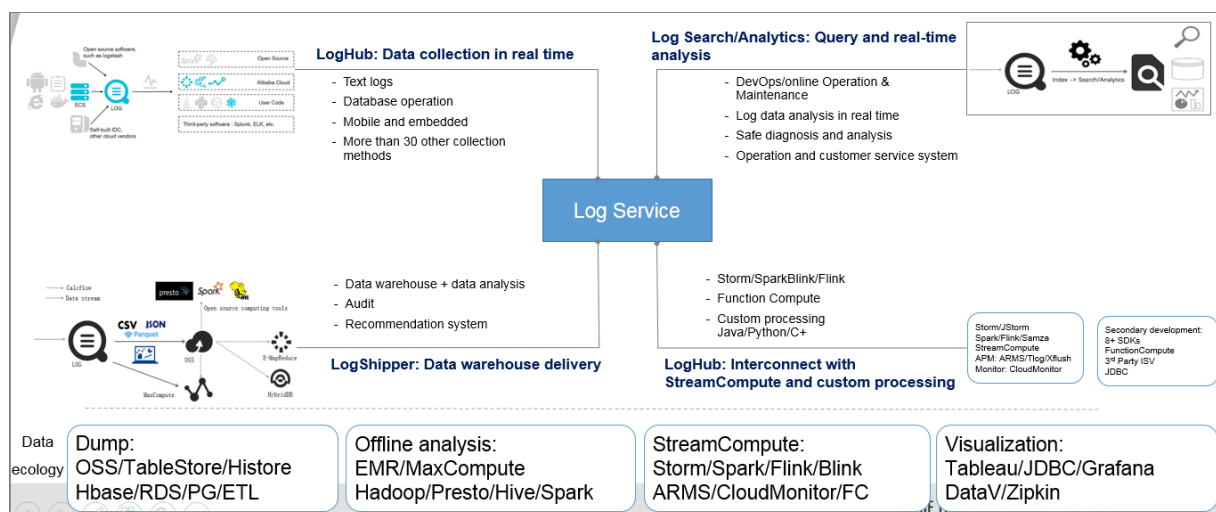
To make it distinct, we made a comparison between C Producer Library and other embedded collection solutions. The following table lists the comparison results.

Type		C Producer Library	Other solutions
Programming	Platform	Mobile + Embedded	Mobile-based
	Context	Supported	Not supported
	Multiple logs	Supported	Not supported (one type of logs)
	Custom format	Supported	Not supported (several limited fields are provided)
	Priority	Supported	Not supported
	Environment parameter	Configurable	Configurable
Stability	Concurrency	High	Medium
	Compression algorithm	LZ4 (balance between efficiency and performance) + Gzip	Optimized
	Low resource consumption	Optimized	Medium
Transmission	Resumable upload	Supported	By default, resumable upload is not supported. Secondary development is required.
	Access point	8 (in China) + 8 (outside China)	Hangzhou
Debugging	Local log	Supported	Supported in manual mode
	Parameter configuration	Supported	Not supported
Real-time performance	Visible on the server side	1 second (99.9%) to 3 seconds (maximum)	1 to 2 hours
Custom processing		More than 15 interconnection modes	Customized real-time and offline solution

C Producer Library + Log Service solution

C Producer Library integrates with Alibaba Cloud [Log Service](#) to form a complete set of log solutions for IoT devices.

- Large scale
 - Supports writing hundreds of millions of log entries on the client in real time.
 - Supports writing petabytes of data every day.
- High speed
 - Fast data collection: Data can be consumed after being written without any latency.
 - Quick query: Billions of data records can be processed and queried within 1 second by using a complex query statement (with five conditions).
 - Rapid analysis: Hundreds of millions of data records can be aggregated and analyzed within 1 second by using a complex analysis statement (aggregated with five dimensions and the GroupBy statement).
- Wide interconnection
 - Seamlessly integrated with various Alibaba Cloud products.
 - Compatible with various open-source storage, computing, and visual systems.

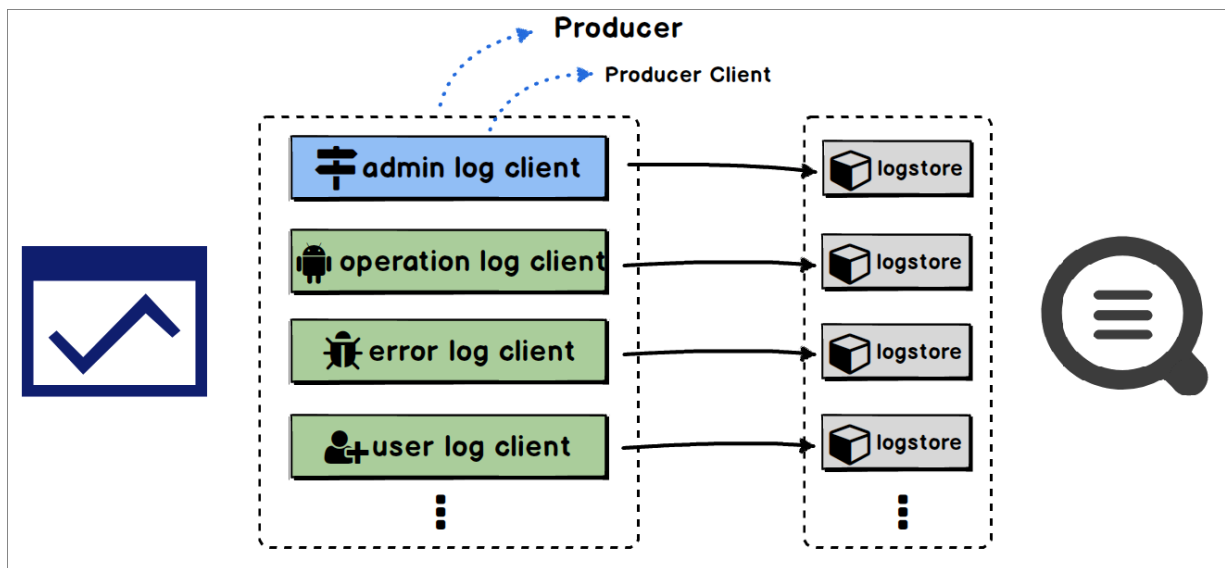


Download and use

Download URL: [GitHub](#)

One application can create multiple producers, and each producer can include multiple clients. Each client can be independently configured with the target address, log level, local debugging, cache size, custom identifier, and topic.

For more information about installation methods and operation steps, see [README](#).



Performance test

Environment configuration

- High-performance scenario: traditional x86 servers.
- Low-performance scenario: Raspberry Pi (low power consumption environment).

The following figure shows the configurations.

High-performance scenario	Low-performance scenario
<ul style="list-style-type: none"> • CPU: Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50 GHz • Memory: 64 GB • Operating system: Linux version 2.6.32-220.23.2.el11113.el5.x86_64 • GCC: 4.1.2 • C-Producer: Dynamic library is 162 KB. Static library is 140 KB. (Use the static library for test. The binary after compilation is 157 KB. All are stripped.) 	<p>Type: Raspberry Pi 3B</p> <p>CPU: Broadcom BCM2837 1.2 GHz A53 64 bit. (Use the host USB for power supply. The frequency is lowered to 600 MHz.)</p> <p>Memory: 1 GB DDR2</p> <p>Operating system: Linux 4.9.41-v7+ #1023 SMP armv71 GNU/Linux</p> <p>GCC: 6.3.0 (Raspbian 6.3.0-18+rpi1)</p> <p>C-Producer: Dynamic library is 179 KB. Static library is 162 KB. (Use the static library for test. The binary after compilation is 287 KB. All are stripped.)</p>

C Producer Library configuration

- ARM (Raspberry Pi)
 - Cache: 10 MB
 - Aggregation time: 3 seconds (If any of the conditions is met, namely, aggregation time, aggregation data package size, and aggregation log quantity, the data is packaged and sent.)
 - Aggregation data package size: 1 MB
 - Aggregation log quantity: 1,000
 - Sending thread: 1
 - Custom tag: 5
- x86
 - Cache: 10 MB
 - Aggregation time: 3 seconds (If any of the conditions is met, namely, aggregation time, aggregation data package size, and aggregation log quantity, the data is packaged and sent.)
 - Aggregation data package size: 3 MB
 - Aggregation log quantity: 4,096
 - Sending thread: 4
 - Custom tag: 5

Sample log

1. The total data volume is approximately 600 Bytes for 10 key-value pairs.
2. The total data volume is approximately 350 Bytes for 9 key-value pairs.

```

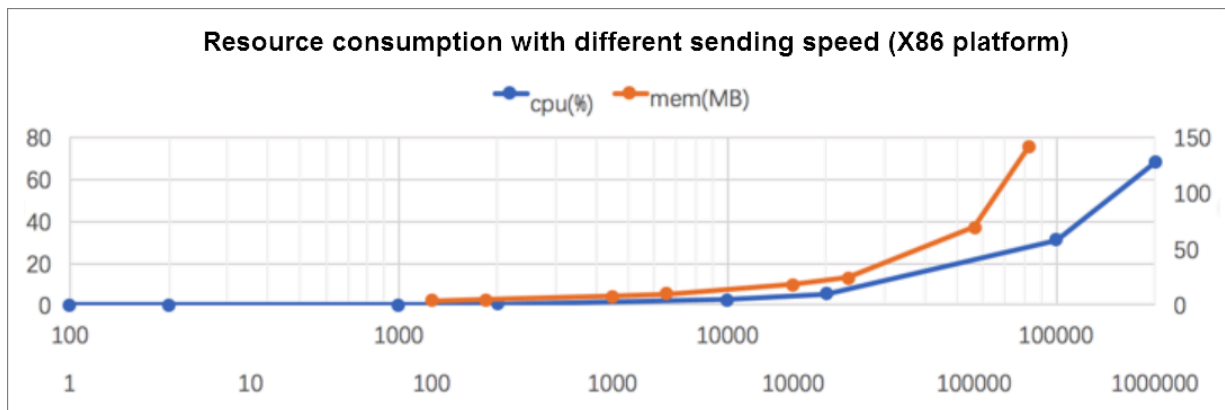
__source__ : 11 . 164 . 233 . 187
__tag__ : 1 : 2
__tag__ : 5 : 6
__tag__ : a : b
__tag__ : c : d
__tag__ : tag_key : tag_value
__topic__ : topic_test
_file_ : / disk1 / workspace / tools / aliyun - log - c - sdk /
sample / log_producer_sample . c
_function_ : log_producer_post_logs
_level_ : LOG_PRODUCER_LEVEL_WARN
_line_ : 248
_thread_ : 40978304
LogHub : Real - time log collection and consumption
Search / Analytics : Query and real - time analysis
Interconnection : Grafana and JDBC / SQL92
Visualized : dashboard and report functions

```

Test results

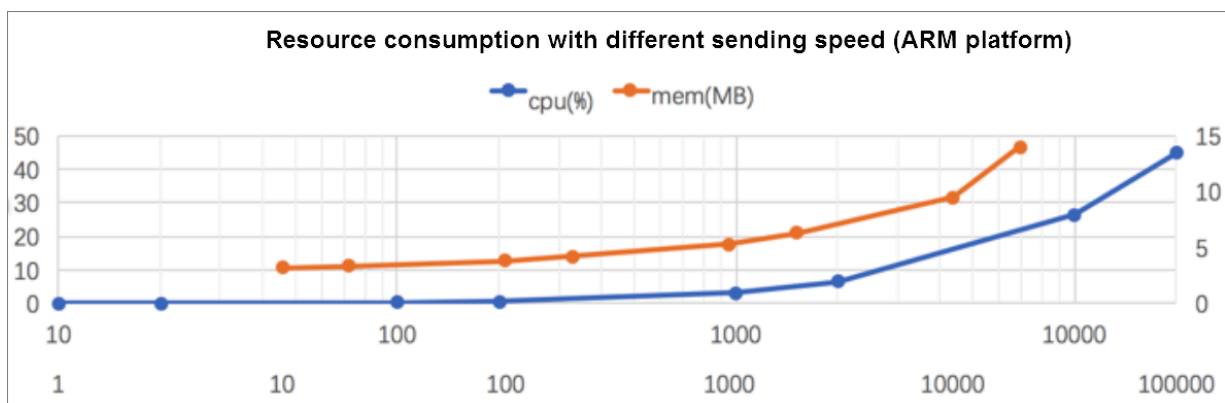
Test results on the x86 platform

- C Producer Library sends up to 90 MB data every second. It consumes only 70% of CPU and 140 MB of memory to upload 200,000 log entries every second.
- When the sending speed of the server is 200 entries every second, data sending basically does not have any impact on CPU (reduced to less than 0.01%).
- The average time it takes the client thread to send an entry of data (or generate a log entry) is 1.2 μ s.



Test results on the Raspberry Pi platform

- In the Raspberry Pi test, the frequency of CPU is only 600 MHz. Therefore, the performance is approximately 10% of that of the server. The highest sending speed is 20,000 log entries every second.
- When the sending speed of Raspberry Pi is 20 entries every second, data sending basically does not have any impact on CPU (reduced to less than 0.01%).
- Raspberry Pi uses a USB to connect to a PC shared network. The average time it takes the client thread to send an entry of data (or generate a log entry) is about 12 μ s.



1.2 Collect logs through Web Tracking

When sending an important email, you will set the `read_receipt` tag in the email to make sure that the recipient has read the email. You will receive a reminder when the recipient reads the email.

The `read_receipt` mode is widely used in many scenarios, such as:

- Checking whether the recipient has read the sent leaflet.
- Checking how many users have clicked a promoted webpage.
- Analyzing user access conditions on a mobile application marketing activity page.

Traditional website- and webmaster-based solutions cannot be used for custom collection and statistics, as they face the following difficulties:

- Hard to meet the individual needs: User behavior data is not generated at the mobile client. The user behavior data includes some parameters out of operations-based custom needs, such as the source, channel, environment, and behavior parameters.
- High development difficulty and cost: To meet the need of data collection and analysis, you must first purchase the cloud host, public network IP address, development data receiving server, and message-oriented middleware. In addition, you must adopt mutual backup to guarantee the high availability of the services. Then, develop the server and perform tests.
- Hard to use: After data is transmitted to the server, engineers must cleanse the results and import them to the database to generate data for operations.
- Inelastic: The usage of users cannot be predicated. Therefore, a large resource pool must be reserved.

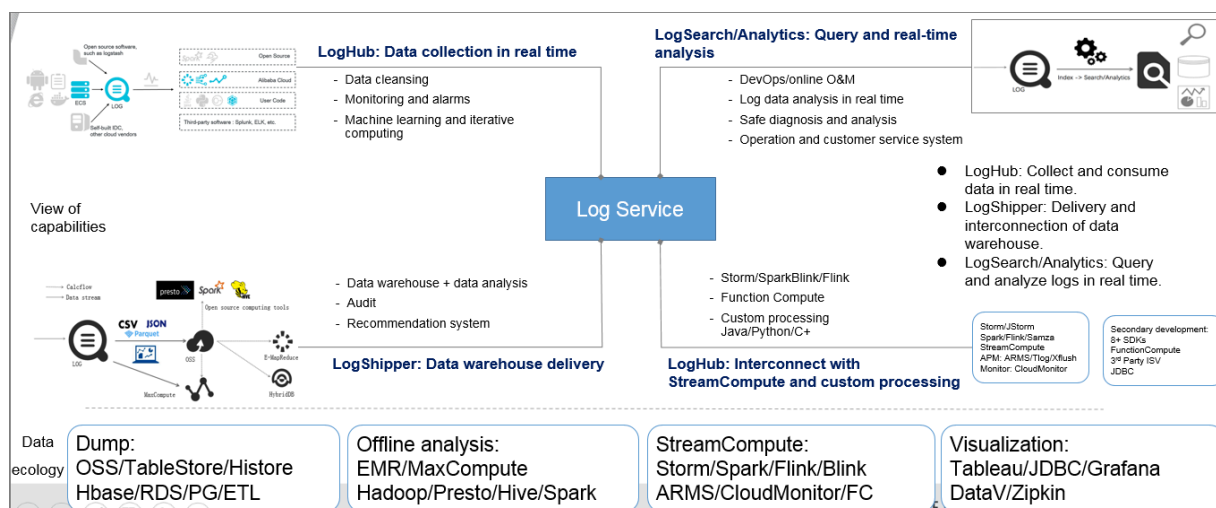
From the preceding aspects, when a content-oriented operations need is raised, a big challenge is how to quickly meet the collection and analysis needs of such user behavior data.

[Log Service](#) provides Web Tracking, JS, or Tracking Pixel SDK for the preceding lightweight tracking collection scenarios. With this feature, tracking and data reporting can be completed within 1 minute. In addition, Log Service provides 500 MB [FreeTier quota](#) per month for each account, allowing you to process business without any cost.

Features

The collection and analysis solution is based on Alibaba Cloud Log Service, a one-stop service for log data. Log Service allows you to quickly complete the collection, consumption, shipping, query, and analysis of large amounts of log data without the need for development. This improves both the O&M efficiency and the operational efficiency. Log Service has the following features:

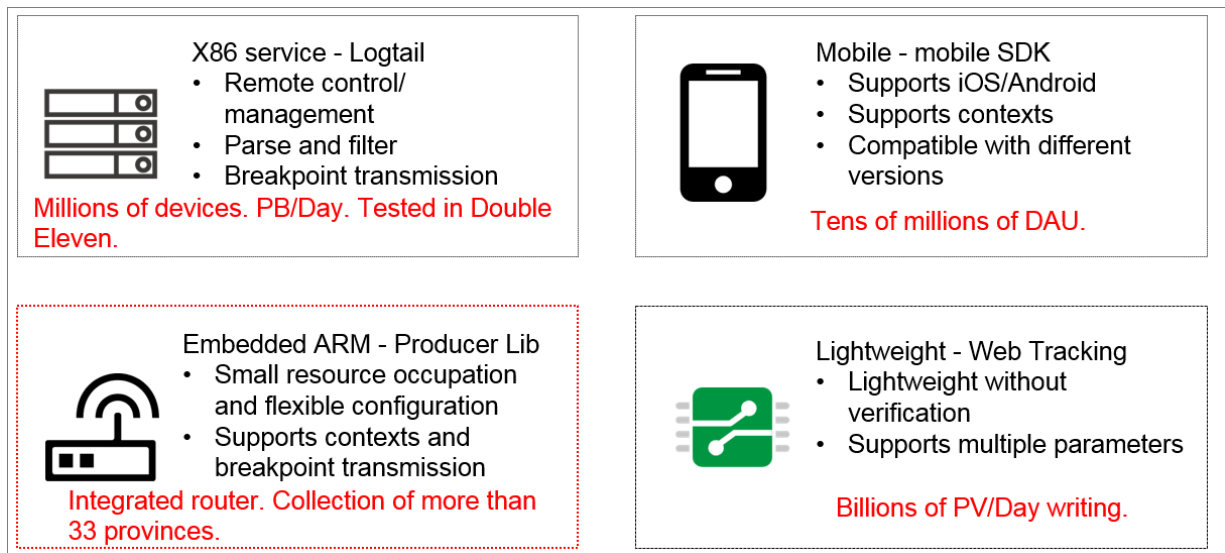
- **LogHub** for real-time collection and consumption. It is interconnected with Blink, Flink, Spark Streaming, Storm, and Kepler.
- **LogShipper** for data shipping. It is interconnected with MaxCompute, E-MapReduce, Object Storage Service (OSS), and Function Compute.
- **LogSearch and Analytics** for query and real-time analysis. It is interconnected with DataV, Grafana, Zipkin, and Tableau.



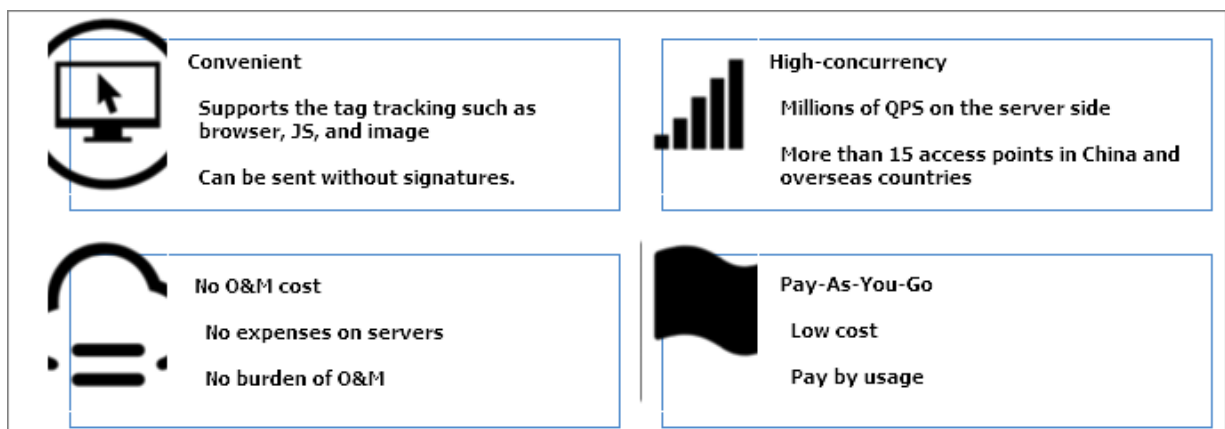
Advantages on the collection side

Log Service provides more than 30 data collection approaches and complete solutions for servers, mobile clients, embedded devices, and various development languages. For example:

- **Logtail:** the log collection agent for x86 servers.
- **Android or iOS SDK:** oriented to mobile clients.
- **C Producer Library:** oriented to devices with limited CPU or memory, and smart devices.



The lightweight collection solution Web Tracking in this topic only uses an HTTP GET request to transmit data to a Log Service Logstore. It is applicable to scenarios where no verification is required, such as static websites, advertising, document promotion, and mobile data collection. The following figure shows the characteristics of Web Tracking, when compared with other log collection solutions.



Web Tracking access process

The term Web Tracking (also named Tracking Pixel) is from the image tag in the HTML syntax. A 0-pixel image can be embedded on a page, which is invisible to users by default. When you access the page and the image is loaded, a GET request is initiated, transmitting parameters to the server.

To use Web Tracking, follow these steps:

1. Enable the Web Tracking feature for the Logstore.

By default, the Logstore does not allow anonymous writing. Before using the Logstore, you must enable the Web Tracking feature for the Logstore.

2. Write data to the Logstore by tracking.

You can write data in any of the following ways:

- Use an HTTP Get request to write data.

```
curl --request GET 'http://${project}.${sls-host}/logstores/${logstore}/track?APIVersion=0.6.0&key1=val1&key2=val2'
```

- Embed an HTML image tag. Data is automatically written to the Logstore when a user visits the page.

```
<img src='http://${project}.${sls-host}/logstores/${logstore}/track.gif?APIVersion=0.6.0&key1=val1&key2=val2' />
or
<img src='http://${project}.${sls-host}/logstores/${logstore}/track_ua.gif?APIVersion=0.6.0&key1=val1&key2=val2' />
track_ua.gif
```

In addition to uploading custom parameters, the server also uses UserAgent and referer in the HTTP header as fields in logs.

- Use the SDK for JavaScript to write data.

```
<script type="text/javascript" src="loghub-tracking.js" async></script>

var logger = new window.Tracker('${sls-host}','${project}','${logstore}');
logger.push('customer','zhangsan');
logger.push('product','iphone 6s');
logger.push('price',5500);
logger.logger();
```

For more information, see [#unique_6](#).

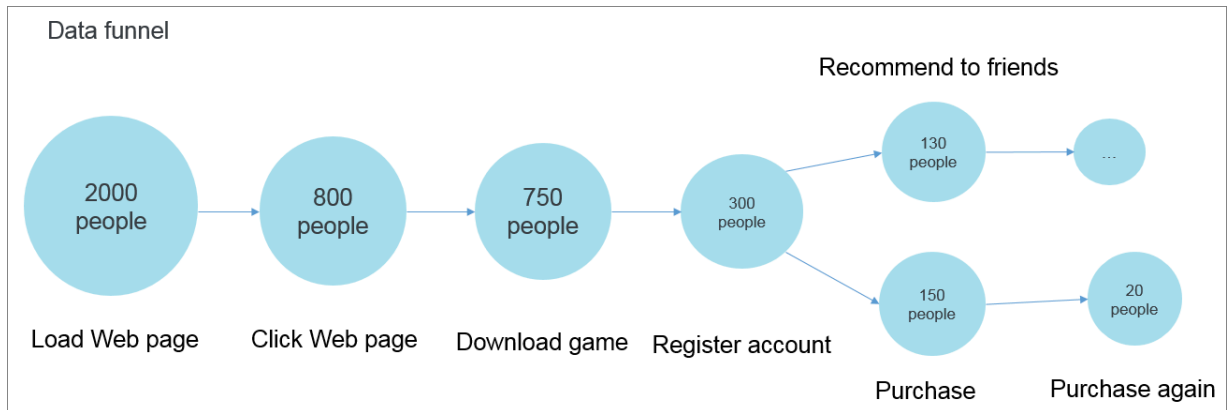
Case about multi-channel content promotion

Scenarios

Operational staff cannot wait to send new contents (such as new functions, activities, games, and articles) to users because this is the first and most important step to gain users.

Taking game release as an example. A great expense is put in game promotion, for example, 10,000 advertisements are invested. About 2,000 people load the advertisements, which accounts for 20% of the total number of advertisements. About 800

people click the advertisement, and fewer people finally download the game, register accounts, and try the game.



Therefore, obtaining the content promotion effectiveness accurately and in real time is important for the business. To reach the overall promotion goal, operational staff usually use various channels for promotion, for example:

- In-site messages, official website blogs, and homepage banners.
- SMS messages, emails, and leaflets.
- New media, such as Sina Weibo, DingTalk group, WeChat public account, Zhihu forum, and TouTiao.



Procedure

Step 1 Enable the Web Tracking feature

Create a Logstore (for example, myclick) in Log Service and enable the Web Tracking feature.

Step 2 Generate a Web Tracking tag

1. Add an identity for the article (article=1001) to be promoted in each promotion channel, and generate a Web Tracking tag as follows (taking the img tag as an example):

- In-site message channel (mailDec)

```
< img src = " http :// example . cn - hangzhou . log . aliyuncs . com / logstores / myclick / track_ua . gif ? APIVersion = 0 . 6 . 0 & from = mailDec & article = 1001 " alt = "" title = "" >
```

- Official website channel (aliyunDoc)

```
< img src = " http :// example . cn - hangzhou . log . aliyuncs . com / logstores / myclick / track_ua . gif ? APIVersion = 0 . 6 . 0 & from = aliyundoc & article = 1001 " alt = "" title = "" >
```

- Email channel (email)

```
< img src = " http :// example . cn - hangzhou . log . aliyuncs . com / logstores / myclick / track_ua . gif ? APIVersion = 0 . 6 . 0 & from = email & article = 1001 " alt = "" title = "" >
```

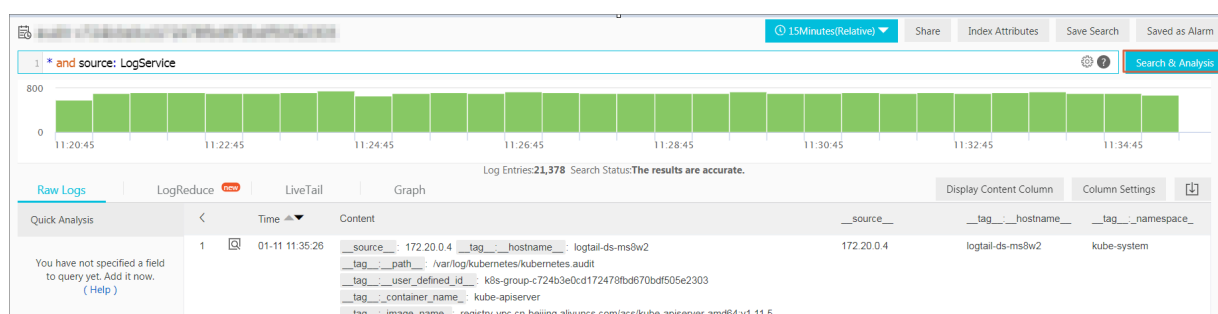
You can add other channels after the from parameter or add more parameters to be collected in the URL.

2. Put the img tag in the promotion content and release it.

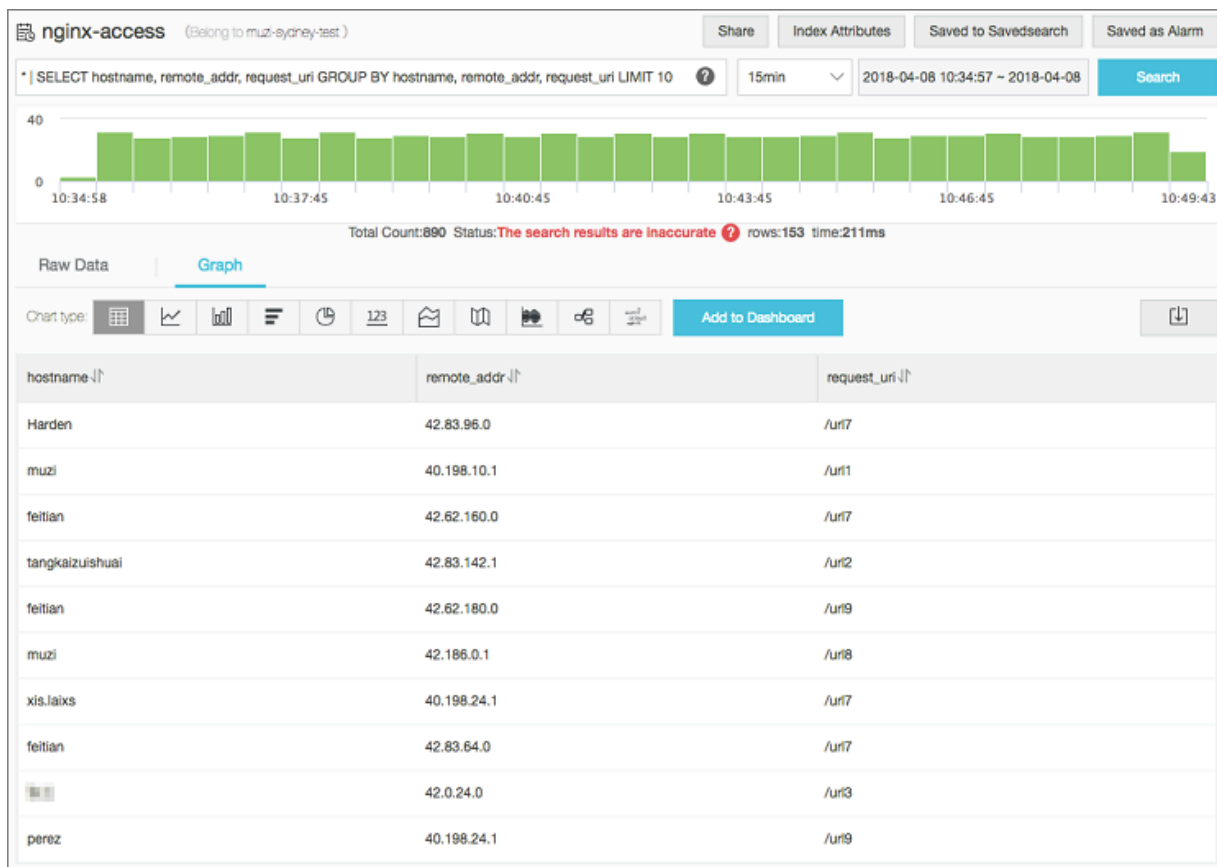
Step 3 Analyze logs

After tracking collection, you can use the [LogSearch and Analytics](#) features of Log Service to query and analyze large amounts of log data in real time. In addition to the [built-in dashboard](#), Log Service supports the interconnection methods such as [DataV](#), [Grafana](#), and Tableau to achieve result analysis visualization.

The following figure displays the collected log data. You can enter a keyword in the search box to query logs.



You can also enter an SQL statement after the query to perform real-time analysis and visualization in seconds.



1. Design the query statements.

The real-time analysis statements for user clicking or reading logs are as follows.

For more information about the fields and analysis scenarios, see [Analysis syntax](#).

- Current total traffic and page views

```
* | select count ( 1 ) as c
```

- Curve of the page views per hour

```
* | select count ( 1 ) as c , date_trunc ( ' hour ',
from_unixtime ( __time__ )) as time group by time
order by time desc limit 100000
```

- Ratios of page views in each channel

```
* | select count ( 1 ) as c , f group by f desc
```

- Devices where the page views are from

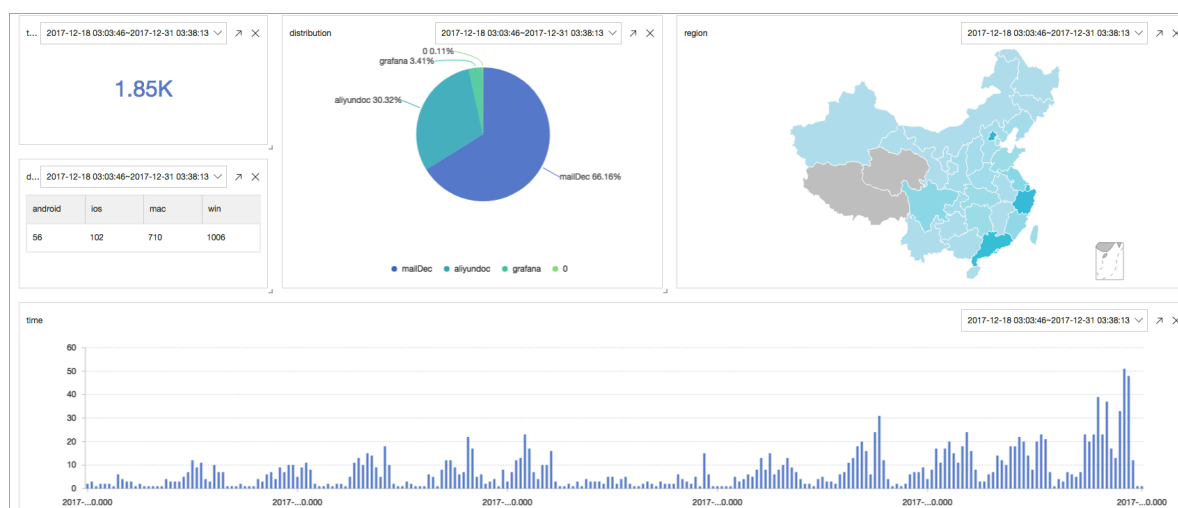
```
* | select count_if ( ua like '% Mac %') as mac ,
count_if ( ua like '% Windows %') as win , count_if (
```

```
ua like '% iPhone %') as ios , count_if ( ua like '%
Android %') as android
```

- Regions where the page views are from

```
* | select ip_to_province ( __source__ ) as province ,
count ( 1 ) as c group by province order by c
desc limit 100
```

2. Configure the real-time data to a dashboard that is refreshed in real time.



Note:

An invisible img tag records your access to this topic. You can try to find it in the source code of the page.

1.3 Build a service to read logs from mobile apps directly

In the era of mobile Internet, it is increasingly common to upload data through mobile apps. We expect that logs in mobile apps can be directly uploaded to Log Service, instead of being transferred by an app server, so that users can focus more on their business logic development.

In normal mode, the AccessKey of the Alibaba Cloud account is required when logs are written to Log Service for authentication and anti-tampering. If a mobile app accesses Log Service in this mode, you need to save your AccessKey information on the mobile client, which poses a data security risk of AccessKey disclosure. Once the AccessKey is disclosed, you must upgrade the mobile app and replace the AccessKey, which is too costly. Another approach for uploading logs from mobile clients to Log Service is to use users' app servers. However, in this mode, if the number of mobile

apps is large, the app servers must carry all the data on mobile clients. This mode has a high requirement on the server specification.

To avoid the preceding problems, Log Service provides a more secure and convenient scheme to collect mobile app logs. It uses RAM to build a direct data transfer service for mobile apps based on mobile services. In contrast to the scheme of directly using the AccessKey to access Log Service, you do not need to store the AccessKey on the app side in this scheme. This eliminates the risk of AccessKey disclosure. A temporary token with a lifecycle gives more safety. You can also configure more complex access control policies for the token, for example, limiting the access permission of the IP address segment. The cost of this scheme is low. You do not need many app servers because the mobile apps are directly connected to the cloud platform and only the control flow is sent to the app server.

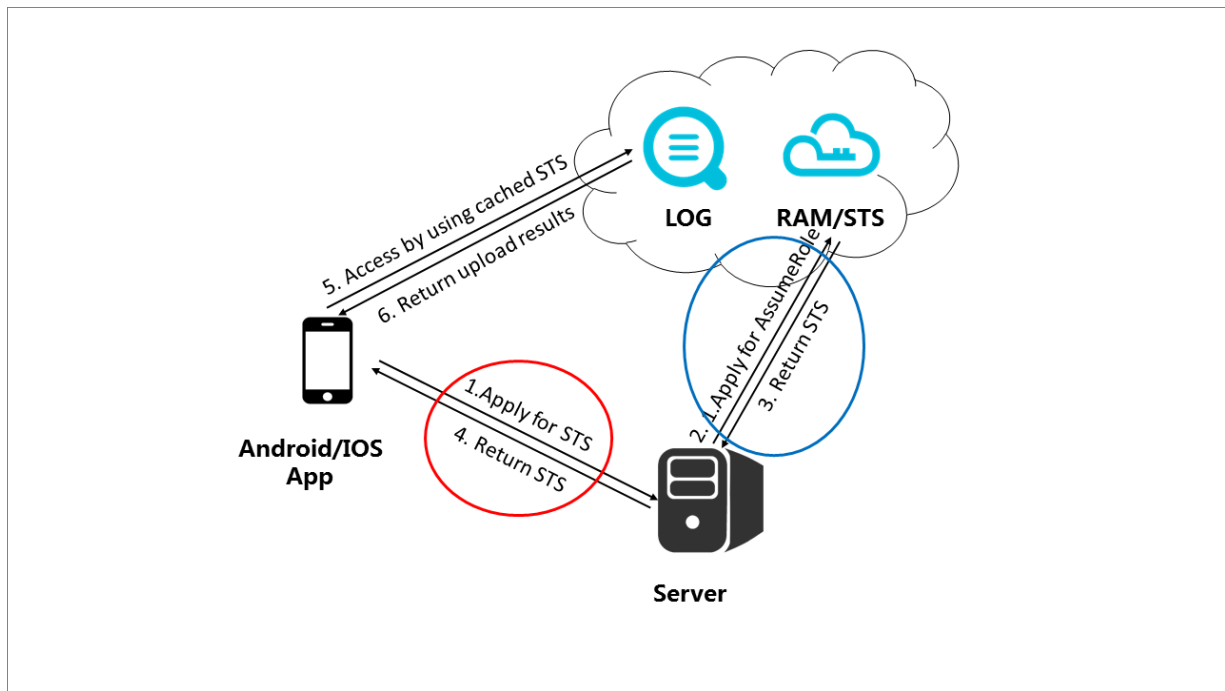
You can create a RAM role for operating Log Service and configure an app as a RAM user to assume this role, so that you can build a Log Service-based direct log transfer service for mobile apps within 30 minutes. Direct data transfer is a service that allows mobile apps to directly access Log Service, with only the control flow sent to the app server.

Advantages

Using RAM to build a Log Service-based direct data transfer service for mobile apps has the following advantages:

- This access mode is more secure, providing flexible and temporary permission assignment and authentication.
- Fewer app servers are required, reducing the cost. The mobile apps are directly connected to the cloud platform and only the control flow is sent to the app server.
- Log Service supports high concurrency. A large number of users can use the service at the same time. Large upload and download bandwidths are provided.
- Log Service supports auto scaling, providing unlimited storage space.

The following figure shows the architecture.



Description

Module	Description
Android or iOS mobile app	The app on the mobile phone of the end user and the log source.
LOG	Alibaba Cloud Log Service, responsible for storing log data uploaded by the app.
RAM or STS	Alibaba Cloud RAM, responsible for providing the user identity management and resource access control services, and generating temporary access credentials .
App server	The app background service developed for the Android or iOS apps. It manages tokens used by the app for data upload and download, and metadata information of data uploaded by users in the app.

Configuration process

1. The app applies for a temporary access credential from the app server.

To avoid the risk of information leakage, the Android or iOS app does not store the AccessKey ID and AccessKey secret. Therefore, the app must request a temporary upload credential (a token) from the app server. The token is only valid for a

certain period. For example, if a token is set to be valid for 30 minutes (which can be specified by the app server), the Android or iOS app can use this token to access Log Service within the next 30 minutes. However, 30 minutes later, the app must request a new token.

2. The app server checks the validity of the request and then returns a token to the app.
3. After obtaining the token, the mobile app can access Log Service.

This topic describes how the app server applies for the token from the RAM service and how the Android or iOS app obtains the token.

Procedure

1. Authorize a RAM role to operate Log Service.

Create a RAM role for operating Log Service and configure an app as a RAM user to assume this role. For more information, see [#unique_13](#).

After the configuration is complete, you can obtain the following information:

- AccessKey ID and AccessKey secret of the RAM user
- Resource path of the role

2. Build an app server.

This topic provides sample programs in multiple languages. The download URLs are listed at the end of this topic.

Each downloaded language package contains the configuration file `config.json`, which contains the following information:

```
{
  "AccessKeyId": "",
  "AccessKeySecret": "",
  "RoleArn": "",
  "TokenExpirationTime": "900",
  "PolicyFile": "policy/write_policy.txt"
}
```



Note:

1. **AccessKeyId**: the ID of your AccessKey.
2. **AccessKeySecret**: the secret of your AccessKey.
3. **RoleArn**: the resource path of the role.

4. **TokenExpireTime**: the expiration time of the token obtained by the Android or iOS app. The minimum value is 900, in seconds. The default value can be left unchanged.
5. **PolicyFile**: The file that lists the permissions that the token can grant. The default value does not need to be changed.

This topic describes two token files that define the most common permissions in the policy directory.

- **write_policy.txt**: specifies a token that grants the write permission for the project to this user.
- **readonly_policy.txt**: specifies a token that grants the read permission for the project to this user.

You can also design your policy file as required.

Response format

```
// Correct response
{
  " StatusCode ": 200 ,
  " AccessKeyId ":" STS . 3p *** dgagdasdg ",
  " AccessKeySecret ":" rpnw09 *** tGdrddgsR2 YrTtI ",
  " SecurityToken ":" CAES + wMIARKAAZh jH0EU0IhJM QBMjRywXq7
MQ / cjLYg80Aho 1ek0Jm63XM hr90c5s `ð`ð 3qaPer8p1Y aX1NTDiCFZ
WFkvlHf1pQ huxfKBc + mRR9KAbHUe fqH + rdjZqjTF7p 2m1wJXP8S6
k + G2MpHrUe6T YBkJ43GhhT VFMuM3BZaj Y3VjZW0XBI ODRIR1FKZj
IiEjMzMzE0 MjY0Nm5MT E4NjkxMS0L Y2xpZGSSDg SDGAGESGTE
Tq0io6c2Rr LWRLbW8vKg oUYWNzOm9z czoq0io6c2 RrLWRLbW9K
EDEXNDg5Mz AxMDcyNDY4 MThSBTI2OD QyWg9Bc3N1 bWVkuUm9sZV
VzZXJgAGoS MzMzMtQyNj Q3MzkxMTg2 OTExcglzZG stZGVtbzI =",
  " Expiration ":" 2017 - 11 - 12T07 : 49 : 09Z ",
}

// Error response
{
  " StatusCode ": 500 ,
  " ErrorCode ":" InvalidAccessKeyId . NotFound ",
  " ErrorMessage ":" Specified access key is not found
."
}
```

Correct response description: The following table describes the five variables that constitute a token.

Variable	Description
StatusCode	The result returned when the app retrieves the token. The app returns 200 if the token is successfully retrieved.
AccessKeyId	The AccessKey ID that the Android or iOS app obtains when initializing LogClient.
AccessKeySecret	The AccessKey secret that the Android or iOS app obtains when initializing LogClient.
SecurityToken	The token that the Android or iOS app uses to access Log Service.
Expiration	The time period before the token expires. The Android SDK automatically determines the validity of the token and then retrieves a new one as needed.

Error response description:

Variable	Description
StatusCode	The result returned when the app retrieves the token. The app returns 500 if the token fails to be retrieved.
ErrorCode	The error cause.
ErrorMessage	The error description.

Running method of the sample code:

For Java 1.7 or later, after downloading and decompressing the package, create a Java project, copy the dependency, code, and configuration to the project, and run the main function. By default, the program listens on port 7080 and waits for the HTTP request. Perform the operations in other languages in a similar way.

3. The mobile client constructs an HTTP request to obtain a token from the app server.

The formats of HTTP request and response are as follows:

```
Request   URL : GET   https :// localhost : 7080 /

Response :
{
  " StatusCode " : " 200 ",
  " AccessKeyI  d " : " STS . XXXXXXXXXXXX XXXXX ",
  " AccessKeyS  ecret " : "",
```

```
" SecurityToken ":"",  
" Expiration ":" 2017 - 11 - 20T08 : 23 : 15Z "  
}
```

**Note:**

All examples in this topic are used to demonstrate how to set up a server. You can implement custom development based on these examples.

Source code download

Sample code of the app server: [PHP](#), [Java](#), [Ruby](#), [Node.js](#).

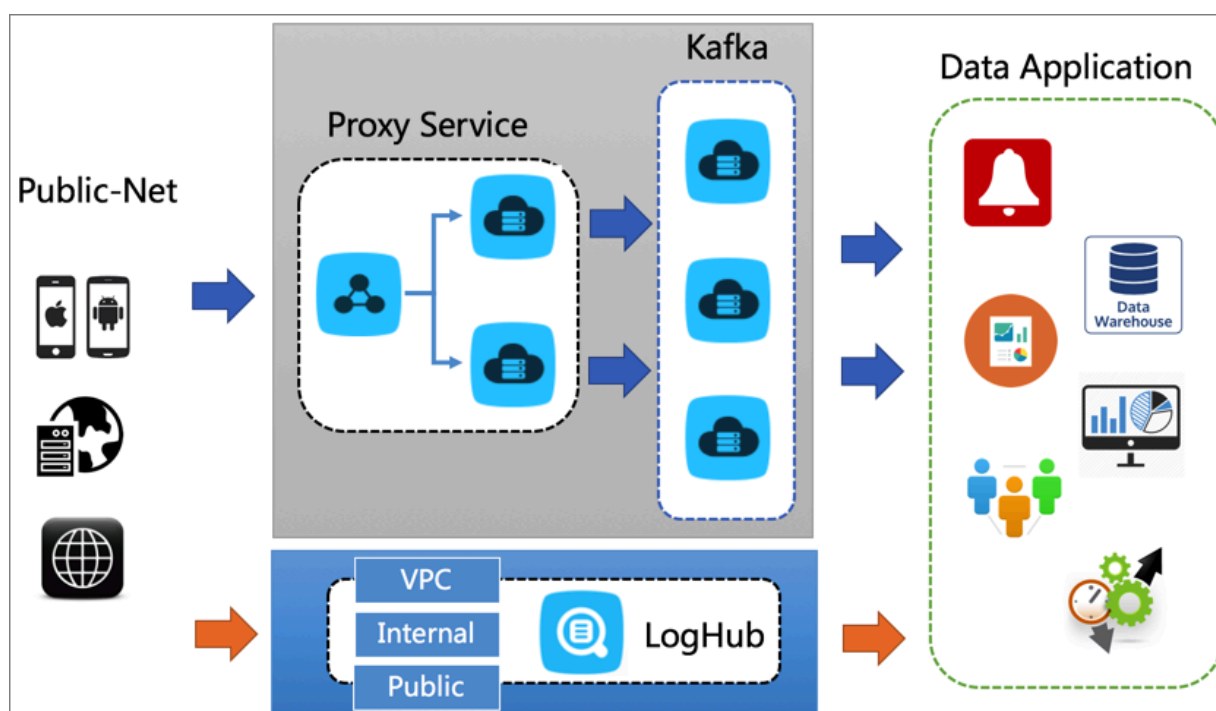
1.4 Collect data over the public network

In some scenarios, you may need to collect data from clients, such as mobile clients, HTML webpages, PCs, servers, hardware devices, and cameras, over the public network for real-time processing.

In a traditional architecture, the preceding feature can be achieved by integrating front-end servers with Kafka. Now, you can use [Log Service](#) LogHub to replace such architecture with a solution that is more reliable, cost-effective, elastic, and secure.

Scenarios

Data can be collected from clients, such as mobile clients, external servers, and webpages, over the public network. After data is collected, data applications, such as real-time computing and data warehouse, are required to store or consume data.



Solution 1: Integrate front-end servers with Kafka

Kafka does not support the RESTful protocol and is more commonly used in clusters. Therefore, you need to set up an NGINX server as a public network proxy, and then use LogStash or API to write data to Kafka through NGINX.

The following table lists the required devices.

Device	Quantity	Configuration	Purpose	Price
ECS instance	2	Single-core CPU, 2 GB memory	Front-end server, load balancing, and mutual backup	RMB 108 per unit per month
Server Load Balancer	1	Standard	Pay-as-you-go instance	RMB 14.4 per month (lease) + RMB 0.8 per GB (data traffic)
Kafka or ZooKeeper	3	Single-core CPU, 2 GB memory	Data writing and processing	RMB 108 per unit per month

Solution 2: Use LogHub

You can use mobile SDKs, Logtail, or Web Tracking JS to write data to the LogHub endpoint.

The following table lists the required devices.

Device	Purpose	Price
LogHub	Real-time data collection	Less than RMB 0.2 per GB. Click here to check the billing method.

Scenario comparison

Scenario 1: Up to 10 GB of data is collected each day, generating about 1 million write requests. The 10 GB in this example is the compressed size. The actual data volume ranges from 50 GB to 100 GB.

Solution 1 :

```

Server Load Balancer ( lease ): 0 . 02 × 24 × 30 = RMB 14 . 4
Server Load Balancer ( data traffic ): 10 × 0 . 8 × 30 = RMB 240
ECS cost : 108 × 2 = RMB 216
Kafka ECS : free if shared with other services
Total : RMB 470 . 4 per month

```

Solution 2 :

```

LogHub traffic : 10 × 0 . 2 × 30 = RMB 60
Number of LogHub requests : 0 . 12 ( assuming 1 million requests per day ) × 30 = RMB 3 . 6
Total : RMB 63 . 6 per month

```

Scenario 2: Up to 1 TB of data is collected each day, generating about 100 million write requests.

Solution 1 :

```

Server Load Balancer ( lease ): 0 . 02 × 24 × 30 = RMB 14 . 4
Server Load Balancer ( data traffic ): 1 , 000 × 0 . 8 × 30 = RMB 24 , 000
ECS cost : 108 × 2 = RMB 216
Kafka ECS : free if shared with other services
Total : RMB 24 , 230 . 4 per month

```

Solution 2 :

```

LogHub traffic : 1 , 000 × 0 . 15 × 30 = RMB 4 , 500 ( tiered pricing )
Number of LogHub requests : 0 . 12 × 100 ( assuming 100 million requests per day ) × 30 = RMB 360
Total : RMB 4 , 860 per month

```

Solution comparison

The comparison between the preceding scenarios shows that you can use LogHub to collect data from the public network at a competitive cost. Furthermore, Solution 2 outperforms Solution 1 in the following aspects:

- **Auto scaling:** LogHub supports collecting MBs or even PBs of data each day.
- **Abundant permission control options:** You can use Access Control List (ACL) to control the read and write permissions.
- **HTTPS compatibility:** Data is encrypted before transmission.
- **Log shipping at no cost:** No additional development is required for shipping logs to data warehouses.
- **Detailed data monitoring:** You can gain more information about your business.
- **Abundant SDKs for connecting to upstream and downstream systems:** Like Kafka, LogHub provides abundant SDKs to connect to downstream systems. It can be deeply integrated with Alibaba Cloud and open-source products.

For information, see the product page of [Log Service](#).

1.5 Collect data from multiple channels

Log Service LogHub supports real-time data collection and consumption. It supports more than 30 real-time collection approaches.

Data is usually collected in two different modes as described in the following table. This topic describes how to use the streaming import feature of LogHub to collect data in real time.

Mode	Advantage	Disadvantage	Example
Batch import	High throughput , focusing on historical data	Poor real-time performance	FTP-based import , uploading from OSS, mailing hard drives, and SQL- based data export to LogHub

Mode	Advantage	Disadvantage	Example
Streaming import	Real-time, what you see is what you get (WYSIWYG), focusing on real-time data	High requirements on the collection terminals	LogHub-based data collection, HTTP-based uploading, Internet of Things (IoT) data collection, and Queue

Background

"I Want Take-away" is an e-commerce website with a platform involving users, restaurants, and couriers. Users can place their take-away orders on the website, app, WeChat, or Alipay. Once receiving an order from a user, a merchant starts preparing the ordered dishes. At the same time, the system automatically notifies the nearest couriers. Then, one of the couriers accepts the order and delivers the dishes to the user.



Operation requirements

During operations, the following issues are identified:

- It is difficult to attract new users. In some other cases, a hefty advertising investment in various marketing channels, such as webpage ads and WeChat push messages, attracts some new users. However, it is difficult to evaluate the effectiveness of each channel.

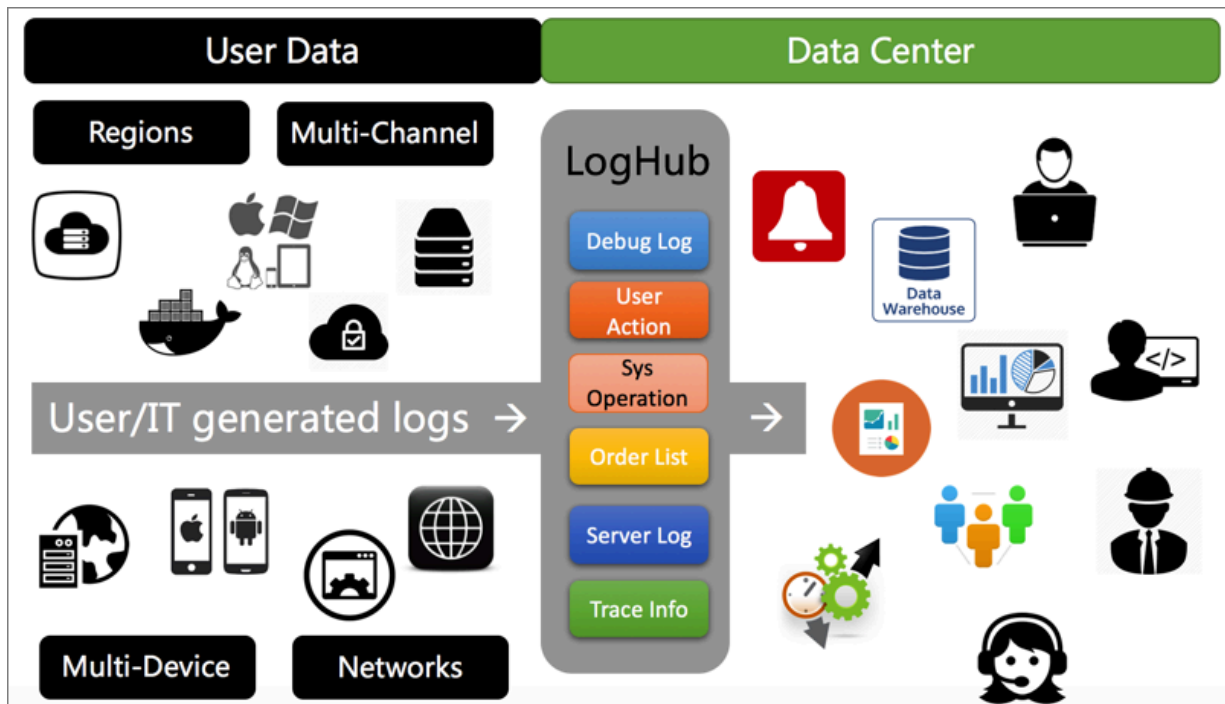
- Users often complain about the slow delivery. However, it is difficult to locate the delayed phase, for example, order taking, delivery, or preparing the order. In addition, "I Want Take-away" has no clue how to improve the situation.
- User operations teams often organize promotions, such as giving away coupons, but cannot get expected results.
- In terms of scheduling, "I Want Take-away" wants to know how to facilitate merchants to stock up food for peak hours and dispatch more couriers to a specific area.
- As for the customer service, "I Want Take-away" wants to analyze the reason why some users fail to place orders, for example, whether any inappropriate actions were performed by the users before they report the failure or whether any system errors occur.

Data collection challenges

In digital operations, the first step is to figure out how to centrally collect the distributed log data. The following challenges are encountered during the process:

- Multiple channels: advertisers and street promotions (leaflets)
- Multiple terminals: webpages, official social media accounts, mobile phones, desktop browsers, and mobile browsers
- Heterogeneous networks: Virtual Private Cloud (VPC), user-created Internet data center (IDC), and Alibaba Cloud Elastic Compute Service (ECS)
- Various development languages: Java for the core system, NGINX for front-end servers, and C++ for the back-end payment system
- Devices: merchants' devices adopting different architectures, such as x86 and ARM

In the past, you need to complete a large amount of diversified work to collect the logs distributed externally and internally for unified management. But now, you can use the collection feature of LogHub for unified access.



Unified log management and configuration

1. Create a log management project, for example, myorder.
2. Create Logstores for logs generated from different data sources, for example:
 - wechat-server (for storing WeChat server access logs)
 - wechat-app (for storing WeChat server application logs)
 - wechat-error (for storing WeChat error logs)
 - alipay-server
 - alipay-app
 - deliver-app (for storing logs related to the courier app status)
 - deliver-error (for storing delivery error logs)
 - web-click (for storing HTML5 webpage click logs)
 - server-access (for storing server-side access logs)
 - server-app (for storing application logs)
 - coupon (for storing application coupon logs)
 - pay (for storing payment logs)
 - order (for storing order logs)
3. To cleanse raw data and run Extract-Transform-Load (ETL) jobs on the raw data, create intermediate Logstores.

User promotion log collection

The following two approaches are commonly used to attract new users:

- Offer coupons upon website registration.
- Offer coupons for scanning QR codes on other channels, such as:
 - QR codes on leaflets
 - QR codes on webpages

Implementation

Define the following registration server link and generate QR codes for leaflets and webpages. When a user scans the QR code on a webpage for registration, you can identify the user as being from a specific source, and create logs accordingly.

```
http :// exampleweb  site / login ? source = 10012 & ref = kd4b
```

When receiving a request, the server generates the following logs:

```
2016 - 06 - 20   19 : 00 : 00   e41234ab34  2ef034 , 102345 , 5k4d ,  
467890
```

The logs contain the following parameters:

- time: the time of registration.
- session: the current session of the browser. It is used for behavior tracking.
- source: the source channels. For example, Campaign A is labeled as 10001, leaflets 10002, and elevator advertisements 10003.
- ref: the account of someone who recommends the user to sign up. If no one recommends the user to sign up, leave this parameter blank.
- params: other parameters.

Collection methods:

- The application generates logs to hard disks by using [Logtail](#).
- The application writes data to Log Service by using the [SDK](#).

Server-side data collection

Alipay or WeChat official account programming is a typical web-side model that generally utilizes the following four types of logs:

- **NGINX or Apache access logs:** used for monitoring and real-time statistics.

```
10 . 1 . 168 . 193 - - [ 01 / Mar / 2012 : 16 : 12 : 07 + 0800 ]
" GET / Send ? AccessKeyId = 8225105404 HTTP / 1 . 1 " 200
5 "-" " Mozilla / 5 . 0 ( X11 ; Linux i686 on x86_64 ; rv
: 10 . 0 . 2 ) Gecko / 20100101 Firefox / 10 . 0 . 2 "
```

- **NGINX or Apache error logs**

```
2016 / 04 / 18 18 : 59 : 01 [ error ] 26671 # 0 : * 20949999
connect () to unix : / tmp / fastcgi . socket failed ( 111
: Connection refused ) while connecting to upstream ,
client : 10 . 101 . 1 . 1 , server : , request : " POST /
logstores / test_log HTTP / 1 . 1 " , upstream : " fastcgi : /
unix : / tmp / fastcgi . socket : " , host : " ali - tianchi - log .
cn - hangzhou - devcommon - intranet . sls . aliyuncs . com "
```

- **Application-layer logs:** capture the event details, such as the time, location, result, delay, method, and parameters. This type of log usually ends with extended parameters.

```
{
  " time ":" 2016 - 08 - 31 14 : 00 : 04 ",
  " localAddress ":" 10 . 178 . 93 . 88 : 0 ",
  " methodName ":" load ",
  " param ":" [ 31851502 ]",
  " result ":" ....",
  " serviceName ":" com . example ",
  " startTime ":" 1472623203 994 ",
  " success ":" true ",
  " traceInfo ":" 88_1472621 445126_109 2 "
}
```

- **Application-layer error logs:** record the error details, such as the time, code line, error code, and reason.

```
2016 / 04 / 18 18 : 59 : 01 : / var / www / html / SCMC / routes
/ example . php : 329 [ thread : 1 ] errorcode : 20045 message
: extractFuncDetail failed : account_hsf_service_log
```

Implementation

- To write logs to a local file, specify regular expressions in the Logtail Config to write the logs to the specified Logstore.
- To collect logs generated in a Docker container, integrate Container Service with Log Service.
- To collect logs for Java programs, use Log4J Appender (without saving logs to hard disks) or LogHub Producer Library (for high-concurrency client-side write).

- To collect logs for C#, Python, Java, PHP, and C programs, use the corresponding SDKs.

Access to end user logs

- Mobile clients: Use the SDK for iOS or Android, or Mobile Analytics (MAN) for log access.
- ARM devices: Use the native C code for cross-compiling.
- Merchant platform devices: Use the corresponding SDK on x86 servers for log access. Use the native C code on ARM devices for cross-compiling.

Collection of user behavior data for desktop or mobile websites

The user behavior can be divided into the following two types:

- User behavior that has interaction with the back-end server, such as placing an order, logging on, and logging out.
- User behavior that has no interaction with the back-end server, including sending requests that are processed directly at the front end, such as scrolling or closing a page.

Implementation

- To collect data of user behavior that has interaction with the back-end server, refer to the implementation approaches for server-side data collection.
- To collect data of user behavior that has no interaction with the back-end server, use Tracking Pixel or JS Library.

Server log O&M

Example:

- System logs

```
Aug 31 11 : 07 : 24 zhouqi - mac WeChat [ 9676 ]:  
setupHotkey_listenning_event NSEvent : type = KeyDown loc  
=( 0 , 703 ) time = 115959 . 8 flags = 0 win = 0x0 winNum =  
7041 ctxt = 0x0 chars = " u " unmodchars = " u " repeat = 0  
keyCode = 32
```

- Application debug logs

```
__FILE__ : build / release64 / sls / shennong_worker /  
ShardDataIndexManager.cpp  
__LEVEL__ : WARNING  
__LINE__ : 238  
__THREAD__ : 31502  
offset : 8161034535 52
```

```

saved_curs or : 1469780553 885742676
seek count : 62900
seek data redo
log : pangu :// localclust er / redo_data / 41 / example /
2016_08_30 / 250_147255 5483
user_curso r : 1469780553 885689973

```

- Trace logs

```

[ 2013 - 07 - 13 10 : 28 : 12 . 772518 ] [ DEBUG ] [ 26064 ]
__TRACE_ID __ : 6613539512 01 __item__ : [ Class : Function ]
_end__ request_id : 1734117 user_id : 124 context :.....

```

Implementation

For more information, see the implementation approaches for server-side data collection.

Data collection in different network environments

LogHub provides access points in each region with the following three access approaches:

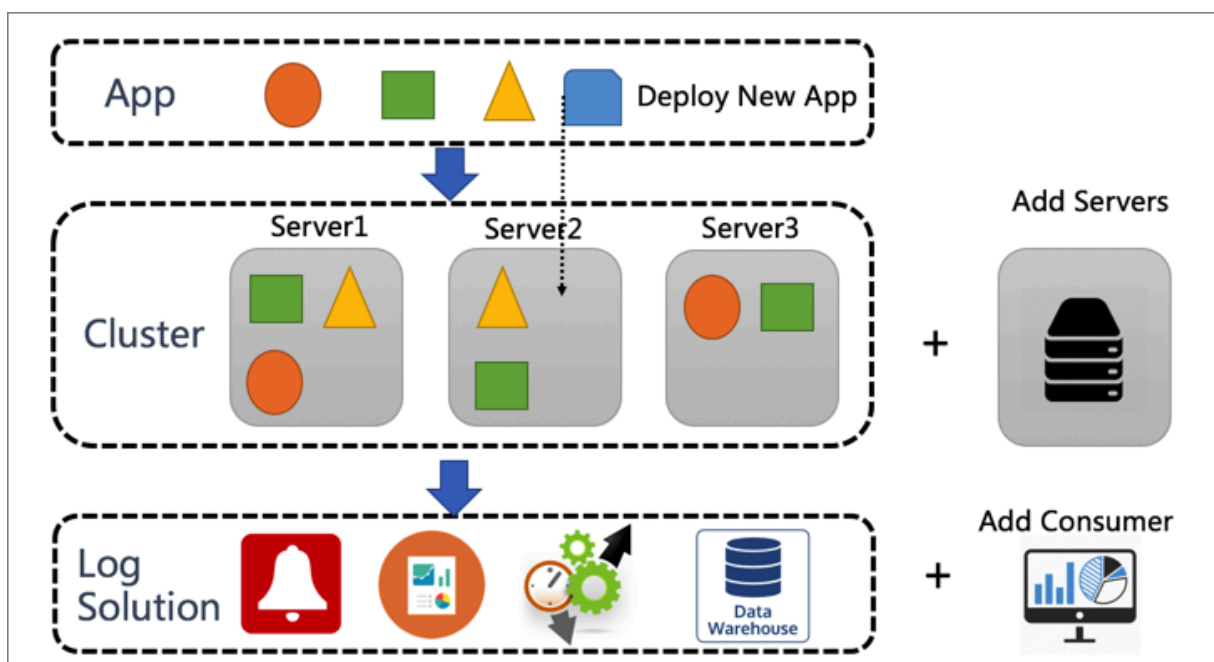
- Internal network (classic network): accessible to services within the current region . We recommend that you use this access approach for its optimal link quality.
- Public network (classic network): accessible without any limits. The transmission speed varies depending on the link quality. We recommend that you use HTTPS to ensure secure transmission of data.
- Private network (VPC): accessible to services in the current region through VPCs.

1.6 Manage logs

Log management

Here is a typical scenario: A server or container stores a large amount of application log data generated in different directories.

- Developers deploy new applications and take them offline.
- The server can scale in or out as needed. Specifically, the servers scale out during the peak periods and scale in during the slack periods.
- The log data is to be queried, monitored, and warehoused depending on the different and ever-changing requirements.



Challenges

1. Fast application deployment and go-live, and a growing number of log types

Each application can generate access, operations, logic, and error logs. When more applications are added and the dependence exists between applications, the volume of logs explodes.

The following table lists the different types of logs collected for a takeaway website.

Category	Application	Log name
Web	NGINX	wechat-nginx (for storing WeChat server NGINX logs)
Web error	NGINX	alipay-nginx (for storing Alipay server NGINX logs)
	NGINX	server-access (for storing server-side access logs)
	NGINX error	alipay-nginx (for storing NGINX error logs)
	NGINX error	...
Web app	Tomcat	alipay-app (for storing Alipay server application logic)
	Tomcat	...

Category	Application	Log name
App	Mobile app	deliver-app (for storing logs related to the courier app status)
App error	Mobile app	deliver-error (for storing delivery error logs)
Web	HTML5	web-click (for storing HTML5 webpage click logs)
Server	Server	Server-side internal logic logs
Syslog	Server	Server system logs

2. Log consumption for different purposes

For example, access logs can be used for billing, and for users to download.

Operations logs can be queried by a database administrator (DBA). These logs require Business Intelligence (BI) analysis and full-link monitoring.

3. Environment changes

With the incredibly fast evolution of the Internet, you need to adapt to the ever-changing business and environment in the real world.

- Application server scale-out
- Servers as machines
- New application deployment
- New log consumers

A perfect management architecture requires:

- A well-defined architecture with low cost
- A stable and highly reliable, preferably unattended mechanism (which, for example, allows for automatically scaling in or out servers as needed)
- Standardized application deployment without complicated configuration
- Easy compliance with log processing requirements

Log Service solution

Log Service LogHub uses Logtail to collect logs, involving the following concepts:

- Project: a management container.

- **Logstore:** indicates a log source.
- **Machine group:** indicates the directory and format of logs.
- **Config:** indicates the path to logs.

The relationships between these concepts are as follows:

- A project includes multiple Logstores, machine groups, and Configs. Different projects can be used for different business purposes.
- An application can have multiple types of logs. Each type of log has a Logstore and a fixed directory (with the same Config).

```
app --> logstore1 , logstore2 , logstore3
app --> config1 , config2 , config3
```

- A single application can be deployed on multiple machine groups, and multiple applications can be deployed on a single machine group.

```
app --> machineGro up1 , machineGro up2
machineGro up1 --> app1 , app2 , app3
```

- The collection directories defined in the Logtail Configs are applied to the corresponding machine groups, and collected into any Logstore.

```
config1 * machineGro up1 --> Logstore1
config1 * machineGro up2 --> logstore1
config2 * machineGro up1 --> logstore2
```

Advantages

- **Convenient:** The web console or SDK is provided for batch management.
- **Large-scale:** Millions of machines and millions of applications can be managed.
- **Real-time:** The collection configuration takes effect within minutes.
- **Elastic:**
 - The machine ID feature is used for auto scaling of servers.
 - LogHub supports auto scaling.
- **Stable and reliable:** No human intervention is required.

- Abundant query capabilities such as real-time computing, offline analysis, and indexing in log processing:
 - **LogHub:** real-time collection and consumption. LogHub uses more than 30 approaches to collect large amounts of data for real-time downstream consumption.
 - **LogShipper:** stable and reliable log shipping. It ships data from LogHub to storage services such as Object Storage Service (OSS), MaxCompute, and Table Store, for storage and big data analysis.
 - **LogSearch:** real-time data indexing and querying. It allows for centralized log query without caring about where active server logs are located.

2 Query and analysis

2.1 Display query and analysis results on multiple pages

Returning too much data for a log query may affect the result displaying speed and query experience. You can use API operations to perform paged queries so that the query result is displayed on multiple pages. This helps limit the volume of data returned each time.

The paged query feature that Log Service provides not only allows you to read raw logs by page but also allows you to write the execution result of SQL statements to local devices by page. Developers can read logs by page through the SDK or Command-Line Interface (CLI) provided by Log Service.

Paging methods

The query and analysis features of Log Service allow you to query logs based on keywords and analyze the query result by running the corresponding SQL statements. The `GetLogstoreLogs` operation is provided by Log Service to query logs. With this operation, you can query raw logs by keyword, compute data by running SQL statements, and obtain the analysis result. The query feature and the analysis feature implemented in query and analysis statements use different paging methods.

- **Query statements:** Search for records in raw logs by keyword. You can use the `offset` and `lines` parameters of the operation to obtain all required records and display them on multiple pages.
- **Analysis statements:** Analyze the query result by running the corresponding SQL statements and obtain the analysis result. You can get the paged result by running the SQL `LIMIT` statement.

Display the query result by page

Use the `offset` and `lines` parameters of the `GetLogStoreLogs` operation to get the paged query result.

- **offset:** the line from which the logs are to be read.
- **lines:** the number of lines to be read for the current request. The maximum value of this parameter is 100. If you set a value greater than 100, only 100 lines are returned.

The value of the offset parameter increases for reading logs by page. When the value reaches a certain number, zero lines are returned and the progress becomes complete. In this case, all data has been read.

Sample code of querying logs by page

- Pseudocode:

```

offset = 0 // Indicates that logs are read from
the zeroth line .
lines = 100 // Indicates that 100 lines are read
at a time .
query = " status : 200 "// Indicates that the logs
where the value of the status field contains 200
are queried . while True :
    response = get_logsto re_logs ( query , offset , lines
) // The response to the read request .
    process ( response ) // Use
the custom logic to process the query result .
    If response . get_count () == 0 && response . is_complet
e ()
        Stop reading logs and exit the current loop
    .
    else
        offset += 100 // The
value of the offset parameter increases by 100 . The
next 100 lines are to be read .

```

- Python code:

For more information about the sample code, see [#unique_24](#).

```

endpoint = '# The endpoint that matches the region
of the project created in the preceding step .
accessKeyI d = '# Your Alibaba Cloud AccessKey ID .
accessKey = '# Your Alibaba Cloud AccessKey secret .
project = '# The name of the project created in
the preceding step .
logstore = '# The name of the Logstore created
in the preceding step .
client = LogClient ( endpoint , accessKeyI d , accessKey )
topic = ""
query = " index "
From = int ( time . time () ) - 600
To = int ( time . time () )
log_line = 100
offset = 0
while True :
    res4 = Nonefor retry_time in range ( 0 , 3 ):
        req4 = GetLogsReq_uest ( project , logstore , From
, To , topic , query , log_line , offset , False )
        res4 = client . get_logs ( req4 )
        if res4 isnotNonea nd res4 . is_complet ed ():
            break
        time . sleep ( 1 )
        offset += 100if res4 . is_complet ed () && res4 .
get_count () == 0 :
            break ;
    if res4 isnotNone :

```



```
res4 . log_print () # Display the execution
result .
```

- **Java code:**

For more information about the sample code, see [#unique_25](#).

```
int log_offset = 0 ;
int log_line = 100 ;// Indicates that 100 lines
are read at a time . The maximum value of this
parameter is 100 . If you need to read more than
100 lines at a time , use the offset parameter
. Note that the offset and lines parameters are
available to keyword - based queries instead of SQL -
based queries . To read more than 100 lines at a
time during a SQL - based query , use the LIMIT
statement .
while ( true ) {
    GetLogsResponse res4 = null ;
    // For each log offset , 100 lines are
read at a time . If a read operation fails , a
maximum of 3 retries are allowed .
    for ( int retry_time = 0 ; retry_time < 3 ;
retry_time ++ ) {
        GetLogsRequest req4 = new GetLogsRequest
( project , logstore , from , to , topic , query , log_offset
,
        log_line , false );
        res4 = client . GetLogs ( req4 );

        if ( res4 != null && res4 . IsCompleted () )
        {
            break ;
        }
        Thread . sleep ( 200 );
    }
    System . out . println ( " Read log count : " +
String . valueOf ( res4 . GetCount () ));
    log_offset += log_line ;
    if ( res4 . IsCompleted () && res4 . GetCount ()
== 0 ) {
        break ;
    }
}
```

Display the analysis result by page

The offset and lines parameters of the GetLogStoreLogs operation are not available to SQL-based analysis. If you traverse the offset parameter based on the preceding method to display the analysis result by page, the SQL-based analysis result is the same for each execution. If you try to obtain all the computing result at a time, the following issues may occur because of the large-sized data in the result:

- The latency of transmitting large-sized data is high.

- The memory usage is high. Storing large-sized data in the analysis result occupies the storage space of the client memory.

To resolve these issues in displaying the SQL-based analysis result by page, Log Service provides the standard SQL [LIMIT statement](#) that uses the following syntax:

```
limit Offset , Line
```

- **Offset:** the line from which the result is to be read.
- **Line:** the number of lines to be read. The value of Line does not have an upper limit . However, reading too many lines at a time may result in a high network latency and an adverse impact on the processing speed of the client.

Assume that 2,000 lines are returned for the analysis statement `* | selectcount t (1) , url group by url` . You can specify that all data is to be read at four intervals, with 500 lines being read at a time.

```
* | selectcount t ( 1 ) , url group by url limit 0 ,
500
* | selectcount t ( 1 ) , url group by url limit 500 ,
500
* | selectcount t ( 1 ) , url group by url limit 1000 ,
500
* | selectcount t ( 1 ) , url group by url limit 1500 ,
500
```

Sample code of displaying the SQL-based analysis result by page

- **Pseudocode:**

```
offset = 0 // Indicates that logs are read from
the zeroth line .
lines = 500 // Indicates that 500 lines are read
at a time .
query = "* | select count ( 1 ) , url group by url
limit " whileTrue :
    real_query = query + offset + "," + lines
    response = get_logsto re_logs ( real_query ) // The
response to the read request .
    process ( response ) // Use the
custom logic to process the analysis result .
    If response . get_count () == 0
        Stop reading logs and exit the current loop
    .
    else
        offset += 500 // The value
of the offset parameter increases by 500 . The next
500 lines are to be read .
```

- **Python code:**

```
endpoint = ''# The endpoint that matches the region
of the project created in the preceding step .
```

```

accessKeyId = '# Your Alibaba Cloud AccessKey ID .
accessKey = '# Your Alibaba Cloud AccessKey secret .
project = '# The name of the project created in
the preceding step .
logstore = '# The name of the Logstore created
in the preceding step .
client = LogClient ( endpoint , accessKeyId , accessKey )
topic = ""
origin_query = "* | select count ( 1 ) , url group
by url limit "
From = int ( time . time () ) - 600
To = int ( time . time () )
log_line = 100
offset = 0
while True :
    res4 = None
    query = origin_query + str ( offset ) + " , " + str
( log_line )
    for retry_time in range ( 0 , 3 ):
        req4 = GetLogsRequest ( project , logstore , From
, To , topic , query )
        res4 = client . get_logs ( req4 )
        if res4 is not None and res4 . is_completed ():
            break
        time . sleep ( 1 )
        offset += 100
    if res4 . is_completed () && res4 .
get_count () == 0 :
        break ;
    if res4 is not None :
        res4 . log_print () # Display the execution
result .

```

• Java code:

```

int log_offset = 0 ;
int log_line = 500 ;
String origin_query = "* | select count ( 1 ) ,
url group by url limit " while ( true ) {
    GetLogsResponse res4 = null ;
    // For each log offset , 500 lines are
read at a time . If a read operation fails , a
maximum of 3 retries are allowed .
    query = origin_query + log_offset + " , " +
log_line ;
    for ( int retry_time = 0 ; retry_time < 3 ;
retry_time ++ ) {
        GetLogsRequest req4 = new GetLogsRequest
( project , logstore , from , to , topic , query ) ;
        res4 = client . GetLogs ( req4 ) ;

        if ( res4 != null && res4 . IsCompleted () )
        {
            break ;
        }
        Thread . sleep ( 200 ) ;
    }
    System . out . println ( " Read log count : " +
String . valueOf ( res4 . GetCount () ) ) ;
    log_offset += log_line ;
    if ( res4 . GetCount () == 0 ) {
        break ;
    }
}

```

```
}
```

2.2 Analyze sales system logs

Bills are the core data of e-commerce companies and the outcome of a series of marketing and promotional activities. This data contains a lot of valuable information. With the information, you can define user profiles, providing guidelines for future marketing plans. The billing data can also serve as a popularity indicator, providing suggestions for subsequent stocking options.

The billing data is stored as logs in Alibaba Cloud Log Service. With a computing capacity of processing hundreds of millions of log entries per second, Log Service supports high-speed queries and SQL-based statistics. This topic uses several examples to describe how to mine useful information.

The following is a complete bill that contains goods information (name and price), deal information (final price, payment method, and discount information), and buyer information (membership information):

```
__source__ : 10 . 164 . 232 . 105      __topic__ : bonus_disc
ount : category : men ' s clothing    commodity : Every
day discount autumn and winter teenager velvet
and thickened skinny jeans men ' s winter slim
pants commodity_id : 443 discount : member_dis count
: member_level : nomember_p oint : memberid : mobile
: pay_transaction_id : 060f0e0d08 0e0b050603 07010c0f02
09010e0e01 0c0a060500 0606050b0c 0400 pay_with : alipay
real_price : 52 . 0 suggest_pr ice : 52 . 0
```

Statistical analysis

Before query and analysis, enable and set indexes. For more information, see [#unique_27](#).

1. View the percentage of the sales of each category of products to the total sales.

```
*| select count ( 1 ) as pv , category group by category
limit 100
```

2. View the sales trends of different women's clothes.

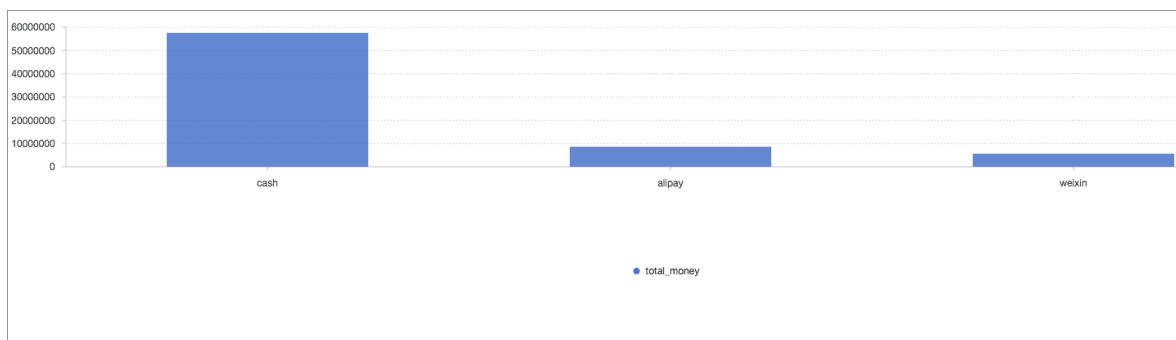
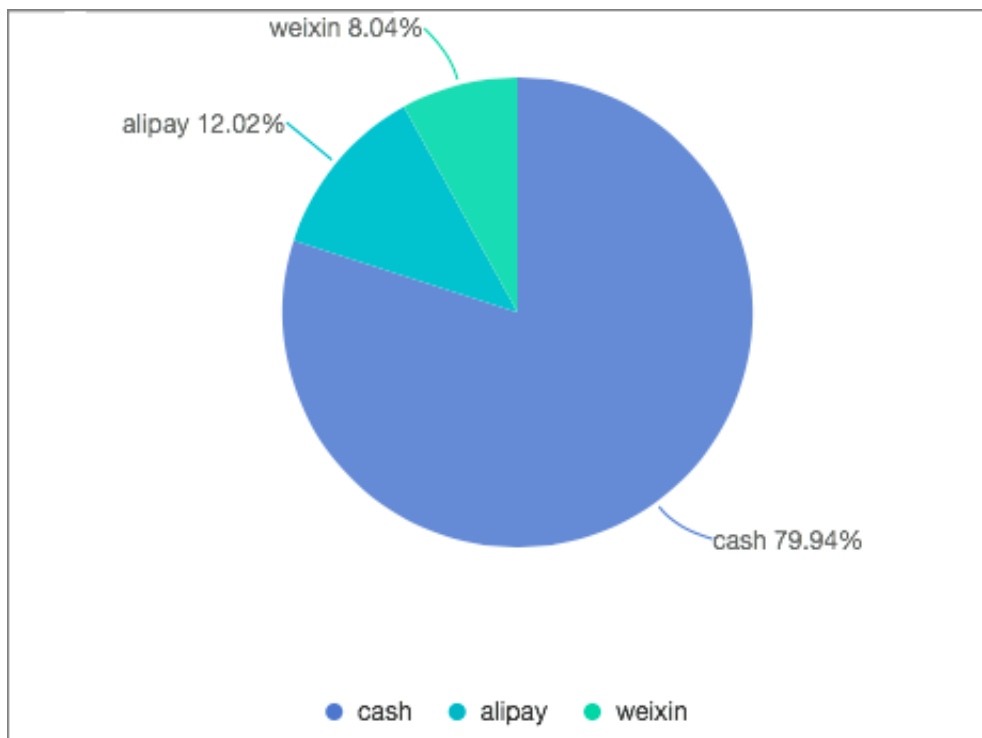
```
category : women ' s clothing | select count ( 1 ) as
deals , commodity
```

```
group by commodity order by deals desc limit 20
```

3. View share and turnover of different payment methods.

```
* | select count ( 1 ) as deals , pay_with group by
pay_with order by deals desc limit 20

* | select sum ( real_price ) as total_mone y , pay_with
group by pay_with order by total_mone y desc
limit 20
```



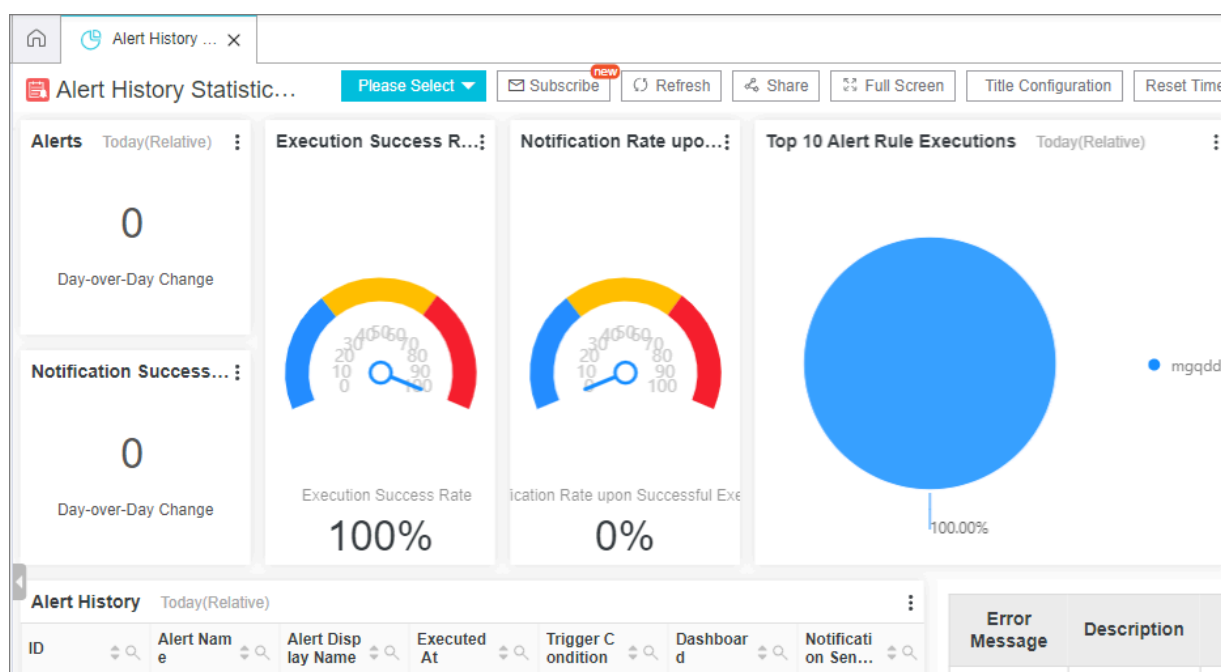
2.3 Analyze website logs

Website access logs are vital to webmasters. The access information of a website includes the page views (PVs), unique visitors (UVs), regions of visitors who access the website, and top 10 pages with most visits. Application logs are essential to application developers. Analysis and optimization of the SQL ORDER BY and LIMIT statements can improve application quality. O&M engineers can monitor server logs

and locate exceptions. By monitoring logs in real time, engineers can obtain the server response time changes in the last hour and check whether the traffic is normal when a request is sent from a client to a machine for load balancing. Engineers can also view the monitoring data and obtain key information through the dashboard.

To meet the requirements of the preceding scenarios, Log Service provides a one-stop solution for log data collection and analysis. The LogSearch and Analytics features allow you to analyze logs in real time by using the query statements and SQL-92 statements. Log Service supports multiple visualization methods, including the built-in dashboards and open-source visualization tools such as [DataV](#), Grafana, Tableau through Java Database Connectivity (JDBC), and QuickBI.

Log Service provides data analysis charts to display the real-time log query and analysis results. In addition, Log Service provides dashboards for displaying the log data analysis result in different scenarios.



[Click here for a trial](#)

Username: sls_reader1@*****

Password: pnX-32m-MHH-xbm

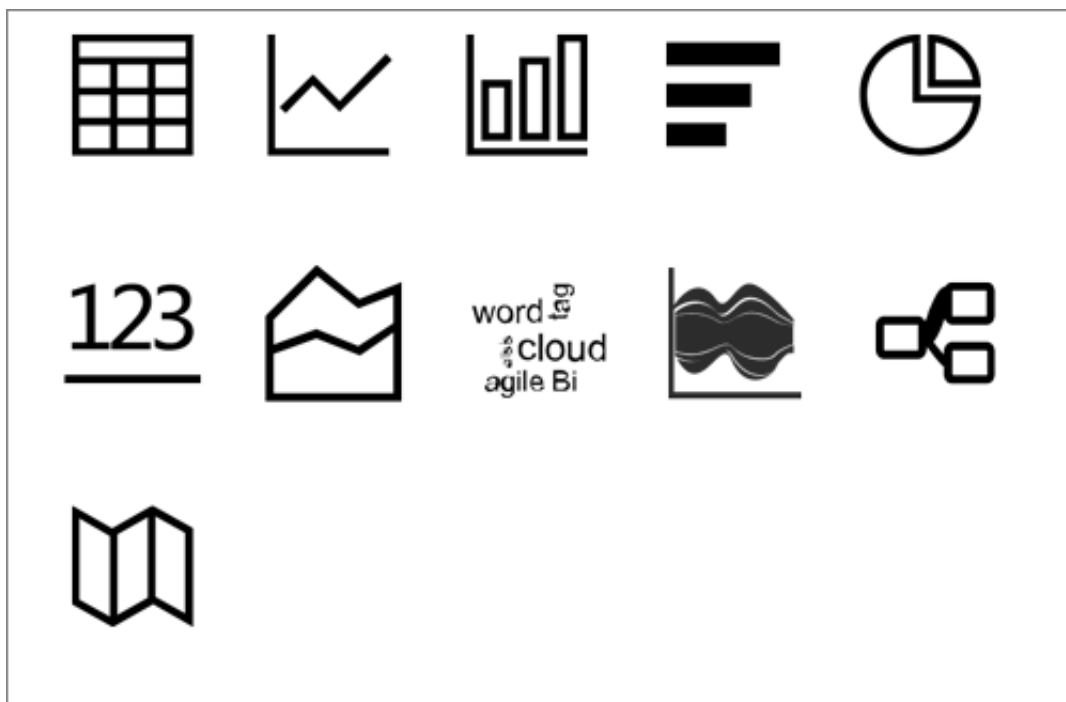
Features

- No need to define charts in advance: You can apply any computing method and any filter condition to logs generated in any time period. The corresponding chart appears within seconds.

- **Interactive analysis:** Seamless switching is supported between charts and raw logs.
- **Scenario-specific solution:** You can view the analysis dashboard through the data import wizard without complex configurations.

Chart types

The following figure shows the chart types that are supported by the visualization feature of Log Service.



Process and architecture

1. **Collect data.** Log Service supports multiple methods to collect logs, such as by using clients, webpages, protocols, and SDKs or APIs (for mobile apps and games). All collection methods are implemented based on RESTful APIs. You can also implement new collection methods by using APIs and SDKs.
2. **Set indexes and use query and analysis statements** to query and analyze logs.
3. **Visually display data.** Log Service provides RESTful APIs. You can use an appropriate method to visualize your log data. The following section describes the visualization and dashboard features of Log Service.



Examples

This section describes the typical use cases of different chart types. Before query and analysis, enable and set indexes. For more information, see [#unique_27](#).

1. Table

A table, as the most common visual representation of data, consists of one or more groups of cells. It is used to display numbers and other items for quick reference and analysis. The items in a table are organized as rows and columns. The first row

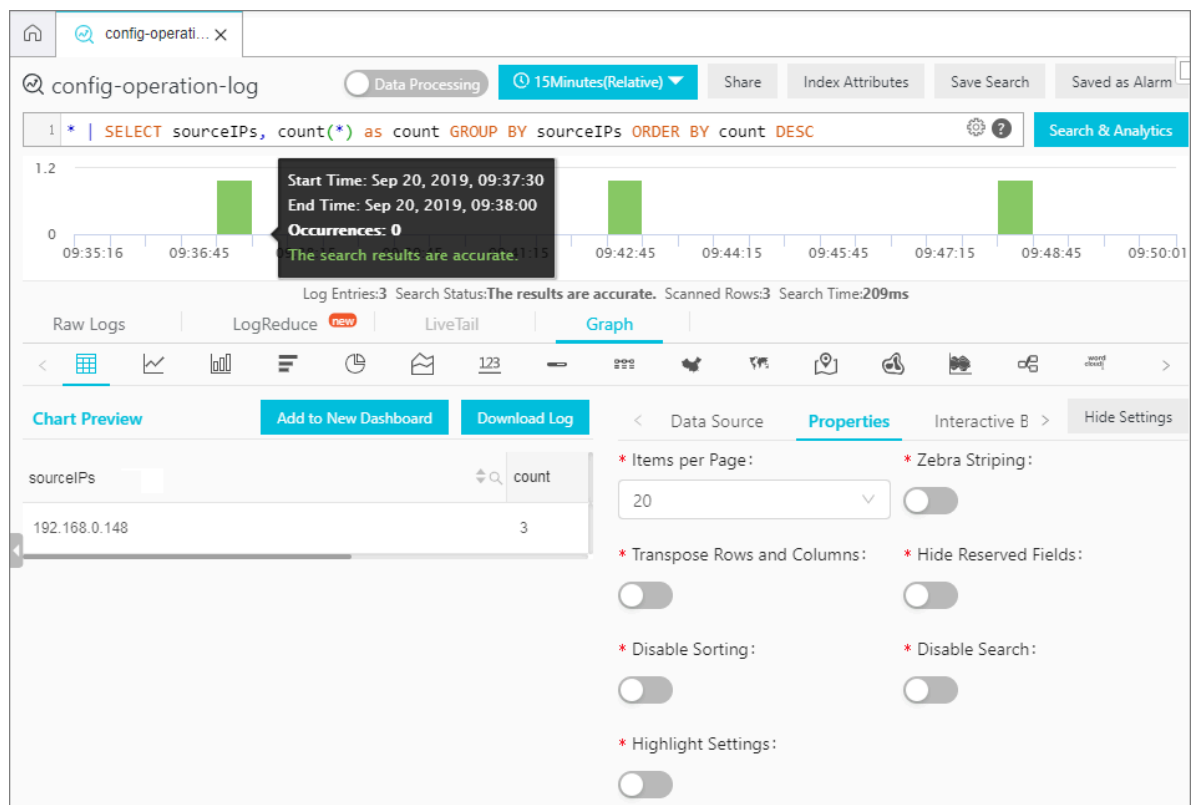
of a table is called the header, indicating the content and meaning of each column in the table.

In Log Service, the result returned by the query and analysis statements is displayed in tables by default.

Run the following statement to view the distribution of source IP addresses in the current time range in descending order:

```
* | SELECT sourceIPs , count (*) as count GROUP BY sourceIPs ORDER BY count DESC
```

The following figure shows the result table. You can click the sort icon in the header to sort a column in a certain order.



2. Line chart

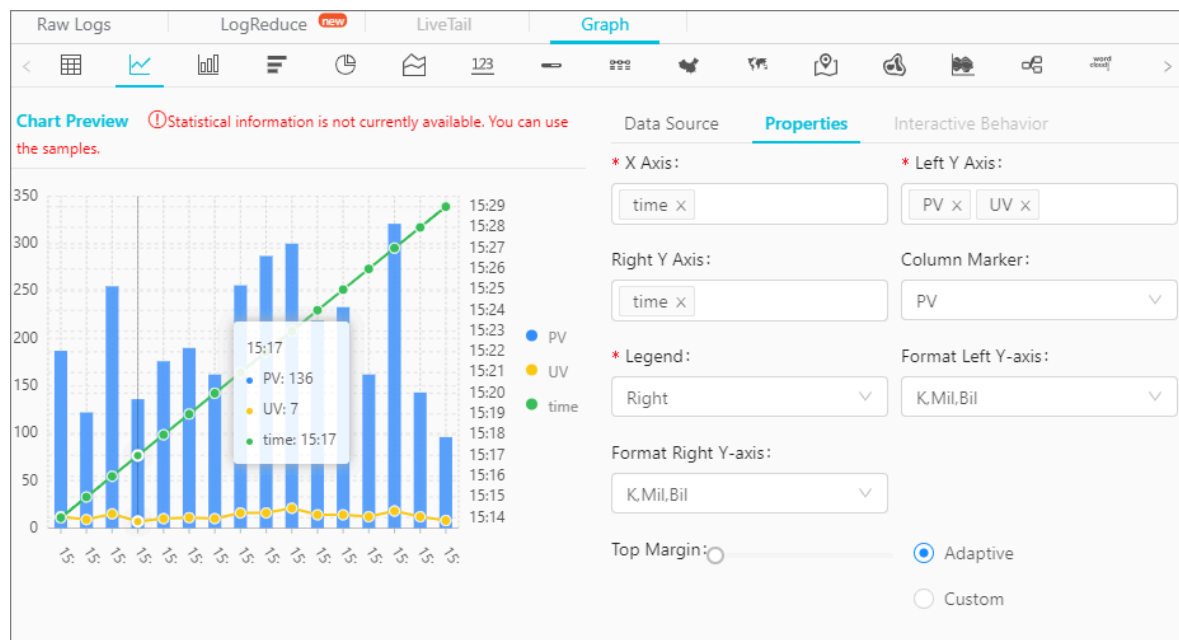
A line chart analyzes trends. It is typically used to indicate the changes of a group of data based on an ordered data type (successive time intervals in most cases) for analyzing the trend of data changes.

Run the following statement to analyze the changes of PVs, UVs, and average response time in the last 15 minutes:

```
* | select date_format ( from_unixtime ( __time__ - __time__ % 60 ), '% H :% i :% S ') as minutes , approx_dis
```

```
tinct ( remote_add r ) as uv , count ( 1 ) as pv , avg (
request_t i me ) as avg_group by minutes order by
minutes asc limit 100000
```

Select minutes for X Axis, PV and UV for Left Y Axis, avg for Right Y Axis, and UV for Column Marker. The following figure shows a sample line chart.



3. Column chart

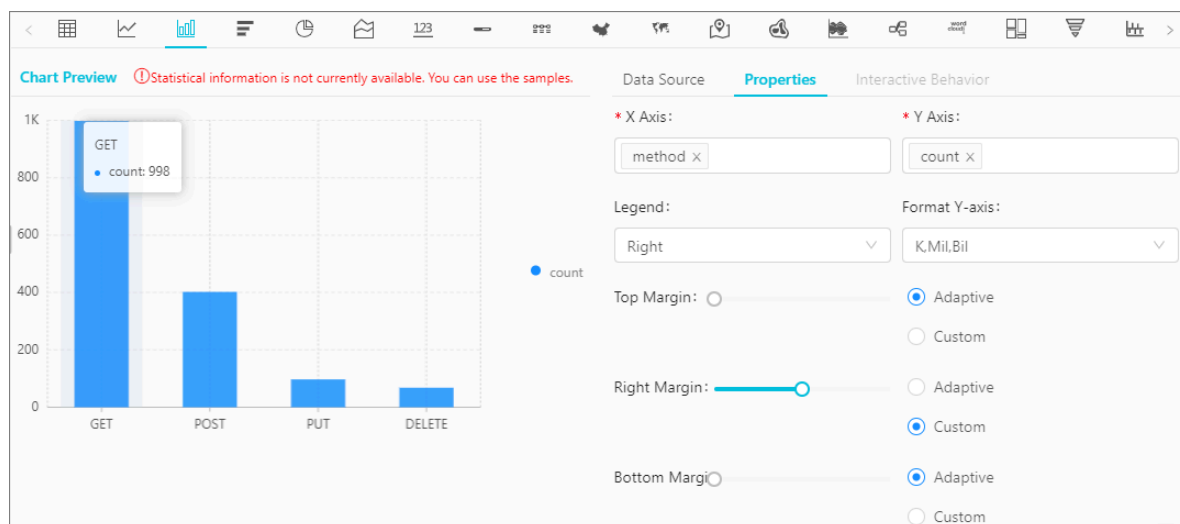
A column chart uses vertical or horizontal columns to compare the numeric data among different types. A line chart describes the ordered data, while a column

chart describes different types of data and counts the number in each data type.

For more information about the line chart, see [#unique_30](#).

Run the following statement to analyze the number of visits for each `http_referer` in the last 15 minutes:

```
* | select http_referer, count ( 1 ) as count group by http_referer
```

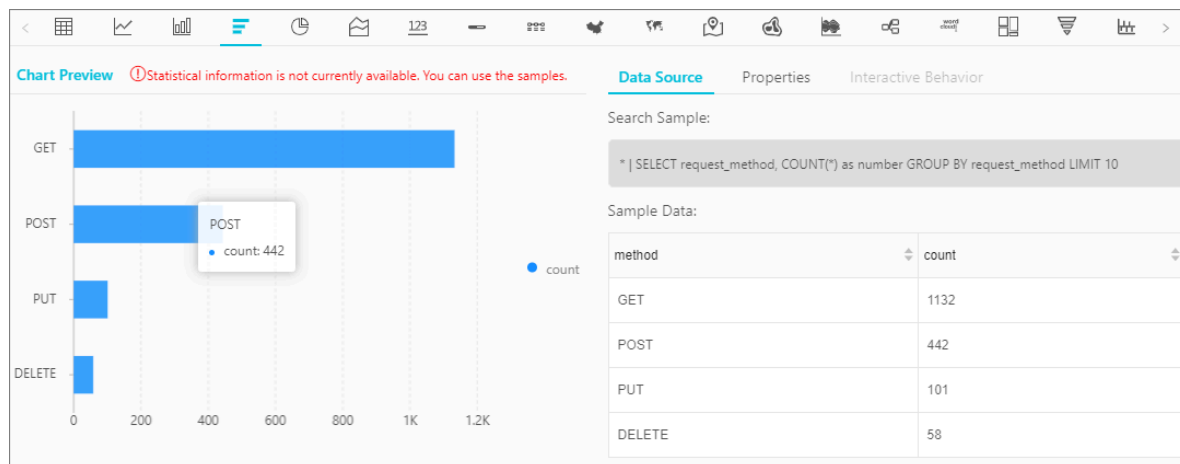


4. Bar chart

A bar chart, also known as a horizontal bar chart, is suitable for analyzing the ranking of different categories.

Run the following statement to analyze the top 10 `request_uri` with the most visits in the last 15 minutes:

```
* | select request_uri, count ( 1 ) as count group by request_uri order by count desc limit 10
```



5. Pie chart

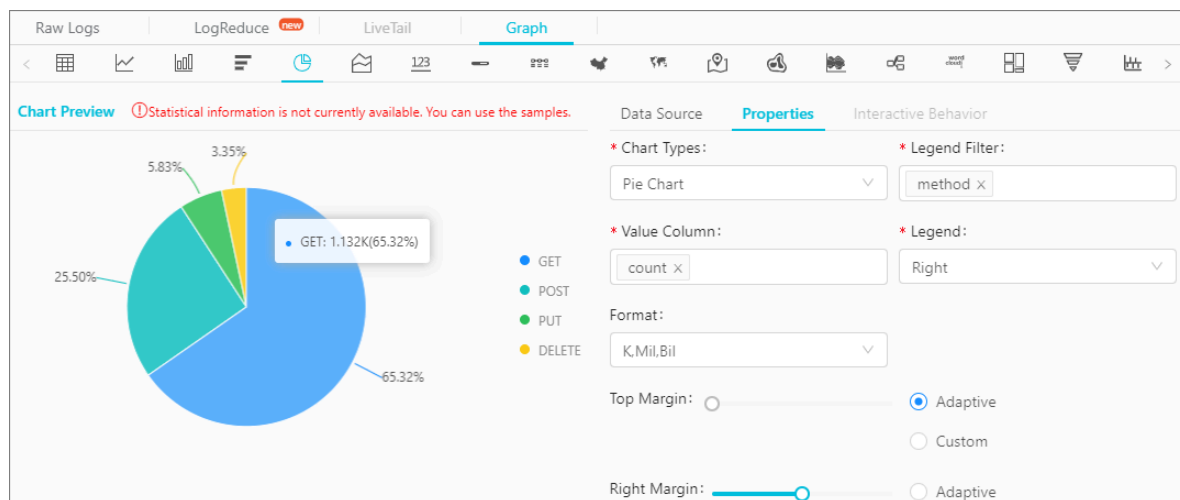
A pie chart is used to indicate the ratios of different data types and compare different data types based on the arc length. A pie chart is divided into multiple sectors based on the percentages of various data types. The entire chart indicates

the sum of data. Each sector (arc-shaped) indicates the ratio of a data type to the sum. The sum of percentages in all sectors is equal to 100%.

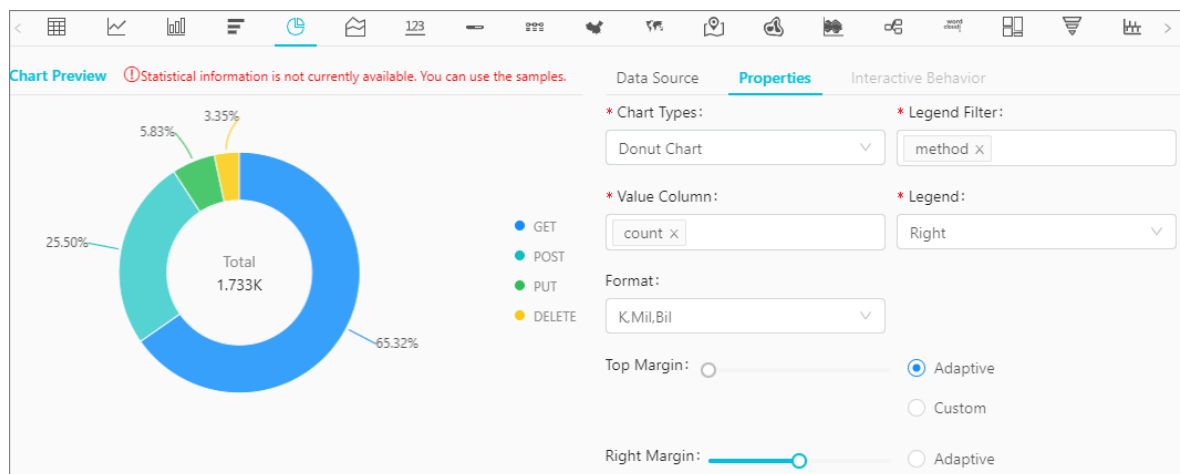
Run the following statement to analyze the visit distribution of different request URIs in the last 15 minutes:

```
* | select requestURI as uri , count ( 1 ) as c group
  by uri limit 10
```

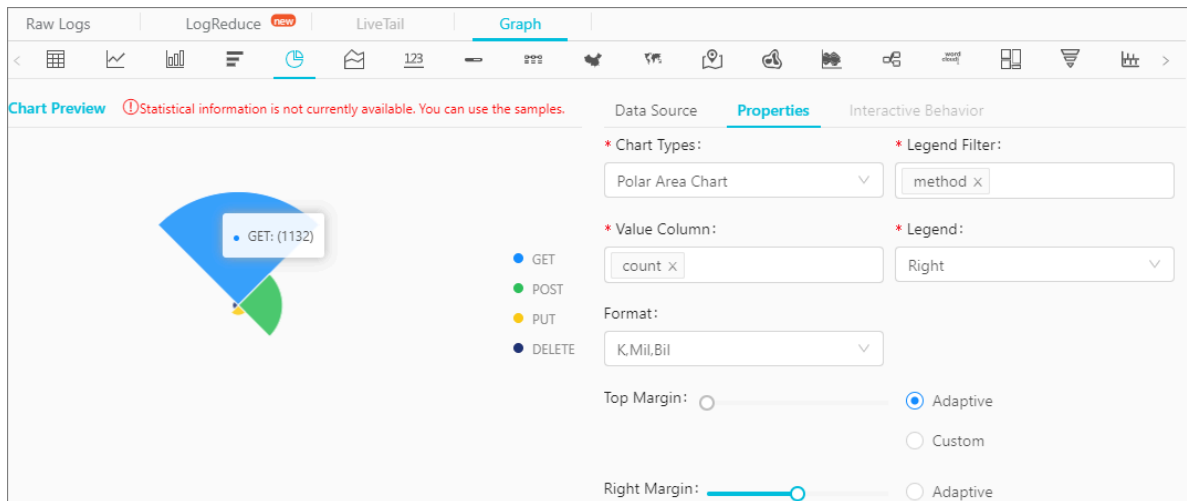
Pie chart:



Donut chart:



Polar area chart:

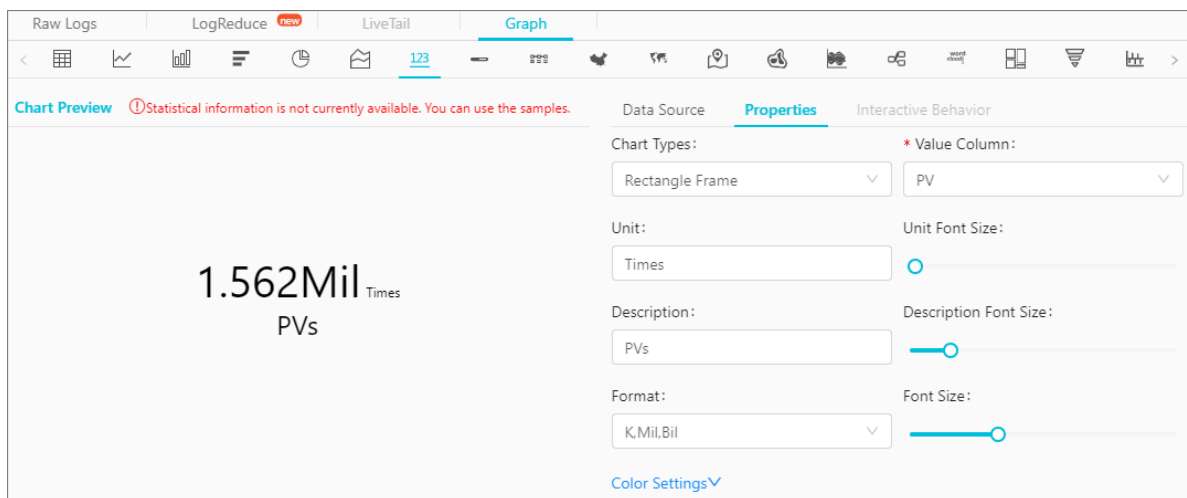


6. Single value chart

A single value chart, as the easiest and most intuitive display type of data, shows the data on a point clearly and intuitively, and is generally used to indicate the key information on a time point.

Run the following statement to analyze the PVs in the last 15 minutes:

```
* | select count ( 1 ) as PV
```



7. Area chart

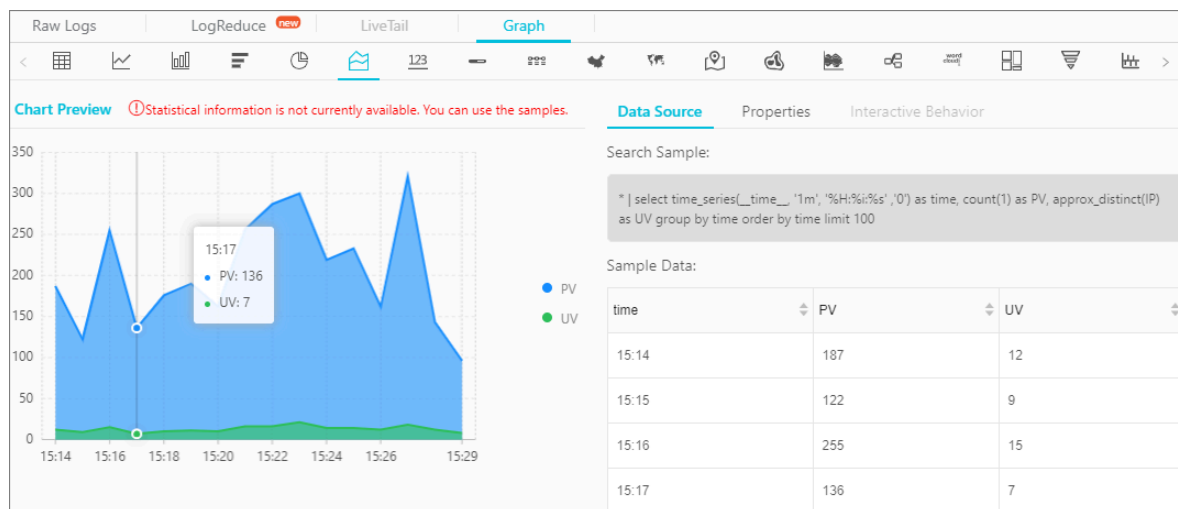
Based on a line chart, an area chart fills the section between a line and the axis with color. The filled section is an area block and the color highlights the trend.

Run the following statement to collect statistics about the PVs of the IP address 10

. 0 . XX . XX on the last day:

```
remote_add r : 10 . 0 . XX . XX | select date_format (
date_trunc (' hour ', __time__ ), '% m -% d % H :% i ') as
```

```
time , count ( 1 ) as PV group by time order by
time limit 1000
```



8. Map chart

You can add color blocks and marks to a map to display geographic data. Log Service provides three kinds of maps: Map of China, World Map, and AMap. Among them, AMap offers the scatter chart and heat map.

You can use the `remote_addr` parameter in the statements to make three types of map charts. Run the following statements to display the top 10 regions with the most visits:

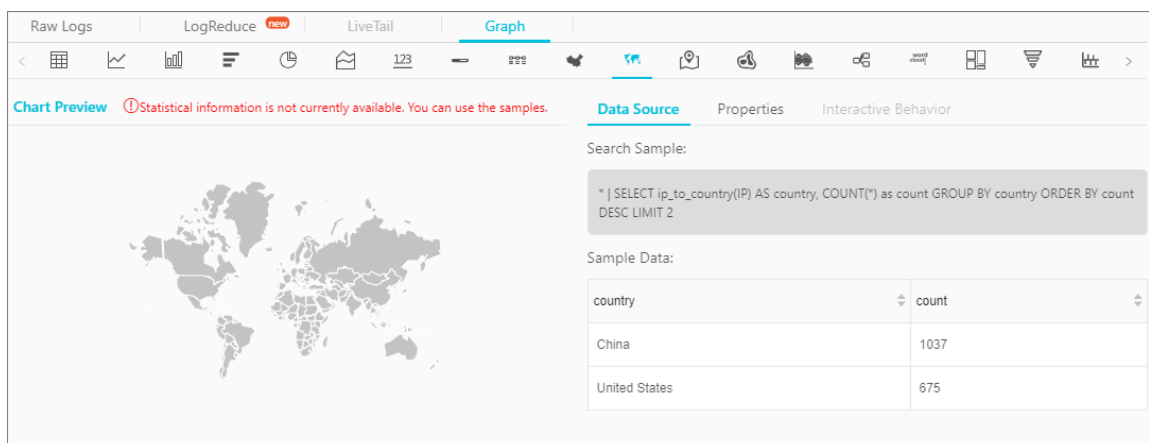
• Map of China

```
* | select ip_to_province ( remote_addr ) as address ,
count ( 1 ) as count group by address order by
count desc limit 10
```



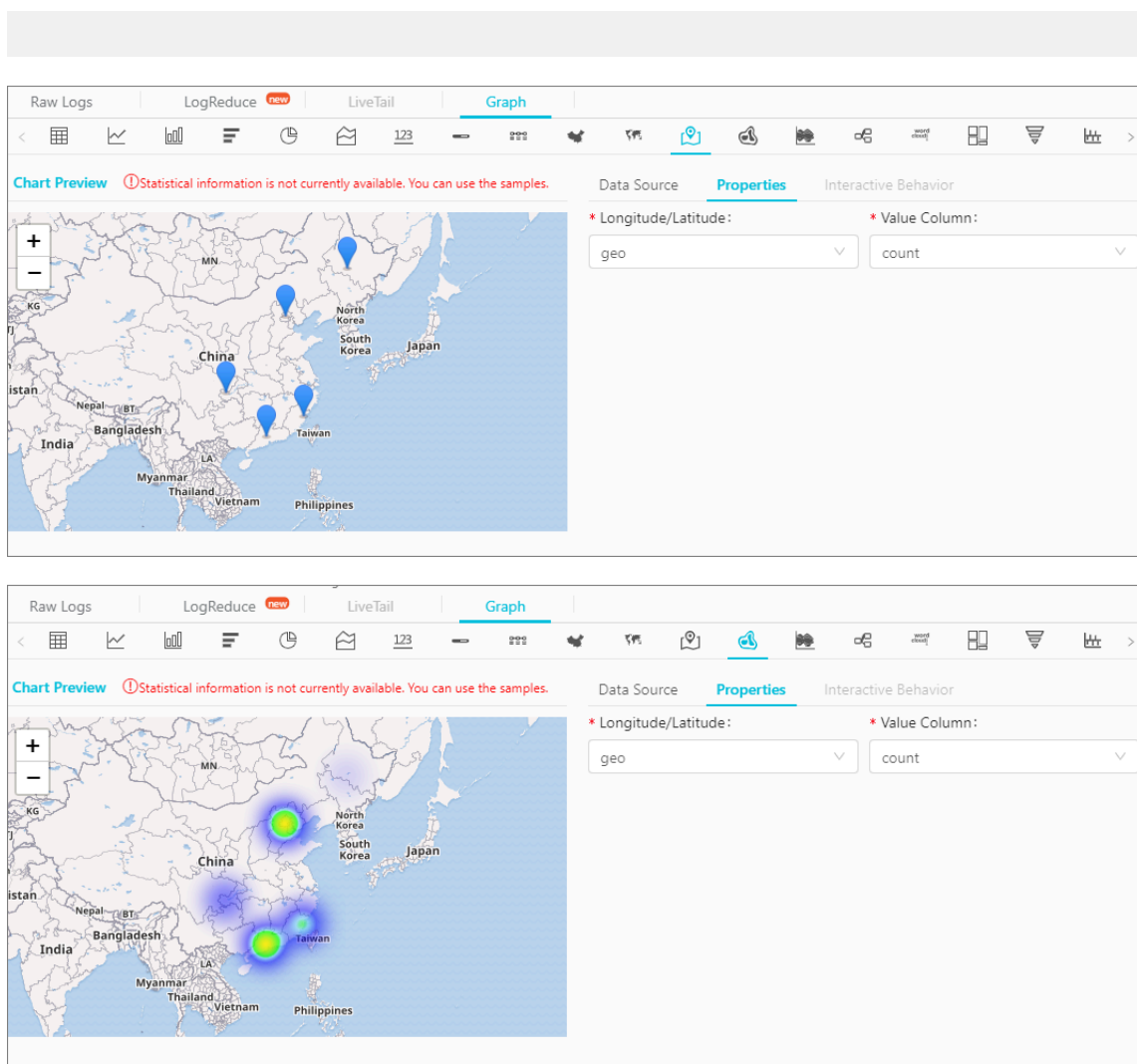
• World Map

```
* | select ip_to_country ( remote_add r ) as address ,
count ( 1 ) as count group by address order by
count desc limit 10
```



• AMap

```
* | select ip_to_geo ( remote_add r ) as address , count
( 1 ) as count group by address order by count
desc limit 10
```

9. Flow chart

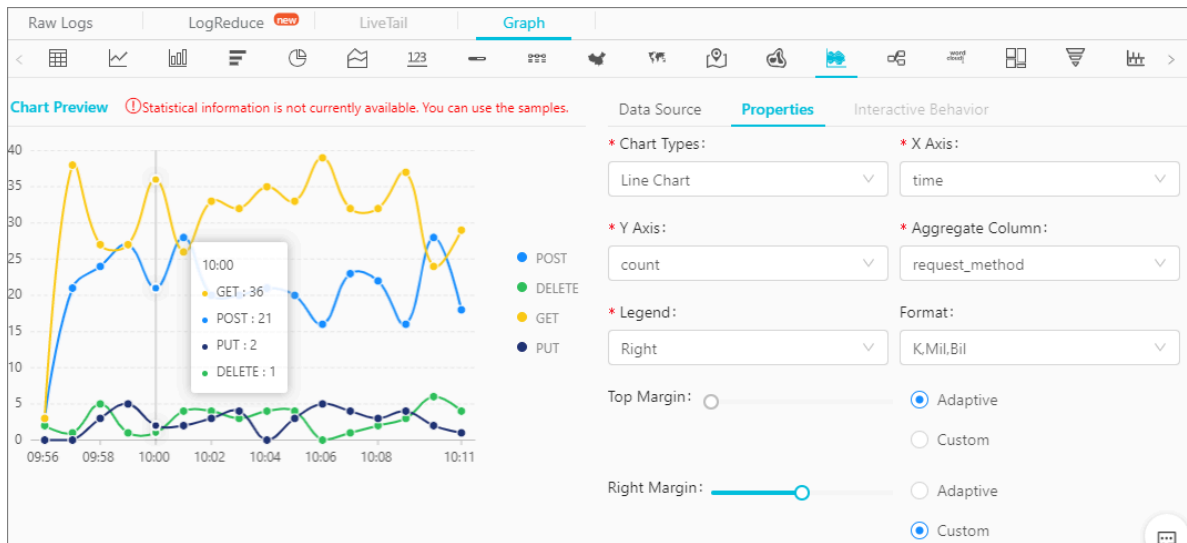
The banded branches with different colors indicate different types of information. The band width indicates the corresponding numeric value. In addition, the centralized time attribute of the original data maps to the X-axis, which forms a three-dimensional relationship.

Run the following statement to display the trends of the requests sent through different methods in the last 15 minutes:

```
* | select date_format ( from_unixtime ( __time__ -
__time__ % 60 ), '% H :% i :% S ' ) as minute , count ( 1 ) as
```

```
c , request_method group by minute , request_method
order by minute asc limit 100000
```

Select minute for X Axis, c for Y Axis, and request_method for Aggregate Column.

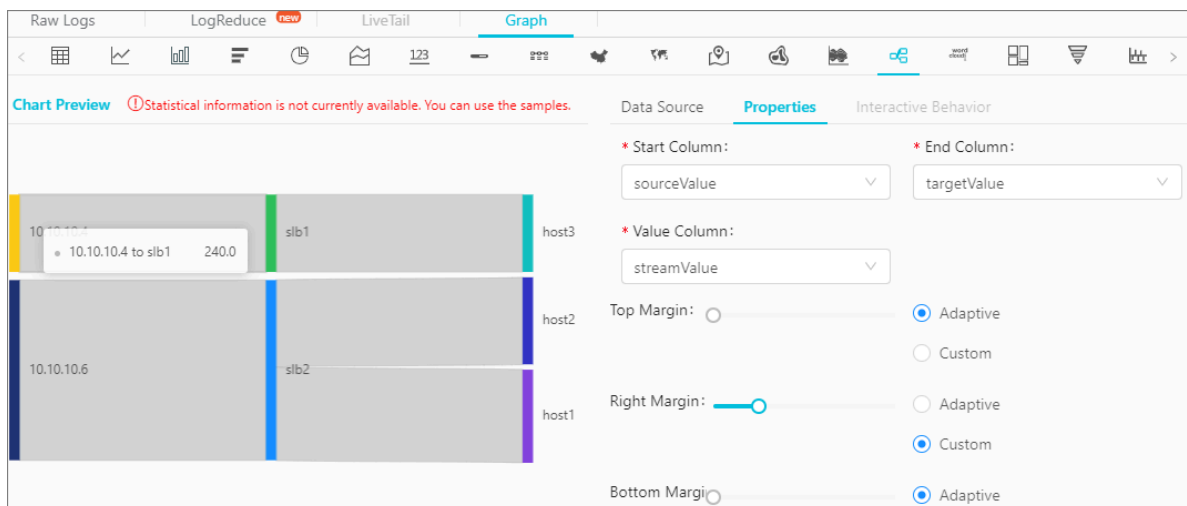


10. Sankey diagram

A Sankey diagram is a specific type of flow chart. It is used to describe the flow from one set of values to another. Sankey diagrams are applicable to scenarios such as network flow data. Generally, a Sankey diagram contains three sets of values: source, target, and value. The source and target describes the edge relationship between nodes, and value describes the relationship between source and target.

Run the following statement to analyze the flow data in a load balancing scenario:

```
* | select sourceValue e , targetValue e , streamValue e
   group by sourceValue e , targetValue e , streamValue e
   order by streamValue e
```



11. Word cloud

A word cloud visualizes text data. It is a cloud-like and colored image composed of words. It can be used to display a large amount of text data. The importance of each word is shown with its font size or color. This allows you to quickly perceive the weight of some keywords.

Run the following statement to analyze the visits to different request URIs in the last 15 minutes:

```
* | select requestURI as uri , count ( 1 ) as c group
  by uri limit 100
```



Add charts to a dashboard

All charts obtained through the query and analysis statement can be saved in a dashboard. Then, you can adjust the layout of these charts to get a comprehensive dashboard.

You can click Add to New Dashboard to create a dashboard. After creating a dashboard, you can open the dashboard based on tags to view data in real time.

2.4 Analyze vehicle track logs

Taxi companies record the details of each trip, including the time when a passenger gets in and out, latitude and longitude, distance of the trip, payment method, payment amount, and tax amount. Detailed data greatly facilitates the operation of taxi companies. For example, with the data, the companies can shorten the running intervals in peak hours or dispatch more vehicles to the areas where more people need taxis. With the help of the data, passengers can get a timely response and drivers can have higher incomes. This improves the efficiency of the whole society.

Taxi companies store the trip logs to Alibaba Cloud Log Service, and pick out useful information with the help of reliable storage and rapid statistical calculations. This topic describes how taxi companies mine useful information from the data stored in Alibaba Cloud Log Service.

Sample data:

```
RatecodeID : 1 VendorID : 2 __source_ _ : 11 . 164 . 232 . 105
__topic__ : dropoff_latitude : 40 . 7439956665 03906
dropoff_longitude : - 73 . 9835052490 23437extra : 0
fare_amount : 9 improvement_surcharge : 0 . 3
mta_tax : 0 . 5 passenger_count : 2 payment_type : 1
pickup_latitude : 40 . 7614669799 80469
pickup_longitude : - 73 . 9624633789 0625 store_and_fwd_flag : N
tip_amount : 1 . 96 tolls_amount : 0
total_amount : 11 . 76 tpep_dropoff_datetime : 2016 - 02 - 14 11 : 03 : 13
tpep_dropoff_time : 1455418993
tpep_pickup_datetime : 2016 - 02 - 14 10 : 53 : 57
tpep_pickup_time : 1455418437 trip_distance : 2 . 02
```

	时间戳	RatecodeID	VendorID	dropoff_latitude	dropoff_longitude	pickup_latitude	pickup_longitude	total_amount	tpep_dropoff_datetime	tpep_pickup_datetime	trip_distance
1	08-31 20:05:53	1	2	40.758163452148438	-73.99129480603844	40.704853057891328	-74.015822546386719	24.3	2016-02-14 14:49:31	2016-02-14 14:17:32	4.85
2	08-31 20:05:53	1	1	40.708518881933594	-74.017219543457031	40.718776702888869	-74.000670016113281	11.15	2016-02-14 14:27:32	2016-02-14 14:17:32	1.50
3	08-31 20:05:53	3	1	40.690466205078125	-74.17755889825781	40.741939544877734	-74.003875732421875	185.95	2016-02-14 14:47:29	2016-02-14 14:17:32	19.60
4	08-31 20:05:53	1	1	40.7266845703125	-73.990483774414063	40.7138892578125	-74.009140014848437	10.8	2016-02-14 14:29:52	2016-02-14 14:17:32	2.00
5	08-31 20:05:53	1	1	40.719020843509859	-73.999252319339538	40.71150088892578	-74.00995639883281	11.76	2016-02-14 14:29:43	2016-02-14 14:17:32	1.00
6	08-31 20:05:53	1	2	40.744297027387891	-73.985468003417969	40.764141082763672	-73.973802294921875	13.8	2016-02-14 14:37:21	2016-02-14 14:17:31	2.11
7	08-31 20:05:53	1	2	40.763916015625	-73.956244323730469	40.770534515388859	-73.948394775398525	7.56	2016-02-14 14:22:37	2016-02-14 14:17:31	.96
8	08-31 20:05:53	1	2	40.750503540039063	-73.989883422851562	40.763473510742188	-73.986414184570313	9.8	2016-02-14 14:29:07	2016-02-14 14:17:31	1.28
9	08-31 20:05:53	1	1	40.748451232910156	-73.988792419433594	40.71227835791916	-73.985231628477969	17.8	2016-02-14 14:43:07	2016-02-14 14:17:31	2.70
10	08-31 20:05:53	1	1	40.720909118852344	-74.00088715820313	40.742031097412109	-73.98307037335156	18.8	2016-02-14 14:30:50	2016-02-14 14:17:31	1.80

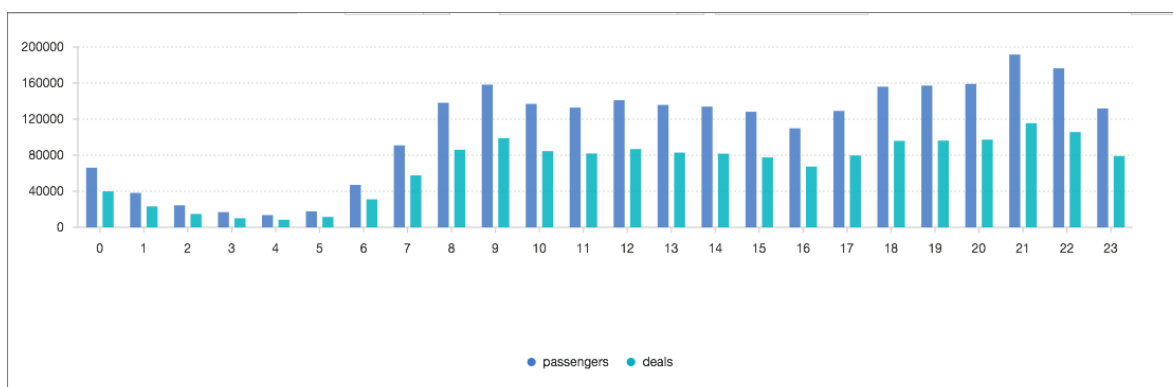
Common statistics

Before query and analysis, enable and set indexes. For more information, see [#unique_27](#).

1. Run the following statement to count the number of passengers boarding the taxis during the day and determine the peak hours:

```
*| select count ( 1 ) as deals , sum ( passenger_ count )
as passengers ,
( tpep_pickup_time % ( 24 * 3600 ) / 3600 + 8 ) % 24 as time

group by ( tpep_pickup_time % ( 24 * 3600 ) / 3600 + 8 ) % 24
order by time limit 24
```

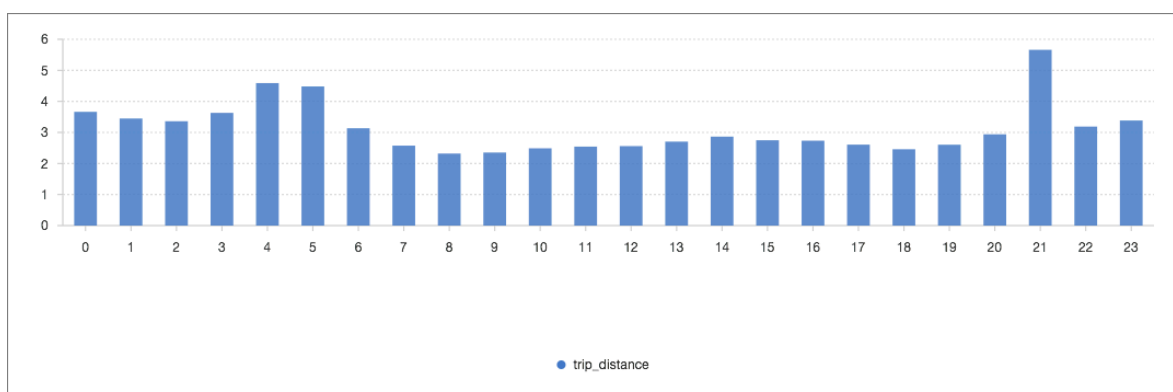


As shown from the preceding figure, the peak hours are generally the morning hours when people go to work and the evening hours when people get off work. With this data, taxi companies can dispatch more vehicles accordingly.

2. Run the following statement to collect statistics about the average trip distance in different time periods:

```
*| select  avg ( trip_distance ) as trip_distance ,
( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 )% 24 as time

group by ( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 )% 24
order by time limit 24
```

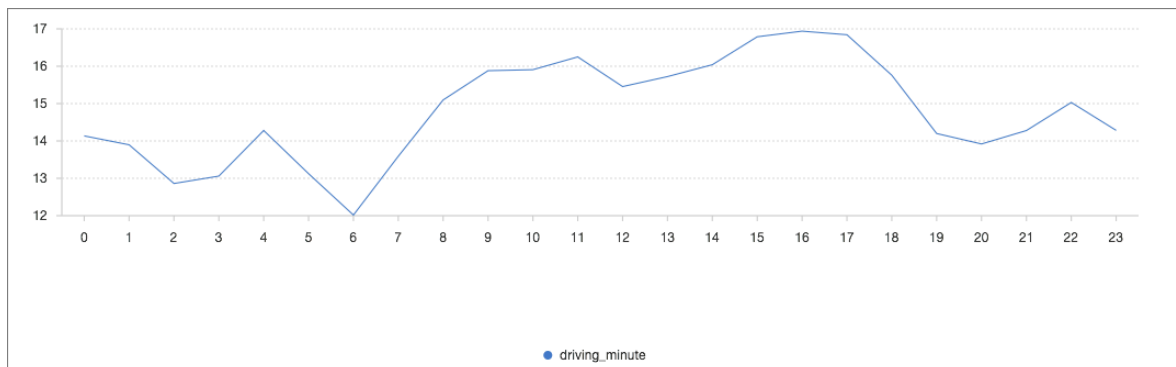


Passengers tend to take a longer trip during certain time periods of the day, so taxi companies need to dispatch more vehicles.

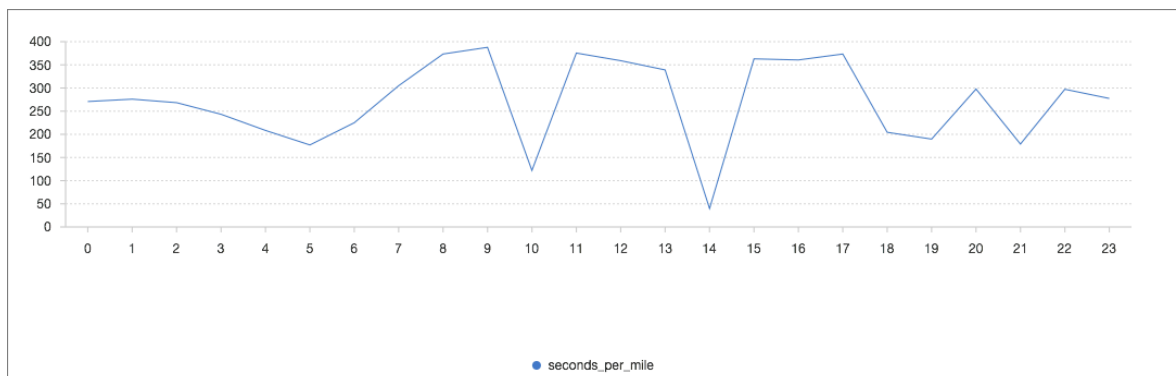
3. Run the following statement to calculate the average trip duration (in minutes) and the time required for per unit of mileage (in seconds), and determine during which time period of the day taxis experience more traffic:

```
*| select  avg ( tpep_dropoff_datetime - tpep_pickup_datetime ) / 60
as driving_minutes ,
( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 )% 24 as time
```

```
group by ( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 ) % 24
order by time limit 24
```



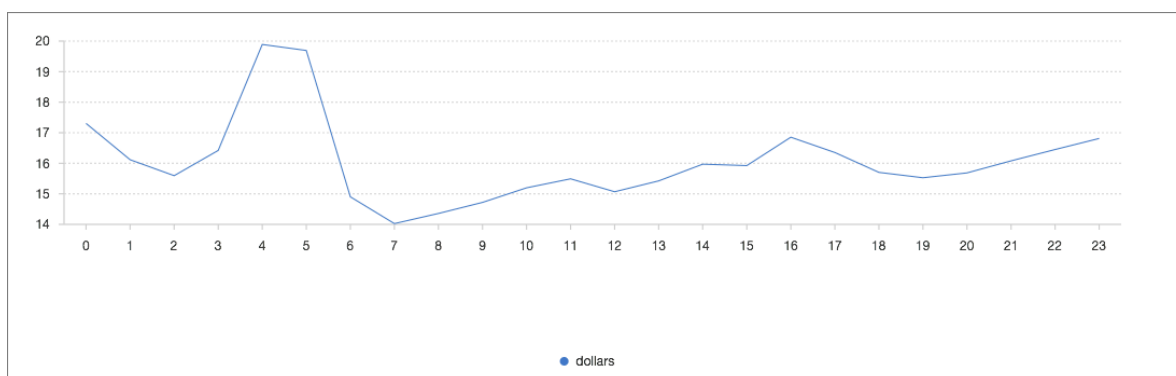
```
*| select sum ( tpep_dropoff_datetime - tpep_pickup_datetime ) / sum
( trip_distance ) as driving_minutes ,
( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 ) % 24 as time
group by ( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 ) % 24
order by time limit 24
```



More vehicles must be dispatched during peak hours.

4. Run the following statement to calculate average taxi fares during different time periods and determine the hours with more incomes:

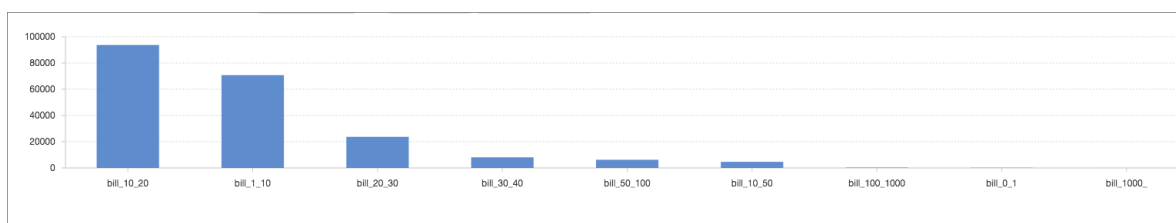
```
*| select avg ( total_amount ) as dollars ,
( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 ) % 24 as time
group by ( tpep_pickup_datetime % ( 24 * 3600 ) / 3600 + 8 ) % 24
order by time limit 24
```



The average taxi fares per customer are higher around 04:00, so financially stressed drivers can consider providing services during this time period.

5. Run the following statement to view the distribution of the payment amount:

```
*| select case when total_amount < 1 then 'bill_0_1'
when total_amount < 10 then 'bill_1_10'
when total_amount < 20 then 'bill_10_20'
when total_amount < 30 then 'bill_20_30'
when total_amount < 40 then 'bill_30_40'
when total_amount < 50 then 'bill_40_50'
when total_amount < 100 then 'bill_50_100'
when total_amount < 1000 then 'bill_100_1000'
else 'bill_1000_+' end
as bill_level, count(1) as count group by
case when total_amount < 1 then 'bill_0_1'
when total_amount < 10 then 'bill_1_10'
when total_amount < 20 then 'bill_10_20'
when total_amount < 30 then 'bill_20_30'
when total_amount < 40 then 'bill_30_40'
when total_amount < 50 then 'bill_40_50'
when total_amount < 100 then 'bill_50_100'
when total_amount < 1000 then 'bill_100_1000'
else 'bill_1000_+' end
order by count desc
```



As shown in the preceding figure, the payment amount of most transactions ranges from USD 1 to USD 20.

2.5 Analyze access logs of Layer-7 Server Load Balancer

Alibaba Cloud Server Load Balancer (SLB) distributes traffic among multiple instances to improve the servicing capabilities of your applications. You can use SLB to prevent single point of failures (SPOFs) and improve the availability and the fault tolerance capability of your applications. SLB is an infrastructure component for most services in the cloud architecture. It is required to monitor, detect, diagnose, and generate reports for SLB in a continuous way. You can learn about the running status of SLB instances through the monitoring reports provided by cloud service providers.

Currently, access logs can be generated for HTTP- or HTTPS-based Layer-7 SLB. The access log can contain about 30 columns such as the time when the request is received, the IP address of the client, processing latency, request URI, back-end RealServer (Alibaba Cloud ECS instance) address, and returned status code. For more information about the columns and features, see [Access logs of Layer-7 Server Load Balancer](#).

This topic describes the optimal scheme for real-time collection, query, and analysis of Layer-7 SLB access logs based on the visualization and real-time query and analysis (OLTP and OLAP) features of Log Service. This topic also describes some typical methods of report statistics and log query and analysis for SLB instances. The scheme combines visual reports and query and analysis engines to analyze the status of SLB instances in real time and interactively.

Currently, Layer-7 SLB access logs are available in all regions.

Prerequisites

1. Log Service and Layer-7 SLB are activated.
2. Layer-7 SLB logs are collected. For more information about how to collect the logs, see [Access logs of Layer-7 Server Load Balancer](#).

Visual analysis

Business overview

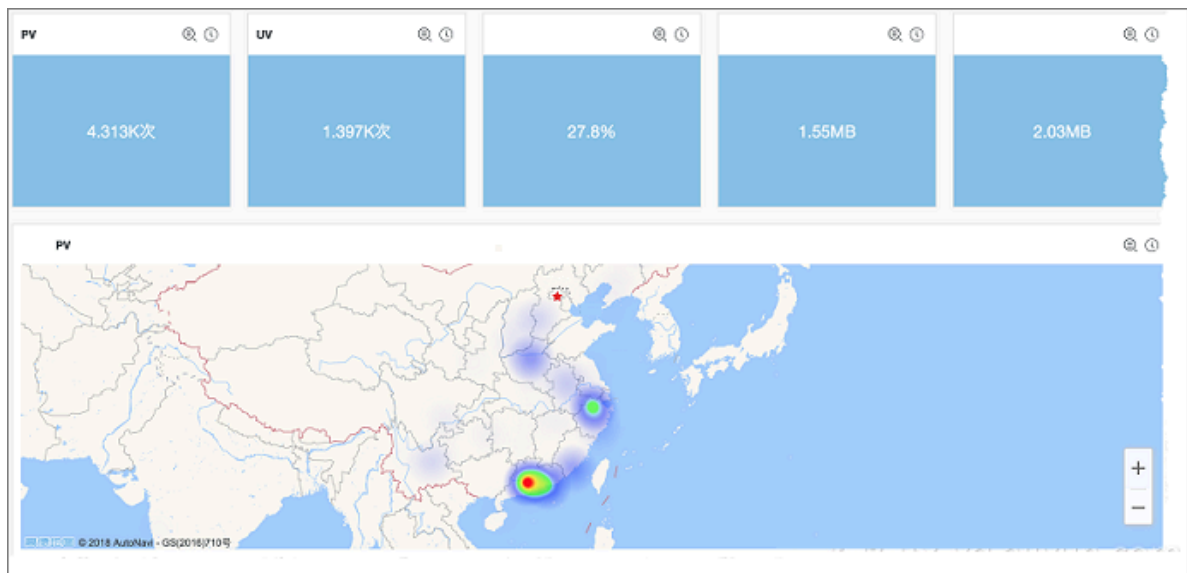
SLB supports horizontal scaling of RealServers and fault redundancy recovery, supporting the processing of large-scale concurrent web access requests and guaranteeing high reliability for applications.

**Note:**

Typical business overview metrics are as follows:

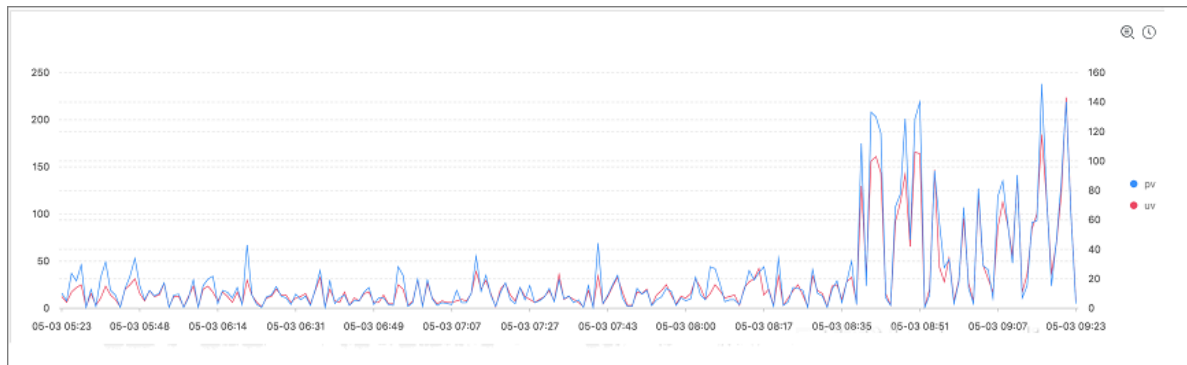
- PV: the number of the HTTP or HTTPS requests sent by the client (the IP address of the request source).
 - UV: the total number of the requests. The requests from the same client IP address are counted as one request.
 - Request success rate: the percentage of the requests whose status code is 2XX to the total PVs.
 - Request traffic: the sum of the length of the requests (specified by the request_length field).
 - Response traffic: the total number of bytes in the HTTP response body (specified by the body_bytes_sent field) returned from SLB to the client.
 - PV heat map of requests: displays the density of PVs in the regions where the IP addresses of the clients reside.
- View the regions from which the requests are sent.

In this example, most requests are sent from the Pearl River Delta and the Yangtze River Delta, as shown in the following figure.



- In the dashboard of Log Service, you can add filter conditions and display the data that meets the filter conditions, such as the IP address of the client and the ID of the SLB instance, in the current chart.

For example, you can view the trend of the PVs and UVs of a specified SLB instance over time.



Request scheduling analysis

The requests sent from clients are first processed by SLB and then dispatched to one of multiple RealServers for processing based on the business logic. SLB can detect unhealthy RealServers and dispatch traffic to other RealServers in a normal state. After the unhealthy RealServers enter a normal state, the traffic continues to be dispatched to these RealServers. The whole process is automatic. Add a listener to the SLB instance. The following three scheduling methods are available to the listener: round robin, weighted round robin (WRR), and weighted least connections (WLC).

- Round robin: External requests are sequentially distributed to back-end ECS instances based on the number of visits.
- WRR: You can set the weight for each back-end RealServer. RealServers with higher weights receive more requests than those with smaller weights.
- WLC: Requests are forwarded based on the weight and load (the number of connections) of each back-end RealServer. If the weights are the same, back-end RealServers with fewer connections receive more requests than those with more connections.

For example, the RealServer whose IP address is `172 . 19 . 39 . **` is also a jump server. Its performance is four times that of the other three RealServers. If you set the weight of this RealServer to 100, the weight of the other three RealServers is set

to 20. Run the following statement to aggregate the traffic of two dimensions based on the access logs of the instance:

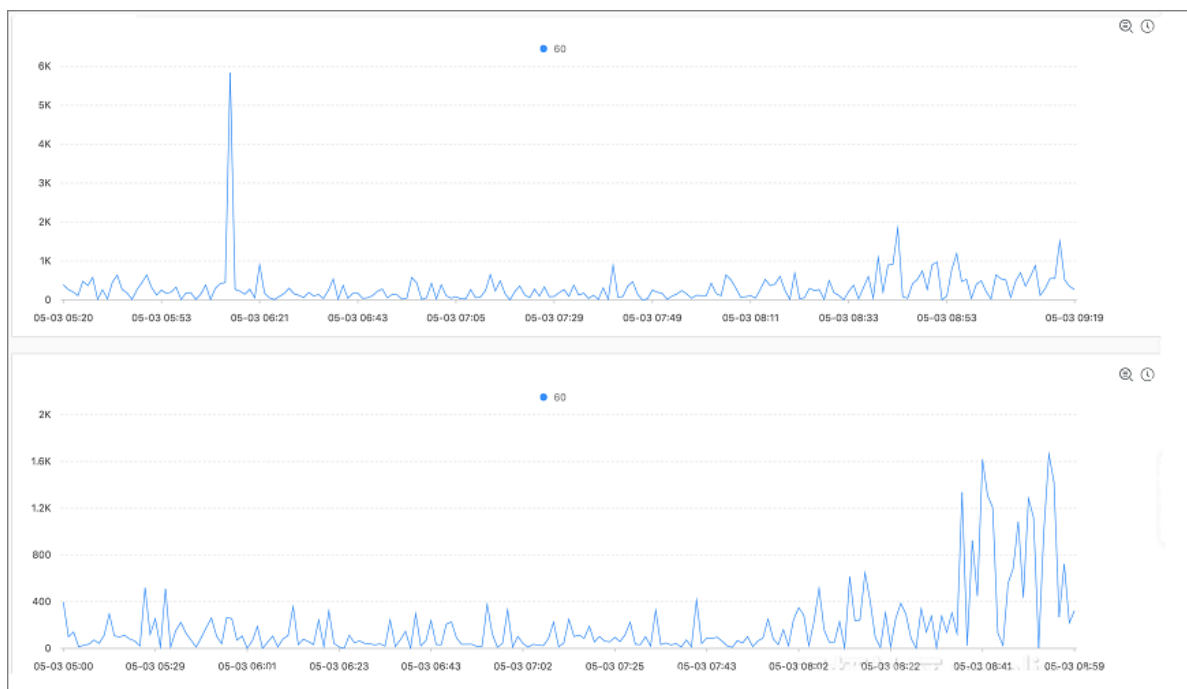
```
* | select COALESCE ( client_ip , vip_addr , upstream_addr )
  as source , COALESCE ( upstream_addr , vip_addr , client_ip )
  as dest , sum ( request_length ) as inflow group
  by grouping sets ( ( client_ip , vip_addr ) , ( vip_addr ,
  upstream_addr ) )
```

Based on the Sankey diagram, you can obtain a request traffic topology by aggregating and visualizing the result returned by the SQL statement from the virtual IP address dimension. If requests are sent from multiple client IP addresses to the SLB virtual IP address (172 . 19 . 0 . **), the request traffic is dispatched to the back-end RealServers in the 20:20:20:100 proportion. The Sankey diagram clearly shows the load of each RealServer.

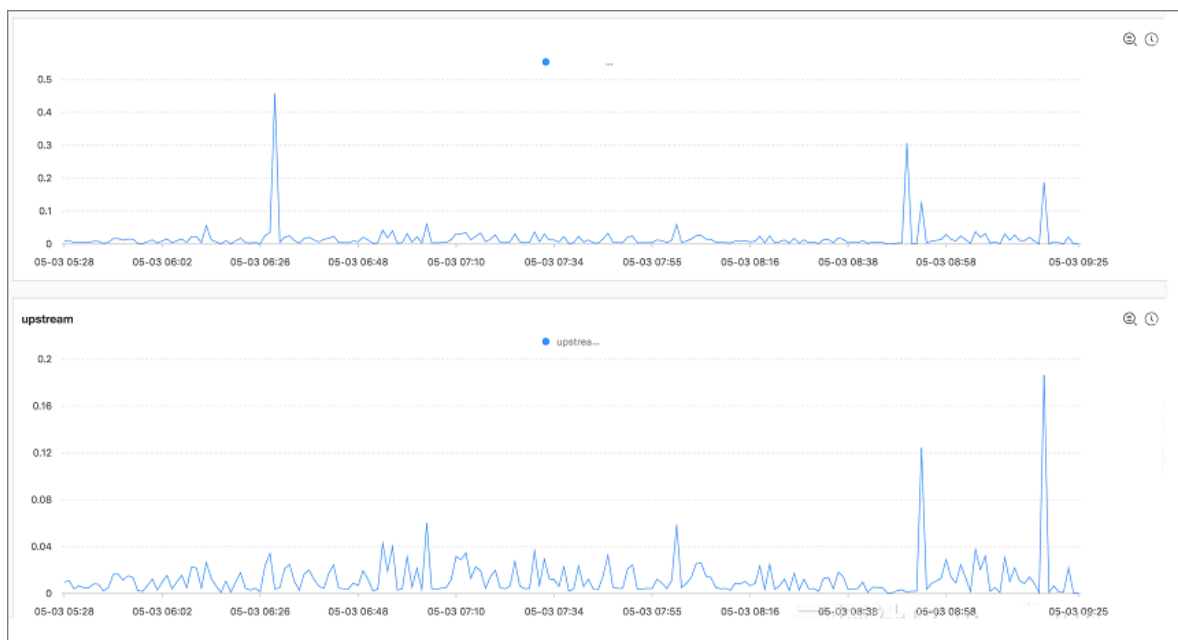
Traffic and latency analysis

Perform aggregate computing on the traffic and latency in each minute.

- Statistics from the request_length and body_bytes_sent dimensions



• Statistics from the request_time and upstream_response_time dimensions



Request overview

You can analyze the HTTP or HTTPS requests in access logs from multiple dimensions, such as request methods, protocols, and status codes.

You can select a time range and specify the status code to locate a RealServer in the logs based on the query and analysis features of Log Service, as shown in the following figure.

internal-alert-history

Data Processing 1Day(Relative) Share Index Attributes Save Search Saved as Alarm

1 Success

2.4

0 09-19 09-19 09-19 09-19 09-19 09-20 09-20 09-20 09-20

Log Entries:96 Search Status:The results are accurate.

Raw Logs LogReduce LogTail Graph Display Content Column Column Settings

Quick Analysis

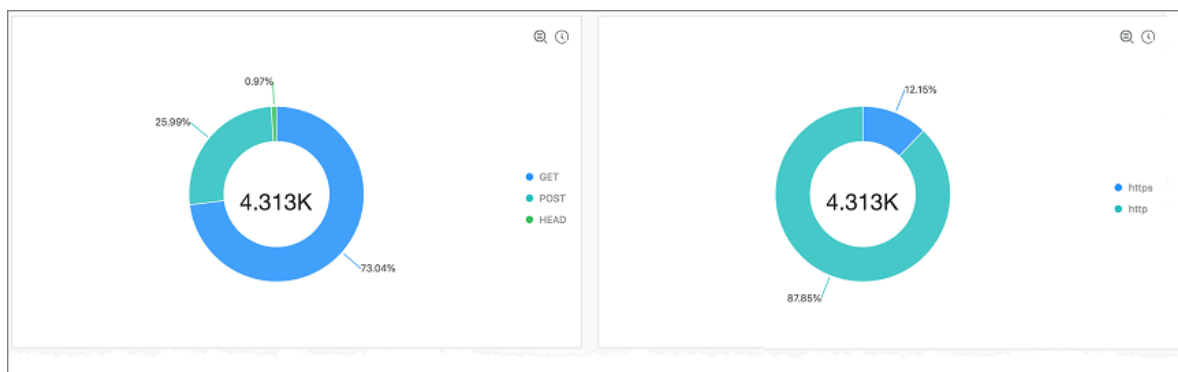
Search

AlertDisplayName AlertID AlertName Condition Dashboard FireCount Fired

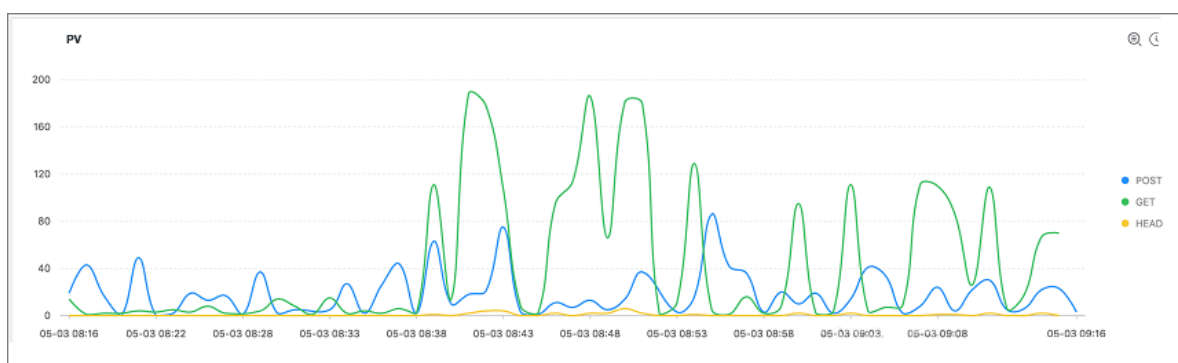
1 Sep 20, 10:15:04

AlertDisplayName: test
AlertID: jid-7ba8c9efe5b3fd-e6f8-4ef5-8236-a3bed3be647f
AlertName: alert-1564568989-103336
Condition: method==1
Dashboard: dashboard-1557481630994-536830
FireCount: 0
Fired: false
LastNotifiedAt: 0
NotifyStatus: NotNotified
Reason: Alert condition not met
Results: [{"EndTime":1568945704,"FireResult":null,"LogStore":"wdproject","Query":"","SELECT method, COUNT(*) as number GROUP BY method LIMIT 10","RawResultCount":0,"RawResults":[],"StartTime":1568945644,"Truncated":false}]
Status: Success
source:
topic: alert

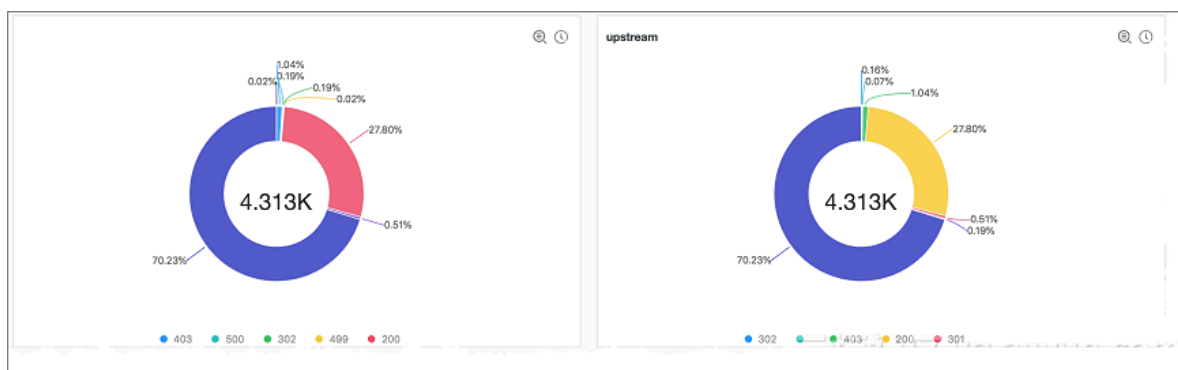
1. You can collect statistics about the PVs for different request methods.



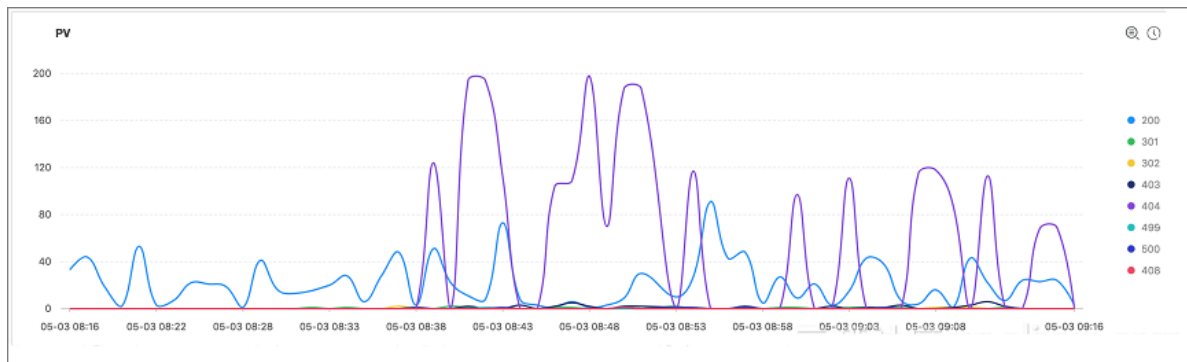
2. With the time dimension, you can view the trend lines of PVs for different request methods over time.



3. You can learn about the service running status from the proportion of the requests with different status codes. If there are a lot of requests with the status code 500, internal errors occur with the applications of the back-end RealServers.



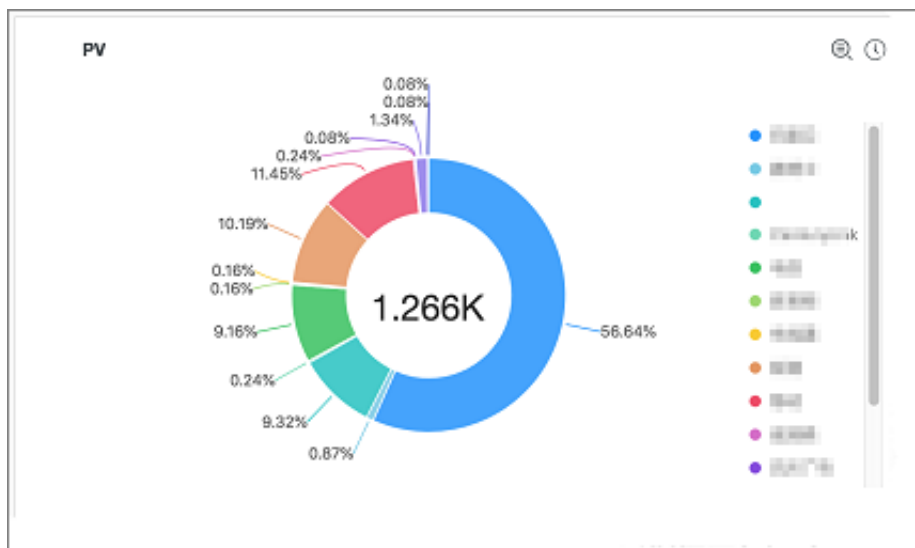
4. You can view the trend lines of PVs for each status code.



Request source analysis

You can obtain the geographic location (country, province, or city) and telecom operator information by analyzing the IP address of the client.

1. You can make a PV distribution chart for the requests from the IP addresses of the different operators.



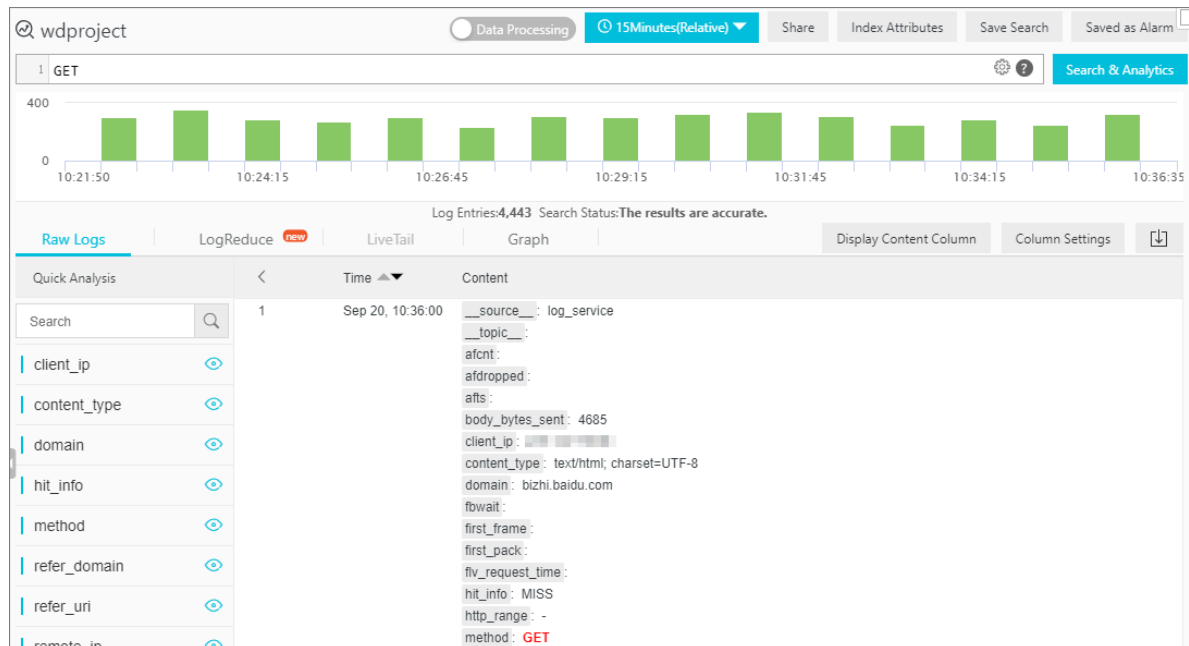
2. You can view the request sources in descending order of the PVs by analyzing the client IP addresses.

3. You can view the user agent information.

The user agent (`http_user_agent`) is a common metric that can be used to identify who is visiting the website or service. For example, a search engine uses web crawlers to scan or download website resources. In general, infrequent crawler access allows the search engine to update website content in a timely manner and facilitates website promotion and search engine optimization (SEO). If the requests with high PVs are sent from the web crawlers, the service performance and server

resources may be wasted. Therefore, you need to monitor the requests with high PVs and take measures to avoid wastes.

You can search the access logs for related records based on the SLB instance ID, application host, and user agent. The following figure shows the log of GET requests sent from the Sogou web crawler. The requests are not frequent and have no adverse impact on applications.

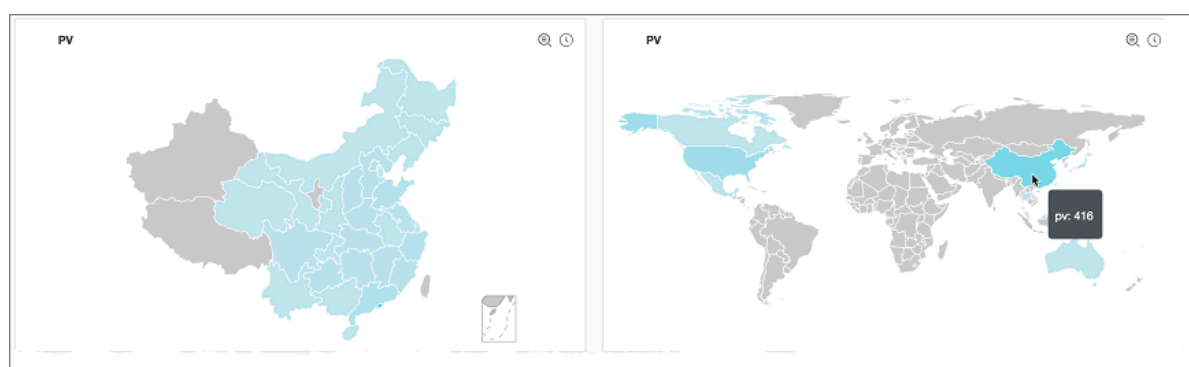


Operations overview

SLB access logs are vital to the marketing team, who can use the access logs for traffic analysis to make business plans.

1. View the PV distribution of different regions.

By viewing the PVs of different regions in the geographical dimension, you can know the areas where the key customers of the services are, and the areas with low PVs that may need further promotion.

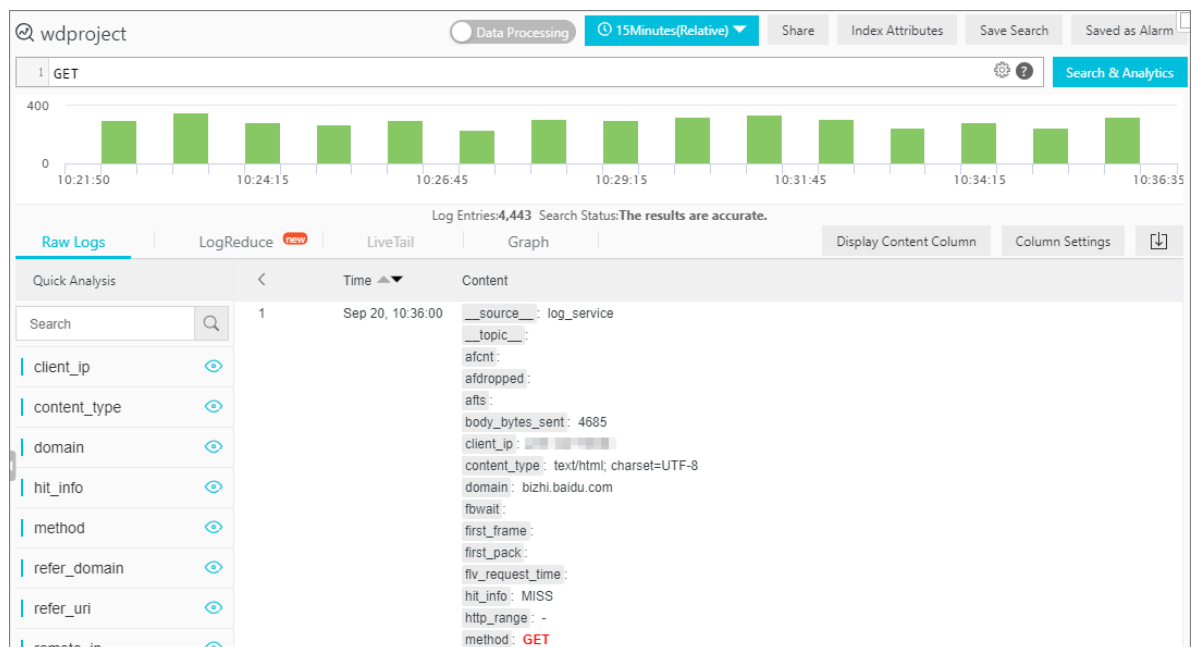


2. View the host and URI of the requests.

For a website, analyzing visitor behavior can provide a powerful reference for website content construction. Whether the content is good or poor can be told by the host or the URI of the top websites with the highest PVs or with the lowest PVs.

3. View the request URI.

For popular resources (specified by the `request_uri` field in access logs), you can pay attention to the `http_referer` field in the log details to view the request source of the website. Strengthen good request sources and enable hotlinking protection. The following figure shows the log details of a page redirection request from Baidu Image.



2.6 Analyze NGINX access logs

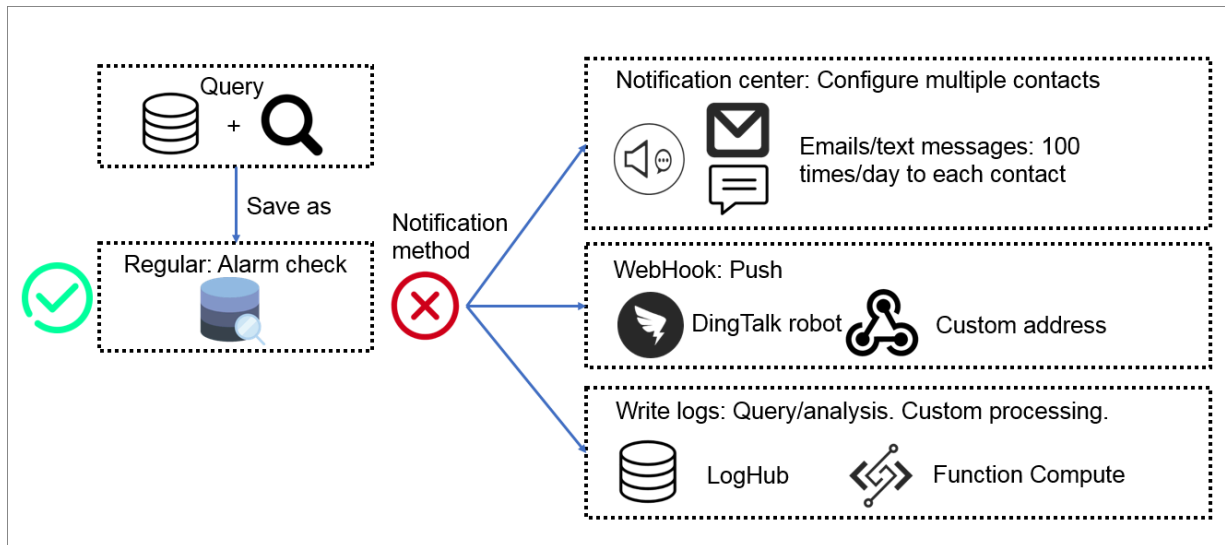
Currently, Log Service supports saving query statements through Saved Search. Log Service also supports setting the trigger cycle (interval) for queries, setting judgment conditions for execution results, and reporting alerts. You can set an alerting action to specify the way to inform you when the execution result of a regular Saved Search operation meets the trigger conditions.

Currently, the following three notification methods are supported:

- **Notification center:** Multiple contacts can be set in the Alibaba Cloud notification center. You can send notifications to contacts through emails and SMS messages.
- **WebHook:** including DingTalk Chatbot and custom WebHook.

- (Coming soon) Writing back to Log Service Logstores: You can subscribe to events through Realtime Compute and Function Compute, or generate views and reports for alerts.

For more information about how to configure the alerting feature, see [#unique_35](#). In addition to the monitoring and alerting features of Log Service, you can also use CloudMonitor to monitor all metrics of Log Service. CloudMonitor can send you a notification when the alerting condition is triggered.



Scenarios

This section takes NGINX logs as an example to describe how to query and analyze collected logs regularly through Log Service and determine the following business issues based on the query result:

- Whether an error exists.
- Whether a performance problem exists.
- Whether a sudden decrease or increase of the traffic exists.

Preparation (NGINX log access)

1. Collect log data.

- a. On the Overview page, click Import Data in the upper-right corner. In the dialog box that appears, click NGINX Access Log.

- b. Select a Logstore.

If you enter the log collection configuration process by clicking the + icon next to Data Import under a Logstore on the Logstores tab, the system skips this step.

- c. Create a machine group.

Before creating a machine group, make sure that you have installed Logtail.

- **Machines of Alibaba Group:** By default, Logtail is installed for these machines. If Logtail is not installed on a machine, contact us as prompted.
- **ECS instances:** Select an ECS instance, and click Install. ECS instances running Windows do not support one-click installation of Logtail. In this case, you need to manually install Logtail. For more information, see [#unique_36](#).
- **User-created machines:** Install Logtail as prompted. For more information about how to install Logtail, see [#unique_37](#) or [#unique_38](#) based on your operating system.

After installing Logtail, click Confirm Installation to create a machine group. If you have created a machine group, click Use Existing Machine Group.

- d. Configure the machine group.

Select a machine group and move the machine from Source Machine Group to Application Machine Group.

- e. Specify the following configuration items: Configuration Name, Log Path, NGINX Log Format, and NGINX Key. You can specify Advanced Options based on your needs.

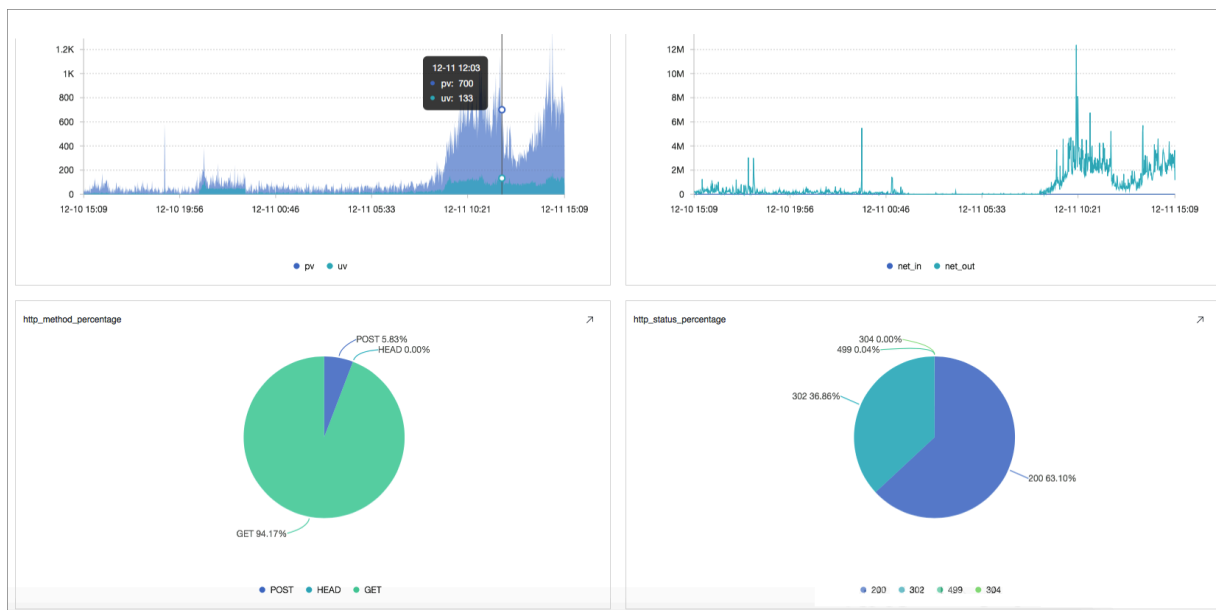
- f. Click Next.

2. Complete query and analysis configurations.

For more information, see [Enable and set indexes](#), [Interconnect with DataV big screen](#), or [Collect and analyze NGINX access logs](#).

3. Set the views and alerts for key metrics.

Sample views:



Procedure

1. Determine if any error exists

The common error codes are as follows: 404 (the request cannot find the address), 502, and 500 (an error occurs with the server). Generally, we only focus on 500 errors.

Determine if a 500 error exists. You can run the following query statement to count the number of errors (c) per unit time. Then, you can set the alert rule as $c > 0$, indicating that an alert will be sent when the number of 500 errors exceeds 0 in the unit time.

```
status : 500 | select count ( 1 ) as c
```

This method is relatively simple but too sensitive. For services facing relatively high business pressure, a few 500 errors are common. In response to this situation, you can set the trigger count to 2 in the trigger conditions so that alerts are only triggered when the conditions are met for 2 times in a row.

2. Determine if any performance problem exists

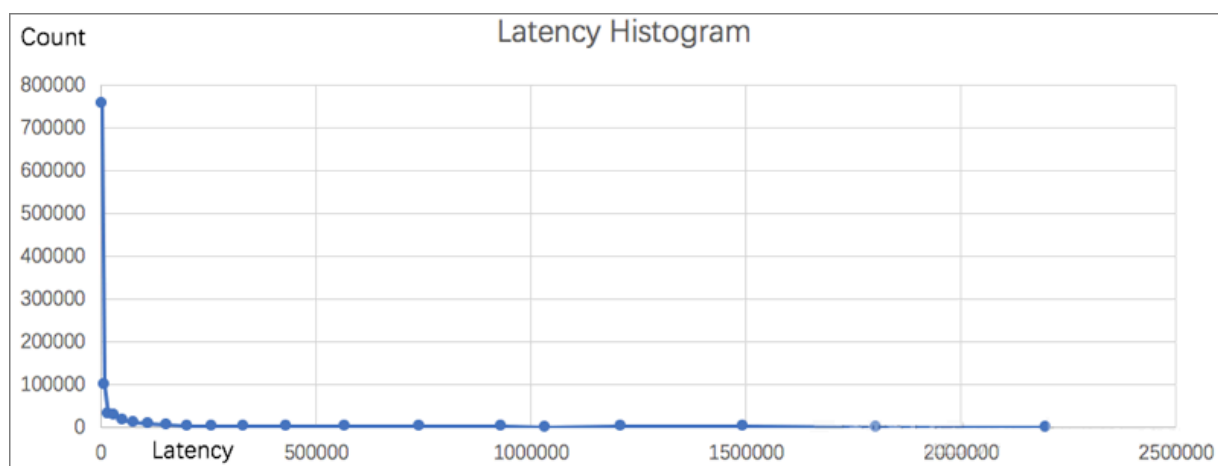
Although no error occurs in server operation, the latency might be increased. You can set an alert for latency.

For example, you can calculate the latency of all the write requests ("Post") of an interface ("/adduser") through the following method: Set the alert rule as $l > 300000$, indicating that an alert will be sent when the average latency exceeds 300 ms.

```
Method : Post    and    URL :"/ adduser " | select  avg ( Latency )  
as    l
```

Sending alerts based on the average latency is simple and direct. However, this method may average the latency of individual requests, making it difficult to detect problems. For example, you can compute a mathematical distribution for the latency in the time period, namely, dividing the latency into 20 intervals and calculating the number in each interval. As shown in the histogram, the latency of most requests is lower than 20 ms, but the highest latency reaches 2.5s.

```
Method : Post    and    URL :"/ adduser " | select  numeric_hi  
stogram ( 20 , Latency )
```



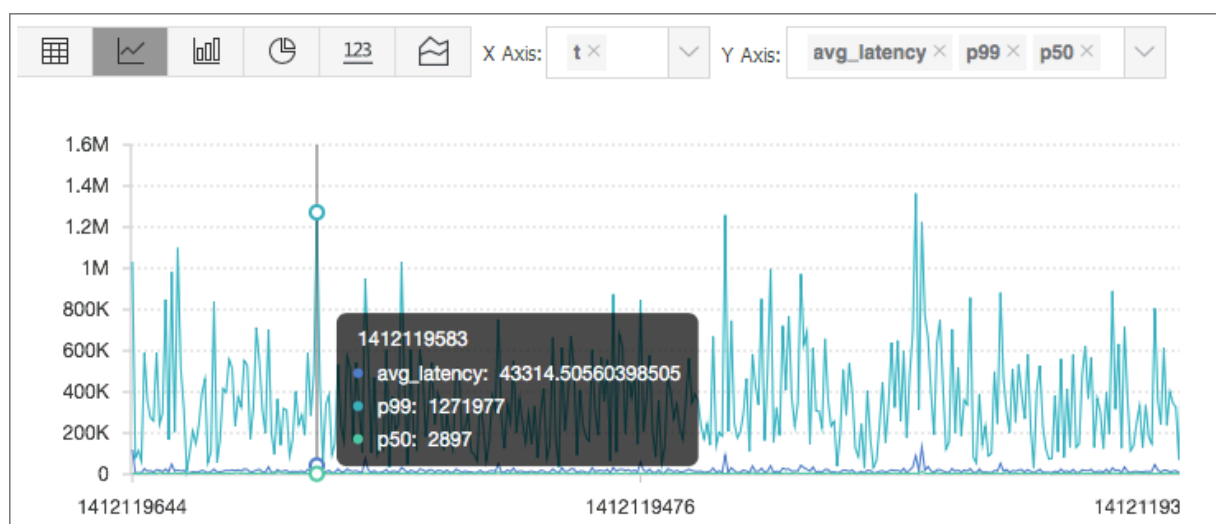
You can use the percentile in mathematical statistics (the maximum latency is 99 %) as the trigger condition. In this way, you can exclude false alerts triggered by accidental high latency and reflect the overall situation of the latency. The following statement calculates the latency of the 99th percentile, `approx_percentile (Latency , 0.99)`. You can also change the second parameter to calculate the latency of other

percentiles, for example, the request latency of the 50th percentile, `approx_percentile (Latency, 0.5)`.

```
Method : Post and URL :"/ adduser " | select approx_percentile ( Latency , 0 . 99 ) as p99
```

In a monitoring scenario, you can chart the average latency, the 50th percentile latency, and the 90th percentile latency. The following figure shows the latency of every minute in a day (1,440 minutes).

```
* | select avg ( Latency ) as l , approx_percentile ( Latency , 0 . 5 ) as p50 , approx_percentile ( Latency , 0 . 99 ) as p99 , date_trunc ( ' minute ' , time ) as t group by t order by t desc limit 1440
```



3. Determine if the traffic has a sudden decrease or increase

The natural traffic on the server is usually in line with probability distribution, which means that a process of slow increase or decrease exists. The sudden decrease or increase of the traffic indicates great changes in a short time period. This phenomenon is usually abnormal and needs special attention.

As shown in the following monitoring chart, the traffic decreases by over 30% within 2 minutes and resumes rapidly within 2 minutes.



The following reference frames are provided for a sudden decrease or increase:

- Last window: compares data in the current time period with that in the previous time period.
- Window of the same time period of the previous day: compares data in the current time period with that in the same time period of the previous day.
- Window of the same time period of the previous week: compares data in the current time period with that in the same time period of the previous week.

This section takes the first reference frame as an example to calculate the change ratio of inbound traffic. You can also calculate other metrics of the traffic such as queries per second (QPS).

3.1 Define a calculation window

Define a window of 1 minute to calculate the inbound traffic size of this minute. The following figure shows the statistic result within a 5-minute interval.

```
* | select  sum ( inflow )/( max ( __time__ )- min ( __time__ )) as
  inflow ,  __time__ - __time__ % 60 as window_time from
log_group  by window_time order by window_time limit
15
```

As shown in the result, the average inbound traffic size specified by `sum (inflow)/(max (__time__)- min (__time__))` in every window is even.

window_time	inflow
1513045740	315574947
1513045800	333233937
1513045860	335821584
1513045920	330556452
1513045980	316785257

3.2 Calculate the difference in the window (max_ratio)

Subqueries are involved. Run a query statement to calculate the change ratio between the maximum value or the minimum value and the average value from the preceding result. In this example, the change ratio between the maximum value and the average value is calculated, for example, 1.02. You can set the alert rule as `max_ratio > 1.5`, indicating that an alert will be sent when the ratio of change exceeds 50%.

```
* | select  max ( inflow )/ avg ( inflow ) as max_ratio from
      ( select  sum ( inflow )/( max ( __time__ )- min ( __time__ )) as
        inflow , __time__ - __time__ % 60 as window_time from
        log group by window_time order by window_time limit
        15 )
```

window_time	inflow
1513045740	315574947
1513045800	333233937
1513045860	335821584
1513045920	330556452
1513045980	316785257

Maximum value

3.3 Calculate the difference in the window (latest_ratio)

In some scenarios, more attention is paid to the fluctuation of the latest value (whether the value is recovered). You can use the `max_by` function to get the inbound

traffic size of the latest window (specified by `window_time`). Then, you can calculate the change ratio between the latest value and the average value, for example, 0.97.

```
* | select max_by ( inflow , window_time ) / 1 . 0 / avg (
  inflow ) as latest_ratio from ( select sum ( inflow ) /
  ( max ( __time__ ) - min ( __time__ ) ) as inflow , __time__ -
  __time__ % 60 as window_time from log group by
  window_time order by window_time limit 15 )
```



Note:

The calculation result of the `max_by` function is of the character type, which must be converted to the numeric type. To calculate the relative ratio of changes, you can replace it with `1.0-max_by(inflow, window_time)/1.0/avg(inflow))` as `latest_ratio`.

window_time	inflow
1513045740	315574947
1513045800	333233937
1513045860	335821584
1513045920	330556452
1513045980	316785257

Latest value

3.4 Calculate the difference in the window which indicates the fluctuation ratio, namely, the change ratio between the current value and the previous value

Another method for calculating the fluctuation ratio is the first derivative in mathematics, namely, the change ratio between the value of the current window and the value of the previous window.

window_time	inflow
1513308660	8138522256
1513308720	8584340710
1513308780	9210706832
1513308840	9684619494

Difference

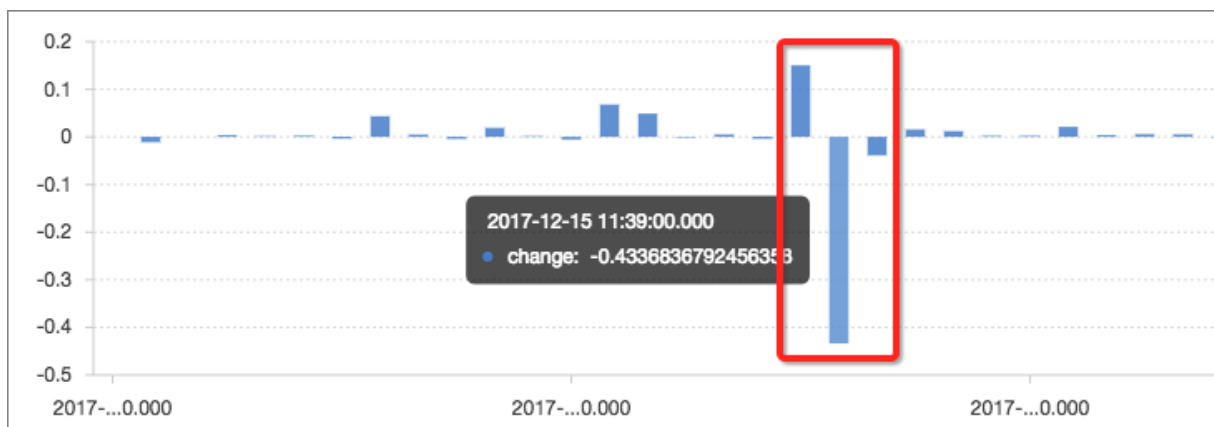
Use the window function (`lag`) for calculation. Extract the current inbound traffic and the inbound traffic of the previous window to calculate the difference by using `lag`(

`inflow, 1, inflow)over()` and divide the calculated difference value by the current value to get the change ratio.

```
* | select ( inflow - lag ( inflow , 1 , inflow ) over ( ) ) * 1
. 0 / inflow as diff , from_unixt ime ( window_tim e ) from
( select sum ( inflow ) / ( max ( __time__ ) - min ( __time__ ) ) as
inflow , __time__ - __time__ % 60 as window_tim e from
log_group by window_tim e order by window_tim e limit
15 )
```

In this example, a relatively major decrease occurs in traffic at 11:39, with a change ratio of over 40%.

To define an absolute change ratio, you can use the `abs` function (absolute value) to unify the calculation result.



Summary

The query and analysis features of Log Service follow the SQL-92 standard and support various mathematical statistics and computing methods. Anyone who can use SQL can perform fast analysis. Have a try!

2.7 Query MNS logs

Alibaba Cloud [Message Service \(MNS\)](#) pushes logs to Log Service. This topic describes how to query specified information in logs pushed from MNS.

MNS supports queue logs and topic logs. These logs contain all information about messages throughout the lifecycle, including the time, location, operation, and context. You can analyze the logs through real-time query, real-time computing, and offline computing.

Real-time query

This section describes several common scenarios of real-time query. You can use multiple keywords to run complex queries.

View the message tracing of a message in a queue

- Procedure

1. Enter the queue name and the ID of the message in the search box. Format: \$ `queuename` and \$ `messageid`.
2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the operation logs of the message.

- Example

View the message tracing of the message whose ID is `12682720A1 B271D0 - 1 - 1635DD12B1 C - 2000000004` in the queue named `loglog`.

- **Statement:** `loglog` and `12682720A1 B271D0 - 1 - 1635DD12B1 C - 2000000004`
- **Result:** As shown in the following figure, the query result displays the process from message sending to message deleting.

	Time	Content
1	05-14 16:43:20	AccountId: 1231579085529123 Action: DeleteMessage MessageId: 12682720A1B271D0-1-1635DD12B1C-2000000004 ProcessTime: 11 QueueName: loglog ReceiptHandleInRequest: 1-ODU4OTkzNDU5Ni0xNTI2Mjg3NDI3LTETOA== RemoteAddress: 106.11.227.98 RequestId: 5AF94C28048A9319D541FD71 Time: 2018-05-14 16:43:20.418258 __source__: 10.152.69.131 __topic__:
2	05-14 16:43:17	AccountId: 1231579085529123 Action: ReceiveMessage MessageId: 12682720A1B271D0-1-1635DD12B1C-2000000004 NextVisibleTime: 1526287427 ProcessTime: 35 QueueName: loglog ReceiptHandleInResponse: 1-ODU4OTkzNDU5Ni0xNTI2Mjg3NDI3LTETOA== RemoteAddress: 106.11.229.156 RequestId: 5AF94C2536AF628D2ABEEFCB Time: 2018-05-14 16:43:17.506714 __source__: 10.152.70.131 __topic__:
3	05-14 16:42:59	AccountId: 1231579085529123 Action: SendMessage MessageId: 12682720A1B271D0-1-1635DD12B1C-2000000004 NextVisibleTime: 1526287379 ProcessTime: 10

View the number of the messages that are sent to Log Service in a queue

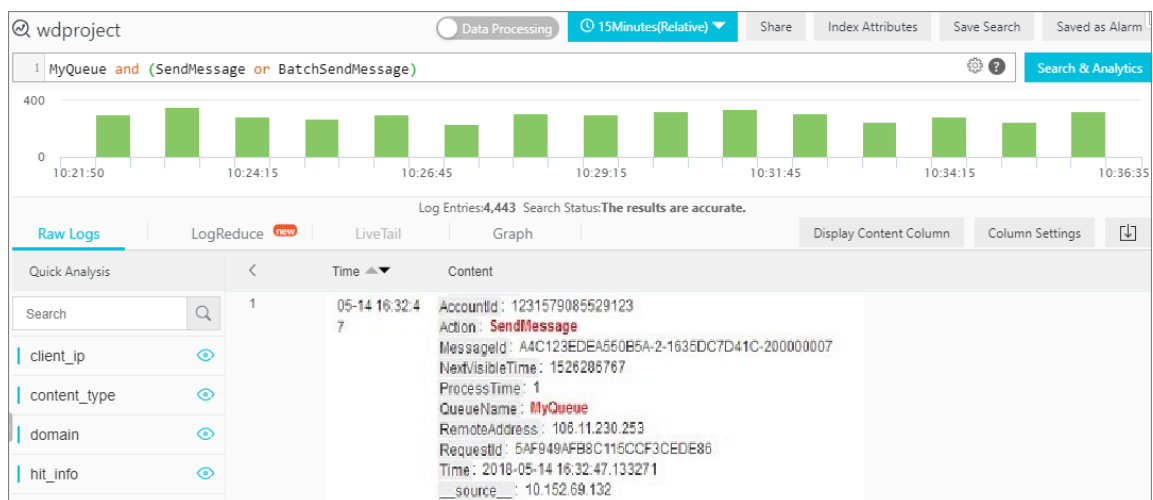
- Procedure

1. Enter the queue name and the write operation in the search box. Format: `$queueName and (SendMessage or BatchSendMessage)`.
2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the logs about all messages sent to Log Service in the queue. Move the pointer over the green column chart. You can view the number of the messages sent in the time period you specified.

- Example

View the number of the messages that are sent to Log Service in the queue named **MyQueue**.

- **Statement:** `MyQueue and (SendMessage or BatchSendMessage)`
- **Result:** As shown in the following figure, four write operations are performed in the time period you specified.



View the number of the messages that are consumed by Log Service in a queue

- Procedure

1. Enter the queue name and the consumption operation in the search box.

Format: `$ queueName and (ReceiveMessage or BatchReceiveMessage)`.

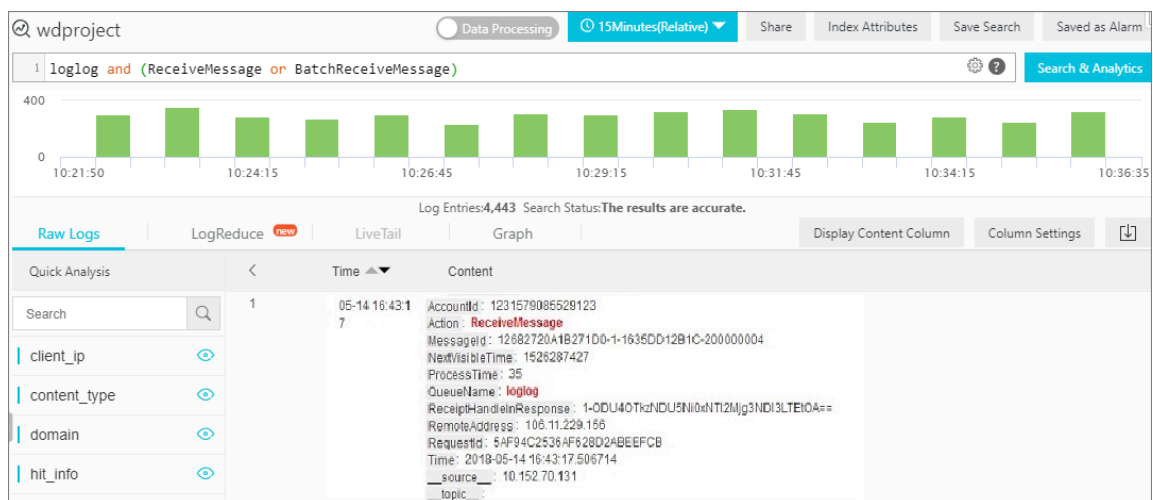
2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the logs about the messages consumed by Log Service in the queue.

- Example

View the number of the messages that are consumed by Log Service in the queue named `loglog`.

- Statement: `loglog and (ReceiveMessage or BatchReceiveMessage)`

- Result: As shown in the following figure, five consumption operations are performed in the time period you specified.



View the number of the messages that are deleted in a queue

- Procedure

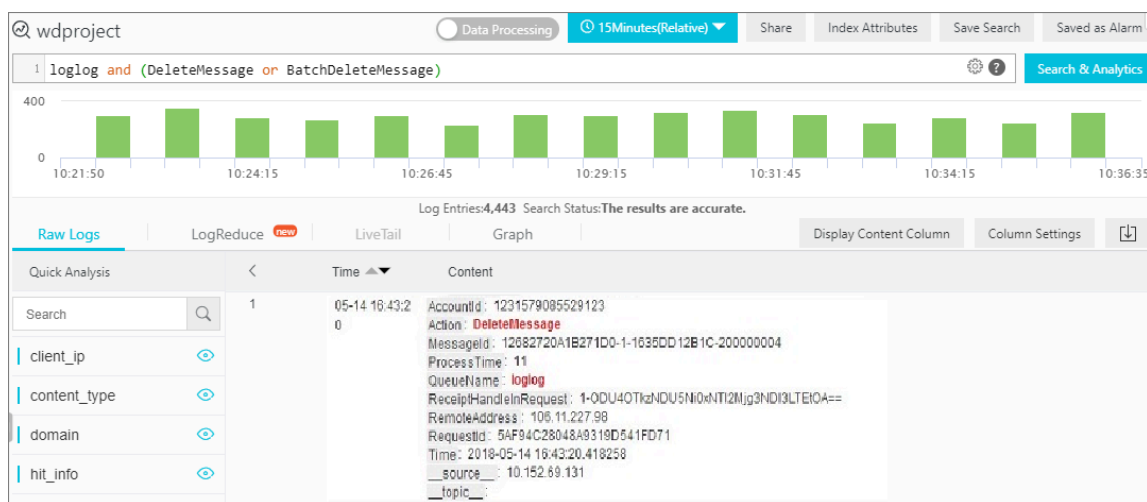
1. Enter the queue name and the deletion operation in the search box. Format: `$ queueName and (DeleteMessage or BatchDeleteMessage)`.

2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the logs about the messages that are deleted in the queue.

• Example

View the number of the messages that are deleted in the queue named loglog.

- **Statement:** `loglog and (DeleteMessage or BatchDeleteMessage)`
- **Result:** As shown in the following figure, the query result displays the logs about all messages that are deleted. You can view the number of the messages that are deleted.



View the message tracing of a message in a topic

• Procedure

1. Enter the topic name and the ID of the message in the search box. Format: `$topicname and $messageid`.
2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the message tracing of the message in the topic.

• Example

View the message tracing of the message whose ID is `BD692F55DE D88AF6 - 1 - 1635DFEAF3 B - 2000000008` in the topic named `logtesttt`.

- **Statement:** `logtesttt` and `BD692F55DE D88AF6 - 1 - 1635DFEAF3 B - 2000000008`
- **Result:** As shown in the following figure, the query result displays the process from message sending to notification in the `logtesttt` topic.

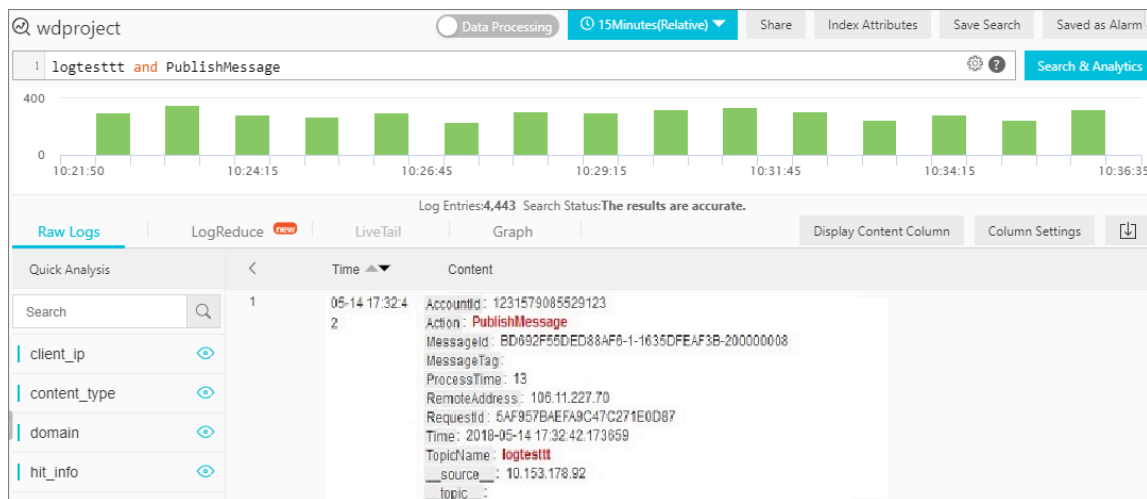
<	Time ▲▼	Content	⬇	⚙
1	05-14 17:32:5 3	AccountId: 1231579085529123 Action: Notify MessageId: BD692F55DED88AF6-1-1635DFEAF3B-2000000008 NotifyLatency: 502183 NotifyStatus: 400 SubscriptionName: logloglog Time: 2018-05-14 17:32:53.224181 TopicName: logtesttt __source__: 10.153.177.113 __topic__:		
2	05-14 17:32:4 2	AccountId: 1231579085529123 Action: PublishMessage MessageId: BD692F55DED88AF6-1-1635DFEAF3B-2000000008 MessageTag: ProcessTime: 13 RemoteAddress: 106.11.227.70 RequestId: 5AF957BAEFA9C47C271E0D87 Time: 2018-05-14 17:32:42.173659 TopicName: logtesttt __source__: 10.153.178.92 __topic__:		
3	05-14 17:32:4 2	AccountId: 1231579085529123 Action: Notify MessageId: BD692F55DED88AF6-1-1635DFEAF3B-2000000008 NotifyLatency: 503831 NotifyStatus: 400 SubscriptionName: logloglog Time: 2018-05-14 17:32:42.685364		

View the number of the messages that are published in a topic

• Procedure

1. Enter the topic name and the publish operation in the search box. Format: `$topicname and PublishMessage`.
2. Select the time range of which you want to query logs, and click Search & Analysis. Then, you can view the logs about the messages that are published in the topic.

- **Example:** View the number of the messages that are published in the topic named `logtestttt`.
 - **Statement:** `logtestttt` and `PublishMessage`
 - **Result:** As shown in the following figure, five publish operations are performed in the time period you specified.



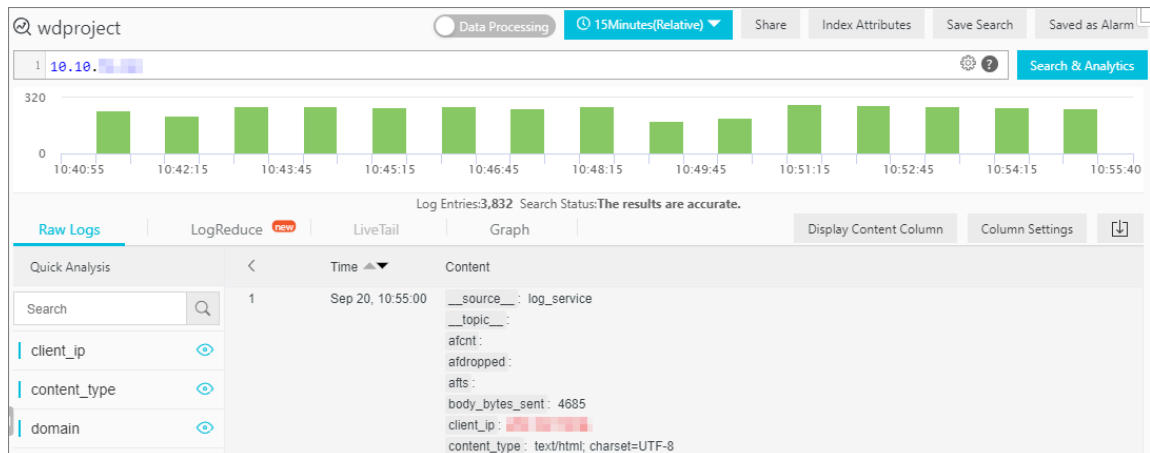
View the number of the messages processed by a client

- **Procedure**
 1. Enter the IP address of the client in the search box. Format: `$ ClientIP` . If you want to query certain types of operation logs, specify the operations in the search box. Example: `$ ClientIP and (SendMessage or BatchSendM essage)`.
 2. Select the time range of which you want to query logs, and click **Search & Analysis**. Then, you can view all the operation logs of the client.

- **Example:**

View the number of the messages processed by the client whose IP address is 10.10.XX.XX.

- **Statement:** `10 . 10 . XX . XX`
- **Result:** As shown in the following figure, three message processing operations are performed in the time period you specified.



Real-time computing and offline computing

- **Real-time computing:** analyzes the MNS logs in real time through Spark, Storm, Realtime Compute, or Consumer Library. The following are some use cases:
 - What are the IP addresses of the top 10 clients that send the most messages or consume the most messages in a queue?
 - Is the balance kept between the speed of message production and consumption? Do any consumers have difficulties in processing latency?
- **Offline computing:** analyzes the logs within a large time span through MaxCompute, E-MapReduce, or Hive.
 - What is the average latency from message publish to message consumption over the last week?
 - How is the performance changed after the upgrade?

2.8 Query and analyze AppLogs

Features of AppLogs

- **Complete information:** AppLogs are provided by programmers, covering key locations, variables, and exceptions. Technically, over 90% of bugs in the production environment can be located based on AppLogs.
- **Arbitrary formats:** One piece of code is often developed by multiple programs who have their preferred formats. Therefore, the formats are difficult to be unified. Style inconsistency is also seen in logs introduced from third-party databases.
- **Similarity among AppLogs:** Despite of the arbitrary formats, different AppLogs share things in common. For example, the following information is required for Log4J logs:
 - Time
 - Level
 - File or class
 - Line number
 - Thread ID

Challenges in processing AppLogs

- **Large data volume**

The size of AppLogs is generally one order of magnitude larger than that of access logs. Assume that a website has one million independent PVs each day, each PV involves about 20 logic modules, and 10 major logic points in each logic module need to be logged.

The total number of the logs is calculated by using the following formula:

$$1,000,000 \times 20 \times 10 = 2 \times 10^8 \text{ entries}$$

Assume that the length of each entry is 200 Bytes. The total storage size is calculated by using the following formula:

$$2 \times 10^8 \times 200 = 4 \times 10^{10} \text{ Bytes} = 40 \text{ GB}$$

The data size grows as the business system becomes increasingly complex. It is common for a medium-sized website to generate 100 to 200 GB of log data every day.

- **Multiple distributed applications**

Most applications are stateless and run in different frameworks, such as:

- Server
- Docker (container)
- Function Compute (Container Service)

The numbers of corresponding instances vary from a few to thousands. Multiple instances require a cross-server log collection scheme.

- **Complex runtime environments**

Programs are running in different environments, such as:

- Application logs are stored in Docker containers.
- API logs are stored in Function Compute.
- Old system logs are stored in traditional IDCs.
- Mobile-side logs are stored in users' mobile devices.
- Mobile web logs are stored in browsers.

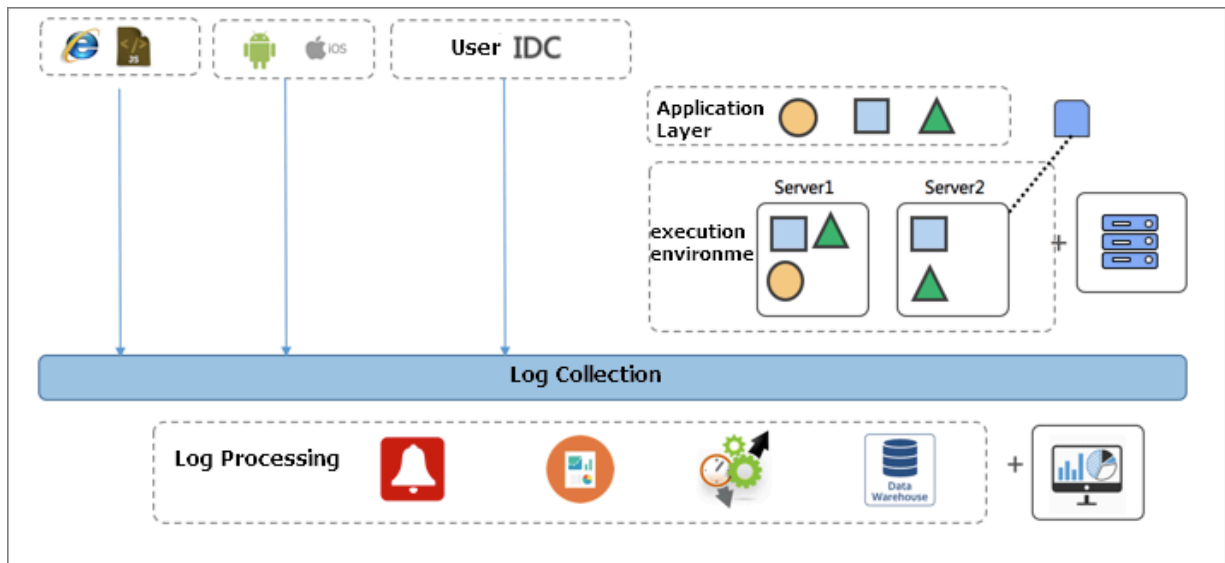
To have a complete picture of the log data, you need to unify and store the logs in a single place.

Schemes

Unified data storage

Purpose: to collect data from different sources into a central place to facilitate subsequent operations.

You can create a project in [Log Service](#) to store AppLogs. Log Service supports over 30 collection methods, such as tracking in physical servers, entering JavaScript code on the mobile web side, and exporting logs on servers.



Apart from writing logs by using methods like SDK, Log Service provides a convenient, stable, and high-performance agent, namely, Logtail, for server logs. Logtail provides two versions for Windows and Linux. Once you have defined the machine group and completed the log collection configuration, server logs can be collected in real time.

After the log collection configuration is complete, you can operate on logs in the project.

Compared with other log collection agents, such as Logstash, Flume, FluentD, and Beats, Logtail has following advantages:

- **Easy to use:** provides APIs and remote management and monitoring capabilities. Logtail is designed with the rich experience of Alibaba Group in million-level server log collection and management. This enables you to configure a collection point for hundreds of thousands of devices in seconds.
- **Adaptive to different environments:** Logtail supports the public network, VPCs, and user-created IDCs. The HTTPS and resumable upload features make it possible to integrate with data on the public network.
- **Great performance with a little consumption of resources:** With years of refinement, Logtail is superior to its open-source competitors in performance and resource consumption.

Quick fault locating

Purpose: to ensure that the time it takes to locate problems is constant, regardless of how the data volume increases and how servers are deployed.

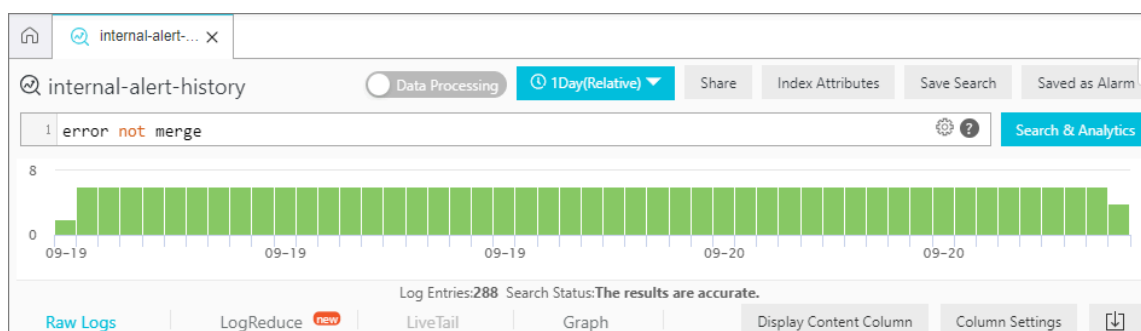
For example, quickly locate an order error and a long latency issue out of TBs of data every week. The process also involves filtering and investigation based on multiple conditions.

1. For example, you can investigate AppLogs with latency details for the requests with a latency of over 1 second and methods starting with Post.

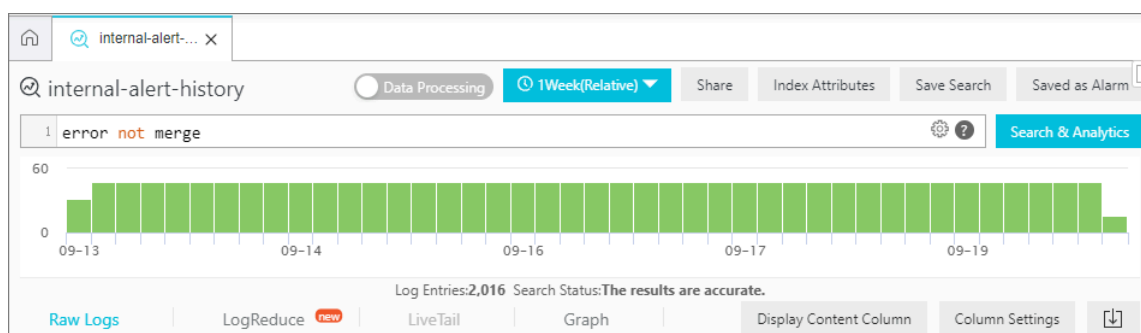
Latency > 1000000 and Method = Post *

2. Search for logs that contain the keyword "error" and do not contain the keyword "merge".

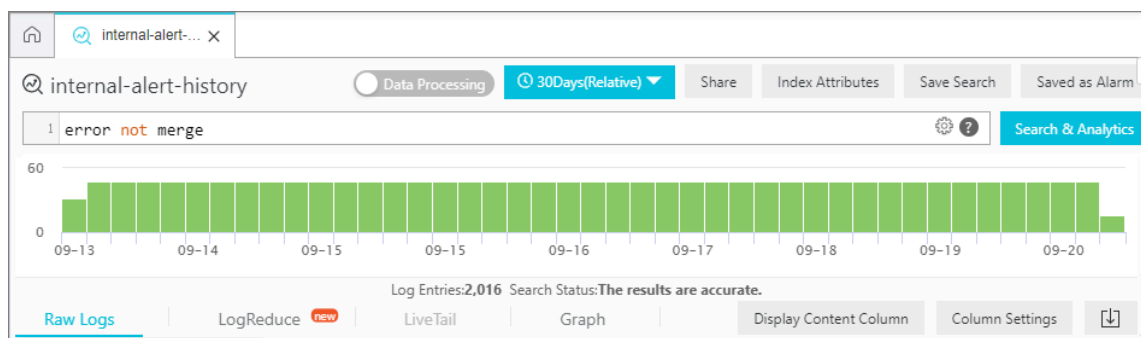
- Result of a day



- Result of a week



- Result a larger time span

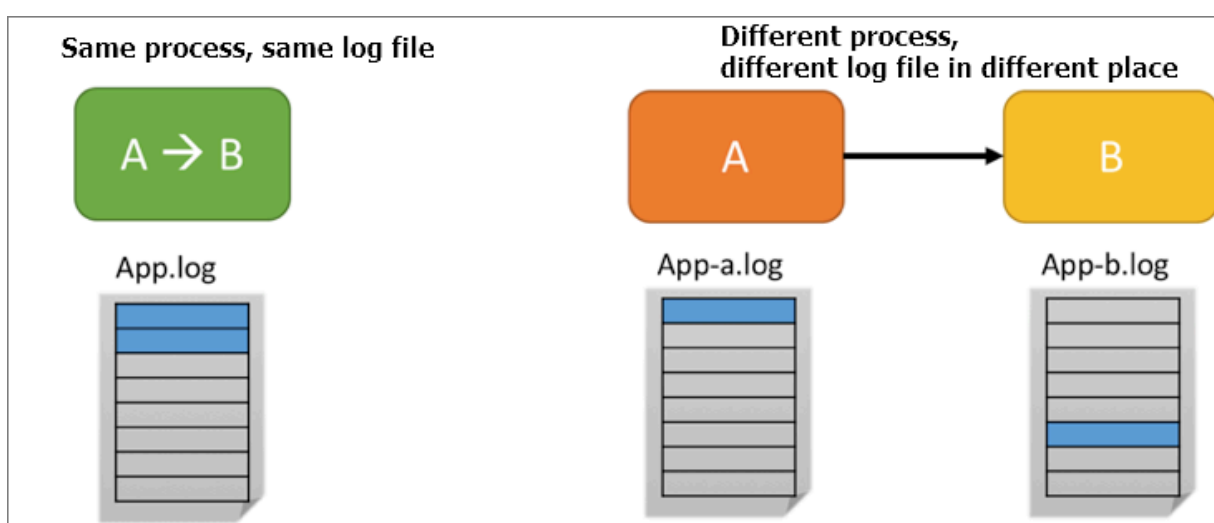


The results of any time span can be returned in less than 1 second.

Association analysis

There are two types of association: intra-process association and inter-process association. The difference between the two types is as follows:

- **Intra-process association:** This type of association is simple because the old and new logs of a function are stored in the same file. In a multi-thread process, you can filter logs by thread ID.
- **Cross-process association:** Usually, it is hard to find clear clues when dealing with logs from different processes. The association is generally performed by passing the TracerId parameter to Remote Procedure Call (RPC).



- **Context association**

Locate an error log by running a keyword-based query in the Log Service console.

Click Context View, and see the preceding and following results.

- You can click Old and New for more results.
- You can also highlight the results by entering keywords.

- **Cross-process association**

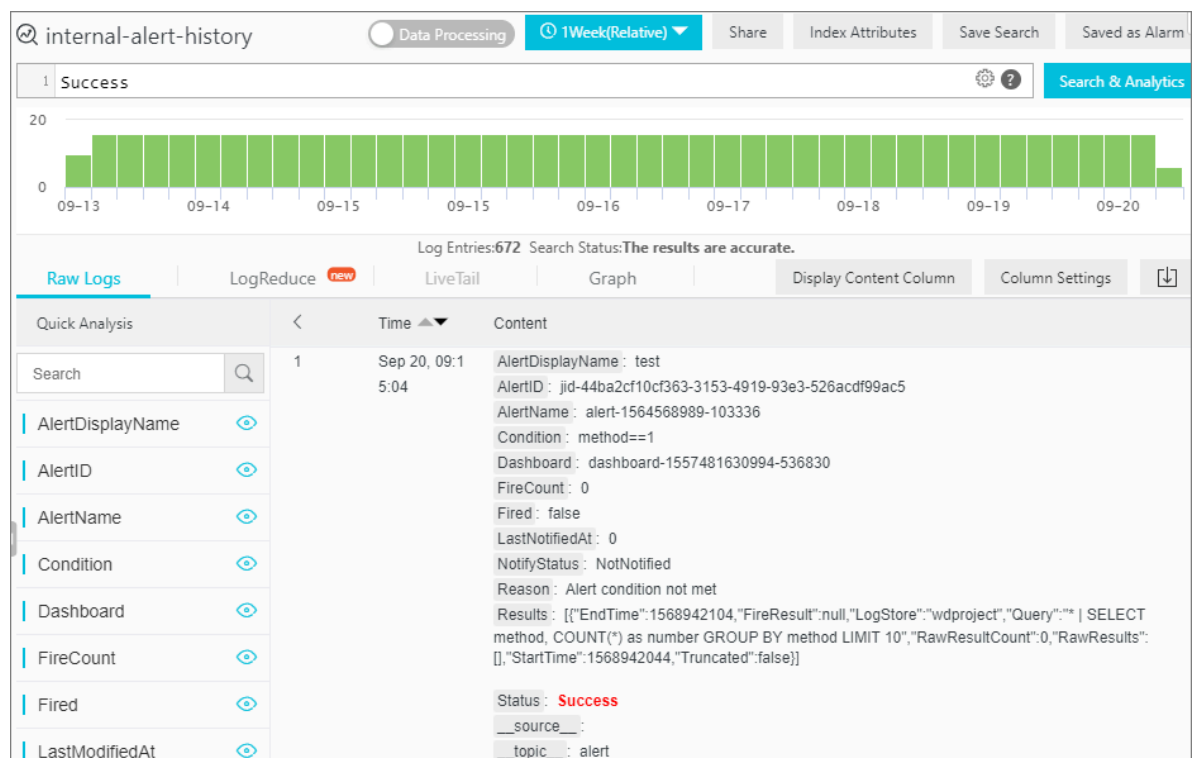
The concept of cross-process association, also known as tracing, can be dated back to the famous article [Dapper, a Large-Scale Distributed Systems Tracing Infrastructure](#) by Google in 2010. Inspired by the article, developers from the open

source sector created many affordable versions of tracers. Currently, the following versions are well-known:

- Dapper (Google): the foundation for all tracers.
- StackDriver Trace (Google): compatible with ZipKin.
- Zipkin: an open-source tracing system by Twitter.
- Appdash: a tracer of the Golang version.
- Hawkeye: developed by the Middleware Technology Department of Alibaba Group.
- X-ray: introduced at AWS re:Invent 2016.

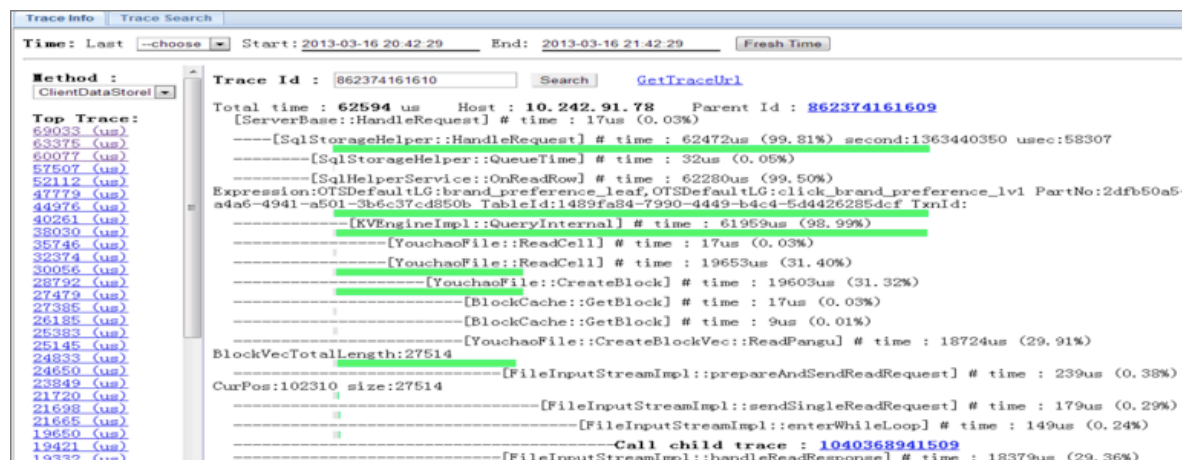
Applying a tracer from scratch is easier than in an existing system because of the costs and challenges in adapting it to the system.

Based on Log Service, you can implement a basic tracing feature. To obtain all required logs, you can record associative fields such as Request_id and OrderId in logs of different modules and search for the fields in various Logstores.



For example, you can search logs of front-end servers, back-end servers, payment systems, and order systems by using SDKs. After the results are returned, you can

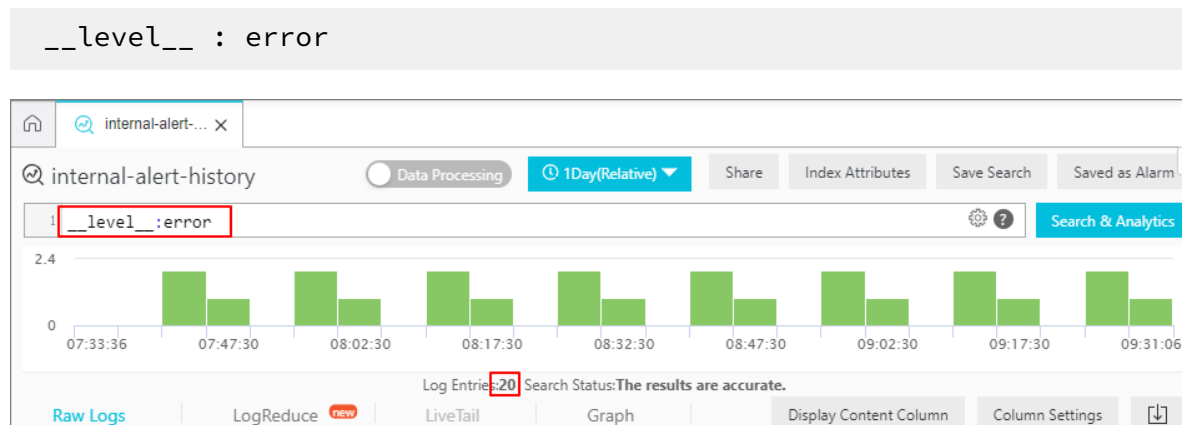
create a page on the front end to associate the cross-process calls. The following figure shows the tracing system built based on Log Service.



Statistical analysis

After the specific logs are located, you can analyze the logs, for example, calculating the types of online error logs.

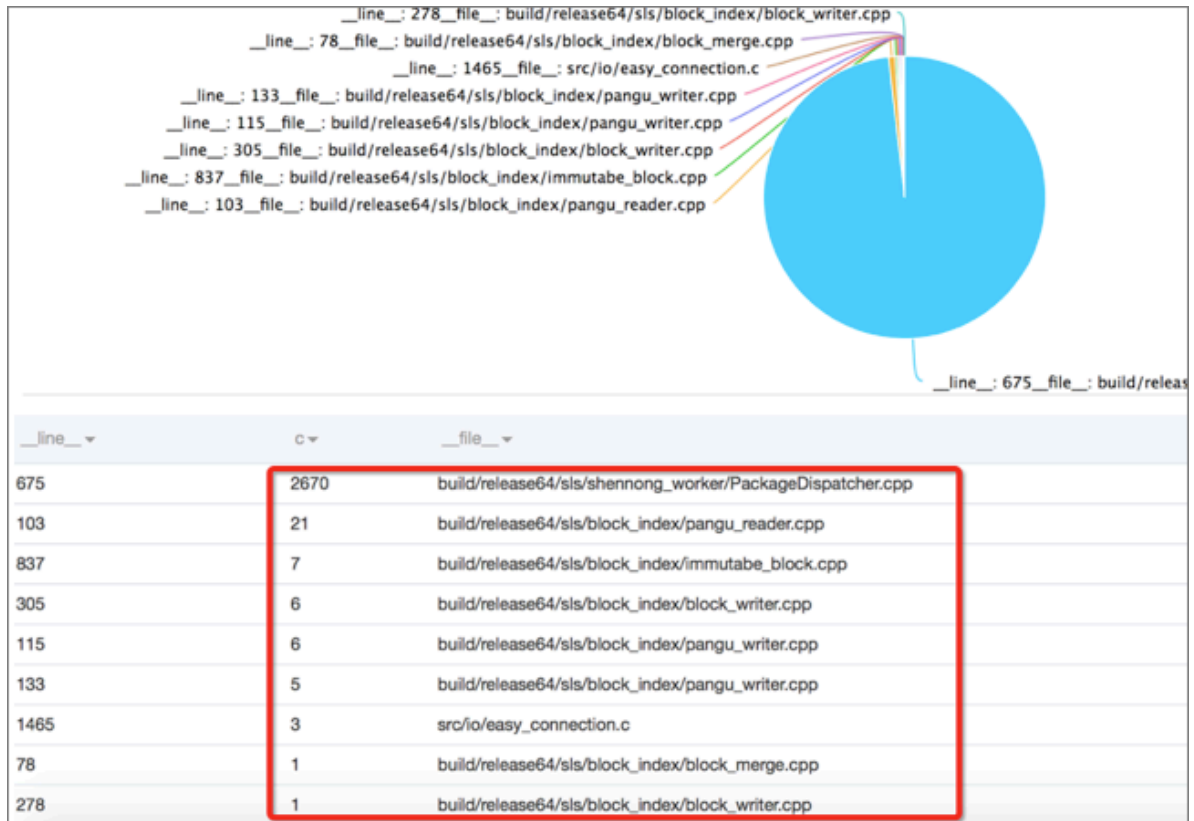
1. You can query logs by `__level__`. For example, 20 errors are found within one day.



2. To determine the unique log type, you can analyze and aggregate data based on the file and line fields.

```
__level__ : error | select __file__, __line__, count (*)
as c group by __file__, __line__ order by c desc
```

Then, the results of all types and locations of errors can be returned.



Besides, you can locate and analyze IP addresses with certain error codes or high latency.

Other features

- Log backup for auditing

You can back up the logs to OSS, Infrequent Access (IA) storage which has a lower storage cost, or MaxCompute.

- Keyword alerting

Alerts can be reported in the following ways:

- [Enable alerting by Log Service.](#)
- [Enable alerting by CloudMonitor.](#)

- Log query permission management

You can grant permissions to RAM users or a user group to isolate the development environment and the production environment.

Price and cost: In this scheme, AppLogs mainly use the LogHub and LogSearch features of Log Service. Compared with an open source scheme, this scheme is an easy-to-use scheme with only 25% of the cost of an open source scheme. This improves the development efficiency.

2.9 Perform association query and analysis on logs and the data from databases

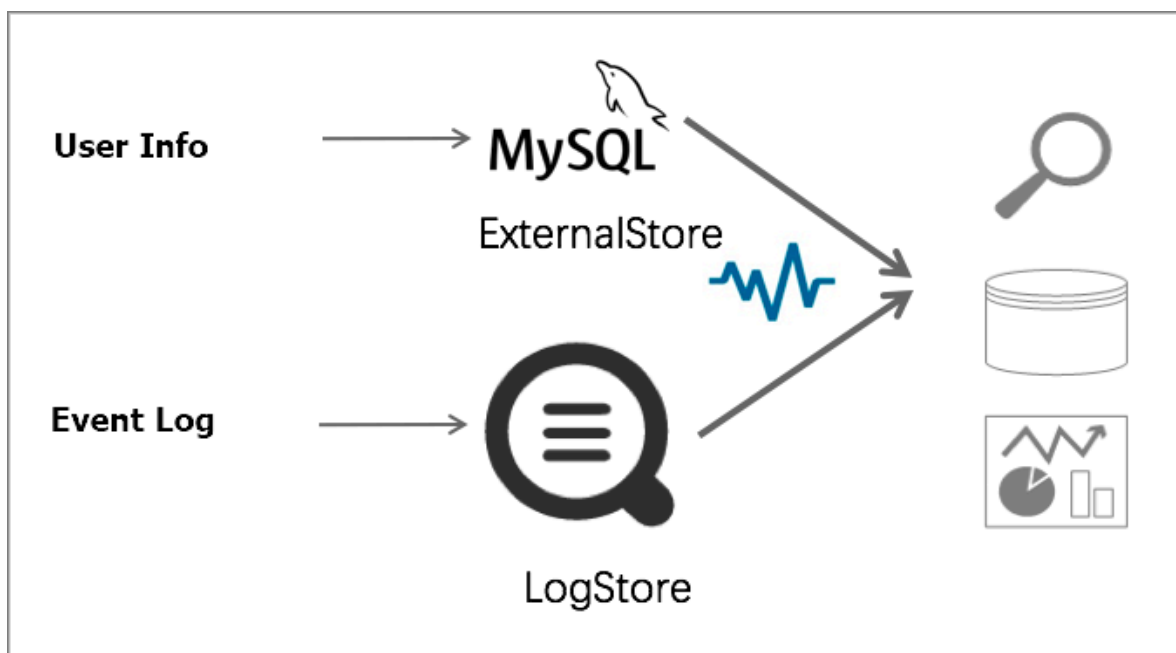
The scenario where data is distributed across different regions is common in log analysis. In this scenario, you need to perform hierarchical analysis on user data based on both the logs and the data from databases. The result is written to databases and can be queried through report systems. Association query of Logstores and databases is required.

Context

- User log data: Taking game logs as an example, a classic game log contains properties such as the operation, target, blood, mana, network, payment method, click location, status code, user ID.
- User metadata: Logs record incremental events. However, static user information, such as the gender, registration time, and region, is fixed and hard to be obtained on the client. The static user information cannot be recorded in logs. The static user information is known as user metadata.
- Association analysis of Logstores and ApsaraDB RDS for MySQL instances: The query and analysis engine of Log Service can perform association query and analysis of Logstores and ExternalStores. You can use the SQL JOIN syntax to associate logs and metadata to analyze metrics related with user properties. In addition to referencing ExternalStores for association query and analysis, Log

Service also supports writing results to ExternalStores such as ApsaraDB RDS for MySQL instances. This allows you to further process the results.

- **Logstore:** allows you to collect, store, query, and analyze logs.
- **ExternalStore:** maps data to ApsaraDB for RDS tables. Developers can store the user information in the tables.



Procedure

1. Collect logs to Log Service.

- Collect logs on mobile clients by using the [Android SDK](#) or [iOS SDK](#).
- Collect logs on servers by using [Logtail](#).

2. Create a user property table.

Create a table named `chiji_user` to store user properties including the ID, username, gender, account balance, registration time, and province.

```
CREATE TABLE `chiji_user` (
  `uid` int(11) NOT NULL DEFAULT '0',
  `user_nick` text,
  `gender` tinyint(1) DEFAULT NULL,
  `age` int(11) DEFAULT NULL,
  `register_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `balance` float DEFAULT NULL,
  `province` text, PRIMARY KEY (`uid`))
```

```
) ENGINE = InnoDB DEFAULT CHARSET = utf8
```

3. Create an ExternalStore.

- a. Install the Alibaba Cloud Command Line Interface (CLI) before creating an ExternalStore.

```
pip install -U aliyun-cli
```

- b. Specify the project and create the configuration file /root/config.json for the ExternalStore.

```
aliyunlog log create_external_store --project_name="log-rds-demo" --config="file:///root/config.json"
```

- c. Set the ExternalStore name and parameters in the configuration file. For an ApsaraDB for RDS instance in a VPC, you need to set the vpc-id, instance-id, host, port, username, password, db, table, and region parameters.

```
{
  "externalStoreName": "chiji_user",
  "storeType": "rds-vpc",
  "parameter": {
    "vpc-id": "vpc-m5eq4irc1p-ucpk85f-****",
    "instance-id": "rm-m5ep2z5781-4qs-****",
    "host": "example.com",
    "port": "3306",
    "username": "testroot",
    "password": "123456789",
    "db": "chiji",
    "table": "chiji_user",
    "region": "cn-qingdao"
  }
}
```

4. Add the specified IP address to the whitelist.

- For an ApsaraDB for RDS instance, add the IP address `100.104.0.0/16` to the whitelist of IP addresses allowed to access the instance.
- For an ApsaraDB RDS for MySQL instance, add the IP address to the security group.

5. Perform association analysis.

- Analyze the gender distribution of active users.

Use the JOIN syntax to associate logs and user properties through the identical value of the userid field in logs and the uid field in the ApsaraDB for RDS instance.

```
* | select case gender when 1 then 'male' else 'female' end as gender, count(1) as pv from log
```

```
l join chiji_user u on l.userid = u.uid group
by gender order by pv desc
```

- Analyze the activity level of users in different provinces.

```
* | select province , count ( 1 ) as pv from log l
join chiji_user u on l.userid = u.uid group by
province order by pv desc
```

- Analyze the consumption trends of users of different genders.

```
* | select case gender when 1 then ' male ' else '
female ' end as gender , sum ( money ) as money from
log l join chiji_user u on l.userid = u.uid
group by gender order by money desc
```

6. Store the result of the association query and analysis.

- a. Create a result table that stores page views (PVs) per minute.

```
CREATE TABLE `report` (
  `minute` bigint ( 20 ) DEFAULT NULL ,
  `pv` bigint ( 20 ) DEFAULT NULL
) ENGINE = InnoDB DEFAULT CHARSET = utf8
```

- b. Create an ExternalStore for the report table by following the instructions in step 3, and save the result in the table.

```
* | insert into report select __time__ - __time__ %
300 as min , count ( 1 ) as pv group by min
```

The execution result of the SQL statement displays the number of the rows that are written to the ApsaraDB for RDS instance.

```
[mysql> select * from report;
+-----+-----+
| minute | pv    |
+-----+-----+
| 1526448600 | 3000 |
| 1526448540 | 9900 |
| 1526448780 | 3100 |
| 1526448480 | 5400 |
| 1526448720 | 3000 |
| 1526448960 | 3000 |
| 1526448900 | 3000 |
| 1526449080 | 3000 |
| 1526449140 | 3000 |
| 1526448660 | 2900 |
| 1526449260 | 3000 |
```

2.10 Perform association query and analysis on logs and tables from OSS

The scenario where the information in logs is incomplete is common in log analysis. For example, logs contains the information about user clicks, but lacks user properties such as the registration and fund information. In some log analysis scenarios, such as analyzing the impact of regions on users' payment habits, you need to analyze the properties and behavior of users.

Context

Log Service works together with Object Storage Service (OSS) to provide the association analysis feature that has the following benefits:

- Low cost
 - Log Service supports disparate data sources. Data can be stored in different storage systems based on the data features. This lowers the cost. Data that involves infrequent changes can be stored in OSS buckets. You only need to pay for the storage service. If the data is stored in an ApsaraDB RDS for MySQL instance, you also need to pay for the computing service.
 - OSS is a storage service provided by Alibaba Cloud. Data can be read from OSS buckets over the internal network without any traffic costs.

- Less workload

With the lightweight association analysis platform, you do not need to store all data in one storage system. This saves your energy.

- Efficiency

- If you run a SQL statement to analyze data, you can get the result within seconds
-
- You can define the common view as a report so that you can view the result directly.

Procedure

1. Upload a CSV file to OSS.

- a. Define a property file that contains the userid, nick, gender, province, and age properties.
- b. Save and name the file as `user . csv`. Use `ossutil` to upload the file to OSS.

```
ossutil put ~/ user . csv oss :/ testosscon nector /  
user . csv
```

2. Define the ExternalStore.

Run the following SQL statement to define a virtual external table named `user_meta`:

```
* | create table user_meta ( userid bigint , nick  
varchar , gender varchar , province varchar , gender  
varchar , age bigint ) with ( endpoint = ' example . com ',  
accessid = '< youraccess id >', accesskey = '< accesskey >', bucket  
= ' testosscon nector ', objects = ARRAY [ ' user . csv ' ], type = '  
oss ')
```

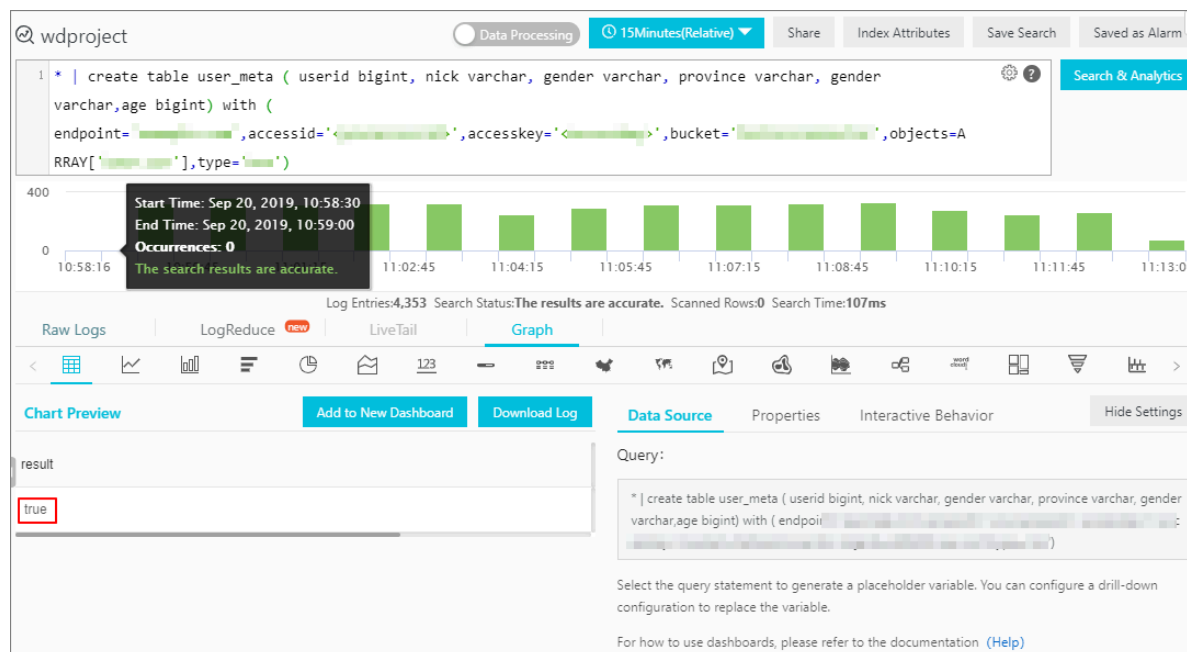


Note:

- In the preceding SQL statement, you need to specify the following information:
 - Table schema: columns involved in the table and the properties of each column.
 - OSS access information: the OSS domain name, AccessKey ID, and AccessKey secret.
 - OSS object information: the bucket where the user.csv object is stored and the object path.

- The value of the objects property is an array of multiple objects.

The result of true indicates that the external table is created.



Run the `select * from user_meta` statement to view the data in the table.

3. Perform association analysis.

The raw logs contain the user ID information. You can run the following SQL statement to associate the ID field in logs with the userid field in OSS objects to complete the information in logs:

```
* | select * from chiji_acce sslog l join user_meta1 u
on l.userid = u.userid
```

- Analyze the access information of users of different genders.

```
* | select u.gender, count(1) from chiji_acce sslog l
join user_meta1 u on l.userid = u.userid
group by u.gender
```

- Analyze the access information of users of different ages.

```
* | select u.age, count(1) from chiji_acce sslog l
join user_meta1 u on l.userid = u.userid
group by u.age
```

- Analyze the access trends of users of different ages in different time periods.

```
* | select date_trunc('minute', __time__) as minute,
count(1), u.age from chiji_acce sslog l join
user_meta1 u on l.userid = u.userid
group by u.age, minute
```

3 Consumption

3.1 Use Function Compute to cleanse log data

LogHub of Log Service is a streaming data center. Logs can be consumed in real time after being written to LogHub. The extract-transform-load (ETL) process of Log Service can process such streaming data in seconds.

Overview

ETL is the process of extracting data from the source, transforming data, and then loading data to the destination. The traditional ETL process is an important part in building a data warehouse. You need to extract required data from the data source, cleanse the data, and then load data to the destination data warehouse based on the pre-defined data warehouse model. Under the ever-growing business requirements, different systems need to exchange large amounts of data. Data flows in different systems. This is helpful in fully exploring the value of log big data.

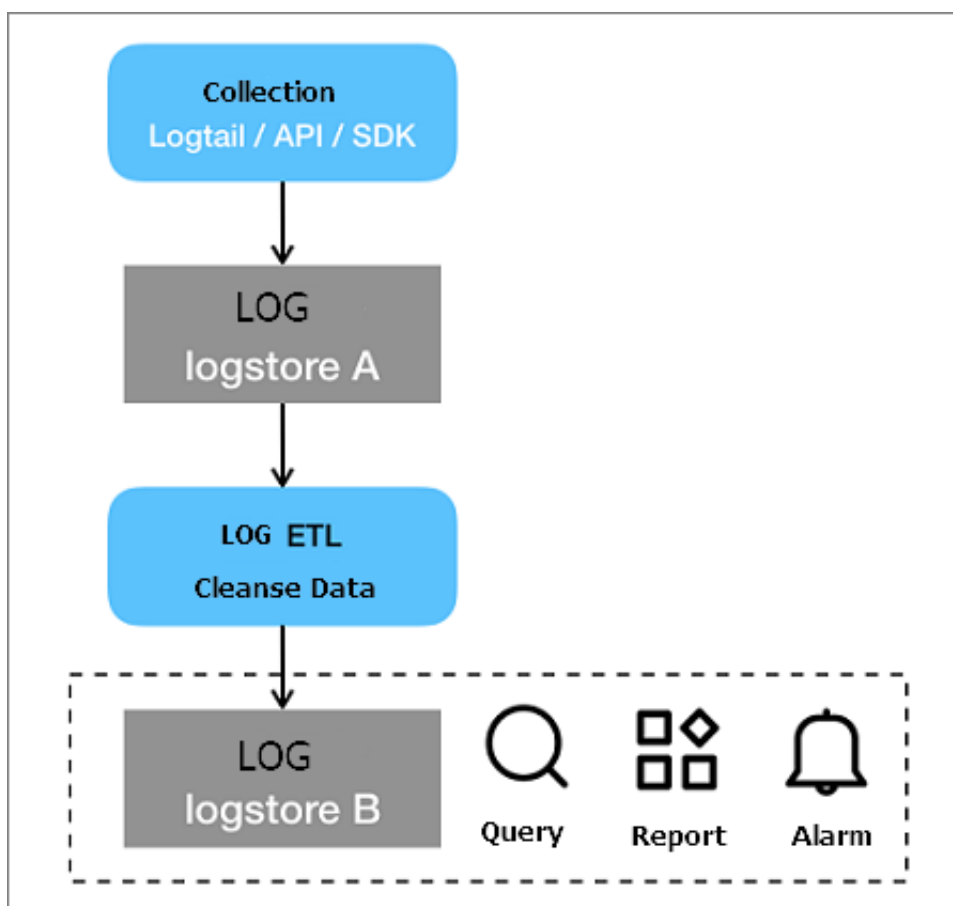
Benefits of the Log Service and Function Compute ETL process

- Data is collected, stored, processed, and analyzed in a uniform manner.
- Data processing is fully managed, triggered by time, and automatically retried.
- Shards can be scaled out to meet the resource requirements of big data.
- Data is processed in Function Compute, which provides elastic resources and supports the pay-as-you-go billing method.
- The ETL process is transparent to users and provides the logging and alerting features.
- Built-in function templates are continuously added to reduce the cost of function development under mainstream requirements.

Scenarios

Data cleansing and processing

Log Service allows you to quickly collect, process, query, and analyze logs.



Data shipping

You can ship data to the destination and build data pipelines between big data products in the cloud.

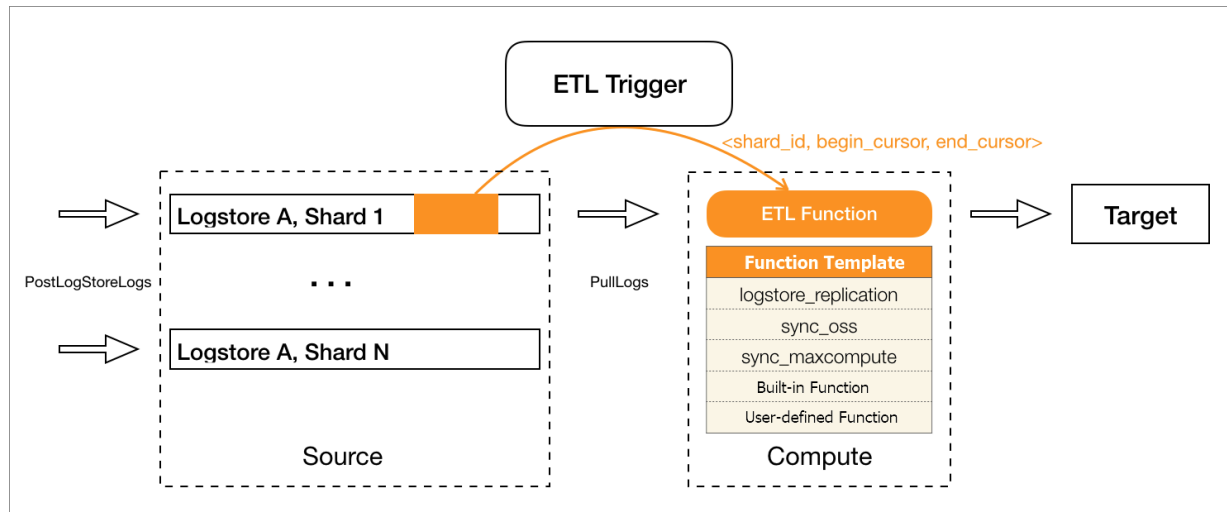
ETL model

The ETL model is a stream-based, real-time data stream processing model. The ETL trigger polls all shards in the source Logstore to detect the position where data is written and regularly generates trituple information to trigger relevant functions. The trituple information is used to identify the range of data to be processed in the current ETL task.

Shards can be scaled out to ensure the auto scaling of the ETL process. Tasks are triggered by a timer to ensure that data is continuously loaded.

During the implementation of the ETL process, in consideration of the flexibility of user-defined functions (UDFs), functions are called in Function Compute to process data. Function Compute provides pay-as-you-go, auto scaling, and custom code execution to meet the requirements of many cloud users. In addition, considering the end-to-end latency of user data, big data throughput, and SQL usability, the Log

Service ETL runtime will involve stream computing engines, such as Alibaba Cloud StreamCompute, in the future to meet more user requirements.



ETL logs

- ETL process logs

ETL process logs record the key points and errors of each step during the implementation of the ETL process, including the start time and end time of a step, completion status of initialization, and module error information. The purpose of recording process logs is to know the status of the ETL process at any time and the causes of errors. Logs generated during the execution of functions record the key points and exceptions of data processing.

- ETL scheduling logs

ETL scheduling logs record only the start time and end time of an ETL task, whether the task is successful, and the information returned when the task is successful. If an error occurs, ETL error logs are generated, and an alert email or SMS message is sent to the system administrator. Based on scheduling logs, reports can be created to display the overall status of the ETL process.

Application examples

Software built based on HTTP servers such as Nginx and Apache can record access logs for each user. Using the Log Service and Function Compute ETL process, you can analyze the regions where your services are used and the links used by users in these regions to access your services.

For data analysis engineers, the workload of the ETL process accounts for 60% to 70% of the overall workload of a project. Log Service can use built-in function templates to shorten the duration of the ETL process to at most 15 minutes.

Step 1: Centralize log storage

You can use the Logtail client of Log Service to quickly collect log files from a machine, for example, Nginx.

Nginx access logs collected by the client are all stored in a Logstore of Log Service. As shown in the following figure, the value of the forward field indicates the IP address from which the user sends the request.

Time	Content
10-16 11:38:21	<pre> source: 192.168.1.100 topic: forward: 192.168.1.100 host: 192.168.1.100 ip: 192.168.1.100 latency: 0.083 method: GET outflow: 232594 port: 36245 protocol: HTTP/1.1 readtime: http_x_readtime ref: - site: status: 200 time: 16/Oct/2017:11:38:21 +0800 ua: Mozilla/5.0 (Linux; Android 6.0.1; X800+ Build/BEXCNFN5902605111S; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/53.0.2785.49 Mobile MQQBrowser/6.2 TBS/043520 Safari/537.36 MicroMessenger/6.5.16.1120 NetType/4G Language/zh_CN upstream_time: 0.071 uri: / url: /?utm_content=se_919625&from=timeline&isappinstalled=0 </pre>

Step 2: Process data in the cloud

1. Use a built-in template to create the ETL function.
2. Create a Log Service trigger for the function.

The following figure shows the configuration of the Log Service trigger.

Set the data source to the Logstore that stores Nginx access logs as described in step 1, that is, `etl - test / logstore : nginx_access_log` in this example.

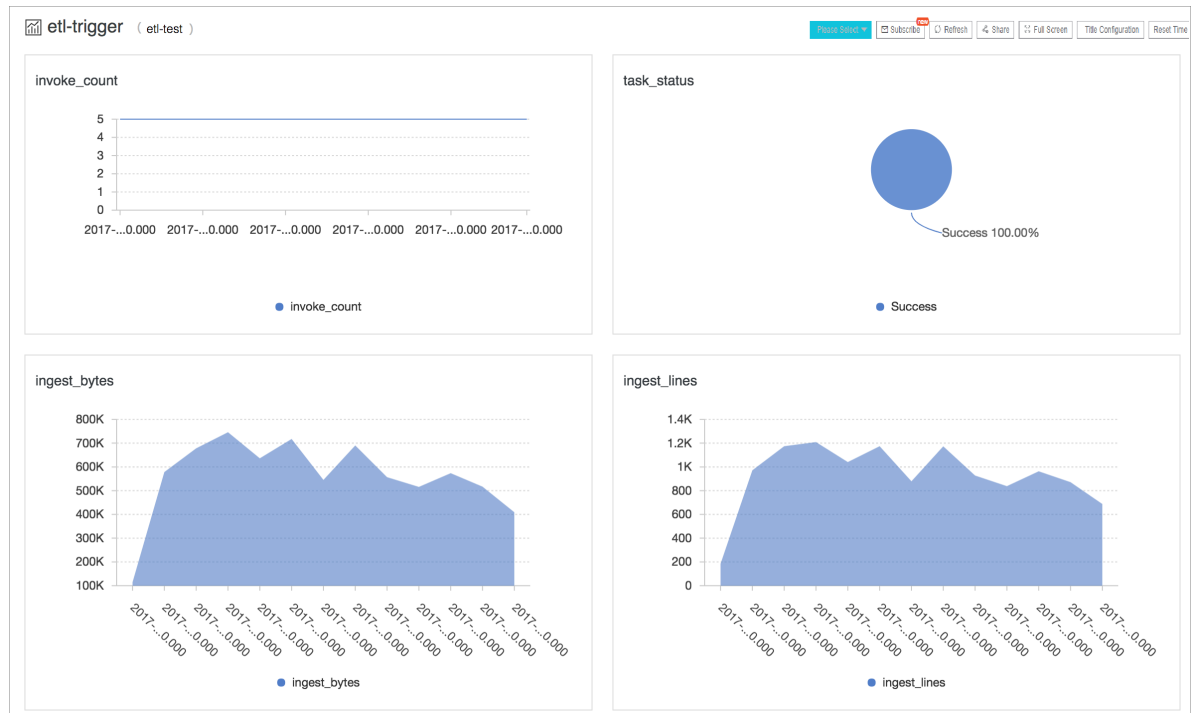
Log Service polls data in the source Logstore. When data is continuously generated, an ETL task is created every 60 seconds to call the ETL function and process data. The trigger and execution results of the function are recorded in the `etl-trigger-log` Logstore that stores trigger logs.

The implementation and features of functions vary with the function configuration. For more information about the configuration items of the `ip-lookup` template, see its README file.

3. Save the configuration and wait 1 minute for the ETL task to start.

Pay attention to the generated ETL process logs and scheduling logs. Based on the preceding configuration, the two types of logs are stored in the `etl-function-log` and `etl-trigger-log` Logstores, respectively.

You can also use query statements to create reports based on ETL logs.



- The chart in the upper-left corner shows the number of times that the ETL function is triggered per minute. This chart is created by using the following query statement:

```
project_name : etl - test and job_name : ceff019ca3
d077f85aca ad35bb6b9b ba65da6717 | select from_unixt
ime ( __time__ - time % 60 ) as t , count ( 1 ) as
invoke_cou nt group by from_unixt ime ( __time__ - time
% 60 ) order by t asc limit 1000
```

- The chart in the upper-right corner shows the percentages of ETL tasks that are successful and fail. This chart is created by using the following query statement:

```
project_name : etl - test and job_name : ceff019ca3
d077f85aca ad35bb6b9b ba65da6717 | select task_statu s ,
count ( 1 ) group by task_statu s
```

- The chart in the lower-left corner shows the number of bytes of logs read from the source Logstore every 5 minutes. This chart is created by using the following query statement:

```
project_name : etl - test and job_name : ceff019ca3
d077f85aca ad35bb6b9b ba65da6717 and task_status :
Success | select from_unixtime ( __time__ - time % 300
) as t , sum ( ingest_bytes ) as ingest_bytes group
by from_unixtime ( __time__ - time % 300 ) order by
t asc limit 1000
```

- The chart in the lower-right corner shows the number of lines of logs read from the source Logstore every 5 minutes. This chart is created by using the following query statement:

```
project_name : etl - test and job_name : ceff019ca3
d077f85aca ad35bb6b9b ba65da6717 and task_status :
Success | select from_unixtime ( __time__ - time % 300
) as t , sum ( ingest_lines ) as ingest_lines group
by from_unixtime ( __time__ - time % 300 ) order by
t asc limit 1000
```

Step 3: Model data after data processing

Nginx access logs on the machine are collected by the Logtail client to the source Logstore in real time, processed by the ETL function in a quasi-real-time manner, and then written to the destination Logstore. The following figure shows the data that is processed by the ETL function and contains the IP address information.

Time	Content
1	<pre>10-16 11:38:21 __source__: 10.177.162.203 __topic__: city: country: china forward: 10.136.90.160 host: [redacted] ip: 172.29.94.109 isp: mobile latency: 0.083 method: GET outflow: 232594 port: 36245 protocol: HTTP/1.1 province: shanxi readtime: http_x_readtime ref: - site: http_x_readtime status: 200 time: 16/Oct/2017:11:38:21 +0800 ua: Mozilla/5.0 (Linux; Android 6.0.1; X800+ Build/BEXCNFN5902605111S; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/53.0.2785.49 Mobile MQQBrowser/6.2 TBS/043520 Safari/537.36 MicroMessenger/6.5.16.1120 NetType/4G Language/zh_CN upstream_time: 0.071 uri: / url: /?utm_content=se_919625&from=timeline&isappinstalled=0</pre>

Compared with the data before processing, the data after processing adds four fields : country, province, city, and isp. Based on the four fields, you can know that the

request whose IP address is 117.136.90.160 comes from Taiyuan, Shanxi, China, and the carrier is China Mobile.

Then, you can use the log analysis feature of Log Service to query the cities from which requests of specific IP addresses come and the distribution of Internet service providers (ISPs) in a period. Use the following two query statements to create reports:

- ```
* | select city , count (1) as c group by city order
by c desc limit 15
```
- ```
* | select isp , count ( 1 ) as c group by isp order  
by c desc limit 15
```

3.2 Build a monitoring system

As the important infrastructure of Alibaba Cloud, Log Service supports the collection and distribution of log data in all clusters of Alibaba Cloud. Many applications, such as Table Store, MaxCompute, and CNZZ, use Logtail of Log Service to collect log data, call API operations to consume log data, and then import the data to the downstream real-time statistics system or offline system for further analysis and statistics. As the infrastructure, Log Service has the following features:

- **Reliability:** After serving the internal users of Alibaba Group and taking on challenges during the Double 11 Shopping Festival for many years, Log Service has proven its capability to ensure data reliability and integrity.
- **Scalability:** Log Service can add shards to quickly and dynamically extend the processing capability when data traffic increases.
- **Convenience:** Log Service supports one-click management. For example, it allows you to collect logs from tens of thousands of machines with one click.

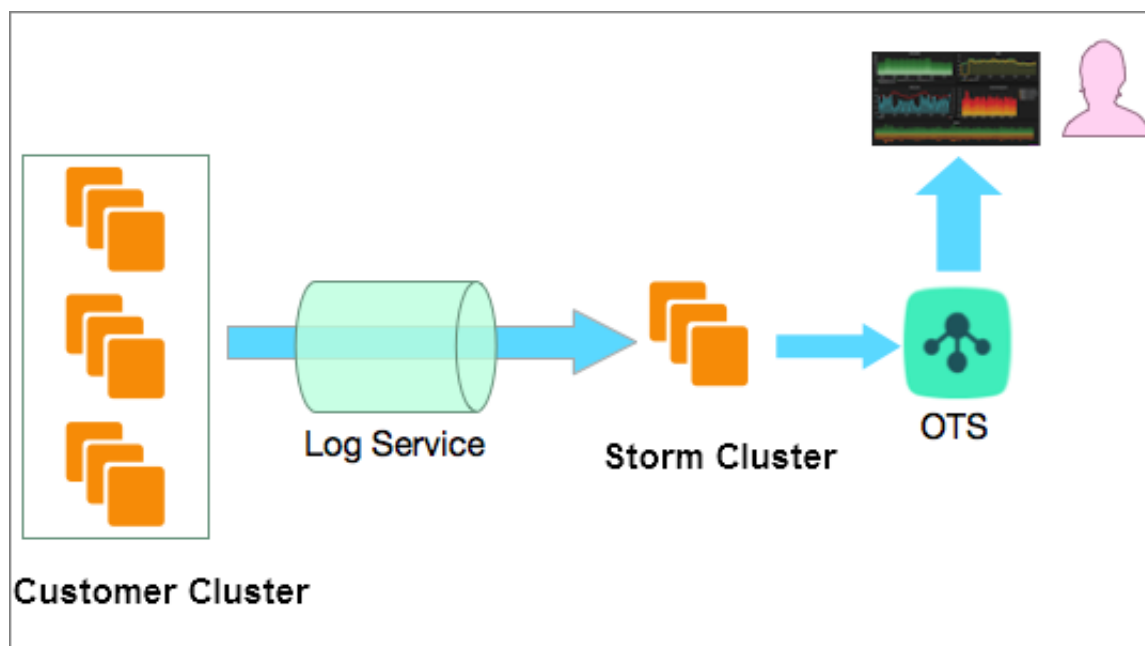
Log Service helps you collect logs, unifies the log format, and provides API operations for downstream systems to consume data. Downstream systems can be connected to multiple systems to consume data in various ways. For example, they can use Spark and Storm to compute data in real time, or use Elasticsearch to query data. In this way, data is collected once and consumed multiple times. Among many data consumption scenarios, monitoring is the most common one. This topic introduces a monitoring system that is provided by Alibaba Cloud based on Log Service.

Log Service collects monitoring data from all clusters as logs. It solves the problems of multi-cluster management and log collection from heterogeneous systems. Monitoring data is transformed into logs of the same format and sent to Log Service.

Log Service provides the following features for the monitoring system:

- **Unified machine management:** After Logtail is installed on each server, all subsequent operations are performed in Log Service.
- **Unified configuration management:** You only need to configure which log files are to be collected in Log Service. The configuration automatically applies to all relevant servers.
- **Structured data:** All data is formatted to fit the data model of Log Service to facilitate downstream consumption.
- **Elastic service capability:** Log Service allows you to read and write a large amount of data.

Figure 3-1: Architecture of the monitoring system



Build the monitoring system

1. Collect monitoring data.

Configure log collection in Log Service to ensure that logs are collected and sent to Log Service.

2. Use middleware and call API operations to consume data.

Use the SDK to call the PullLogs operation to consume log data in Log Service in batches and synchronize the data to the downstream real-time computing system.

3. Build a Storm real-time computing system.

Select Storm or another type of real-time computing system, configure computing rules, select the metrics for computing, and then write computing results to Table Store.

4. Display monitoring information.

Read the monitoring data stored in Table Store and display the monitoring data on the front-end GUI. Alternatively, read the monitoring data and configure alerts based on the data results.

3.3 Consume metering and billing logs

The biggest advantage of cloud services is the pay-as-you-go billing method, with no need to reserve resources. Therefore, all cloud products have metering and billing demands. This topic describes a metering and billing solution that is developed based on [Log Service](#) and used by many cloud products. You can use this solution to process hundreds of billions of metering logs every day.

Bill generation from metering logs

Metering logs record billing items. The back-end billing module computes the expenses based on the billing items and rules to generate the final bill. For example, the following raw access log records the usage of a project:

```
microtime : 1457517269 818107 Method : PostLogSto reLogs Status
: 200 Source : 10 . 145 . 6 . 81 ClientIP : 112 . 124 . 143 .
241 Latency : 1968 InFlow : 1409 NetFlow : 474 OutFlow : 0
  UserId : 44 AliUid : 1264425845 ***** ProjectNam e : app -
myapplicat ion ProjectId : 573 LogStore : perf UserAgent : ali
- sls - logtail APIVersion : 0 . 5 . 0 RequestId : 56DFF2D58B
3D939D6913 23C7
```

The metering and billing programs read the raw log and generate use data from various dimensions based on relevant rules. The following figure shows the generated use data, including the inbound traffic, number of use times, and outbound traffic.

C	D	E	F	G	H	I	J	K	L
uid	project	region	inflowsize	writecount	readcount	outflowsize	network_out	shard_size	index_size
1.47543E+15	aquilapreproductionenviro	cn-hangzhou	437	0	0	0	0	48	790
1.76991E+15	ali-tbosstest-log	cn-hangzhou-corp	0	0	0	0	0	92	0
1.72535E+15	ali-icbu-janus-log	cn-shanghai-corp	229879259	0	0	0	0	96	#####
1.62572E+15	corp-scmg-admin	cn-hangzhou-stg	15709	0	0	0	0	16	82344
1.19214E+15	dtboost	cn-hangzhou	0	0	0	0	0	48	0
1.63404E+15	md-oa	cn-beijing	0	0	0	0	0	240	0
1.85233E+15	ots_e2e_test_2nd	cn-hangzhou-stg	0	0	0	0	0	8	0
1.2358E+15	wxb-log	cn-hangzhou	466323	0	0	0	0	240	1394118
1.26443E+15	portal-b8568f751df75214dc	cn-hangzhou-stg	0	73811	0	500	73811	600	0
1.26854E+15	ali-pangumaster-log-hangz	cn-hangzhou-stg	98041	0	0	0	0	4	633933
1.85386E+15	daily	cn-hangzhou	205159	0	0	0	0	96	0
1.59853E+15	ali-alipay-siteprobe-test	cn-hangzhou-corp	0	0	0	0	0	184	0
34762362	1111111	cn-qingdao	0	0	0	0	0	96	0

Typical scenarios of billing by using metering logs

- An electric power company receives a log every 10 seconds. The log records the power consumption, peak power consumption, and average power consumption for each user ID within the 10 seconds. Based on such logs, the company generates bills for users on an hourly, daily, or monthly basis.
- A carrier receives a log from a base station every 10 seconds. The log records the services used by a mobile number within the 10 seconds, such as Internet access , voice calls, SMS messages, and voice over Internet Protocol (VoIP) calls. The log also records the data usage or duration of each service. Based on this log, the back-end billing service computes the expenses incurred during this period.
- A weather forecast API service charges user requests based on the type of the requested API operation, city, query type, and size of the query result.

Requirements and challenges

A metering and billing solution must meet the following basic requirements:

- **Accuracy and reliability:** Computation results must be accurate.
- **Flexibility:** Data can be supplemented. For example, if some data is not pushed in time, data can be supplemented and corrected to re-compute the expenses.
- **Timeliness:** Services can be billed in seconds. Services are immediately stopped if an account has any overdue payments.

Other requirements:

- **Bill correction:** If real-time billing fails, bills can be generated from metering logs.
- **Details query:** Users can view their consumption details.

Two challenges in reality:

- **Increasing data size:** The data size keeps increasing as the number of users and calls grows. One challenge is to maintain auto scaling of the architecture.

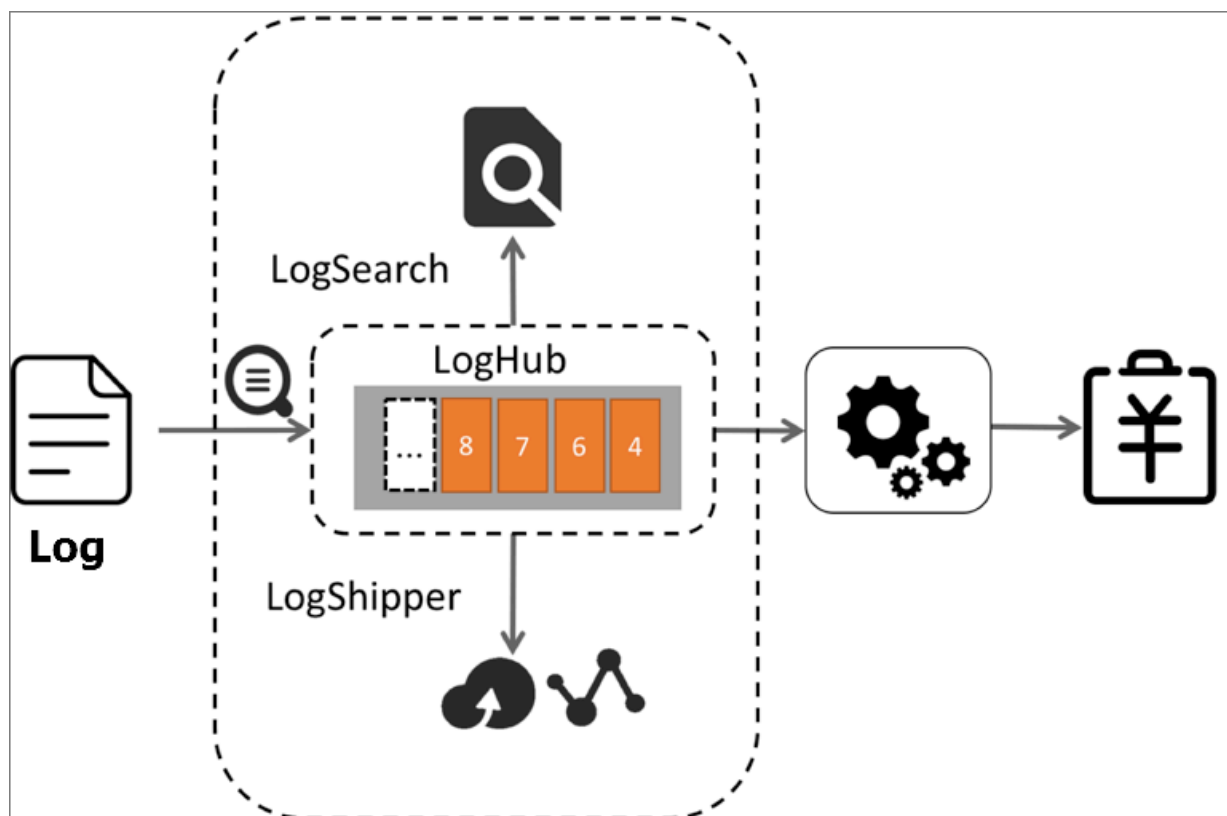
- **Fault tolerance:** The billing program may have bugs. The other challenge is to keep the metering data independent of the billing program.

The metering and billing solution described in this topic is developed by Alibaba Cloud based on Log Service. The solution has been in stable operation online for many years without any error or latency.

System architecture

In the system architecture of the metering and billing solution, LogHub of [Alibaba Cloud Log Service](#) works as follows:

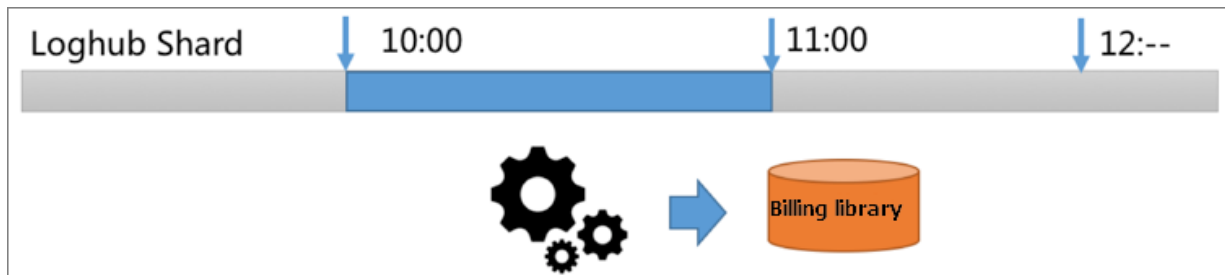
1. Collects metering logs in real time and writes metering logs to the metering program. LogHub supports more than 30 API operations and access methods to help you collect and write metering logs.
2. Allows the metering program to consume the LogHub data of a step size at regular intervals, and then compute the expenses in the memory to generate billing data.
3. Creates indexes for metering logs to meet details query requirements.
4. Pushes metering logs to Object Storage Service (OSS) for offline storage. Allows you to check accounts and take statistics of data on a T+1 basis.



The internal logic of the real-time metering program is as follows:

1. Calls the GetCursor operation to obtain the cursor of a log in a specified period, for example, 10:00 to 11:00, from LogHub.
2. Calls the PullLogs operation to consume data in this period.
3. Takes statistics of data and computes data in the memory, obtains the result, and then generates billing data.

Similarly, the specified period can be 1 minute or 10 seconds.



The performance of the metering and billing solution is analyzed as follows:

- Assume that 1 billion metering logs are generated every day, each of which is 200 bytes. The total data size is 200 GB.
- In LogHub, the SDK or agent can compress data by default. Therefore, the size of stored data is 40 GB after compression, which is at most one-fifth of the original size. The size of data generated in an hour is calculated as follows: $40/24 = 1.6$ GB.
- LogHub allows you to read a maximum of 1,000 log groups each time. The maximum size of each log group is 5 MB. On a GE network, all data generated in an hour can be read within 2 seconds.
- After the metering program takes statistics of data and computes data in the memory, metering logs generated in an hour can be consumed within 5 seconds.

Solution to a large data size

In some billing scenarios of carriers and the Internet of Things (IoT), a large number of metering logs are generated, for example, 10 trillion logs with a data size of 2 PB per day. After compression, the size of data generated in an hour is 16 TB. On a 10-GE network, it takes 1,600 seconds to read all data generated in an hour, which cannot meet the quick billing requirements.

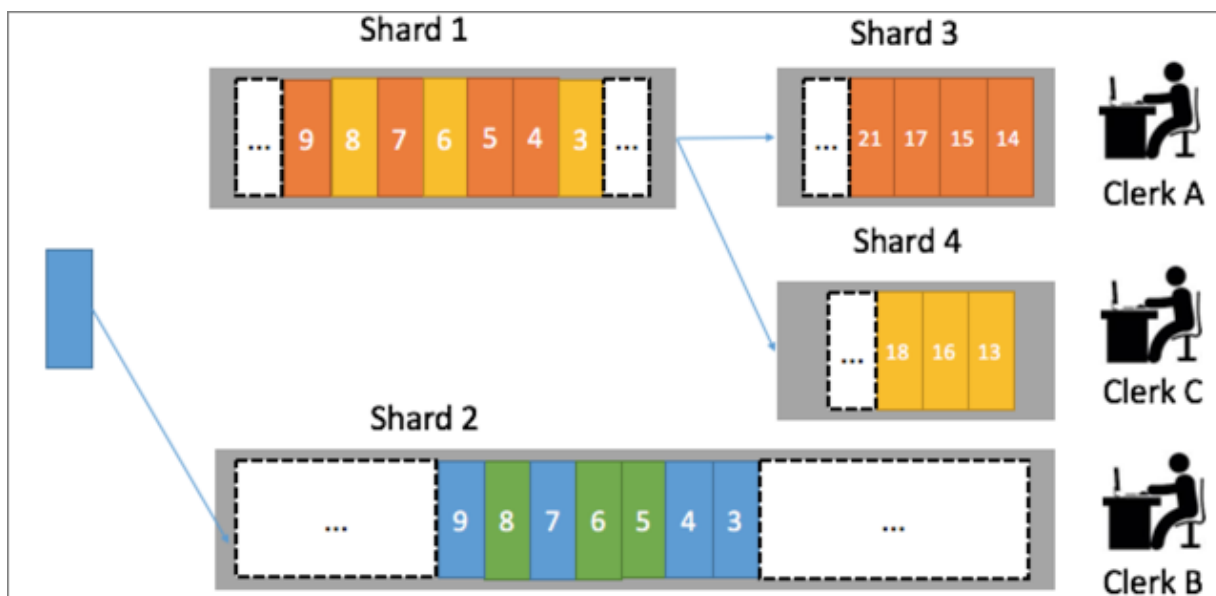
1. Control the size of generated billing data

Modify the program that generates metering logs, such as Nginx, to aggregate metering logs in the memory first and dump the aggregated metering logs once every minute. In this way, the data size is associated with the total number of users

. Assume that Nginx processes data for 1,000 users per minute. The size of metering logs generated in an hour is calculated as follows: $1,000 \times 200 \times 60 = 12$ GB. The data size is only 240 MB after compression.

2. Parallelize metering log processing

In LogHub, each Logstore can have several shards. For example, a Logstore contains three shards, and three metering programs are available. To ensure that the metering data of a user is always processed by the same metering program, use the hash function to map users to shards by user ID. Then, the metering data of the same user can be allocated to a fixed shard. For example, the metering data of users in the Xihu District of Hangzhou is written to shard 1, and the metering data of users in the Shangcheng District of Hangzhou is written to shard 2. In this case, the back-end metering programs can process the metering data in two shards in parallel.



FAQ

- How do I supplement data?

In LogHub, you can set the lifecycle of each Logstore in the range of 1 to 365 days. If the billing program needs to consume data again, it can compute data by period in the lifecycle of a Logstore.

- What can I do if metering logs are scattered on many front-end servers?
 1. Use a Logtail agent to collect the logs in real time from each server.
 2. Use the machine IDs of servers to define a dynamic machine group for auto scaling.

- How do I query details?

You can create indexes for LogHub data to support real-time query and statistical analysis. For example, if you want to query metering logs with a large data size, you can use the following query statement:

```
Inflow > 300000 and Method = Post * and Status in [ 200
300 ]
```

After enabling the indexing feature for LogHub data, you can query and analyze data in real time.

You can also add a statistical analysis statement to the end of the query statement as follows:

```
Inflow > 300000 and Method = Post * and Status in [ 200
300 ] | select max ( Inflow ) as s , ProjectName group
by ProjectName order by s desc
```

ProjectName ▼	s ▼
rel-stat-staff-benefits	1207132
rel-acticle-user-action	816968
tteduhaproxy	688385
noc	583006
xiaobinggd	539798
ludashi-stat	534489
sis-welooop	526956
ykd-testalipay	524523
fc-monitor-ol	524515
syt-accesslog	486647

- How do I store logs and check accounts on a T+1 basis?

Log Service can ship LogHub data to other systems. You can customize shards and the storage format to store logs in OSS. Then, you can use E-MapReduce, HybridDB, Hadoop, Hive, Presto, and Spark to compute log data.

3.4 Use a consumer library to consume logs in high reliability mode

Log processing has a wide scope, covering real-time computing, data warehouses, and offline computation. This topic describes how to process logs in order without data loss or repetition in real-time computing scenarios, even when upstream

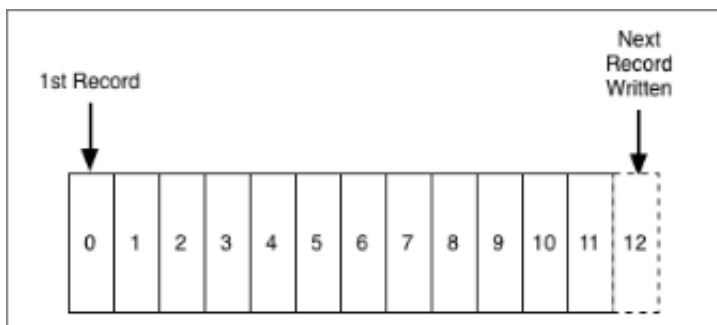
and downstream business systems are unreliable (for example, have faults) or the business traffic fluctuates.

For ease of understanding, this topic uses one day in a bank as an example to illustrate how to process logs. This topic also introduces the LogHub features of Log Service, based on which you can use Spark Streaming and Storm spouts to process logs.

Definitions

What is log data?

Jay Kreps, a former LinkedIn employee, defines a log as "an append-only, totally-ordered sequence of records ordered by time" in [The Log: What every software engineer should know about real-time data's unifying abstraction](#).



- **Append only:** Log entries are appended to the end of the log. They cannot be modified after they are generated.
- **Totally ordered by time:** Log entries are strictly ordered. Each log entry is assigned a unique sequential log entry number to indicate its timestamp. Different log entries may be generated at the same timestamp in seconds. For example, a GET operation and a SET operation are performed at the same timestamp. However, the two operations are still performed in order on a computer.

What kind of data can be abstracted into logs?

Half a century ago, captains and operators kept logs in thick notebooks. Today, computers enable logs to be generated and consumed everywhere. Servers, routers, sensors, GPS, orders, and various devices reflect our lives from different perspectives. In addition to a timestamp used to record the time of a log, captains kept anything they wanted in logs, such as text, an image, weather conditions, and sailing directions. After half a century, logs are generated in a variety of scenarios, such as an order, a payment record, a user access record, and a database operation.

In the computer field, common logs include metrics, binary logs for relational and NoSQL databases, events, audit logs, and access logs.

In this topic, a user operation in the bank is regarded as a log entry. It contains the name, account, operation time, operation type, and transaction amount of the user.

For example:

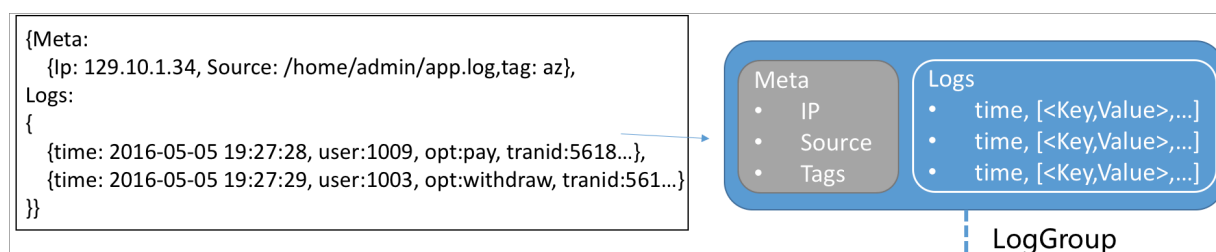
```
2016 - 06 - 28  08 : 00 : 00  Zhang  San  Deposit  RMB  1 ,
000
2016 - 06 - 27  09 : 00 : 00  Li    Si    Withdrawal RMB  20 , 000
```

LogHub data model

To help you understand abstract concepts, this section uses the LogHub data model of [Alibaba Cloud Log Service](#) for demonstration. For more information, see [Basic concepts](#) in Log Service Product Introduction.

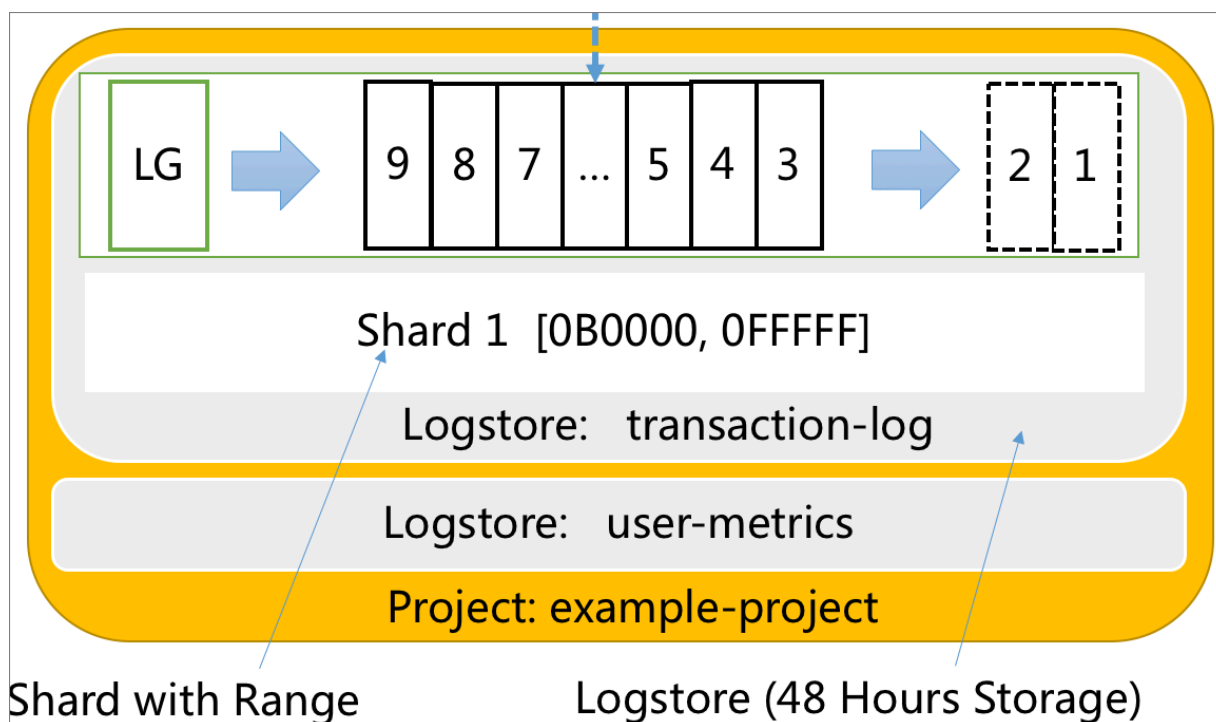
- A log consists of time and a group of key-value pairs.
- A log group is a collection of logs that have the same metadata such as the IP address and source.

The following figure shows the relationships between the logs and log group.



- A shard is the basic read/write unit of a log group. It can be regarded as a 48-hour first in, first out (FIFO) queue. Each shard allows you to write data at 5 MB/s and read data at 10 MB/s. The logical range of a shard is specified by the BeginKey and EndKey. This range enables the shard to contain a type of data different from other shards.
- A Logstore stores log data of the same type. Each Logstore is a carrier that consists of one or more shards whose range is [0000, FFFF..).
- A project is a container for Logstores.

The following figure shows the relationships among the logs, log group, shards, Logstores, and project.



One day in a bank

For example, in the nineteenth century, several users in a city went to a bank to deposit or withdraw their money. Several clerks were working in the bank. At that time, no computers were available to synchronize data in real time. Each clerk recorded data in an account book and used the account book to check the money every night in the bank. In this example, users are producers of data, money deposit and withdrawal are user operations, and clerks are consumers of data.

In a distributed log processing system, clerks are standalone servers that have fixed memory and computing capabilities, users are requests from various data sources, and the bank hall is a Logstore where users can read and write data.



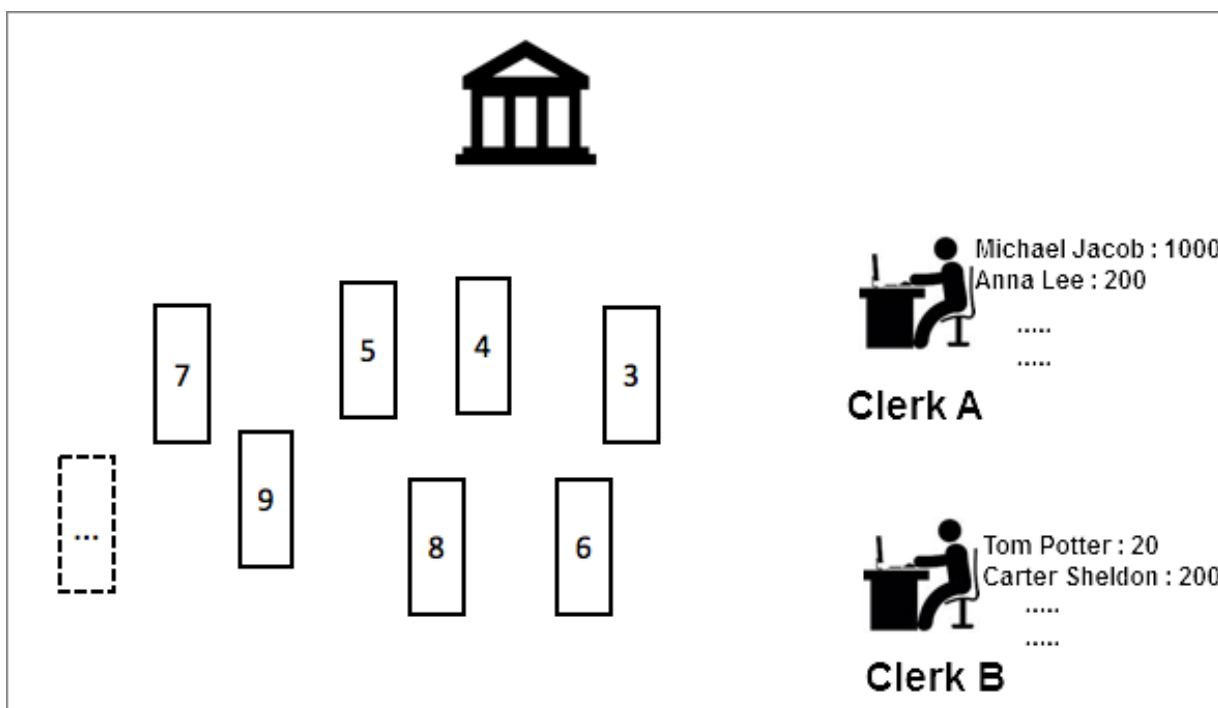
- Logs or log group: the user operations such as money deposit and withdrawal.
- User: the producer of operations.
- Clerk: the employee who handles user requests in the bank.

- Bank hall (Logstore): the place where user requests are received and then assigned to clerks for handling.
- Shard: the way in which the bank manager sorts user requests in the bank hall.

Issue 1: Ordering

Two clerks A and B were working in the bank. Zhang San visited the bank and deposited RMB 1,000 at counter A. Clerk A recorded the transaction amount in account book A. In the afternoon, due to a shortage of money, Zhang San went to counter B to withdraw the money. At counter B, clerk B found no deposit record after checking account book B.

Based on this example, money deposit and withdrawal must be strictly ordered. Requests from the same user must be handled by the same clerk to ensure that the status of user operations is consistent.



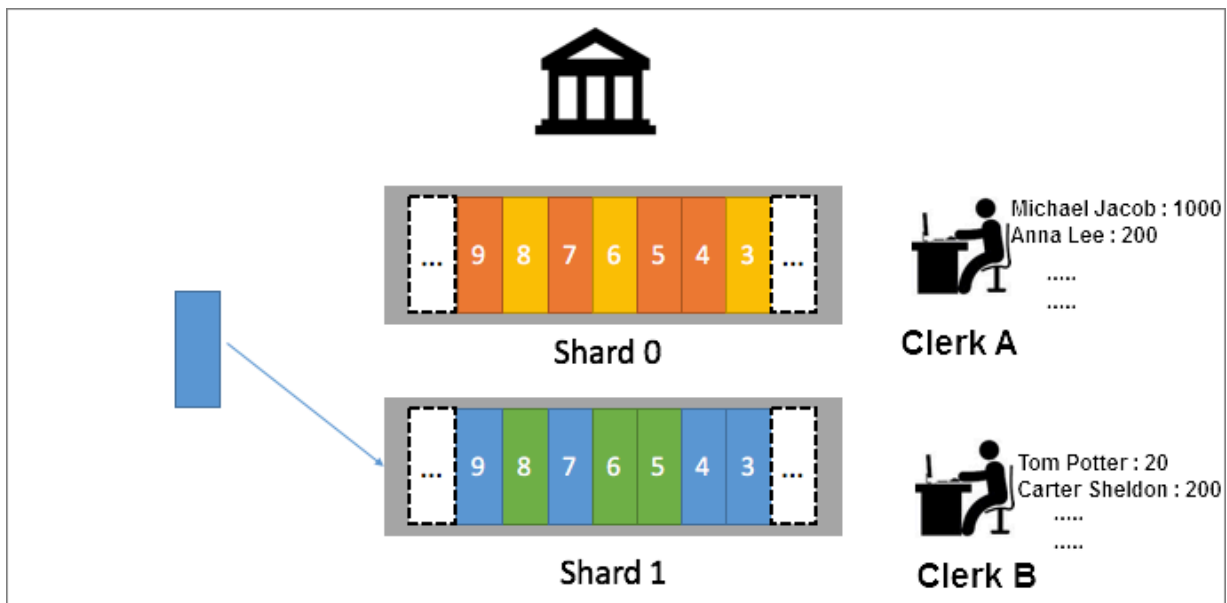
To ensure ordering, users can queue up to submit requests. A shard can be created, where only clerk A is assigned to handle user requests based on the FIFO principle. However, this method leads to low efficiency, even when 10 clerks are assigned to handle requests from 1,000 users.

To improve efficiency in this scenario, you can use the following solution:

1. Create 10 shards for 10 clerks. Assign a clerk to work in each shard.

2. Ensure that operations for the same account are ordered: Map users by using consistent hashing. For example, map users to specific shards by bank account or user name. In this case, by using the formula $\text{hash}(\text{Zhang}) = Z$, requests from Zhang San are always mapped to the specific shard whose range contains Z. A clerk, for example, Clerk A, is assigned to handle requests in this shard.

If many users whose surname is Zhang, the solution can be adjusted. For example, use the hash function to map users to shards by account ID or zip code so that user requests can be more evenly distributed to each shard.



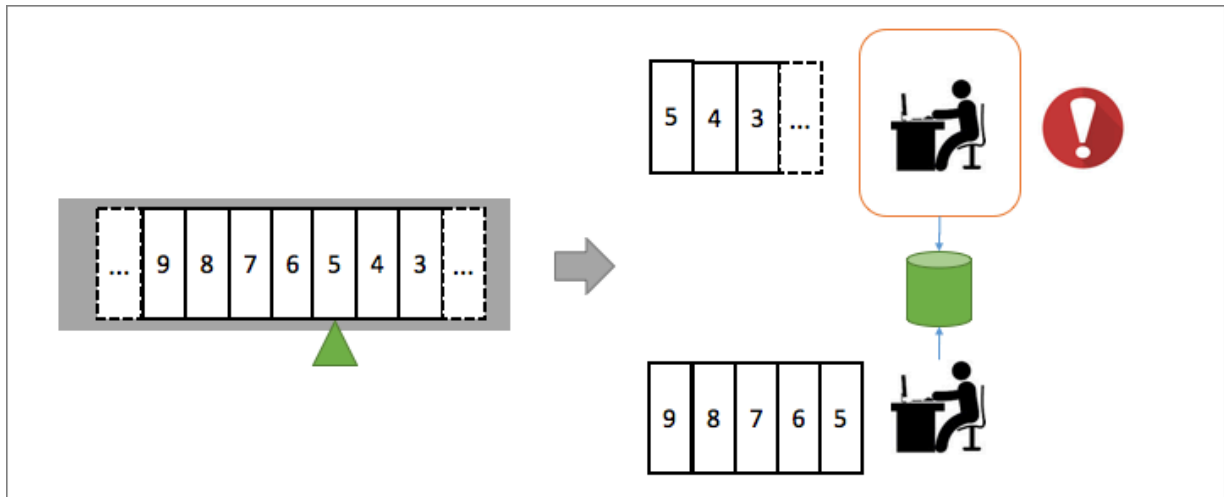
Issue 2: At least once

Zhang San deposited money at counter A. When handling this deposit request, clerk A received a call. After the call ended, clerk A incorrectly considered that the deposit request of Zhang San was handled and started to handle the request from the next user. However, the deposit request of Zhang San was lost in this case.

Computers do not make mistakes like clerks and can work more reliably for a longer time. However, computers may still fail to process data due to failure or overload. Deposit loss for such reasons is not allowed.

To avoid data loss in this scenario, you can use the following solution:

Clerk A records the position of the current request in the shard as the progress of the request in a notebook (not account book A). In this case, clerk A calls the next user only after the deposit request of Zhang San is handled.



However, this solution may lead to repetition. For example, after handling the deposit request of Zhang San and updating data in account book A, clerk A was called away but did not record the position of the current request in the shard in the notebook. When clerk A came back and did not find the progress of the request from Zhang San in the notebook, clerk A may handle the request again.

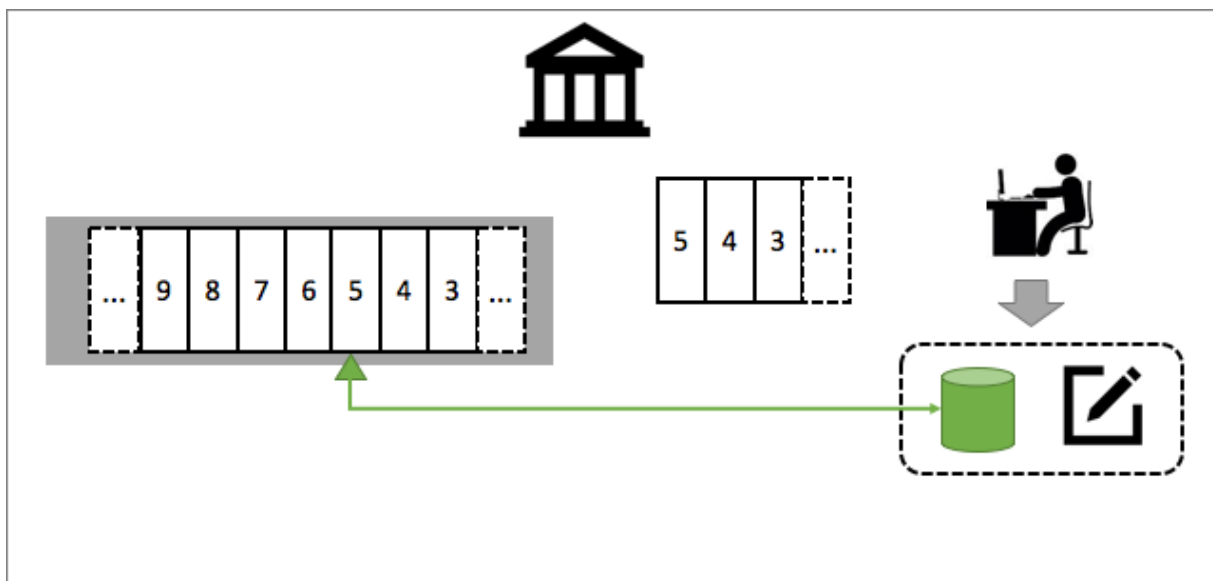
Issue 3: Exactly once

Will repetition certainly cause problems? The answer is no.

If you perform an idempotent operation more than once, you may waste time and energy. However, such repetition does not affect the result. For example, balance inquiry is a read-only operation performed by a user. The repetition of this operation does not affect the inquiry result. Some non-read-only operations, such as user logoff, can also be performed twice consecutively.

In reality, most operations, such as money deposit and withdrawal, are not idempotent. If you perform these operations repeatedly, the impact on the results can be fatal. What is the solution to repetition? After handling a user request, clerk A needs to update data in account book A, record the position of the current request in the shard in the notebook, and then combine two records into a checkpoint.

If clerk A leaves temporarily or permanently, other clerks can continue as follows: If a checkpoint exists for the current user request, proceed to the next user request. If no checkpoint exists for the current user request, handle this request. Guarantee the atomicity of operations.



A checkpoint is a persistent object in which you can record the position or time of an element in a shard as the key to indicate that the element is processed.

Business challenges

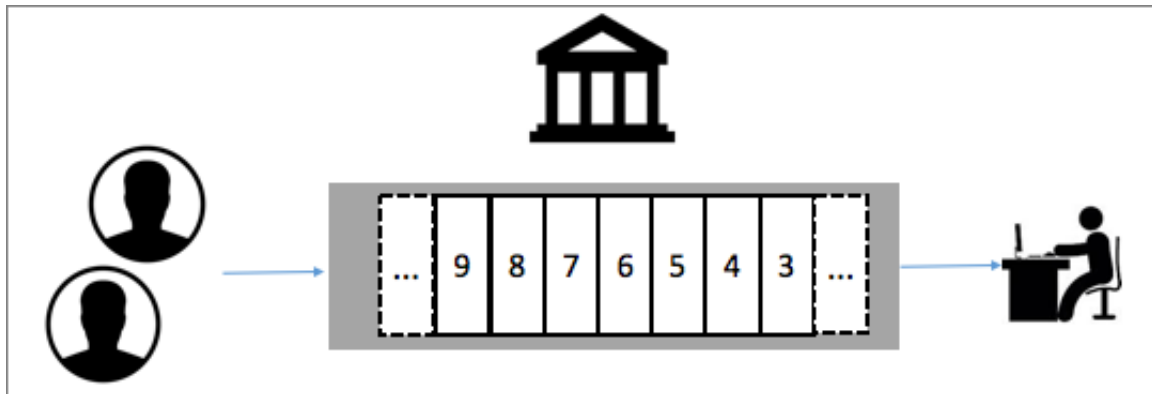
The principles are not complex. However, in the real world, changes and uncertainty make the three issues more complex. For example:

1. The number of users soars on the pay day.
2. Unlike computers, clerks need a break and lunch time.
3. To improve service experience, the bank manager needs to request clerks to work faster at the right time. Can the bank manager determine the right time based on the speed of request processing in a shard?
4. Clerks need to easily and properly transfer account books and checkpoints during the handover.

One day in reality

Bank opening at 08:00

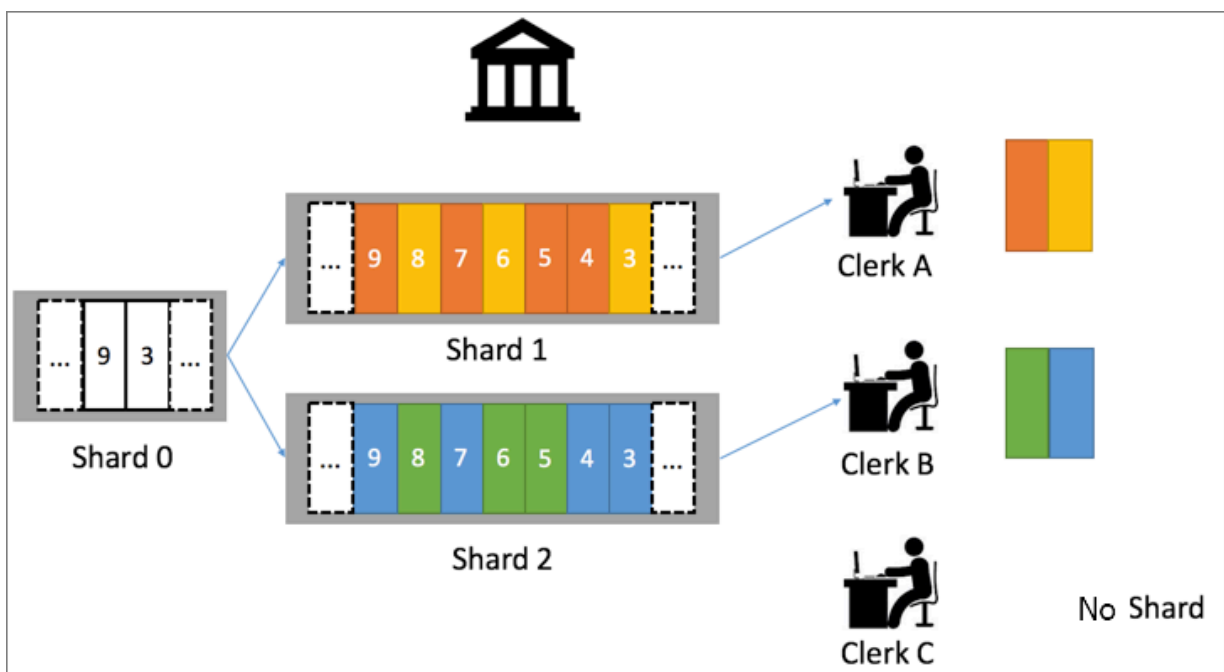
All user requests are assigned to the only shard, shard 0. Clerk A is responsible for handling such requests.



Peak hours after 10:00

The bank manager decides to split shard 0 into shard 1 and shard 2 after 10:00. In addition, the bank manager assigns user requests to the two shards based on the following rules: If the first letter of the surname of a user falls in the range of A to W, the user request is assigned to shard 1. If the first letter of the surname of a user is X, Y, or Z, the user request is assigned to shard 2. The reason why the ranges of the two shards are different is that most surnames start with X, Y, or Z. This mapping method can ensure a balanced workload.

The following figure shows the status of user requests in shards from 10:00 to 12:00.

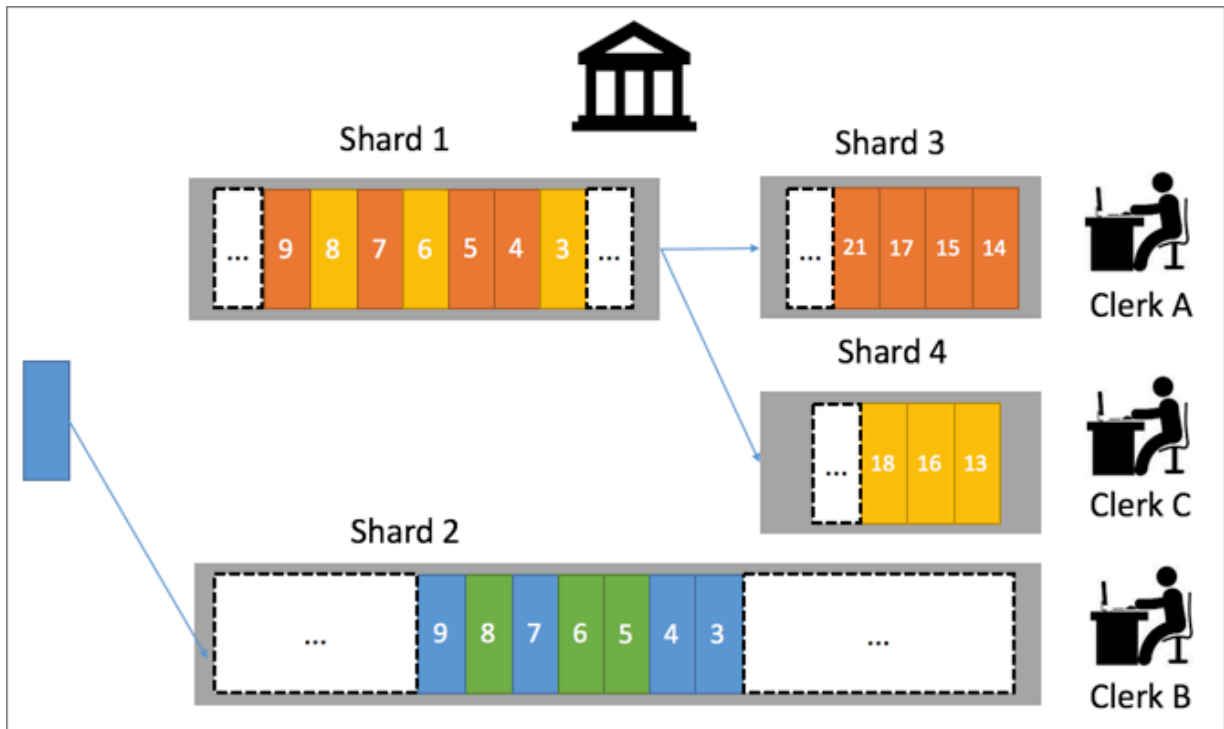


When clerk A has difficulty in handling requests in two shards, the bank manager dispatches clerk B and clerk C. Clerk B takes over one of the shards, whereas clerk C is currently idle.

More and more users at 12:00

The bank manager thinks that clerk A is under much pressure for handling requests in shard 1 and splits shard 1 into shard 3 and shard 4. Then, clerk A handles requests in shard 3, and clerk C handles requests in shard 4. After 12:00, the bank manager assigns user requests originally assigned to shard 1 to shard 3 and shard 4.

The following figure shows the status of user requests in shards after 12:00.



Fewer and fewer users after 16:00

The bank manager asks clerk A and clerk B to have a break, and arranges for clerk C to handle requests in shard 2, shard 3, and shard 4. Later, the bank manager combines shard 2 and shard 3 to form shard 5, and then combines shard 5 and shard 4 to form shard 6. After all user requests in shard 6 are handled, the bank is closed.

Actual log processing

The preceding process can be abstracted into a typical log processing scenario. To meet the business requirements of banks, an auto scaling and flexible log framework can be used to provide the following features:

1. Automatically scales in or out shards.
2. Automatically adapts shards to the consumers of a consumer group when consumers are added to or removed from the consumer group. In this process, guarantees data integrity and processes logs in order.

3. Processes logs only once, which requires cooperation with consumers.
4. Monitors the consumption progress to properly allocate computing resources.
5. Supports logs from more sources. For banks, users can send requests from various channels such as online banking, mobile banking, and checks.

You can use LogHub and the LogHub consumer library to process logs in real time in typical scenarios. Using a consumer library, you only need to focus on the business logic and do not need to worry about traffic scaling or failover.

3.5 Cleanse data through ETL

Data is not flawless, so it requires processing. There is always a gap between raw data and final results. You can run Extract-Transform-Load (ETL) jobs to cleanse, transform, and sort data.

Use case

"I Want Take-away" is an e-commerce website with a platform involving users, restaurants, and couriers. Users can place their take-away orders on the website, app, WeChat, or Alipay. Once receiving an order from a user, a merchant starts preparing the ordered dishes. At the same time, the system automatically notifies the nearest couriers. Then, one of the couriers accepts the order and delivers the dishes to the user.



The operations team has two tasks:

- Obtain the locations of couriers so that the couriers can be dispatched to specific areas as required.
- Obtain the information about the usage of coupons and cashes so that coupons can be given away to specific areas for interactive promotion.

GPS data processing

The GPS data (X, Y) is reported through the courier app every minute. The data format is as follows:

```
2016 - 06 - 26 19 : 00 : 15 ID : 10015 DeviceID : EXX1234567 8
Network : 4G GPS - X : 10 . 30 . 339 GPS - Y : 17 . 38 . 224 .
5 Status : Delivering
```

Each record contains the reporting time, courier ID, network, device ID, and coordinate position including GPS-X and GPS-Y. The longitude and latitude information provided by GPS is accurate. However, the operations team only needs to obtain information about the courier status in each area. Therefore, such accurate data is not required. In this case, the coordinate position can be transformed to a readable field such as City, District, or ZipCode.

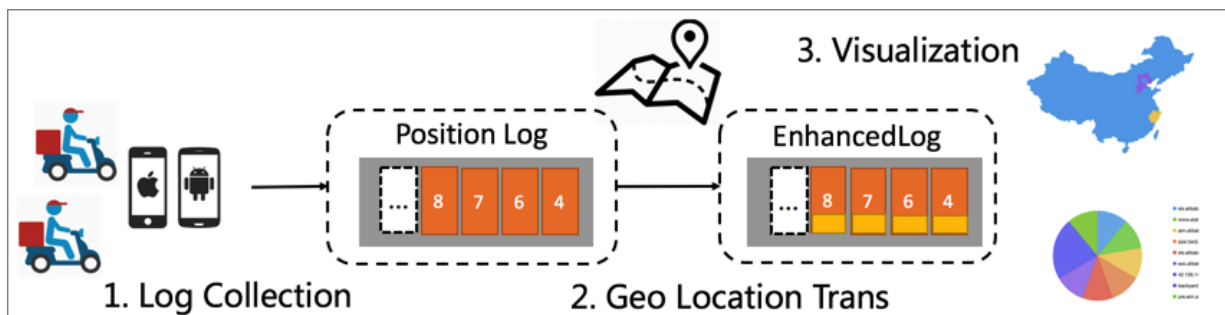
```
( GPS - X , GPS - Y ) --> ( GPS - X , GPS - Y , City , District ,
ZipCode )
```

This is a typical ETL requirement. Use LogHub to create two Logstores, namely, the PositionLog Logstore for raw data and the EnhancedLog Logstore for transformed data. Run an ETL program, such as Spark Streaming and Storm, or start Consumer Library in the container to read data from the PositionLog Logstore in real time, transform the coordinate position, and write the transformed data to the EnhancedLog Logstore. You can connect the EnhancedLog Logstore to Realtime Compute to visualize the log data. You can also query log data in the Logstore by creating indexes.

The whole process is as follows:

1. Use event tracking on the courier app to report the current GPS location to the PositionLog Logstore every minute.
 - We recommend that you use the LogHub SDK for iOS or Android, or Mobile Analytics (MAN) to access the mobile device logs.

2. Use a program to read data from the PositionLog Logstore in real time. Then, transform the log data and write the transformed data to the EnhancedLog Logstore in real time.
 - We recommend that you use Spark Streaming, Storm, Consumer Library (which adopts a programming mode that supports automatic load balancing), or SDK for subscription.
3. Process the log data in the EnhancedLog Logstore, for example, computing the data and visualizing the result. We recommend that you use:
 - LogHub for integrating Realtime Compute.
 - LogShipper for data shipping to OSS, E-MapReduce, Table Store, and MaxCompute.
 - LogSearch for querying orders.



Order masking and analysis

The payment service receives payment requests, including the payment account, method, amount, and coupons.

- The payment requests include sensitive information. Therefore, order masking is required.
- The coupon and cash data involved in the payment information needs to be separated.

The whole process is as follows:

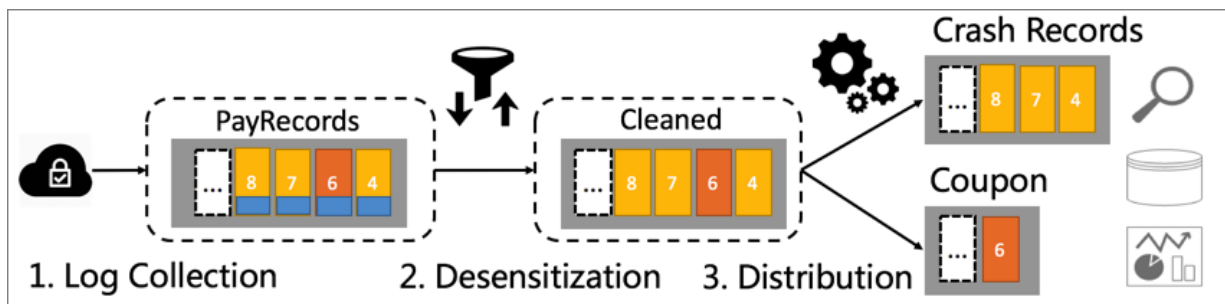
1. Create four Logstores, namely, PayRecords, Cleaned, Cash, and Coupon. The app writes the order data to the PayRecords Logstore through Log4J Appender.

We recommend that you use Log4J Appender to avoid sensitive data from being saved to hard disks.

2. The masking program consumes the data in the PayRecords Logstore in real time . After the account information is masked, the data in the PayRecords Logstore is written to the Cleaned Logstore.

The division program consumes the data in the Cleaned Logstore in real time, and stores the coupon and cash data respectively in the Cash Logstore and Coupon Logstore through the business logic.

We recommend that you use Spark Streaming, Storm, Consumer Library, or SDK for real-time data masking and division.



Other features

- You can control each Resource Access Management (RAM) user's permissions on each Logstore in LogHub in the RAM console.
- You can obtain the current read and write data through CloudMonitor and view the consumption progress in the console.

4 Shipping

4.1 Connect to a data warehouse

The LogShipper feature of Log Service ships logs to storage services such as Object Storage Service (OSS), Table Store, and MaxCompute. It cooperates with E-MapReduce (Spark and Hive) and MaxCompute for offline computing.

Data warehousing (offline computing)

Data warehousing (offline computing) is the supplement to real-time computing, but they are used for different purposes.

Mode	Advantage	Disadvantage	Application scope
Real-time computing	Fast	Simple computing	Mainly used for incremental computation in monitoring and real-time analysis
Offline computing (data warehousing)	Accurate and powerful	Relatively slow	Mainly used for full computation in Business Intelligence (BI), and data statistics and comparison

To satisfy the current data analysis requirements, you need to perform both real-time computing and data warehousing (offline computing) on the same set of data. For example, you need to perform the following operations when processing access logs:

- Use Realtime Compute to display the data, including the current PV, UV, and operator information, on the dashboard in real time.
- Conduct detailed analysis of the full data every night to obtain the growth, year-on-year or month-on-month growth, and top ranking data.

In the world of Internet, there are two classic data processing architectures:

- **Lambda architecture**: When data comes in, the architecture can stream and at the same time save the data to the data warehouse. However, when you initiate a query,

the results are returned from real-time computing and offline computing based on query conditions and complexities.

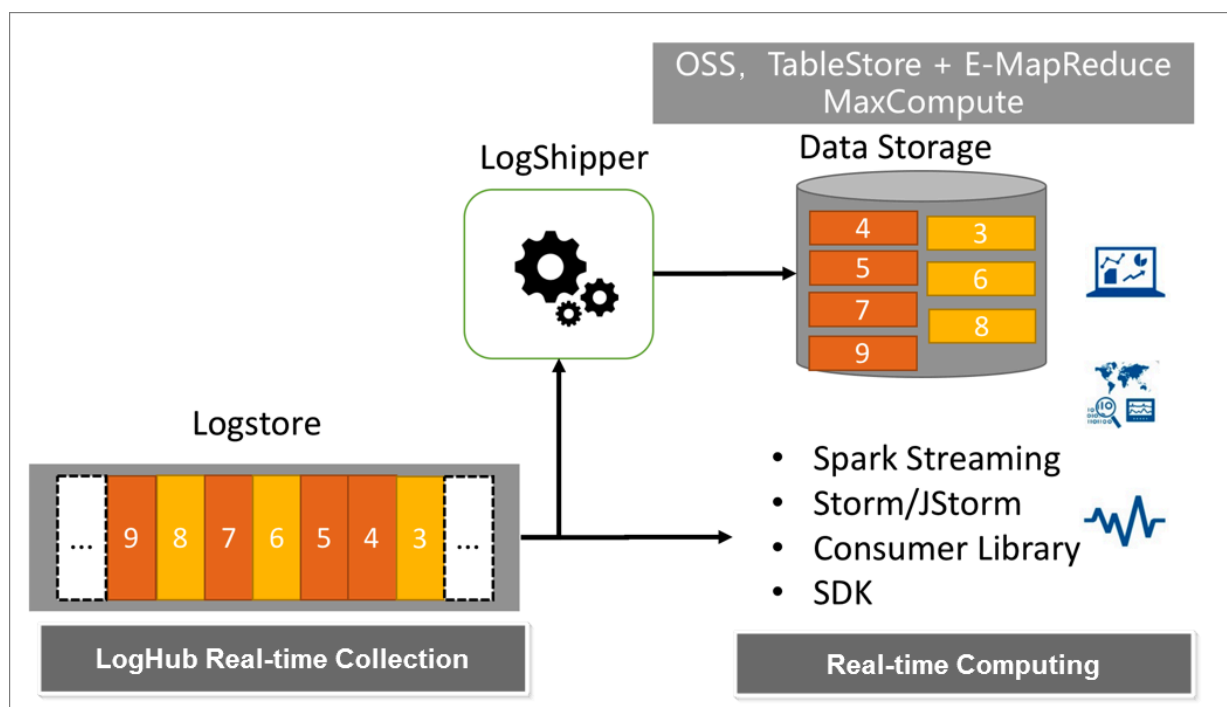
- **Kappa architecture:** Kafka-based architecture. The offline computing feature is weakened. All data is stored in Kafka, and all queries are fulfilled with real-time computing.

Log Service provides an architecture that is more similar to the Lambda architecture.

LogHub and LogShipper for both real-time and offline computing

Create a Logstore first, and configure LogShipper in the console to enable data warehouse connection. Currently, the following data warehouses are supported:

- **OSS:** large-scale object storage service
- **Table Store:** NoSQL data storage service
- **MaxCompute:** data computing service



LogShipper provides the following features:

- **Quasi real-time:** connects to a data warehouse in minutes.
- **Enormous data volume:** does not need to worry about concurrency.
- **Retry on error:** performs automatic retries or API-based manual retries in case of faults.
- **Task API:** uses APIs to acquire the log shipping status for different time periods.

- **Automatic compression:** supports data compression to reduce the storage bandwidth.

Typical scenarios

Scenario 1: Log auditing

A is responsible for maintaining a forum and part of his job is to conduct audits and offline analysis of all access logs on the forum.

- Department G wants A to capture user visits over the past 180 days, and to provide the access logs within a given period of time on demand.
- The operations team must prepare an access log report on a quarterly basis.

A uses Log Service to collect log data from the servers and enables the LogShipper feature, allowing Log Service to automatically collect, ship, and compress logs. When an audit is required, the logs within the time period can be authorized to a third party. To conduct offline analysis, use E-MapReduce to run a 30-minute offline task. In this way, two jobs are done at minimal cost. In addition, Data Lake Analytics (DLA) can be used to analyze the log data shipped to OSS.

Scenario 2: Real-time and offline analysis of log data

As an open-source software enthusiast, B prefers to use Spark for data analysis. His requirements are as follows:

- Collect logs from the mobile client by using APIs.
- Use Spark Streaming to conduct real-time log analysis and collect statistics on online user visits.
- Use Hive to conduct offline analysis in T+1 mode.
- Grant downstream agencies access to the log data for analysis in other dimensions.

With a combination of Log Service, OSS, E-MapReduce or DLA, and Resource Access Management (RAM), you can fulfill such requirements.

5 Operations logs

5.1 Activate, monitor, and consume operations logs

Log Service provides the operations log feature to help you understand the usage of Log Service in real time and improve O&M efficiency.

Enable the operations log feature

Operations logs are divided into detailed logs and important logs (including Logtail-related logs, consumer group latency logs, and metering logs), respectively stored in `internal-operation_log` and `internal-diagnostic_log`. The `internal-diagnostic_log` Logstore is not charged, while the `internal-operation_log` Logstore is charged as common Logstores. Detailed logs record each operation or API request of a user. Multiple operations logs are generated when there are multiple read and write requests. You can enable the operations log feature as required. We recommend that you select Automatic creation (recommended) for Log Storage. In this way, the operations logs in the same region can be stored in the same project, thus facilitating log management and statistics.

Monitor the Logtail heartbeat

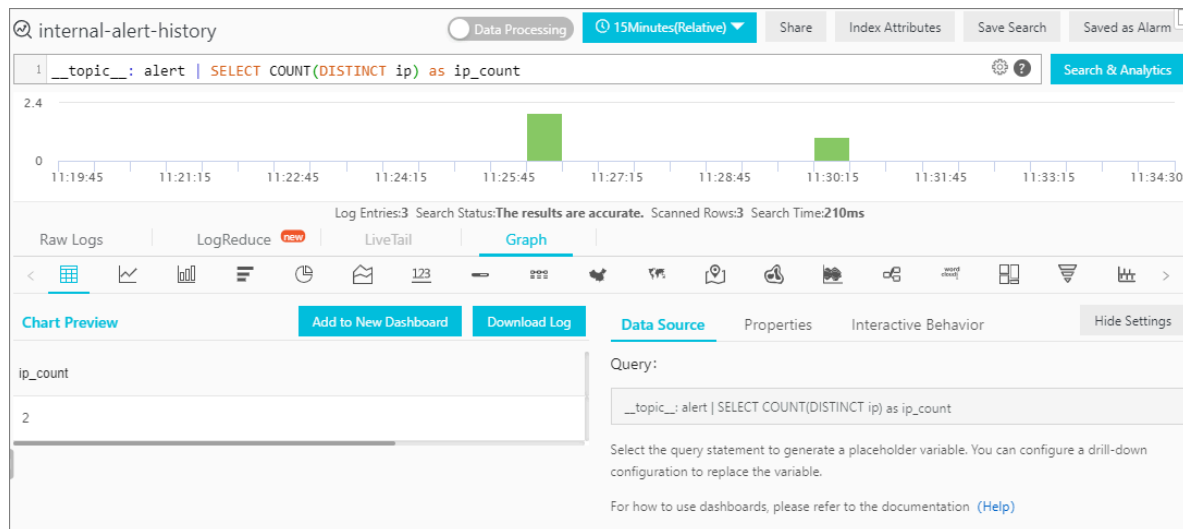
After Logtail is installed, you can use Logtail [status logs](#) in the operations logs to check the working status of Logtail.

Logtail status logs are stored in the `internal-diagnostic_log` Logstore. On the `internal-diagnostic_log` page, run the query statement `__topic__ : logtail_status` to search for Logtail status logs. You can obtain the number of machines for a recent period of time and compare it with the number of machines in the Logtail application machine group. In addition, you can configure the alert rules. For example, an alert is triggered when the counted number is smaller than the number of machines in the machine group.

- Query statement

```
__topic__: logtail_st atus | SELECT COUNT ( DISTINCT ip )  
as ip_count
```

- Query snapshot



- Alert rule configuration (assume that the number of machines in the machine group is 100)

Create Alert

Alert Configuration | Notifications

* Alert Name: test 4/64

* Add to New Dashboard: Select Existing... dashboard-01

* Chart Name: test 4/64

Query: __topic__: alert | SELECT COUNT(DISTINCT ip) as ip_count

* Search Period: 15Minutes(Relative)

* Check Frequency: Fixed Interval 15 Minutes

* Trigger Condition: ip_count < 100

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~. [Documentation](#)

[Advanced >](#)

If an alert is triggered, you can go to the console to view the status of the machine group and check the heartbeat information of the machines.

View metering data

After you write logs to Log Service, you can view the metering information such as the log traffic, read and write times, and storage space to learn about the usage and billing item details of Log Service.

A Logstore generates a metering log per hour, including the read and write traffic and times in the statistical time period, and the storage size for raw logs and indexes at the current time point. For more information about the fields of metering logs, see [Log types](#). Metering logs of Log Service are stored in the internal-diagnostic_log

Logstore. On the internal-diagnostic_log page, run the query statement `__topic__ : metering` to search for metering logs.

Use the `max_by` function to compute the storage size of each Logstore.

- Query statement

```
__topic__ : metering | SELECT max_by ( storage_in dex +
storage_ra w , __time__ ) as total_stor age , project ,
logstore GROUP BY project , logstore
```

The default dashboard of operations logs provides abundant charts for metering logs. For more information, see [Service log dashboards](#).

View the consumer group latency

After logs are written to Log Service, you can consume logs in addition to querying and analyzing them. Log Service provides [consumer groups](#) supported by multiple programming languages.

When you use consumer groups to consume logs, the latency for consuming logs is one of the most concerning problems. By monitoring the consumption latency, you can learn about the consumption progress. In case of a high latency, you can adjust the consumption speed by changing the number of consumers.

As a kind of operations logs, consumer group latency logs are also stored in the internal-diagnostic_log Logstore, and are generated every 2 minutes. On the internal-diagnostic_log page, run the query statement `__topic__ : consumergr oup_log` to search for all consumer group latency logs.

Query the consumption latency of the consumer group test-consumer-group.

Monitor Logtail exceptions

The proper running of Logtail guarantees the log integrity. If Logtail exceptions can be found in time, you can adjust the Logtail configurations to avoid log missing.

You can run the query statement `__topic__ : logtail_al arm` to search for the exception logs of Logtail.

Query the number of exceptions of each type within 15 minutes.

- Query statement

```
--topic__ : logtail_alarm | select sum ( alarm_count ) as
errorCount , alarm_type GROUP BY alarm_type
```

Monitor the data traffic written to a Logstore

You can use metering logs to obtain information about the read and write traffic within 1 hour. However, if you need to obtain the information within a narrower time period, such as 15 minutes, the operations logs are required. Each operation of a user corresponds to a request. Each request generates an operations log. For example, you can query the total traffic for write operations within 15 minutes.

1. Query the traffic of raw logs and compressed logs written to a Logstore within 15 minutes.

- Query statement

```
Method : PostLogStoreLogs AND Project : my - project
and LogStore : my - logstore | SELECT sum ( InFlow ) as
raw_bytes , sum ( NetInflow ) as network_bytes
```

2. Query the traffic decline ratio of the logs written to a Logstore within 15 minutes.

- Query statement

```
Method : PostLogStoreLogs AND Project : my - project
and LogStore : my - logstore | select round (( diff [ 1
]- diff [ 2 ])/ diff [ 1 ], 2 ) as rate from ( select
```

```
compare ( network_by tes , 900 ) as diff from ( select
sum ( NetInflow ) as network_by tes from log ))
```

3. Create an alert.

Set the alert rule so that an alert is triggered when the traffic decline ratio of the logs written to a Logstore exceeds 50%.

Create Alert

Alert Configuration | Notifications

* Alert Name: test 4/64

* Add to New Dashboard: Create 0/64

* Chart Name: test 4/64

Query: Method: PostLogStoreLogs AND Project: my-project and LogStore: my-logstore | select round((diff[1]-diff[2])/diff[1],2) as rate from (select compare(network_bytes, 900) as diff from (select sum(NetInflow) as network_bytes from log))

* Search Period: 15Minutes(Relative)

* Check Frequency: Fixed Interval 15 Minutes

* Trigger Condition: rate>0.5

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~. [Documentation](#)

[Advanced >](#)

Audit operations logs

The operations logs of all resources in the project are stored in the internal-operation_log Logstore. The logs record the operation information, for example, the name of a machine group is recorded when you create the machine group, and the name of a Logstore is recorded when you operate the Logstore. In addition, the logs also record the information about the user who performs the operations. Currently, the user information includes three types, as listed in the following table.

Type	Field
Alibaba Cloud account	<ul style="list-style-type: none">· InvokerUid: the AliUid of the Alibaba Cloud account.· CallerType: the parent account.
RAM user	<ul style="list-style-type: none">· InvokerUid: the AliUid of the RAM user.· CallerType: the subuser account.
STS	<ul style="list-style-type: none">· InvokerUid: the AliUid of the Alibaba Cloud account.· CallerType: STS.· RoleSessionName: the name of the session.

Based on the preceding table, you can obtain the user information corresponding to an operations log.

6 Alerting

6.1 Configure an alert

Log Service enables you to configure alerts based on the charts in a dashboard to monitor the service status in real time.

Set the query time range and execution interval

The configured query statement is executed regularly based on the execution interval to query logs generated in the specified query time range. The query result is used as a parameter in the alert condition. If the calculation result of the alert condition is true, an alert is triggered.

Do not set the query time range to the same time period as the execution interval. For example, the execution interval is 1 minute, and the query time range is also 1 minute. The reason is as follows (taking the execution interval of 1 minute as an example):

- After data is written to Log Service, there is a latency before it can be queried. Even if the latency is low, data may fail to be queried. Assume that the alert execution time is 12:03:30 and the query time range is 1 minute, that is, [12:02:30, 12:03:30). For logs written at 12:03:29, they may not be queried at 12:03:30.
- If you have high requirements on alert accuracy (including no repeated alerts and no missing alerts), you can move the query time range forward, for example, from 70 seconds ago to 10 seconds ago. For example, set the query time range to [12:02:20, 12:03:20) if the alert execution time is 12:03:30. By setting a buffer period of 10 seconds, you can avoid missing alerts caused by indexing latency.
- If you have high requirements on real-time performance (that is, you want to receive alerts once they are generated regardless of repeated alerts), you can move the query time range forward, for example, from 70 seconds ago to now. For example, set the query time range to [12:02:20, 12:03:30) if the alert execution time is 12:03:30.
- When logs at different time points within the same minute are written to Log Service, the index of a later log may fall into the time point of the earlier log due to the indexing method of Log Service. Assume that the alert execution time is

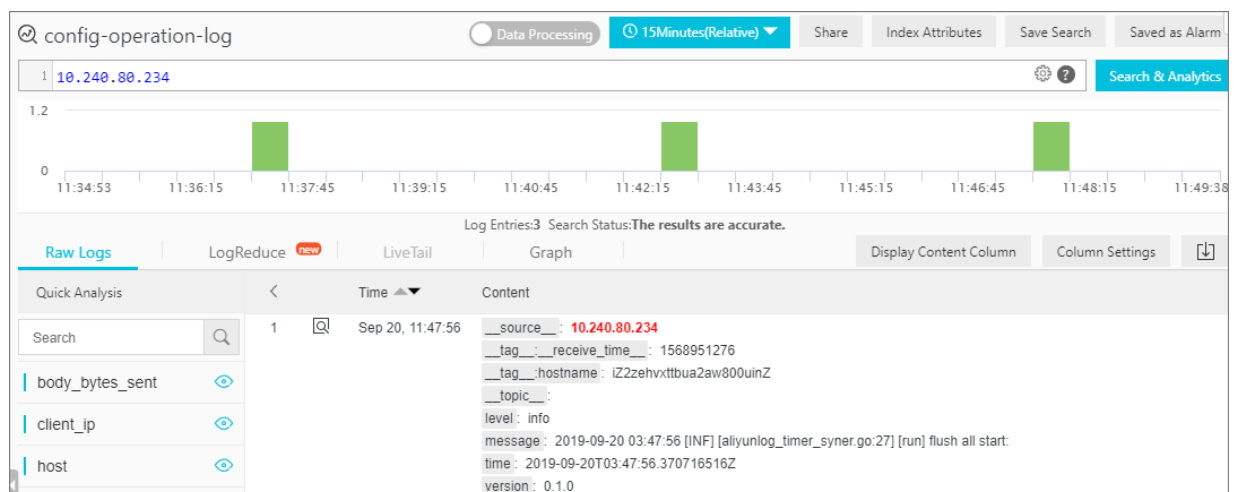
12:03:30 and the query time range is 1 minute, that is, [12:02:30, 12:03:30). Multiple logs are written at 12:02:50 and these logs are generated at different time points within the same minute, such as 12:02:20 and 12:02:50. In this case, the index of the logs may fall into the time point 12:02:20, and the logs cannot be queried in the specified query time range.

- If you have high requirements on alert accuracy (including no repeated alerts and no missing alerts), you can use the integral minute as the query time range, such as 1 minute, 5 minutes, and 1 hour, and set the execution interval to the same time period, correspondingly, 1 minute, 5 minutes, and 1 hour.
- If you have high requirements on real-time performance (that is, you want to receive alerts once they are generated regardless of repeated alerts), you can include at least 1 minute before the alert execution time to the query time range. For example, set the query time range to [12:02:00, 12:03:30) and the execution interval to 1 minute if the alert execution time is 12:03:30.

Trigger an alert based on the query result

Assume that the alert condition is met as long as the result of a query is not empty.

You can set the alert rule so that an alert will be triggered if a certain field exists. For example, search for logs containing the IP address 10.240.80.234.



If you find logs that contain the IP address 10.240.80.234, an alert is triggered. You can set an alert condition that is always true based on any field. Assume that the client_ip field exists in each log and cannot be an empty string. You can set the alert rule so that an alert will be triggered as long as the client_ip field is not empty.

Create Alert

Alert Configuration

Notifications

* Alert Name

test

4/64

* Add to New Dashboard

Create

0/64

* Chart Name

test

4/64

Query

10.240.80.234

* Search Period

🕒 15Minutes(Relative)

* Check Frequency

Fixed Interval

15

+

-

Minutes

* Trigger Condition

client_ip != "

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~.

[Documentation](#)

Advanced

Trigger an alert based on analysis result

Triggering alerts based on analysis results is one of the most common alerting scenarios. For example, an alert is triggered after aggregation of specific fields. Run the following query statement to collect statistics on the number of logs that contain the ERROR keyword. Set the alert rule so that an alert will be triggered if the number of logs that contain the ERROR keyword is greater than the threshold value.

```
ERROR | select count ( 1 ) as errorCount
```

The alert condition can be set to that errorCount is greater than a threshold value, for example, `errorCount > 0`.

Trigger an alert based on association query

When you create an alert in a dashboard, you can select multiple charts as the input for alert query.

- Configure an alert that is triggered based on combined query results in different time ranges.

For example, set the alert rule so that an alert will be triggered if the PV within 15 minutes is greater than 100,000 and the UV within 1 hour is smaller than 1,000.

Create Alert

Alert Configuration

Notifications

* Alert Name

test

4/64

* Associated Chart

0

Chart Name

PV

✕

Query

* | select count(*) as pv

✎

Search Period

15Minutes(Relative)

1

Chart Name

UV

✕

Query

* | select count(distinct client_ip) as uv

✎

Search Period

15Minutes(Relative)

2

Add

* Frequency

Fixed Interval

15

Minutes

* Trigger Condition

\$0.pv > 100000 && \$1.uv < 1000

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~.

Documentation

Advanced



Note:

When multiple charts are selected, the query time ranges are independent of each other. In the trigger condition, use the `${Number}.${Field}` format to reference the field in the query result. For example, `$0.pv > 100000 && $1.uv < 1000`.

- Configure an alert that is triggered based on some charts. Query results of other charts are used as auxiliary information.

Run the following statement to collect statistics on the number of logs whose log level is ERROR:

```
level : ERROR | select count ( 1 ) as errorCount
```

Set the alert rule so that an alert will be triggered if the number of logs whose log level is ERROR is greater than the threshold value.

```
errorCount > 10
```

If you want to view the logs whose log level is ERROR in the alert notification, run the following query statement:

```
level : ERROR
```

Set the alert notification as follows:

```
${ results [ 1 ]. RawResults AsKv }
```

You can view the logs whose log level is ERROR.

Suppress alerts

When an alert is triggered, you may receive the notification multiple times within a period of time. To prevent false alerts and repeated alerts caused by data jitter, you can suppress the alerts in the following two ways:

- Set the trigger condition.

An alert is triggered only when the alert conditions are met for each of the multiple consecutive detections.

For example, if the alert execution interval is 1 minute and the trigger threshold is 5, the notification is sent only when each of the five consecutive alert detections meets the alert conditions. If any one of the five consecutive alert detections cannot meet the alert conditions, the count is reset.

- Set the notification interval.

When you set a small alert execution interval, you can set the minimum interval between two notifications to prevent frequent notifications. For example, if the alert execution interval is 1 minute and the notification interval is 30 minutes, no notification will be received even if an alert is triggered within 30 minutes.

Create Alert

Alert Configuration

Notifications

* Alert Name

test

4/64

* Associated Chart

0

Chart Name

PV

Query

* | select count(*) as pv

Search Period

15Minutes(Relative)

1

Add

* Frequency

Fixed Interval

15

Minutes

* Trigger Condition

\$0.pv > 100000 && \$1.uv < 1000

Support the addition (+), subtraction (-), multiplication (*), division (/), and modulo (%) operations and comparison operations including >, >=, <, <=, ==, !=, =~, and !~. [Documentation](#)

Advanced

* Notification Trigger Threshold

1

* Notification Interval

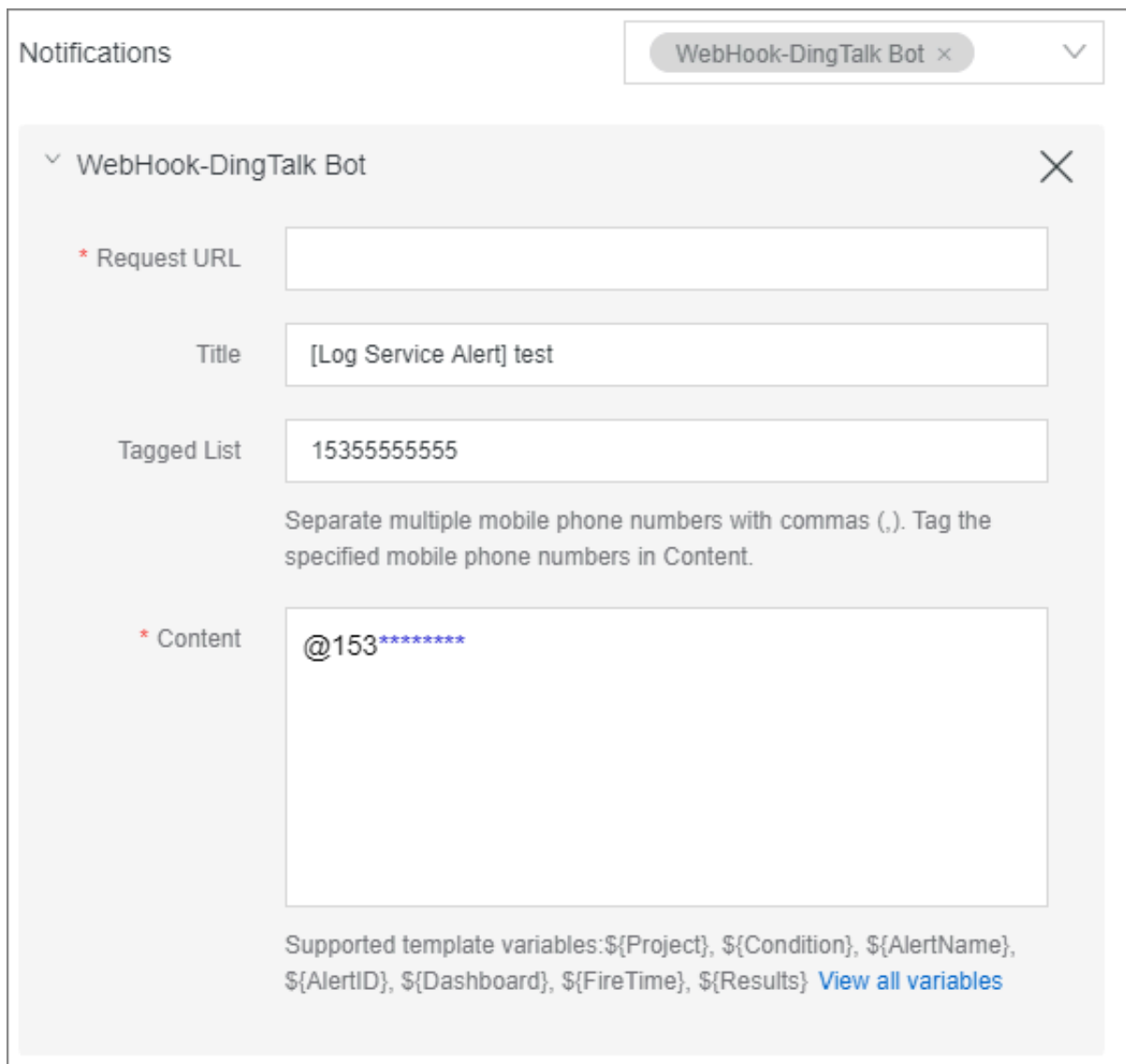
No Interval

Disable alert notifications

After receiving an alert notification, you can go to the alert overview page for temporarily disabling the notification feature, as shown in the following figure. In the Disable Alert Notifications dialog box, select a duration for which notifications are disabled, for example, 30 minutes. No notification will be sent within 30 minutes, even if an alert is triggered. After 30 minutes, the notification feature is automatically restored.

Allow DingTalk group members to process alerts

A DingTalk group is one of the most common alert notification channels. When configuring a DingTalk notification, you can @ a DingTalk group member to process the corresponding alert, as shown in the following figure.



The screenshot shows a configuration window for a 'WebHook-DingTalk Bot' notification channel. The window has a title bar 'Notifications' and a dropdown menu showing 'WebHook-DingTalk Bot'. Inside the window, there is a close button (X) in the top right corner. The configuration fields are as follows:

- * Request URL**: An empty text input field.
- Title**: A text input field containing '[Log Service Alert] test'.
- Tagged List**: A text input field containing '1535555555'.
- Content**: A text input field containing '@153*****'. Above this field, there is a note: 'Separate multiple mobile phone numbers with commas (,). Tag the specified mobile phone numbers in Content.'

At the bottom of the window, there is a note: 'Supported template variables: \${Project}, \${Condition}, \${AlertName}, \${AlertID}, \${Dashboard}, \${FireTime}, \${Results} [View all variables](#)'.



Note:

You must specify the mobile phone number of the corresponding member in both the `Tagged List` and `Content` fields. The `Tagged List` field is used to determine whether the at sign (@) in the `Content` field is a reminder or a common character.

Use template variables to enrich notifications

When configuring the notification method, you can use the template variables to enrich the notifications. Template variables can be used for the email title, DingTalk message title, and message content. Each time when an alert is triggered, an alert context is generated. Each variable in the context can be used as a template variable. For more information, see [#unique_65](#).

- You can reference the `Project`, `AlertName`, and `Dashboard` variables in the `${project}` format without case sensitivity.
- The context of each query is included in the `Results` array. Each element in the array corresponds to a chart associated with the alert. In most cases, there is only one element.

```
{
  "EndTime": "2006 - 01 - 02 15 : 04 : 05 ",
  "EndTimeTs": 1542507580,
  "FireResult": {
    "time": "1542453580",
    "field": "value1",
    "count": "100"
  },
  "FireResult Askv": "[ field : value1 , count : 100 ]",
  "Truncated": false,
  "LogStore": "test - logstore",
  "Query": "* | SELECT field , count ( 1 ) group by field",
  "QueryUrl": "http :// xxxx ",
  "RawResultCount": 2,
  "RawResults": [
    {
      "time": "1542453580",
      "field": "value1",
      "count": "100"
    },
    {
      "time": "1542453580",
      "field": "value2",
      "count": "20"
    }
  ],
  "RawResults Askv": "[ field : value1 , count : 100 ],[ field : value2 , count : 20 ]",
  "StartTime": "2006 - 01 - 02 15 : 04 : 05 ",
  "StartTimeTs": 1542453580
}
```

```
}
```

For more information about the fields, see [#unique_66](#). You can reference the fields in the Results array as follows:

- The fields of the array type are referenced in the `${fieldName[index]}` format. The value of index starts from 0. For example, `${results[0]}` indicates that the first element in the Results array is referenced.
- The fields of the object type are referenced in the `${object.key}` format. For example, the result of `${results[0].StartTimeTs}` is 1542453580.

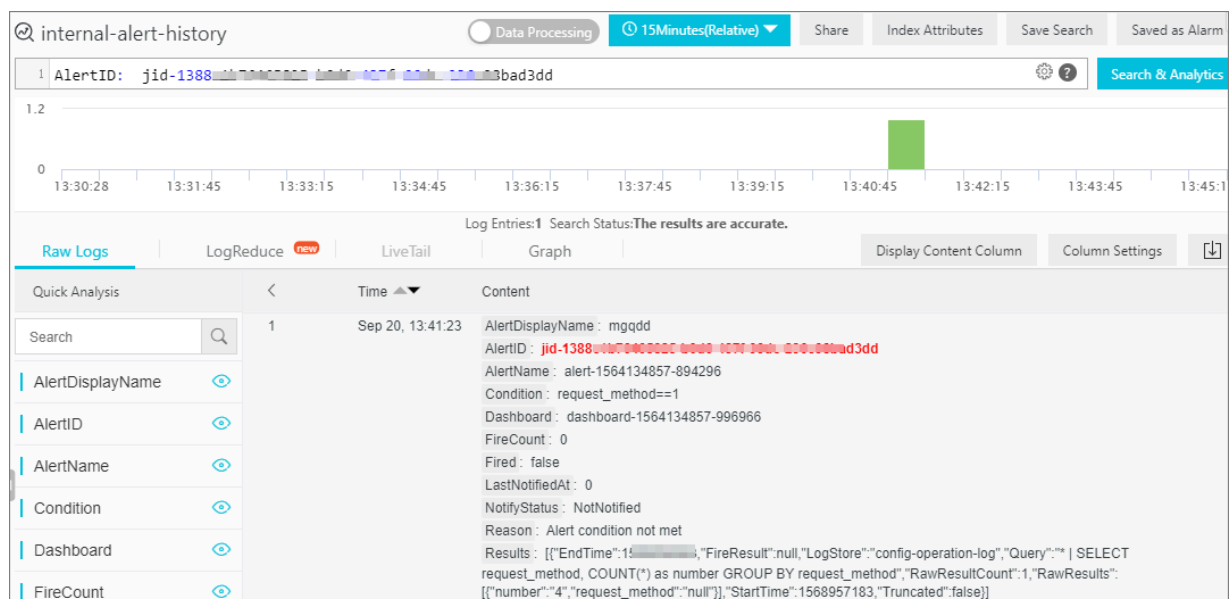


Note:

Only the fields in RawResults and FireResult are query results. These fields are case-sensitive. Other fields are case-insensitive.

Troubleshoot why the alert is not triggered

After an alert is configured, you can view alert statistics. For more information, see [#unique_67](#). You can view the context of a single alert in the internal-alert-history Logstore, as shown in the following figure.



For more information about the fields, see [#unique_66](#).

Each execution generates a unique alert ID and a corresponding log. The log contains the alert execution status and query result. If the query result exceeds 2 KB, it will be truncated. Logs can be used to troubleshoot why the alert is not triggered.