Log Service

Best Practices

Best Practices

Typical application scenarios

Log-service-based Solution

The Log Service can access various cloud products and third-party open-source ecosystems, greatly lowering your threshold for use to the maximum extent. Examples of the products include StreamCompute, data warehouse, and monitoring. In addition, the Log Service introduces ISV into security and other related areas, bringing the service of log analysis experts to users through the security cloud market.

Example

- Data collection: public network data
- Data cleaning and ETL
- Warehouse access

Typical Scenarios

- Log and big data analysis
    - Events generated by real-time collection systems like agents and APIs, such as visits and clicks.
    - Steaming computing conducted through the LogHub interface, such as targeted operations on the basis of analyzing users' favorite shows, channels with the largest viewership and on-demand views of different provinces.
    - Offline archiving of logs in data warehouse, including the detailed daily and weekly operational data and bills.
    - Applicable fields: Stream media, e-commerce, mobile analytics, game operations and so on. For example, the website CNZZ, is also a user of the Log Service.

  Log auditing

    - Logs are collected to the Log Service through agents in real time, removing the worries about deletion by mistake or intentional deletion by hackers.
    - The Log Query function can be used to quickly analyze access behaviors, such as the operational records to show queries of a certain account, object or operation.

- Logs are posted to OSS or MaxCompute for long-term storage to meet the compliance auditing requirements.
- Applicable fields: E-commerce websites, government platforms, websites and so on.

Troubleshooting

- During the process of development, add logs into clients, mobile devices, servers and modules and associate the logs using IDs.
- Collect the logs from various modules and receive real-time access statistics through CloudMonitor and StreamCompute.
- When there is a request or order error, instead of logging on to the server, the development team can use the log query function to directly check keywords, occurrences, and relevant impact of the error, quickly locate faults and limit the scale of impact.
- Applicable fields: Trading system, order system, mobile network and so on.

- Operation and maintenance (O&M) management
  - Collect logs from different applications that are deployed on hundreds and even thousands of machines (including errors, access logs, operation logs, and so on).
  - Centrally manage applications through different log libraries and machine groups.
  - Process different types of logs. For example, conduct steaming computing on access logs for real-time monitoring; index and query operational logs in real time; keep offline archives of important logs.
  - The Log Service offers a complete set of APIs for configuration management and integration.
  - Applicable fields: Users who have many services to manage.
- Others
  - Billing, business system monitoring, vulnerability detection, operation analysis, and mobile client analysis. In Alibaba Cloud, the Log Service is ubiquitous. All cloud products are using the Log Service for log processing and analysis.

The Log Service LogHub function supports real-time data collection and consumption. The real-time collection feature supports over 30 collection methods.

Data is usually collected in two different ways as described below. This document primarily discusses collecting data through LogHub streaming import (real-time).

Method	Pros	Cons	Example
Batch import	Large throughput, focusing on historical data	Poor real-time performance	FTP, OSS uploads, mailing hard drives, and SQL data export
Streaming import	Real-time, WYSIWYG (what you see is what you get), focusing on real-time data	Higher requirements on collection terminals	Loghub, HTTP upload, IOT and Queue

Background

"I Want Take-away" is an e-commerce website with its own platform involving users, restaurants and couriers. Users can place their take-away order through webpage, the App, WeChat and Alipay; when receiving an order, a merchant starts to prepare the food and the couriers nearby are automatically notified; then, one of the couriers picks up and delivers the take-away foods to the users.

Operational Requirements

During operation, the following issues are identified:

- Difficulty in getting users; despite a hefty advertising investment in various marketing channels (webpage ads and WeChat push messages), it is still impossible to evaluate the effects of these channels except for adding some new users.
- Users often complain about slow delivery, but what makes it slow? Order-taking, distribution or food preparation? How to improve?
- The user operation team organized some promotions from time to time (giving away coupons), but to little avail.
- In terms of scheduling, how to help merchants stock up food for the peak hours? How to send more couriers to a specific area?
- From the perspective of customer service, when users reported that they failed to place an order, what operations had they performed? Was there a system error?

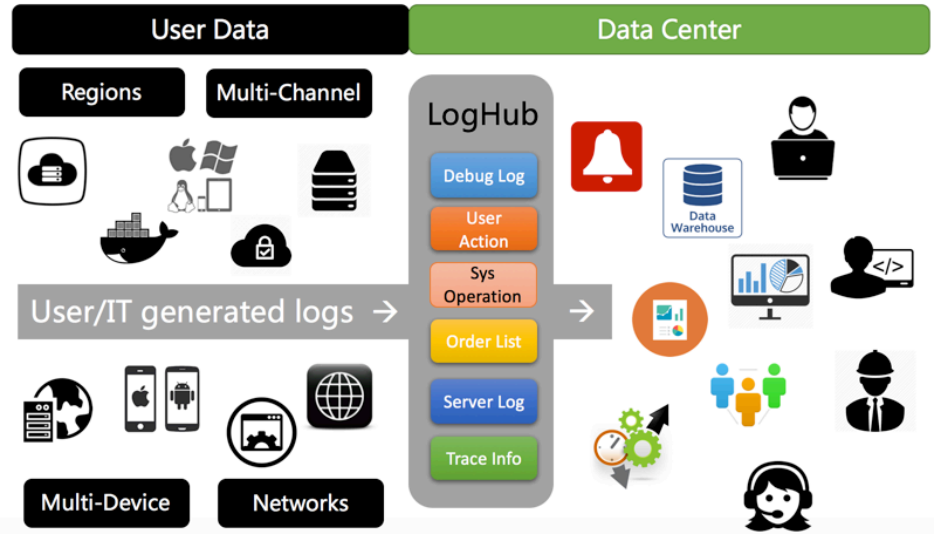
Data Collection Challenges

In digital operation, the first step is to figure out how to centrally collect the distributed log data, but there are a few challenges:

- Multiple channels: For example, advertisers, street promotions (leaflets), and so on

- Multiple terminals: Webpage, official social media account, mobile phone, web browser (desktop and mobile websites), and other terminals
- Heterogeneous networks: VPC, user-built IDC, Alibaba Cloud ECS, and so on
- Various development languages: Java for the core system, Nginx server for the front end and C ++ for the backend payment system
- Devices: Merchants' devices are running on different platforms (X86, ARM, etc.)

We must gather the logs distributed externally and internally for unified management. In the past, this took massive and diversified work. Now we can use LogHub's collection feature for unified access.



Unified Log Management and Configuration

1. Create a log management project, for example, MyOrder.
2. Create the log library LogStore 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 (error logs)
    - alipay-server
    - alipay-app
    - Deliver-app (courier app status)
    - Deliver-error (error logs)
    - Web-click (H5 webpage clicks)
    - Server-access (service-side Access-Log)
    - Server-app (application)
    - Coupon (application coupon logs)
    - Pay (payment logs)
    - Order (order logs)
3. For example, some intermediate LogStores can be created, if it is necessary to cleanse and run ETL jobs on the raw data.

- Refer to Data Cleansing and ETL

For more operations, refer to Quick Start/Management Console.

Collection of User Promotion Logs

To attract new users, there are two common methods:

- Directly give away coupons upon sign-up on the website
- Offer coupons for scanning QR codes through other channels
  - QR codes on leaflets
  - Scan QR codes on a webpage to log on

Implementation

Define the following sign-up server link and generate QR codes (for leaflets and webpages) for users to scan and sign up. When a user scans a QR code on a webpage to sign up, we can identify the user as being from a specific source and create logs accordingly.

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

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

```
2016-06-20 19:00:00 e41234ab342ef034,102345,5k4d,467890
```

In the preceding command:

- time: Time of registration.
- Session: The current browser session, used for behavior tracking.
- Source: Source channels. For example, Campaign A is labelled as “10001”, leaflets “10002”, and elevator ads “10003”.
- Ref: Referral account, which means whether someone else recommends the user to sign up; left blank if there is none.
- Params: Other parameters.

Collection method:

- The application exports the logs to the hard drive, which are collected through the Logtail Collection.
- The application writes the logs using SDK. Refer to the SDK.

Service-side Data Collection

Alipay/WeChat official account programming is a typical web-side model that generally utilizes three

types of log:

Nginx/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/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: The application layer logs capture details about events that occurred, like time, location, result, delay, method and parameter, and usually end with extended fields

```
{
"time":"2016-08-31 14:00:04",
"localAddress":"10.178.93.88:0",
"methodName":"load",
"param":["31851502"],
"result":....
"serviceName":"com.example",
"startTime":1472623203994,
"success":true,
"traceInfo":"88_1472621445126_1092"
}
```

Application-layer error logs: Time of the error code, and the error code, the reason, and others

```
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

- Logs are written to local files using the Logtail and the configuration regular expressions are written to a specified LogStore.
- Logs generated in Docker can be collected using the Container-Service-Integrated Log Service.
- For Java programs, use the Log4J Appender (without saving logs on hard drives), LogHub Producer Library (for high-concurrency client-side write), or Log4J Appender.

- For C #, Python, Java, PHP and C, use the SDK Write.
- For Windows Server, use the LogStash Collection.

Access to end user logs

- Mobile client: Use the mobile terminal SDK IOS, Android or MAN (mobile analytics) for log access.
- ARM devices: The ARM platform can use the Native C for cross-compiling.
- Merchant platform devices: Devices running on an X86 platform can use SDK, and those running on the ARM platform can use Native C for cross compiling.

Desktop website/mobile website users' page actions

The users' page actions for collection are divided into two types:

- Page actions with backend server interaction: For example, placing an order, logging on, and logging out.
- Page actions without backend server interaction: Requests that are processed directly at the front end, for example, scrolling a page, closing a page, and so on.

Implementation

- For the first type, refer to the service-side collection method.
- For the second type, use Tracking Pixel/JS Library to collect page actions. Refer to the Tracking Web Interface.

Server Log Maintenance

For example:

Syslog logs

```
Aug 31 11:07:24 zhouqi-mac WeChat[9676]: setupHotkeyListenning 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:816103453552
saved_cursor:1469780553885742676
seek count:62900
seek data redo
log:pangu://localcluster/redo_data/41/example/2016_08_30/250_1472555483
user_cursor:1469780553885689973
```

Trace logs

```
[2013-07-13 10:28:12.772518]    [DEBUG] [26064]  __TRACE_ID__:661353951201
__item__:[Class:Function]_end__  request_id:1734117   user_id:124 context:.....
```

Implementation

Refer to the service-side collection method.

Data Collection in Different Network Environments

LogHub provides access points in each Region that provides three access methods:

- Intranet (classic network): For service access within the current Region, offering the best bandwidth link quality (recommended).
- Internet (classic network): Since it can be accessed by anyone, the access speed may vary with link quality, and HTTPS is recommended for security and protection.
- Private network (VPC): For accessing VPC network within the current Region.

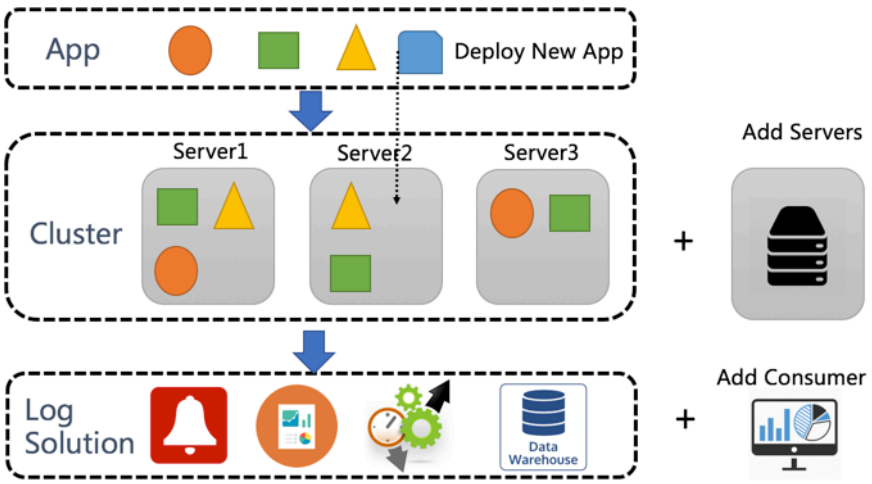
For more information, please refer to the Network Access. You can always find a suitable solution.

Others

- Refer to the Complete LogHub Collection Methods.
- Refer to the Real-time Log Consumption, which involves functions like streaming computing, data cleansing, data warehousing and index query.

Here is a typical scenario: A server (container) stores a huge volume of application log data generated in different directories.

- Developers deploy and deprecate new applications.
- The server can scale out as needed, for example, scaled out during the peak periods and scaled in during the slack periods.
- The log data is to be queried, monitored and warehoused depending on the different and ever-changing requirements.



Challenges in the process

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

Each application can generate Access, OpLog, Logic and Error logs. When more applications are added and the dependence exists between applications, the volume of logs explodes.

Here is an example of an online takeaway website:

Category	Application	Log Name
Web	nginx	wechat-nginx (WeChat server nginx log)
	nginx	alipay-nginx (Alipay server nginx log)
	nginx	server-access (server Access-Log)
Web-Error	nginx-error	alipay-nginx (nginx error log)
	nginx-error	...
Web-App	tomcat	alipay-app (Alipay server application logic)
	tomcat	...
App	Mobile App	deliver-app (delivery app status)
App-Error	Mobile App	deliver-error (error log)
Web	H5	Web-click (H5 page click)
server	server	server internal logic log
Syslog	server	server system log

2. Logs are consumed for different purposes

For example, AccessLog can be used for billing, and for users to download; OpLog is to be queried by a DBA, which also requires BI analysis and full-link monitoring.

3. Environment and changes

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

- Application server resizing
- 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 auto-scaling - adding and removing servers as needed)
- Standardized application deployment without complicated configuration
- Easy compliance with log processing requirements

Log service solution

The LogHub feature of the Log Service defines the following concepts on log access, and uses Logtail to collect logs:

- Project: a management container
- LogStore: represents a log source
- Machine group: represents the directory and format for logs
- Config: indicates the path to logs

The relationships between these concepts are as follows:

- A project includes multiple LogStores, machine groups and configs, with different projects meeting different business requirements.

  An application can have multiple types of logs. There is a LogStore and a fixed directory (with the same config) per log type.

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

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

```
  app --> machineGroup1, mahcineGroup2
machineGroup1 --> app1, app2, app3
```

The collection directory defined in the config is applied to machine groups, and collected into any LogStore.

```
  config1 * machineGroup1 --> Logstore1
config1 * machineGroup2 --> logstore1
config2 * machineGroup1 --> logstore2
```

Advantages

Convenient: It provides WebConsole/SDK and other tools for batch management.

Large-scale: It manages machines and applications in the millions.

Real-time: Collection configuration takes effect just in minutes.

Elastic:

- The machine ID function supports auto scaling up of servers.
- LogHub supports auto scaling. For details, refer to the shard overview.

Stable and reliable: No human intervention is required.

For information on real-time computing, offline analysis, indexing and other query capabilities in log processing, refer to Service Introduction.

- LogHub: Real-time collection and consumption. Uses 30+ methods to collect massive data for real-time downstream consumption.
- LogShipper: Stable and reliable log shipping. It delivers data from LogHub to storage services (OSS/MaxCompute/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.

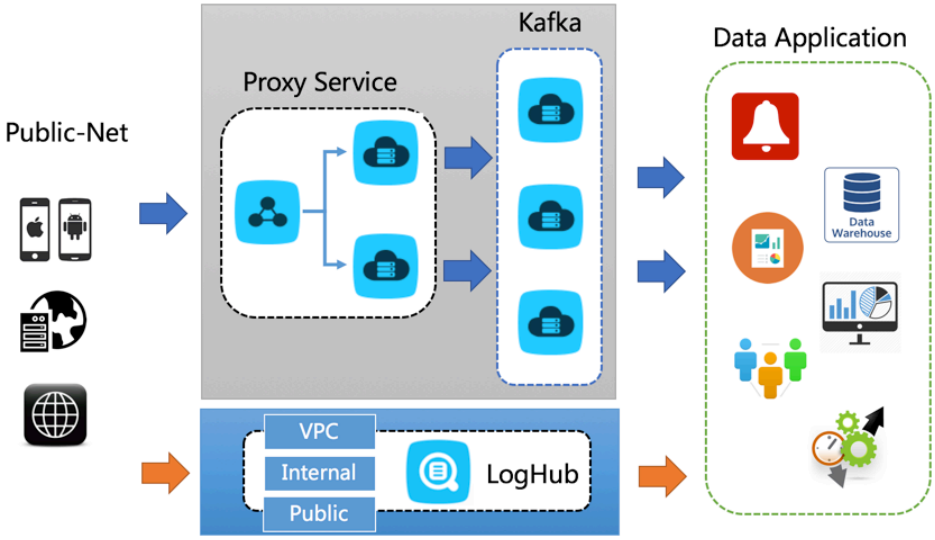
Collect public network data

In some application scenarios, it is required to collect data from a public network (for example, mobile client, HTML webpage, PC, server, hardware devices, camera, and so on) for real-time processing.

In a traditional architecture, the function above can be achieved by using a combination of front-end server and Kafka. But now, such architecture can be replaced by the Log Service with solutions that are more reliable, cost-effective, elastic and secure.

Scenarios

In the public network, data can be collected from mobile client, external servers, webpages and various devices. The data, once collected, needs to be used for applications like real-time computing and data warehousing.



Solution 1: Front-end server + Kafka

Kafka does not support the RESTful Protocol and is used in clusters in most cases. Therefore, it is generally required to set up a Nginx server as the public network proxy, and then use LogStash or API to write data in the message middleware like Kafka through Nginx.

The required infrastructure is:

Device	Quantity	Purpose	Price	
ECS server	2 units	1 core, 2 GB	Front-end host, load balancing, and mutual backup	22.26 USD per unit * month
Load balancer	1 unit	Standard	Pay-as-billing instance	3.6 USD per month (lease) + 0.078 USD per GB (data traffic)
Kafka/ZK	3 units	1 core, 2 GB	Data write and	22.26 USD per

			processing	unit * month

Solution 2: Use LogHub

Use Mobile SDK, LogtTil or Web Tracking JS to directly write data into LogHub EndPoint.

The required infrastructure is:

Device	Purpose	Price
LogHub	Real-time data collection	<0.003125 USD per GB

Scenario Comparison

Scenario 1: Up to 10 GB of data is collected each day, which generates around 1 million write requests. (The 10 GB in this example is the compressed size. So the actual size of data ranges from 50 GB to 100 GB.)

```
Solution 1:
--------------
Load balancer (lease): 0.005 * 24 * 30 = R3.6 USD
Load balancer (traffic): 0.078 * 10 * 30 = 23.4 USD
ECS cost: 22.26 * 2 = 44.52 USD
Kafka ECS: Free, if shared with other services
Total: 71.52 USD per month

Solution 2:
--------------
LogHub traffic: 10 * 0.05 * 30 = 15 USD
Number of LogHub requests: 0.03 (assuming there are 1 million requests per day) * 30 = 0.9 USD
Total: 15.9 USD per month
```

Scenario 2: Up to 1 TB of data is collected each day, which generates around 100 million write requests.

```
Solution 1:
--------------
Load balancer (lease): 0.005 * 24 * 30 = 3.6 USD
Load balancer (traffic): 10.078 * 1000 * 30 = 2340 USD
ECS cost: 22.26 * 2 = 44.52 USD
Kafka ECS: Free, if shared with other services
Total: 2388.12 USD per month

Solution 2:
--------------
LogHub traffic: 0.045 * 1000 * 30 = 1350 (tiered pricing)
Number of LogHub requests: 0.03 * 100 (assuming there are 100 million requests per day) * 30 = 90 USD
Total: 1440 USD per month
```

Comparison of Solutions

The two scenarios above show that, you can use LogHub to collect data from the public network at a very competitive cost. In addition, Solution 2 outperforms Solution 1 in the following aspects:

- Auto scaling: MB-PB/Day traffic that can be controlled freely
- Abundant permission control options: use ACL to control the read and write permissions
- HTTPS compatibility: encrypted transmission
- Log post at no cost: Access to data warehouse without additional development
- Detailed metric data: know your business
- A rich set of SDK interfaces with upstream and downstream systems: complete downstream interfaces just like Kafka, and deep integration with Alibaba Cloud and open-source products

In the era of mobile Internet, data upload using mobile apps is increasingly common. It is expected that logs in mobile apps are directly uploaded to Log Service, instead of being transferred by an app server, so that users can focus more on their service logic development.

In normal mode, the AccessKey of the primary account is required when logs are written into Log Service for authentication and anti-tampering. If a mobile app accesses Log Service in this mode, your AccessKey must be stored on the mobile end, causing a data security risk of AccessKey disclosure. Once the AccessKey is disclosed, your must upgrade the mobile app and replace the AccessKey, which is too costly. Another way to upload logs on the mobile end to Log Service is to use the user’ s app server. In this mode, however, if the number of mobile apps is large, the app server must carry all data on the mobile end. This mode has a high requirement on the server size.

To avoid the preceding problems, Log Service provides a safer and more convenient scheme to collect mobile app logs. It uses RAM to set up direct data transfer for mobile apps based on mobile services. Different from the scheme of directly using the AccessKey to access Log Service, this scheme does not require AccessKey storing on the app end, eliminating the risk of AccessKey disclosure. A temporary token with a life cycle is used, which is safer. You can also configure more complex access control policies for the token, for example, limiting the access permission of the IP segment. The cost of this scheme is low. You do not require many app servers because mobile app data is stored on the cloud platform and only the control flow is sent to the app server.

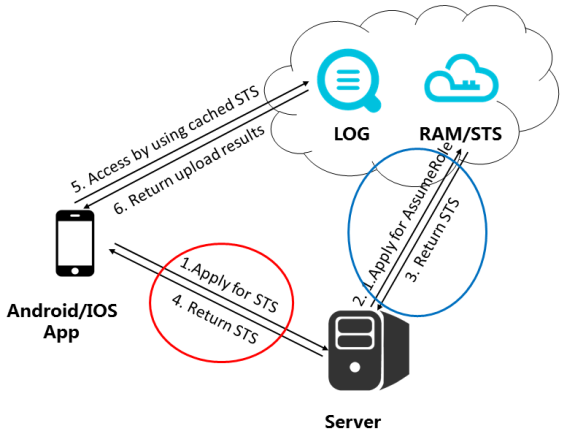
You can create a RAM user role of Log Service and configure an app as a RAM sub-user to assume this role, so that you can set up Log Service-based direct log transfer for mobile apps within 30 minutes. Direct data transfer is a service that allows mobile apps to directly connect to Log Service, with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

Advantages

By using RAM to set up Log Service-based direct data transfer for mobile apps, this scheme has the following advantages:

- Safer access and temporary and flexible permission assignment and authentication.
- Low cost with fewer app servers. The mobile apps are directly connected to the cloud platform and only the control flow is sent to the app server.
- High concurrency and supporting massive users. Log Service provides large upload and download bandwidths.
- Elastic. Log Service allows unlimited storage space resizing.

The architecture diagram is as follows:



NOTE:

Module	Description
Android/iOS mobile app	The app running on the end-user’ s cell phone, and the source of logs.
LOG	Alibaba Cloud Log Service, responsible for storing log data uploaded by the app. For details, see Log Service description on Alibaba Cloud website.
RAM/STS	Alibaba Cloud RAM, which provides the user identity management and resource access control services, and generates temporary access credentials.
App server	The app background service developed for the Android/iOS mobile app and used to manage the tokens used for data upload/download by the app and the metadata of the app-uploaded data.

Configuration process

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

To avoid the risk of information leakage, the Android/iOS app does not store the

AccessKeyID and AccessKeySecret. 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 (editable by the app server), the Android/iOS app can use this token to access Log Service within the next 30 minutes. 30 minutes later, the app must request a new token again.

The app server checks the validity of the above request and then returns a token to the app.

After obtaining the token, the mobile app can access Log Service.

This document mainly describes how to apply for the token from RAM using the app server and how to obtain the token for Android/iOS apps.

Procedure

1. Authorize a user role to operate Log Service.

Create a RAM user role of Log Service and configure an app as a RAM sub-user to assume this role. For details, see Authorize a user role to operate Log Service.

After configuration, you can obtain the following parameters.

- AccessKeyID and AccessKey of the RAM sub-user
- Resource path RoleArn of the role

2. Set up an app server.

For easy development, this tutorial provides sample programs in multiple languages. The download URLs are listed at the bottom of this document.

Each downloaded language package contains the following configuration file config.json:

```
{
"AccessKeyID" : "",
"AccessKeySecret" : "",
"RoleArn" : "",
"TokenExpireTime" : "900",
"PolicyFile": "policy/write_policy.txt"
}
```

NOTE:

1. **AccessKeyID**: The ID of your AccessKey.
2. **AccessKeySecret**: The secret of your AccessKey.

3. **RoleArn**: The RoleArn of the user role.
4. **TokenExpireTime**: The expiration time of the token obtained by the Android/iOS app. The minimum value is 900s and does not need to be changed.
5. **PolicyFile**: The file that lists the permissions the token grants. The default value does not need to be changed.

This document provides two token files defining the most common permissions in the policy directory.

- write_policy.txt: Specifies a token that grants the write permission for the project of this account.

  readonly_policy.txt: Specifies a token that grants the read permission for the project of this account.

  You can design your policy file as required. For details about the permissions, see Permission control of Log Service.

Format of the returned data:

```
//Returned correct result
{
"StatusCode":200,
"AccessKeyId":"STS.3p***dgagdasdg",
"AccessKeySecret":"rpnwO9***tGdrddgsR2YrTtI",
"SecurityToken":"CAES+wMIARKAAZhjH0EUOIhJMQBMjRywXq7MQ/cjLYg80Aho1ek0Jm63XMhr9Oc5s˙∂˙∂3qaPer8p
1YaX1NTDiCFZWFkvlHf1pQhuxfKBc+mRR9KAbHUefqH+rdjZqjTF7p2m1wJXP8S6k+G2MpHrUe6TYBkJ43GhhTVFMu
M3BZajY3VjZWOXBIODRIR1FKZjIiEjMzMzE0MjY0NzM5MTE4NjkxMSoLY2xpZGSSDgSDGAGESGTETqOio6c2RrLWRlb
W8vKgoUYWNzOm9zczoqOio6c2RrLWRlbW9KEDExNDg5MzAxMDcyNDY4MThSBTI2ODQyWg9Bc3N1bWVkUm9sZ
VVzZXJgAGoSMzMzMTQyNjQ3MzkxMTg2OTExcglzZGstZGVtbzI=",
"Expiration":"2017-11-12T07:49:09Z",
}

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

Description of the returned correct result: (The following five variables comprise a token)

Variable	Description
StatusCode	The result that the app retrieves the token. The app returns 200 if the token is successfully retrieved.
AccessKeyId	The AccessKeyID that the Android/iOS app obtains when initializing the Log client.

AccessKeySecret	The AccessKeySecret that the Android/iOS app obtains when initializing the Log client.
SecurityToken	The token the Android/iOS app initializes.
Expiration	The time that the token expires. The Android SDK automatically determines the validity of the token and then retrieves a new one as needed.

Description of the returned error:

Variable	Description
StatusCode	The result that 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 and later versions, after downloading and decompressing the package, create a Java project, copy the dependency, code, and configuration to the project, and run the main function. The program listens to port 7080 and waits for the HTTP request by default. Perform the operations in other languages in a similar way.

3. The mobile app creates an HTTP request to obtain the token from the app server.

The formats of the HTTP request and response are as follows:

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

Response:
{
"StatusCode":"200",
"AccessKeyId":"STS.XXXXXXXXXXXXXXX",
"AccessKeySecret":"",
"SecurityToken":"",
"Expiration":"2017-11-20T08:23:15Z"
}
```

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

Download the source code

Sample code of the app server

PHP, JAVA, Ruby, and Node.js

Processing-Data cleaning_ETL

An assumption during log processing is: Data is not perfect. There is a gap between the raw data and the final results, so the raw data needs to be cleansed, converted and sorted using methods like ETL (Extract Transformation Load) to get the final results.

Example

"I Want Take-away" is an e-commerce website with its own platform involving users, restaurants and couriers. User can place their take-away orders through webpage, the App, WeChat and Alipay; when receiving an order, a merchant starts to prepare the food and the take-away couriers nearby are automatically notified; then, one of the couriers picks up and delivers the food to the users.

The operation team has two jobs:

- Determine couriers' locations and assign orders by location.
- Understand how coupons and cash are used and distribute coupons by locations as part of interactive operations.

Process the courier's location information (GPS)

GPS data (X and Y) is reported once every minute through the courier's app in the following format:

```
2016-06-26 19:00:15 ID:10015 DeviceID:EXX12345678 Network:4G GPS-X:10.30.339 GPS-Y:17.38.224.5
Status:Delivering
```

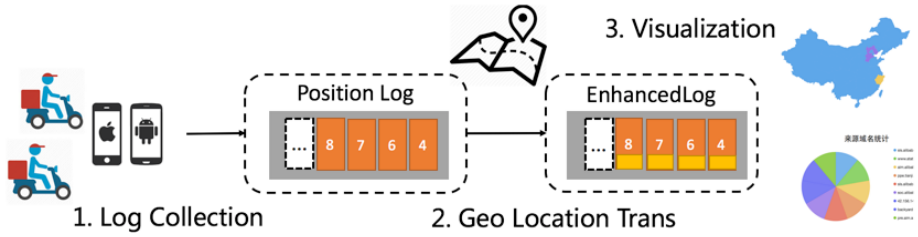
The data feed records the reporting time, courier ID, network in use, device serial number, and coordinates (GPS-X and GPS-Y). The longitude and latitude given by GPS are very accurate, but the operation team actually does not need such accurate data to understand the current status statistics for each region. Therefore, it is necessary to transform the raw data and convert the coordinates into readable fields like city, region, ZIP code and so on.

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

This is a typical ETL requirement. Use the LogHub function to create two LogStores (PositionLog), and the transformed LogStore (EnhancedLog). Run the ETL application (for example, Spark Streaming, Storm, or Consumer Library enabled in a container) to subscribe to the real-time PositionLog, convert the coordinates, and then write the EnhancedLog. Carry out real-time computing operations to visualize, or create an index to query the EnhancedLog data model repository.

The recommended architecture for the entire process is as follows:

1. Each courier’ s location is shown on the app and their GPS locations are reported every minute and written into the first LogStore (PositionLog)
   - We recommend using the LogHub Android/IOS SDK and MAN (mobile analytics) for accessing the mobile device log.
2. Use the real-time application to subscribe to the real-time PositionLog data, and write the processed data into EnhancedLog LogStore.
   - We recommend using the Spark Streaming, Storm, Consumer Lib (an auto-balancing programming mode) or SDK subscription.
3. Process the enhanced log, for example, visualization of computed log data. Recommendations:
   - LogHub Accessible to StreamCompute
   - LogShipper posts (OSS, E-MapReduce, Table Store, and MaxCompute)
   - LogSearch: order query etc.



Payment Order Desensitization and Analysis

The Payment Service receives a payment request which includes the payment account, payment method, amount, and coupon.

- Part of the sensitive information needs to be desensitized.
- Two types of information, coupon and cash, need to be stripped from the payment information.

The entire process is as follows:

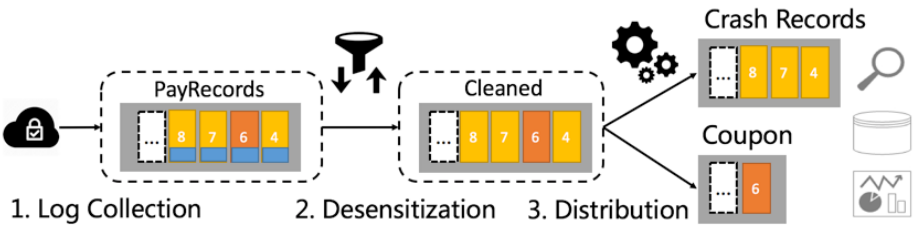
Create four LogStores for raw data (PayRecords), desensitized data (Cleaned), cash order (Cash), and coupon (Coupon) respectively. The application uses Log4J Appender to write the order data into the raw data LogStore (PayRecords).

We recommend using the Log4J Appender or Producer Library, with which the sensitive data is not written to the disk.

The desensitization application consumes the PayRecords LogStore in real time and writes the desensitized data into the Cleaned LogStore after stripping off the account-related information.

The traffic delivery application consumes the Cleaned LogStore in real time and saves the two types of information, coupon and cash, through business logics into the corresponding LogStore (Cash and Coupon) for subsequent processing.

We recommend using the Spark Streaming, Storm, Consumer Lib, an auto-balancing programming mode) or SDK subscription for real-time desensitization and traffic delivery.



Others

- Under the LogHub function, the account permission of each LogStore can be controlled using RAM. For details, refer to the RAM.
- LogHub current read and write capabilities can be obtained from the Acquisition through Monitoring, and the consumption status can be viewed through the console View.

Log Service Overview

As an important infrastructure for Alibaba Cloud, the Log Service supports the collection and distribution of all cluster log data on Alibaba Cloud. Applications like Table Store, MaxCompute and CNZZ use the Log Service Logtail to collect log data and consume data using API for export to a downstream real-time statistics system or offline system for statistics and analysis. As an infrastructure, the Log Service provides the following features:

- Reliability: Proven by Alibaba Group’ s internal users and tested by the enormous traffic during each Single’ s Day shopping festival over the years, the Log Service can ensure data reliability and no data loss.
- Scalability: When data traffic goes up, the number of shards can be increased to quickly and dynamically scale up the processing capabilities.
- Accessibility: Manages the collection of logs from tens of thousands machines with one key.

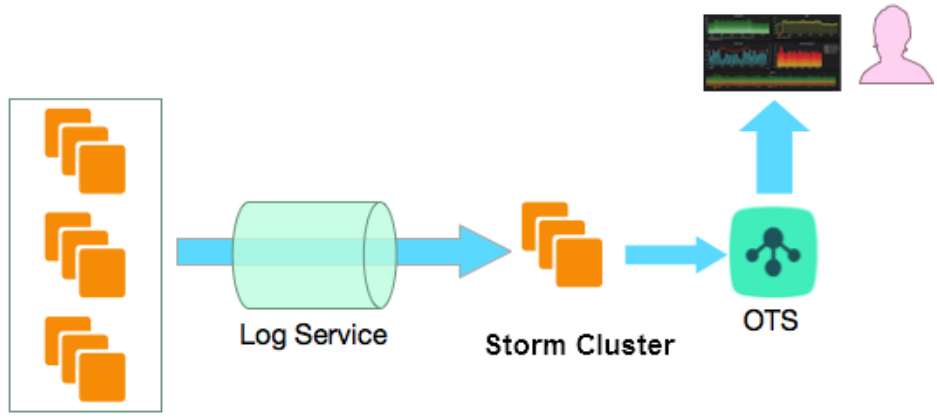
The Log Service helps users collect logs, unify log format and offers APIs for downstream consumption. Downstream systems can be connected to multiple other systems for repeated log consumption, such as using imports from Spark or Storm for real-time computing, or using imports from Elasticsearch for searching, allowing users to collect once and consume multiple times. Among the various data consumption scenarios, monitoring is the most common one. This article introduces Alibaba Cloud’ s log-service-based monitoring system.

The Log Service collects the metric data of all clusters as logs to the server. In this solution, logs are collected from multiple clusters and heterogeneous systems, and monitoring data is unified into the same format and sent to the Log Service.

The Log Service brings the following capabilities to the monitoring system.

- Unified machine management: Once Logtail is installed, all the subsequent operations can be performed on the log server.
- Unified configuration management: You only need to configure what logs files you want to collect at the server once, the configuration can be automatically distributed to all machines.
- Structured data: All data can be formatted to fit the Log Service’ s data model to facilitate downstream consumption.
- Elastic serviceability: The ability to process massive data read and write.

Monitoring System Architecture



How to Set Up a Monitoring System

1. Collect the metric data

Refer to Quick Start to learn how to configure Log Service log collection and ensure that the logs have been collected by the Log Service.

2. For API consumption data used by the middleware

Refer to the How to Use SDK, and select a suitable SDK version. Consume log data in batches from the Log Service using the SDK PullLog interface, and synchronize the data to the downstream real-time computing system.

3. Set up a Storm real-time computing system

Select Storm or other types of real-time computing system, configure the computing rules, choose the monitoring metrics for computing, and then write the computing result into Table Store.

4. Display the monitoring information

Read the metric data stored in Table Store for front-end display; or read the metric data and trigger alarms based on the data results.

Given that cloud services are advantageous for supporting Pay-As-You-Go without reserving resources, all cloud products have billing demands. This article describes a billing method based on Log Service, which can process hundreds of billions of logs every day and is applied to many cloud products.

Process of metering logs generating billing results

The metering log records possible billing items, and the backend billing module calculates the results based on the billing items and rules, and the final bill is generated. For example, the following original access log records the use of a project:

```
microtime:1457517269818107 Method:PostLogStoreLogs Status:200 Source:10.145.6.81 ClientIP:112.124.143.241 Latency:1968 InFlow:1409 NetFlow:474 OutFlow:0 UserId:44 AliUid:1264425845278179 ProjectName:app-myapplication ProjectId:573 LogStore:perf UserAgent:ali-sls-logtail APIVersion:0.5.0 RequestId:56DFF2D58B3D939D691323C7
```

The billing program reads the original log and generates the usage data in various dimensions (including traffic, number of use, and outbound traffic) as defined in the rules:

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	aquilapreproductionenviron	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 billing scenarios based on metering logs

- For electric power companies: A log is generated and sent to these companies every 10 seconds, which records the power consumption, peak value, and average value of each user ID during the 10 seconds. Also, daily, hourly, and monthly bills are provided to the users.
- For ISPs: The base station sends the behaviors (surfing the Internet, making phone calls, sending SMS messages, and using VoIP), consumed traffic, and their durations of a mobile number every 10 seconds, and the backend billing service calculates the fees incurred during this period.
- For weather forecast API services: The user requests are billed based on the types of the called API, the city of the user, the query type, and the result size.

Requirements and challenges

The billing system has the following requirements:

- Accurate and reliable: The billing result must be precise.
- Flexible: The system supports data completing. For example, recalculation is supported for data correction when some data was not pushed.
- Real-time: The system supports billing in seconds and quick disconnection for arrear scenarios.

Other demands:

- Bill correction: Ideal billing is supported for reconciliation when real-time billing fails.
- Detail query: The users can view their own consumption details.

Also, the following two challenges exist:

- Increasing data volume: With the growth of users and calls, the data volume expands, and how to maintain the auto scaling of the system architecture becomes a challenge.
- Error tolerant processing: Bugs may exist in the billing program, and how to ensure the metering data is isolated from the billing program becomes another challenge.

This article describes the billing method developed by Alibaba Cloud based on Log Service. This method has been running online reliably for several years without any miscalculation and latency and provides reference for unit prices.

System architecture

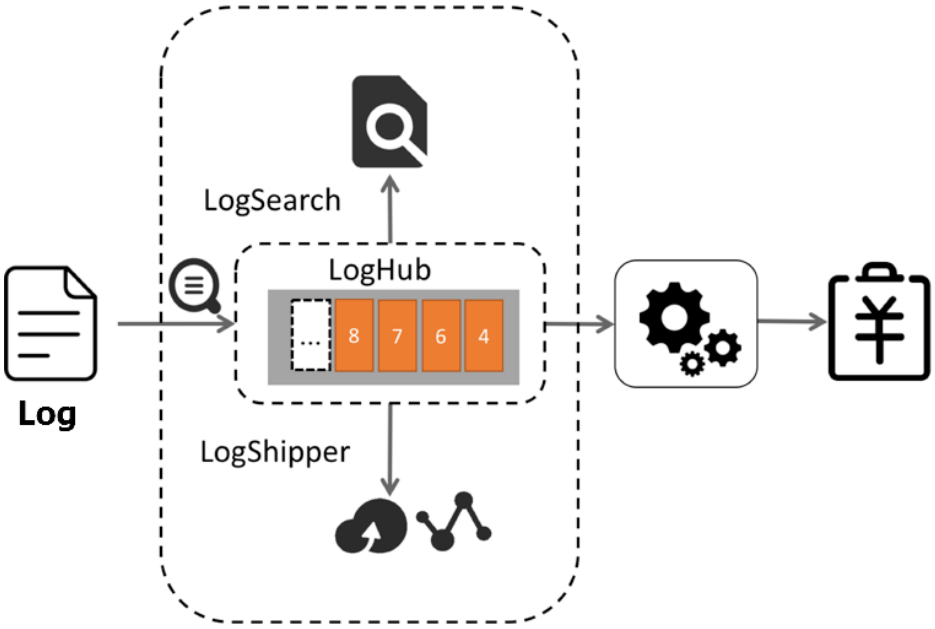
The following uses the LogHub feature of Alibaba Cloud Log Service as an example:

Use LogHub to collect metering logs in real time and connect with the metering program: LogHub supports more than 30 APIs and access methods for easy access to metering logs.

The metering program regularly consumes the incremental data in LogHub, calculates the result and generates billing data in the memory.

(Additional) The index query of metering logs can be configured for detailed data queries.

(Additional) The metering logs are pushed to OSS and MaxCompute to be stored offline for T+1 reconciliation and statistics.



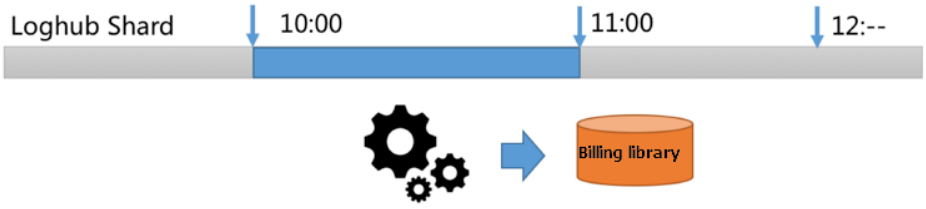
Internal structure of the real-time metering program:

Select the logs within a period (such as 10:00 to 11:00) using the GetCursor feature of the LogHub reading API.

Consume the data of this period through the PullLogs API.

Collect and calculate the data in the memory and generate the billing data.

Similarly, the calculation logic of the selected period can be changed to 1 minute, or 10 seconds, etc.



Performance analysis:

- Assume that one billion metering logs are introduced per day with each log containing 200 bytes, and the total data volume is 200 GB.
- The default LogHub SDK or Agent provides the compression feature. Thus, the actually stored data volume is 40 GB (generally at least a five-time compression rate is available) and the hourly data volume is 40/24 = 1.6 GB.
- The LogHub reading API can read up to 1,000 packages at a time (each of which is limited to 5 MB). The full data can be read within two seconds under the gigabyte network.

- The data of the metering log for one hour can be read within five seconds, which include the data accumulation and calculation time in the memory.

Solutions to large data volume scenarios

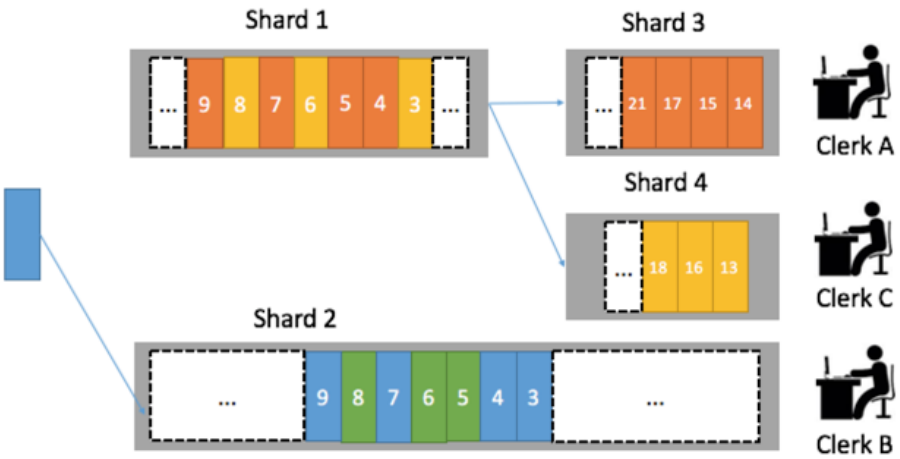
In some billing scenarios (such as the ISP and IoT scenarios), the volume of metering logs is extremely large (such as, for 10 trillion metering logs, the data volume is 2 PB per day). Namely, 16 TB data after compression is to be read per hour, which takes 1,600 seconds under the 10-gigabit network. Thus, quick bill generation cannot be implemented.

1. Control the volume of the billing data to be generated

To do this, modify the metering log generation program (for example, Nginx) by implementing aggregation in the memory and dumping the aggregated metering log results every one minute. In this way, the data volume is related to the total number of users. Assume that 1,000 users exist during the period, then the hourly data volume is 1000 *200* 60 = 12 GB (240 MB after compression).

2. Process metering logs concurrently

Each Logstore of LogHub can be assigned with different number of shards. In this case, three shards and three metering consumption programs are assigned. To ensure that the metering data of a single user is always processed by the same consumption program, the ID of the user can be hashed to the corresponding constant shard. For example, the user data for the West Lake District of Hangzhou can be hashed to Shard 1 while that for the Shangcheng District of Hangzhou can be hashed to Shard 2. By doing this, the backend metering programs can be expanded horizontally.



Other issues

1. How to perform data completing?

Each Logstore of LogHub can be configured with a lifecycle (1 to 365 days). If a billing program needs to consume the data again, the program can calculate the data based on any time period within the

lifecycle.

2. How to deal with the billing logs dispersed on multiple servers (front-end machines)?

1. Collect those logs with Logtail Agent in real time.
2. Define a group of dynamic machines for auto scaling with machine identification.

3. How to implement detail query?

Create an Index for the data in LogHub, which supports Real-Time Query and Statistical Analysis. For example, to query extremely large metering logs:

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

Once indexing is enabled for the data in LogHub, real-time query and analysis features are available.

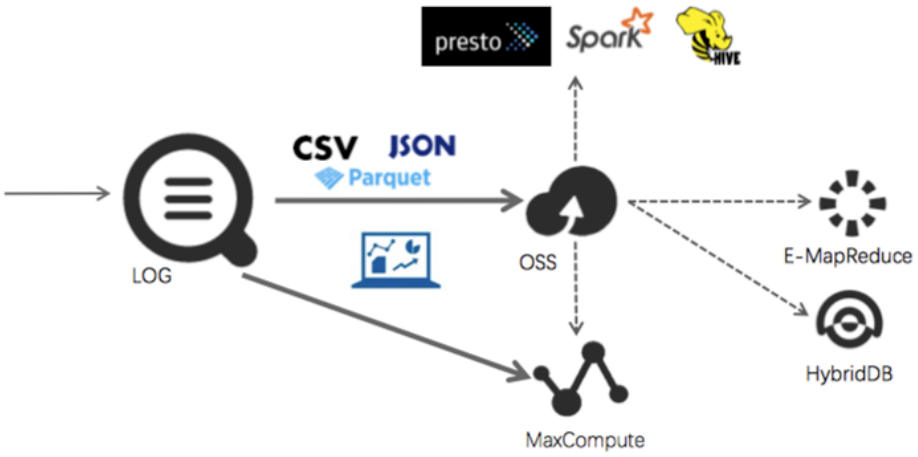
Also, you can perform statistical analysis after the query:

```
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
sls-weloop	526956
ykf-testalipay	524523
fc-monitor-ol	524515
syt-accesslog	486647

4. Store logs and perform T+1 reconciliation

The data delivery feature in LogHub provided by the Log Service supports storing logs on the OSS or MaxCompute in custom shards and storage formats, and calculates using E-MapReduce, MaxCompute, HybridDB, Hadoop, Hive, Presto, and Spark.



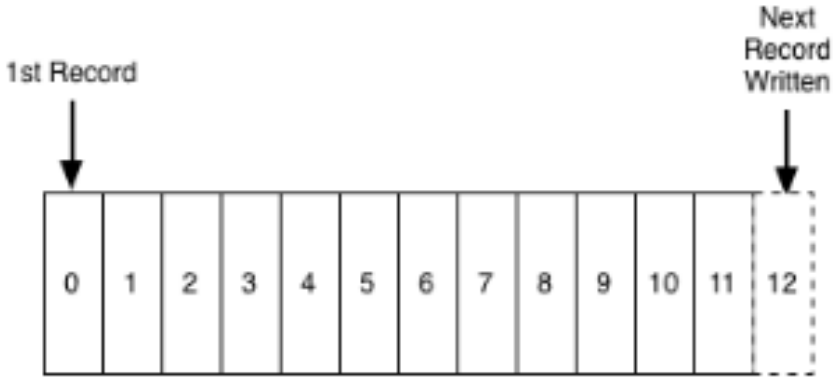
Log processing applies a great of technologies, including real-time computing, data warehouse, and offline computation. This article discusses how to make and guarantee logs to be processed in order, at least once, and exactly once in such scenarios as real-time computing, break-down of upstream/downstream service system and dramatic fluctuation of service traffic.

For easy understanding, I use a day at a bank as an example to explain related concepts. We will talk about the LogHub model of the Log Service as well as the use of LogHub with Spark Streaming and Storm Spout to complete log data processing.

Definitions

What is log data?

Jay Kreps, a former LinkedIn employee, says in The Log: What every software engineer must know about real-time data' s unifying abstraction that log data is "append-only, totally-ordered sequence of records ordered by time."



- Append only: Log works in append mode. Log entries cannot be modified once being generated.
- Totally ordered by time: Log entries are strictly ordered by time. Every log entry is generated at a specific time point. Different log entries may seem to be generated at the same time, for

instance, a GET method and a SET one. For the computer, however, they were performed in sequence.

What type of data can be abstracted into logs?

50 years ago, the term “log” was associated with a thick notebook written by a ship captain or operator. Now, with the rise of computers, logs are produced and consumed everywhere: The world we live in is described in different ways, such as servers, routers, sensors, GPS, purchase orders, and various devices. Using the example of a ship captain’s log, we can see that, besides a recorded timestamp, a log may contain all sorts of information. For example: A text record, an image, weather conditions, or sailing course. Half a century has passed. The Captain’s log is extended to other fields, for example, a purchase order, a payment record, a user access, and a database operation.

In the computer world, we usually use these log types: Metric, Binlog (Database and NoSQL), Event, Auditing, and Access Log.

In this demo, we take each operation that a user performs at the bank as a log entry. The entry consists of user, account name, operation time, operation type, amount and so on.

For example:

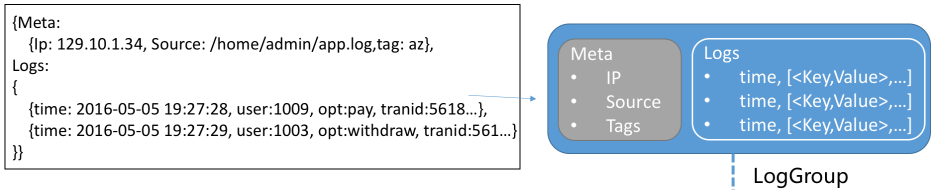
```
2016-06-28 08:00:00 Michael Jacob Deposit US$1,000
2016-06-27 09:00:00 Shane Lee Withdraw US$2,0000
```

LogHub data model

To answer this abstract question, we use Alibaba Cloud Log Service LogHub as the model for demonstration. For details, refer to Basic Concepts of Alibaba Cloud Log Service.

- Log: Composed of time, and a pair of key and value
- LogGroup: A collection of logs that share the same metadata (IP address, source, etc.)

Their relationship is as follows:



- Shard: A partition, as the basic unit for reading and writing logs in a LogGroup, or in other words a 48-hour-cycle FIFO queue. Every Shard offers read/write speeds of 5 MB/s and 10 MB/s respectively. A Shard uses logical segments (BeginKey and EndKey) to sort different types of data.
- LogStore: A log library that stores log data of the same type. LogStore is a carrier constructed from Shards with [0000, FFFF..) segments. One LogStore may contain one or

more Shards.
- Project: A container for storing LogStores.

The relationship among these concepts is as follows:

A day at a bank

Let’ s use the example of a 19th-century bank. Several users (producers) in a city made withdrawals (user operations) from a bank, where several clerks (consumers) were at service. Computer system was not yet available for real-time synchronization in the 19th century. Each clerk had to keep related information in an account book and brought it along with the cash back to the company for reconciliation.

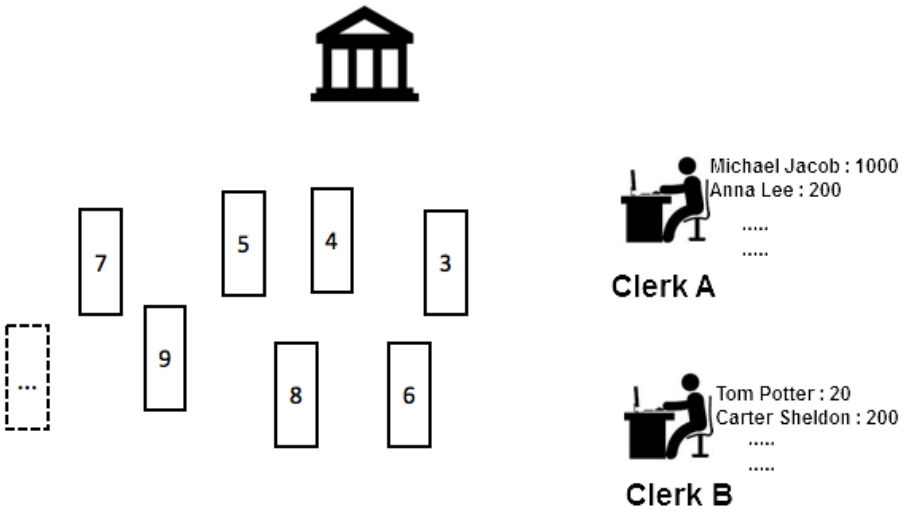
In the world of distributed system, we take a clerk as a single server with fixed memory size and computing capacity. Users are regarded as requests from different data sources, and the bank’ s lobby as the log database (LogStore) that processes users’ access data.

- Log/LogGroup: Operations like deposit and withdrawal initiated by users.
- User: - Log/LogGroup producer.
- Clerk: Bank employees responsible for processing user requests.
- Bank lobby (LogStore): A user’ s operation request goes to the bank’ s lobby before being handled by a clerk.
- Partition (Shard): The way that the bank lobby organizes user requests.

Question 1: Ordering

There were two clerks (Clerk A and Clerk B) at the bank. Michael Jacob entered the bank and asked Clerk A to deposit US$1,000 into his account, and Clerk A made the deposit and recorded it in her account book. Michael Jacob, who was in need of money in the afternoon, went back to the bank and tried to withdraw some money at Clerk B’ s counter. Clerk B checked her account book and found that there was no record of Michael Jacob’ s deposit.

This example shows that deposit and withdrawal are operations in strict sequence. It requires the same clerk (processor) to handle these operations for the same user to maintain state consistency.

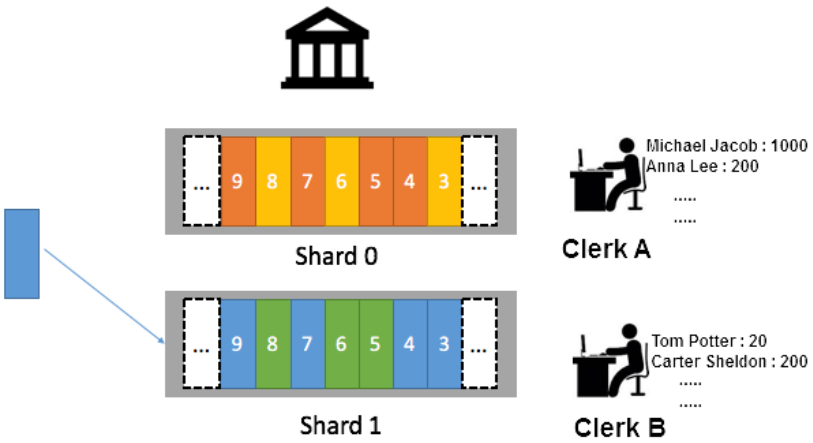


It is easy to achieve ordering: Queue user requests, create a Shard, and assign Clerk A as the only clerk who handles the requests. User requests are handled on a First-In, First-Out (FIFO) basis. Everything works fine except for low efficiency. In the case of 1,000 users, it won't improve the efficiency in any way even if the bank assigns 10 clerks instead of one.

What can we do in this case?

1. Let's assume that there are 10 clerks, and in turn we create 10 Shards.
2. How to ensure that the operations on the same account are in order? Users can be mapped using consistent hashing. For example, we set up 10 queues (Shards), and have every clerk handle one Shard. Different bank accounts or user names are mapped to a specific Shard. In this case, Michael Jacob's hash value, Jacob or J, is always be mapped to a specific Shard (in a segment of the Shard), and the processing end is always Clerk A.

If many users' surnames start with J, you can always switch to another policy. For example, users can be hashed by AccountID, ZipCode or other attributes, for a better balance of operation requests among the Shards.



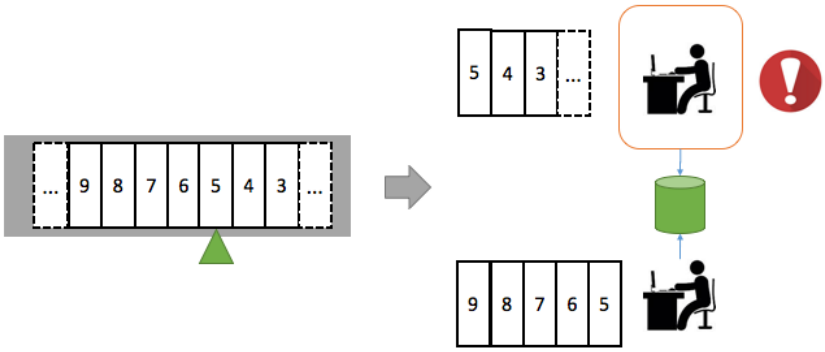
Question 2: At-least once

Michael Jacob went to Counter A to make a deposit. Clerk A stepped out to answer a call halfway while she was handling the request of Michael. When she was back from the call, she thought that Michael's deposit was already made and started to handle the request of the next user. Hence, Michael's deposit request was lost.

Although machines do not make mistakes as men do with longer uptime and higher reliability, However, a business may still be interrupted in case that the system breaks down or encounters a heavy workload. It is absolutely unacceptable to lose users' deposits in such a case.

How to solve this problem?

Clerk A can record an entry in her notebook (rather than an account book) to indicate the current segment in which the request being handled is. Only when Michael Jacob's deposit request is entirely confirmed, can Clerk A proceed to handle the next request.



What is the downside? The same request may be handled twice. In another scenario, when Clerk A completed handling Michael Jacob's request (with the account book updated) and was ready to make a record in her notebook, she was unexpectedly wanted and left away. Upon returning, she found that Michael Jacob's request was not recorded in her notebook and therefore handled Michael Jacob's request for a second time, which led to a repeated entry.

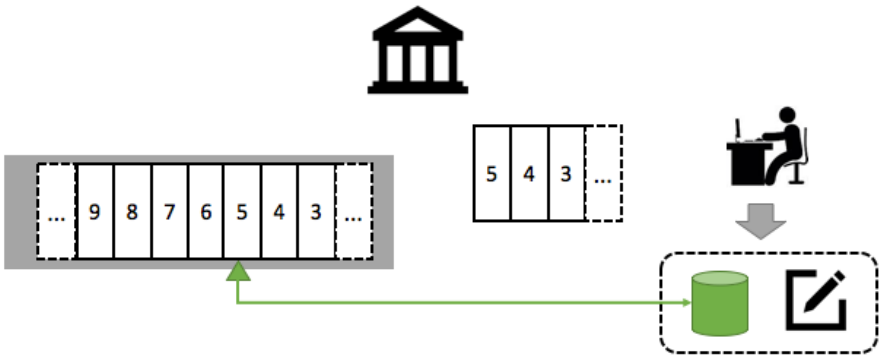
Question 3: Exactly once

Will repeated entries cause problems? Not necessarily.

In the case of idempotence, repeated entries do not impact the results except for wasting a bit of resources. What is idempotence? An operation of duplicate consumption that does not impact the results is idempotence. For example, a user's checking the balance is a read-only operation and does not impact the results even if being repeated. Non read-only operations, such as logging off a user, can be performed twice in a row.

Most operations in the real world are not idempotent, like deposit and withdrawal. Repeat of such entries may cause catastrophic results. What is the solution then? Clerk A must treat "updating the account book" and "recording completion of Shard processing in the notebook" as one operation and write down the progress (CheckPoint).

If Clerk A leaves temporarily or permanently, any other clerk who takes over the request just follows the same rule. If the request is recorded as completed, move to the next request; if not completed, repeat it. It is imperative to maintain atomicity during the process.



CheckPoint can use the element position (or time) in a Shard as a key and put it into an object that can be persisted. It means that the current element has been processed.

Business challenges

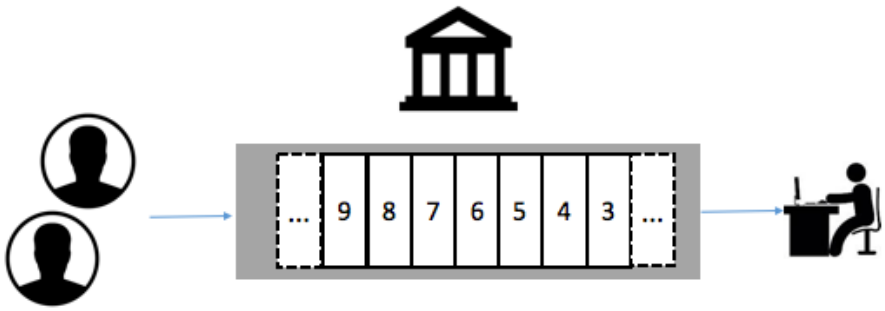
Now we have explained the three questions, which seem easy to resolve. However, in the real world, scale changes and uncertainty may further complicate the three questions above. For example:

1. The number of users will surge on pay day.
2. Clerks are not robots after all, and they need to take leaves and have lunch.
3. To improve the overall service experience, bank managers have to improve clerks' efficiency. What are the criteria of being efficient? What is the processing speed in a Shard?
4. Can a clerk easily pass the clerk' s account book and notebook to another during a handover?

A day in the real world

At 8:00, the bank opened

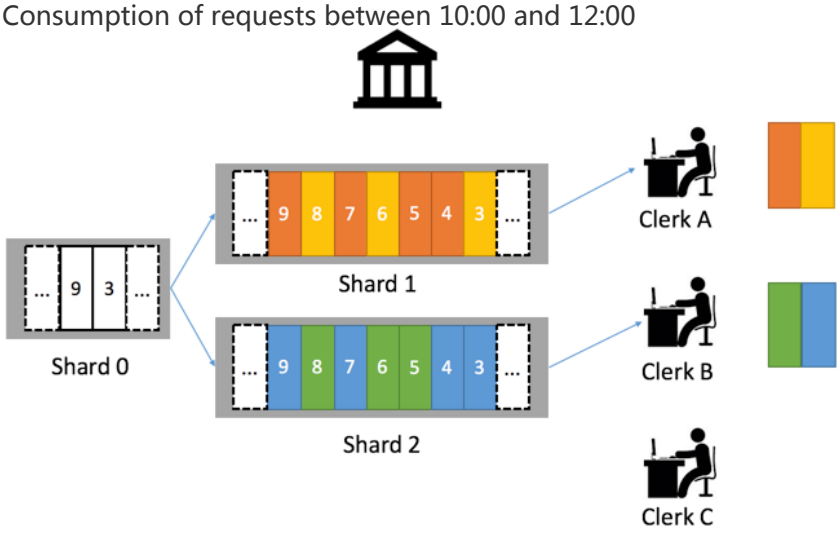
At that point, there was only one Shard, Shard0. All the user requests went to the queue for Shard0, and Clerk A was comfortable to handle the workload alone.



At 10:00, the peak hour started

The bank manager decided to split Shard0 into two new Shards (Shard1 and Shard2) after 10:00 am, and executed a rule that assigns users to a queue for Shard1 if their surnames start with a letter in the range from A to W, and users to a queue for Shard2 if their surnames start with X, Y or Z. Why are the two Shard segments not divided equally? It was because the surnames are not evenly distributed in terms of the first letter. Such a mapping ensures workload balance between the two Shards.

Consumption of requests between 10:00 and 12:00

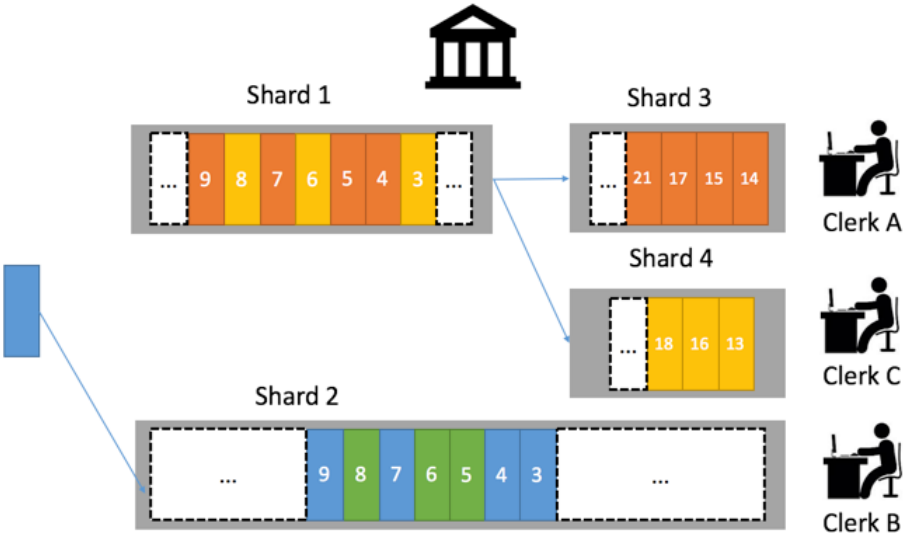


Seeing that Clerk A was difficult to handle two Shards at the time, the bank manager sent Clerks B and C. Since there were only two Shards, Clerk B took over one Shard from Clerk A and Clerk C stood by.

At 12:00, users were getting more

Clerk A handled the requests in Shard1 under high pressure. The bank manager split Shard1 into two new Shards (Shard3 and Shard4). Clerk A was responsible for Shard3 and Clerk C responsible for Shard4. All the requests assigned to the queue for Shard1 after 12:00 were diverted to Shard3 and Shard4 respectively.

Consumption of requests after 12:00:



At 16:00, the user traffic began to wind down

The bank manager relieved Clerks A and B and tasked Clerk C to handle requests in Shard2, Shard3 and Shard4. Then, the bank manager merged Shard2 and Shard3 into Shard5, and at last Shard5 and Shard4 into one Shard. The bank was closed when all the requests in the last Shard were handled.

Log processing in the real world

The process described above can be abstracted into typical scenarios of log processing. To address the business needs of a bank, we need to provide a log foundation framework capable of automatic scaling and flexible adaptation, that can:

1. Automatically scale Shards (for details, refer to LogHub Auto Scaling (Merge/Split).
2. Support automatic adaptation of consumers when they log on/off and prevent data loss during the handling (for details, refer to LogHub Consumer Library-Auto Load-balancing for Collaborative Consumer Group).
3. Support ordering during the handling (for details, refer to Ordering Write and Consumption in LogHub).
4. Prevent repeated entries during the handling (which requires consumers' cooperation).
5. Observe the consuming progress for reasonable allocation of computing resources (for details, refer to Using Console to View Collaborative Consumer Group Progress).
6. Support incoming logs from more channels (in the banking sector, more channels like online banking, mobile banking and cheques can bring in more user requests) (for details, refer to Several LogHub Data Access Modes).

LogHub and LogHub Consumer Library can help you resolve the typical problems in real-time log processing. All you need to do is focusing on the business logic, without worrying about traffic, resizing, failover, and other nuances.

In addition, APIs for Storm and Spark Streaming have been made available with Consumer Library. Why not try them out? You will also find many useful information in Log Service Homepageand Log Processing Community.

What are the characteristics of AppLogs?

- The most comprehensive data: AppLogs are provided by programmers, covering key locations, variables, and exceptions. Technically, over 90% of online bugs are located by AppLogs.

*Arbitrary formats: One piece of code is often developed more than one programmer. Every programmer has their own preferred formats, which are difficult to uniform. Style inconsistency is also seen in logs introduced from third-party databases.

*Share things in common: Despite of the arbitrary formats, different logs share things in common. For example, the following fields are required for Log4J logs:

- Time
- Level
- File or class
- Line number
- ThreadID

What are the challenges in processing AppLogs:

Large data volume

Generally, AppLogs are larger than access logs by an order of magnitude. For example, if a website has one million independent accesses every day. Each access has 20 logic modules, and 10 main logic points in each module need to be logged.

Then, the total number of logs is:

```
1,000,000 * 20 * 10 = 2 * 10^8
```

The length of each log is 200 bytes, meaning a storage size of:

```
2 * 10^8 * 200 = 4 * 10^10 = 40 GB
```

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

Distributed in numerous servers

Most applications are running in a stateless mode under different frameworks, including:

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

The numbers of corresponding instances vary from a few to thousands, which requires a cross-server log collection solution.

Complex runtime environments

Programs are running in different environments, so logs are stored in various places, for example:

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

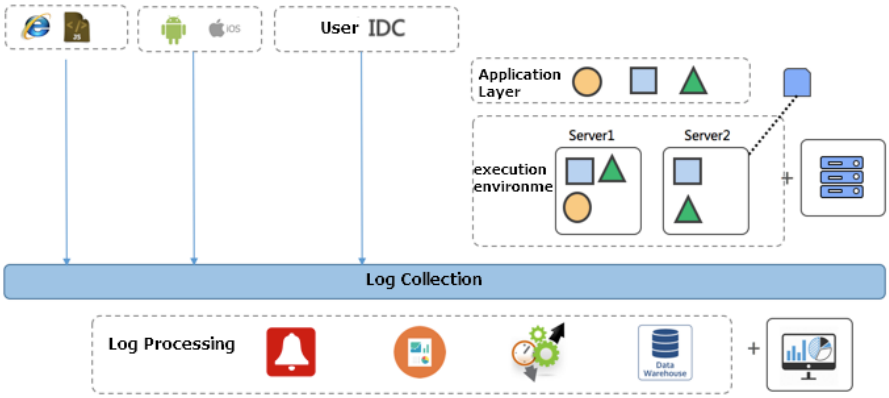
To have a full picture of this data, we must bring it together and save it in a single place.

How to respond to the demand for AppLogs

Unified data storage

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

You can create a project in Log Service to store AppLogs. Log Service supports over 30 collection methods, such as tracking in physical severs, JS on the mobile web side, and outputting logs on servers, all of which can be found in the Real-time Collection List.



Apart from writing logs using methods like SDK, Log Service offers a convenient, stable, and high performance Agent called Logtail for server logs. Logtail comes with the Linus version and the Windows version. Once you have defined the machine group and made the log collection configuration, real-time collection of sever logs occurs in real time. See a 5-minute video for details.

Once a log collection configuration is created, you can operate on logs in the project.

You may wonder how Logtail is different from the various other Agents, such as Logstash, Flume,

FluentD, and Beats. Here is the answer.

- Easy to use: Featuring API access, remote management and monitoring capabilities, Logtail is designed with Alibaba Group’ s rich experience in million-level server log collection and management, allowing you to configure a collection point to hundreds of thousands of devices in seconds.
- Adaptive to different environments: Logtail supports public networks, VPCs and, user-defined IDCs. The https and resumable data transfer functions make it possible to integrate with public network data.
- Great performance with a little consumption of resources: With years of refinement, Logtail is superior to its open-source competitors in terms of performance and resource consumption. See Comparison Tests for details.

Quick searching and locating

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

For instance, how can we locate an order error and a long latency issue out of terabytes of data every week? The process also involves filtering and investigating based on various criteria.

For example, for AppLogs with latency details, we investigate request data with latency of more than one second and methods starting with Post:

```
Latency > 1000000 and Method=Post*
```

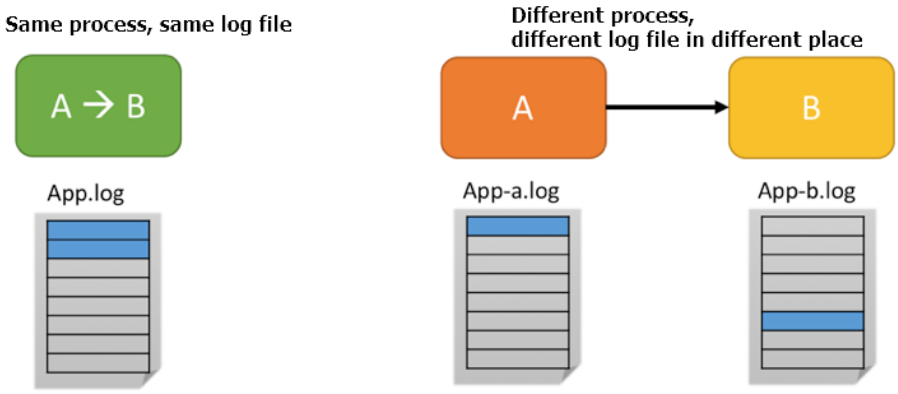
Search logs that contain the keyword “error” and not the keyword “merge”.

Results of one day, one week, or a longer timespan can be returned in less than one second.

Association analysis

There are two types of association: intra-process association and inter-process association. Here are the differences between the two:

- Intra-process association: This is a simple type because the previous and new logs of a function are stored in one file. In multi-thread cases, we can filter logs by ThreadID.
- Cross-process association: Normally, it is hard to find clear clues when dealing with logs from different process. The association is generally performed by passing TracerID into RPC.



3.1 Context-sensitive association

Locate an error log with the keyword query in the Log Service console.

Click Context View and then you can see the preceding and following results.

*You can click **OLD** and **NEW** for more results.* Or you can filter the results by ThreadID to improve the filtering accuracy.

For more information, see Context Query.

3.2 Cross-process association

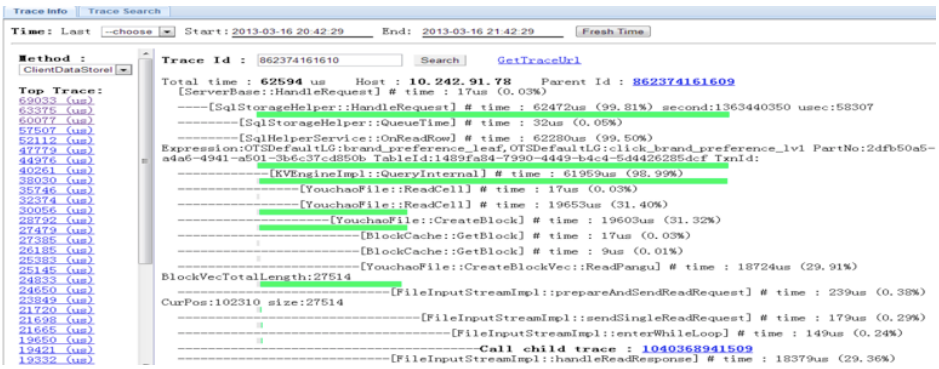
The concept of the cross-process association, or Tracing, can be dated back to the famous paper Dapper, a Large-Scale Distributed Systems Tracing Infrastructure by Google in 2010. Inspired by the paper, developers from the open source sector created many affordable versions of Tracer. Here are some well-known Tracers:

- Dapper (Google): basis of different Tracers
- StackDriver Trace (Google): ZipKin-compatible currently
- Zipkin: an open source Tracing system by Twitter
- Appdash: Golang version
- Hawkeye: by Alibaba’ s Middleware Technology Department
- X-ray: introduced at AWS re:Invent 2016

Applying Tracer from scratch is easier than in an existing system, due to the costs and challenges in adapting it to the system.

Based on Log Service, we can now implement a basic tracing feature, which is to access logs by outputting associative fields such as Request_id and OrderID in logs from different modules and searching them in various log stores.

For example, we can query logs of frontend servers, backend servers, payment systems, and order systems using SDKs. After we obtain the results, we can create a page on the frontend to associate the cross-process calls. Here is the tracing system built quickly based on Log Service.



Statistical analysis

After we locate the specific log, we can perform the analyses on the log such as calculating the types of online error logs.

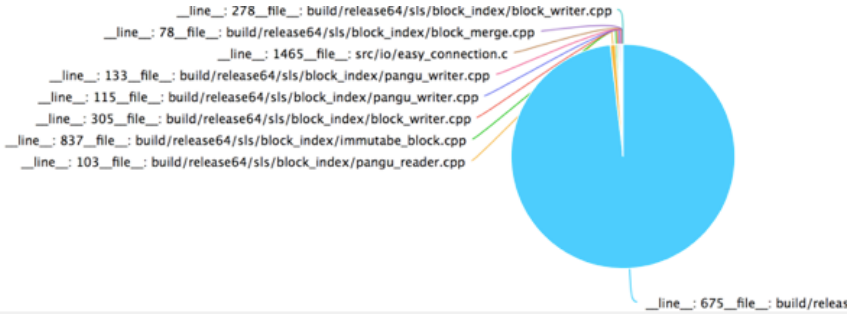
1.We can query logs by __level__, and 2,720 errors are found within one day.

```
__level__:error
```

1. Then, we can perform the analysis and aggregation of data by file and line fields (to determine the unique log type).

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

Then, we can know the type and location of the errors.



__line__	c	__file__
675	2670	build/release64/sls/shennong_worker/PackageDispatcher.cpp
103	21	build/release64/sls/block_index/pangu_reader.cpp
837	7	build/release64/sls/block_index/immutabe_block.cpp
305	6	build/release64/sls/block_index/block_writer.cpp
115	6	build/release64/sls/block_index/pangu_writer.cpp
133	5	build/release64/sls/block_index/pangu_writer.cpp
1465	3	src/io/easy_connection.c
78	1	build/release64/sls/block_index/block_merge.cpp
278	1	build/release64/sls/block_index/block_writer.cpp

Besides, we can locate IPs and perform analyses by fields such as error codes and high delay. For

more information, see Best Practices of Log Analysis

Others

1. Log backup for auditing

You can back up the logs to OSS, IA (with a lower storage cost), or MaxCompute. See Log Shipper for more information.

2. Keyword alarm

Alarms can be performed in the following ways:

1. Saving the log query as a scheduled task in Log Service to alarm the results. Click here for more information.
2. Implementing the CloudMonitor Log Alarm feature. Click here for more information.

3. Log query permission management

You can grant different permissions to your team members by setting sub-accounts or groups. Click here for more information.

Price and cost: AppLog mainly adopts LogHub and LogSearch features of Log Service. Compared with an open source solution, AppLog is an easy-to-use solution with only 25% cost of an open source solution, thus improving the development efficiency.

Introduction

The taxi company records the details of each trip, including the time when a passenger gets in and out, latitude and longitude, distance of the trip, payment option, payment amount, tax amount and other information. Detailed data greatly facilitates the operation of taxi companies. For example, with the data, the companies can shorten the running intervals in busy 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, while drivers can have higher incomes, thus making the entire society more efficient.

Taxi companies storage the trip log on to the log service of Alibaba Cloud, and pick out useful information with the help of reliable storage and rapid statistical calculations. This article describes how taxi companies dig out useful information from the data stored in the Alibaba Cloud Log Service.

Data example:

```
RatecodeID: 1VendorID: 2__source__: 11.164.232.105 __topic__: dropoff_latitude: 40.743995666503906
dropoff_longitude: -73.983505249023437extra: 0 fare_amount: 9 improvement_surcharge: 0.3 mta_tax: 0.5
passenger_count: 2 payment_type: 1 pickup_latitude: 40.761466979980469 pickup_longitude: -
73.96246337890625 store_and_fwd_flag: N tip_amount: 1.96 tolls_amount: 0 total_amount: 11.76
```

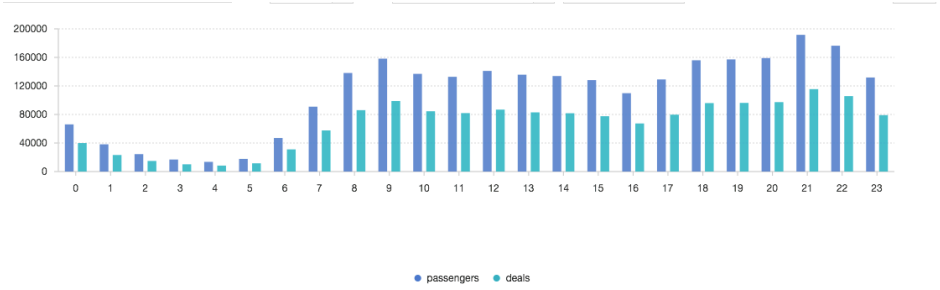
```
tpep_dropoff_datetime: 2016-02-14 11:03:13 tpep_dropoff_time: 1455418993 tpep_pickup_datetime: 2016-02-14 10:53:57 tpep_pickup_time: 1455418437 trip_distance: 2.02
```

Query link

Common statistics

Number of passengers in different time periods, to learn about the busy hours.

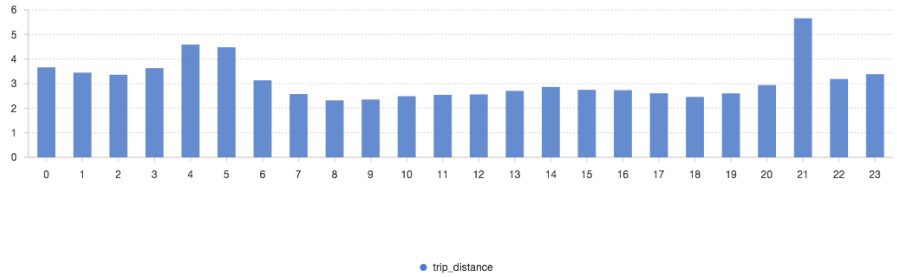
```
*| select count(1) as deals, sum(passenger_count) as passengers,
(tpep_pickup_time %(24*3600)/3600+8)%24 as time
group by (tpep_pickup_time %(24*3600)/3600+8)%24 order by time limit 24
```



Based on the results, the time periods when people go to work in the morning, and get off work in the evening, are the busiest hours within one day, thus taxi companies can dispatch more vehicles accordingly.

Average trip distance in different time periods

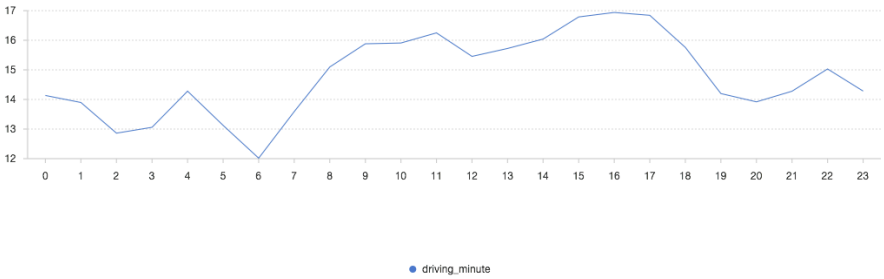
```
*| select avg(trip_distance) as trip_distance,
(tpep_pickup_time %(24*3600)/3600+8)%24 as time
group by (tpep_pickup_time %(24*3600)/3600+8)%24 order by time limit 24
```



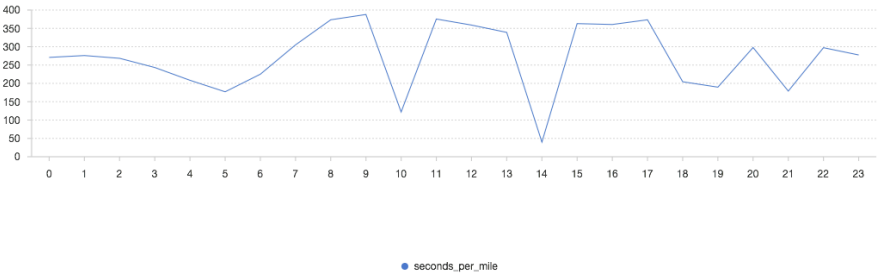
Passengers tend to take a longer trip during certain time periods of a day, so taxi companies also need to dispatch more vehicles.

Average trip time (in minutes), time required for per unit of mileage (in seconds), to see the time periods when traffic jams tend to happen more easily.

```
*| select  avg(tpep_dropoff_time-tpep_pickup_time)/60  as driving_minutes,
(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
```




```
*| select  sum(tpep_dropoff_time-tpep_pickup_time)/sum(trip_distance)  as driving_minutes,
(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
```

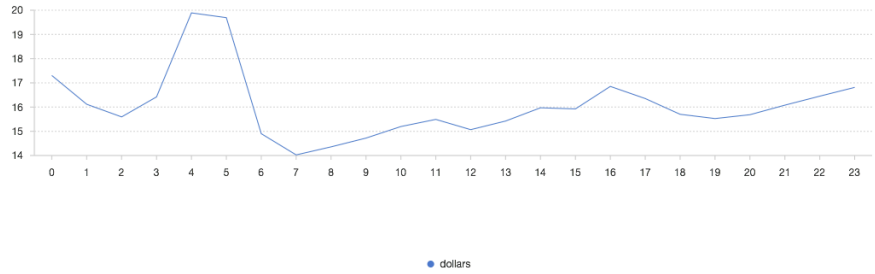


More vehicles need to be dispatched during these time periods.

Average taxi fares in different time periods, to highlight the hours with more income.

```
*| select  avg(total_amount)  as dollars,
(tpep_pickup_time %(24*3600)/3600+8)%24 as time
```

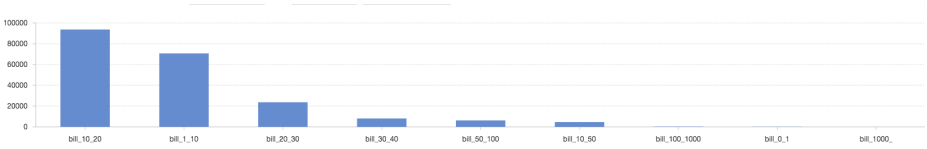
```
group by (tpep_pickup_time %(24*3600)/3600+8)%24 order by time limit 24
```



The per customer transaction is higher around 4 o’ clock in the morning, so financially stressed drivers can consider providing service in this time period.

Range of 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_10_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_10_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
```



We can see that the payment amount of most transaction falls between 1 to 20 USD.

Introduction

Bills are the core data of e-commerce companies and the outcome of a series of marketing and

promotional activities. They contain a wealth of important information, based on which you can define the user profiles, providing guidelines for future marketing plans. Billing data can also serve as a popularity indicator, providing suggestions for subsequent stocking options.

Billing information are stored as logs in Alibaba Cloud Log Service. Log Service ensures high-speed queries and SQL statistics, with a computing capability of hundreds of millions of log entries per second. This article explains how to pick out useful information.

A complete bill containing goods information (the name and price), deal information (final price, payment method, and discount information), and the buyer’ s information (membership information) is shown as follows:

```
__source__: 11.164.232.105 __topic__: bonus_discount: category: men's clothing commodity: ********** commodity_id: 443 discount: member_discount: member_level: nomember_point: memberid: mobile: pay_transaction_id: 060f0e0d080e0b05060307010c0f0209010e0e010c0a0605000606050b0c0400 pay_with: alipay real_price: 52.0 suggest_price: 52.0
```

Statistical analysis

Determine the hot categories

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

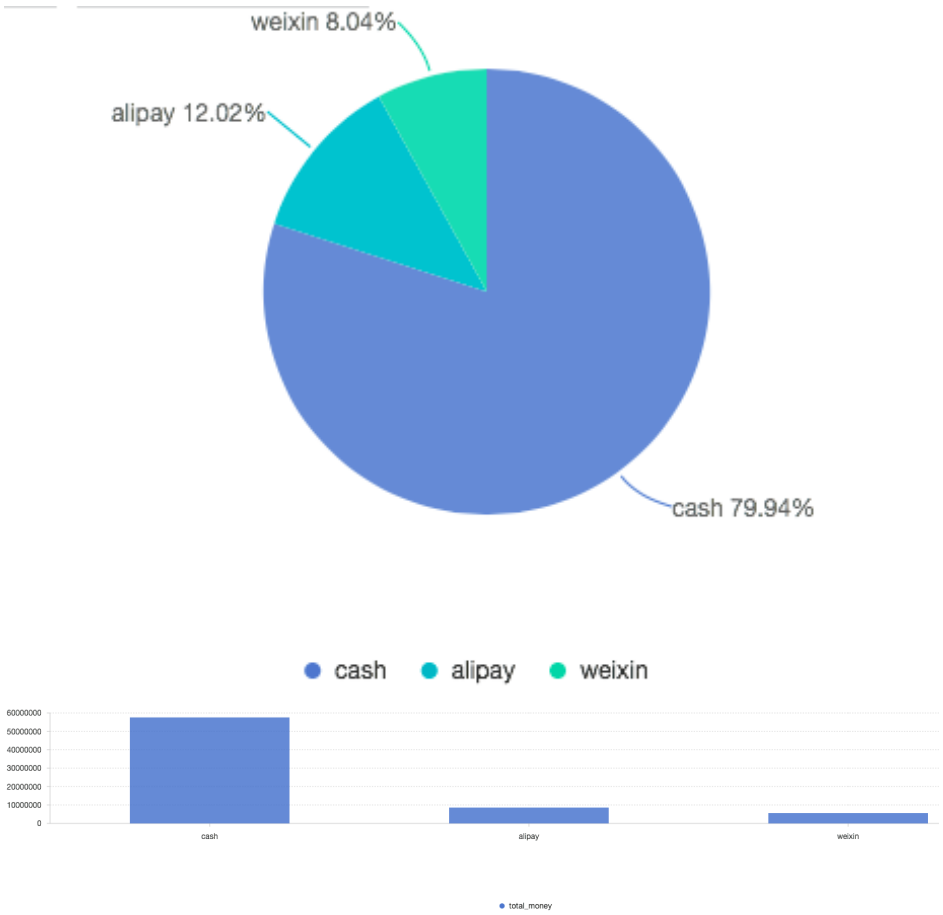
Determine the hot women’ s clothing

```
category: women's clothing/ladies’ collection | select count(1) as deals , commodity group by commodity order by deals desc limit 20
```

Share and turnover: Alipay versus WeChat

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

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



Many webmasters use Nginx as the server to build websites. When they want to obtain the website traffic data, they can perform a statistical analysis on Nginx access logs to obtain such data as the page views and the access time periods of the website. In the traditional CNZZ method, a js is inserted in the frontend page and will be triggered once a user accesses the website. However, this method can only log access requests. StreamCompute or offline statistics can also be used to analyze Nginx access logs, which however requires a particular environment, and is subject to imbalance between timeliness and analytical flexibility.

In addition to log query, Log Service also supports SQL real-time log analysis, which decreases the analytical complexity of Nginx access logs and streamlines the statistical analysis of website access data. This document provides the detailed procedure for analyzing Nginx access logs using SQL real-time log analysis.

Use cases

A webmaster builds a personal website using Nginx as the server. The PV, UV, popular pages, hot methods, bad requests, client types, and referer tabulation are obtained by analyzing Nginx access logs to assess the website access status.

Log format

```
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" $http_host '
'$status $request_length $body_bytes_sent "$http_referer" '
'"$http_user_agent" $request_time';

access_log access.log main;
```

The fields are described as follows.

Field	Description
remote_addr	Client address
remote_user	Client user name
time_local	Server time
request	Request content, including method name, address, and HTTP protocol
http_host	HTTP address used by the user request
status	HTTP status code returned
request_length	Request size
Body_bytes_sent	Data size returned
http_referer	Referer page
Http_user_agent	Client name
request_time	Overall request latency

Configuration steps

1. Collect Nginx server logs

Collect Nginx access logs to Log Service. For detailed steps, see 5-minute Quick Start and Nginx Logs.

2. Create an index query analysis

1. In the Log Service console, click the Project name.
2. Select the target LogStore and click **Search** under the log index column.
3. Click **Enable** in the upper-right corner.

Enter the index configuration information as needed.

3. Analyze the access logs

After the index feature is enabled, enter a statistical statement in the keyword index box to perform a statistical analysis on matched results in the way predefined in the statement. The statements are shown as follows:

PV statistics

With PV statistics, you can view not only the total PVs for a period of time, but also the PVs for a specific period of time (for example, the PVs every 5 minutes).

Statistical statement:

```
*|select from_unixtime( __time__- __time__% 300) as t,
count(1) as pv
group by t
order by t limit 60
```

UV statistics

UVs every 5 minutes within an hour.

Statistical statement:

```
*|select from_unixtime( __time__- __time__% 300) as t,
approx_distinct(remote_addr) as uv
group by t
order by t limit 60
```

Popular page statistics

Ten most frequently accessed pages within an hour.

Statistical statement:

```
*|select count(1) as pv, split_part(url,'?',1) as path  group by path order by pv desc limit 20
```

Request method statistics

Ratio of each request method used within an hour.

Statistical statement:

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

HTTP status code statistics

Ratio of each HTTP status code returned within an hour. Statistical statement:

Statistical statement:

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

Statistics for remote client types

Ratio of each browser used within an hour.

Statistical statement:

```
*| select count(1) as pv, case when http_user_agent like '%Android%' then 'Android' when http_user_agent like '%iPhone%' then 'iOS' else 'unKnown' end as http_user_agent  group by http_user_agent   order by pv desc limit 10
```

Referer statistics

Ratio of various domain names from which the referer comes within an hour.

Statistical statement:

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

Statistics of every provinces

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

4. Access diagnosis and optimization

In addition to some access indicators, webmasters often need to diagnose some access requests to check the latency of request processing, how many long latencies, and on what pages long latencies occur.

Calculate the average latency and the maximum latency

With the average latency and the maximum latency every 5 minutes, you can get a picture of the latency issue.

Statistical statement:

```
 *|select from_unixtime(__time__ -__time__% 300) as time,
avg(request_time) as avg_latency ,
max(request_time) as max_latency
group by __time__ -__time__% 300
limit 60
```

Identify the request page with the maximum latency

After knowing the maximum latency, you need to identify the corresponding request page to optimize page response.

Statistical statement:

```
*|select from_unixtime(__time__ - __time__% 60) ,
max_by(url,request_time)
group by __time__ - __time__%60
```

Measure the distribution of request latencies

Measure the distribution of all request latencies occurring on the website, place the request latencies in ten buckets, and count the requests in each latency interval.

Statistical statement:

```
 *|select  numeric_histogram(10,request_time)
```

Calculate the ten longest latencies

In addition to the maximum latency, the second to the tenth longest latencies and their values also need to be calculated.

Statistical statement:

```
*|select max(request_time,10)
```

Tune the page with the maximum latency

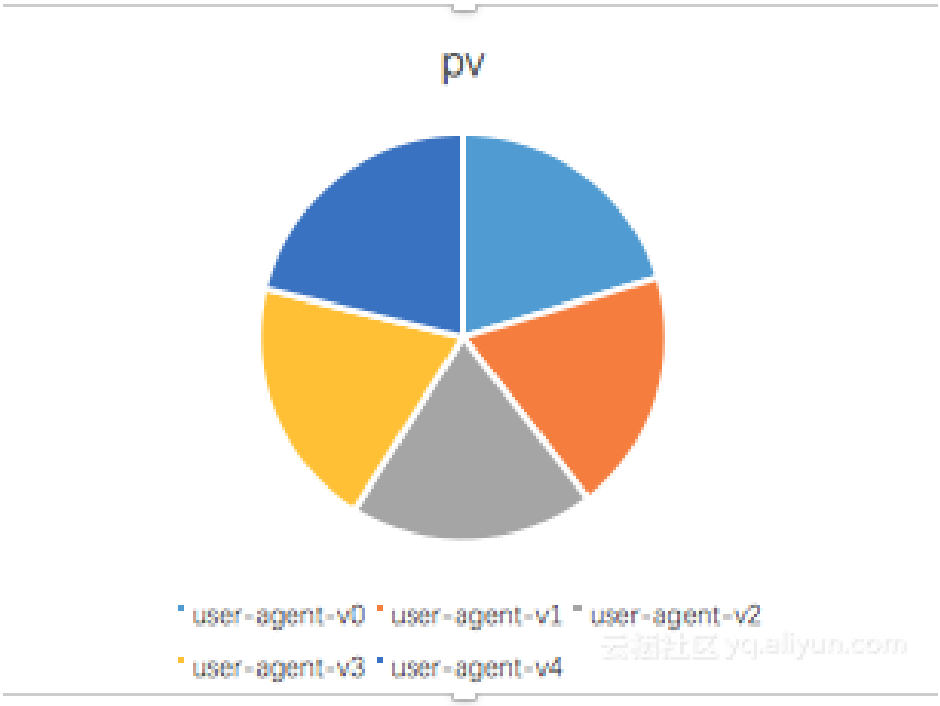
According to the statistics, the maximum latency occurs on the /0 page. To tune the /0

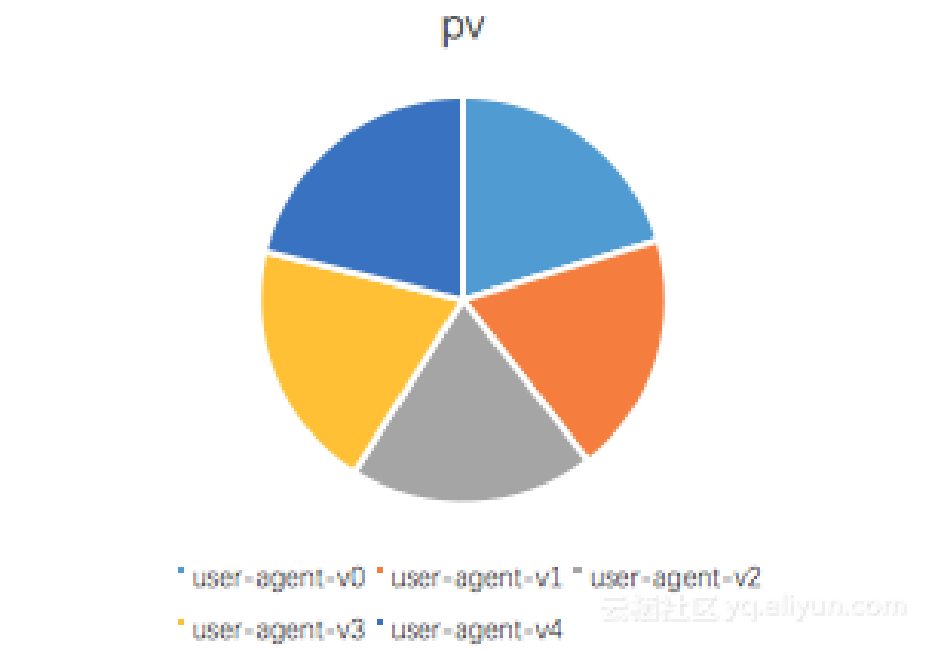
page, you must calculate the PV, UV, counts of various methods, statuses, browsers, and the average latency, and the maximum latency.

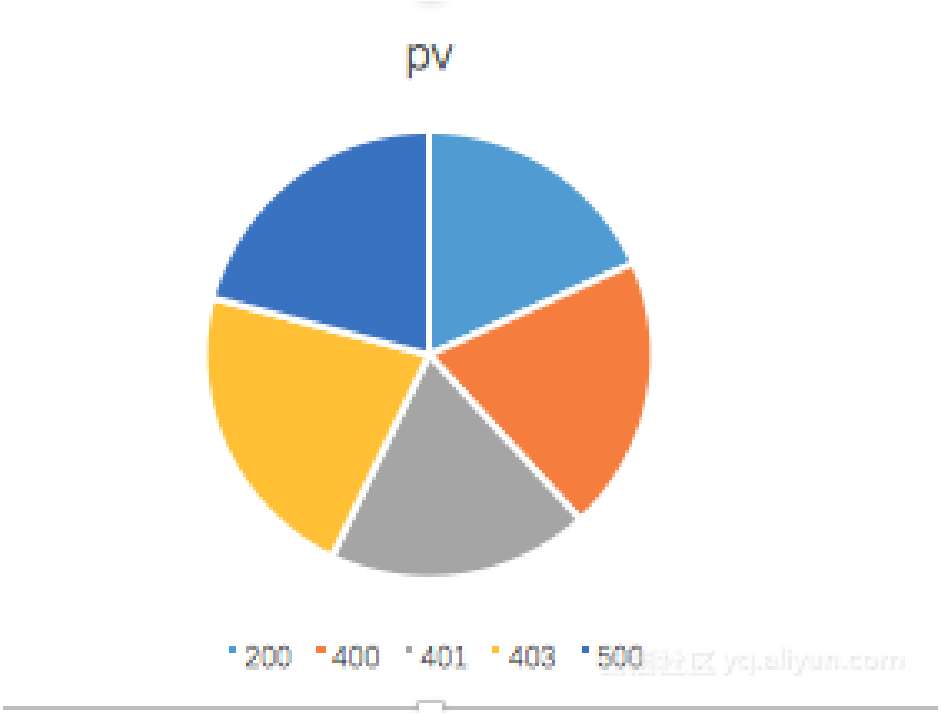
Statistical statement:

```
 url:"/0"|select count(1) as pv, approx_distinct(remote_addr) as uv, histogram(method) as method_pv,histogram(status) as status_pv, histogram(user_agent) as user_agent_pv, avg(request_time) as avg_latency,  max(request_time) as max_latency
```

Statistical results:







With these results, you can make targeted and detailed assessments on the access status of this website.

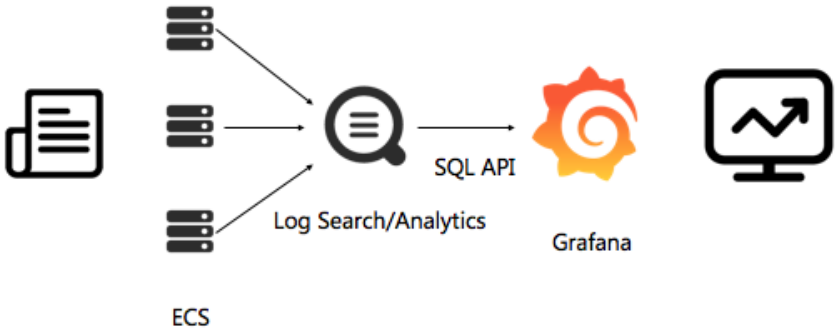
Visualization - Interconnection with Grafana

Alibaba Cloud Log Service is an allinone service for logtype data. Users can leave trivial jobs like data collection, storage computing interconnection, and data index and query to Log Service to focus on the analysis. In September 2017, Log Service upgraded the LogSearch/Analytics, allowing real-time log analysis using query + SQL92 syntax.

Besides the built-in Dashboard, interconnections with DataV, Grafana, Tableua, and Quick are also available to visualize the results analysis. This article demonstrates how to analyze and visualize Nginx logs using Log Service by giving an example of Grafana.

Process structure

Process from log collection to analysis is structured as follows.



Configuration process

1. Log data collection. For detailed procedures, see Getting Started in 5 Minutes.
2. Index settings and console query configuration. For detailed procedures, see Index Settings and Visualization or Website Log Analysis Case.
3. Install the Grafana plug-in to convert real-time query SQL into views.

This document demonstrates Step 3.

After completing Step 1 and 2, you can view the raw log on the query page.

Configuration steps

Install Grafana

For detailed installation steps, see Grafana official document.

To Ubuntu, for example, the installation command is:

```
wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_4.5.2_amd64.deb
sudo apt-get install -y adduser libfontconfig
sudo dpkg -i grafana_4.5.2_amd64.deb
```

To use the pie chart, you must install the pie chart plug-in. For detailed procedures, see Grafana official document.

Run the following command:

```
grafana-cli plugins install grafana-piechart-panel
```

Install the Log Service plug-in

Confirm the directory location of Grafana plug-in. The location of Ubuntu plug-in is /var/lib/grafana/plugins/. Restart the grafana-server after installing the plug-in.

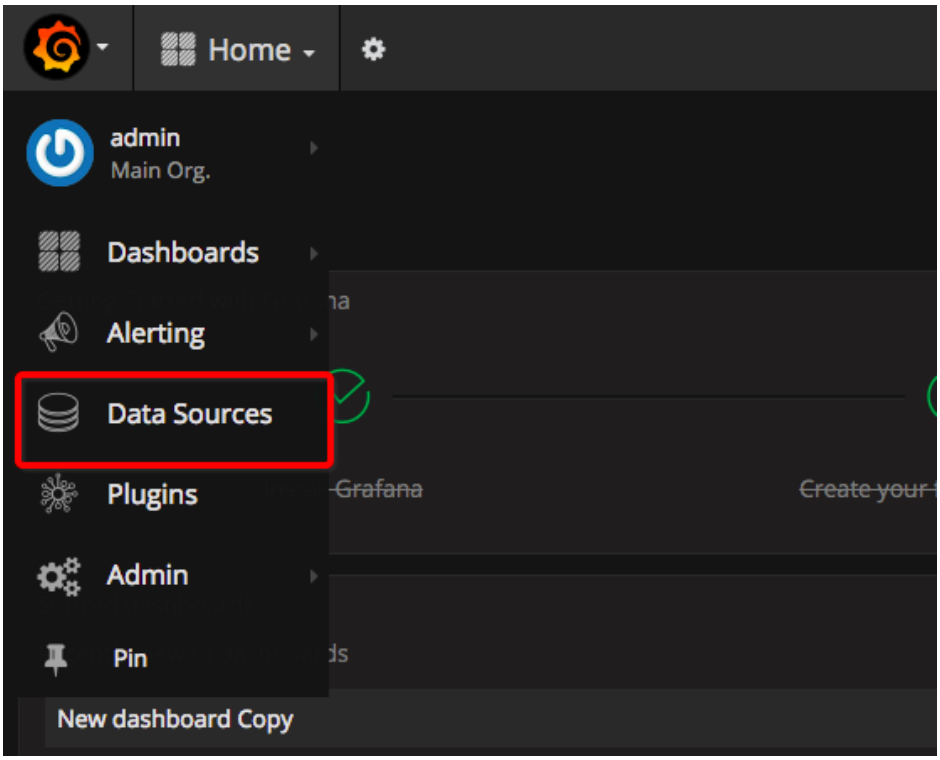
Taking Ubuntu system as an example, run the following command to install the plug-in and restart the grafana-server.

```
cd /var/lib/grafana/plugins/
git clone https://github.com/aliyun/aliyun-log-grafana-datasource-plugin
service grafana-server restart
```

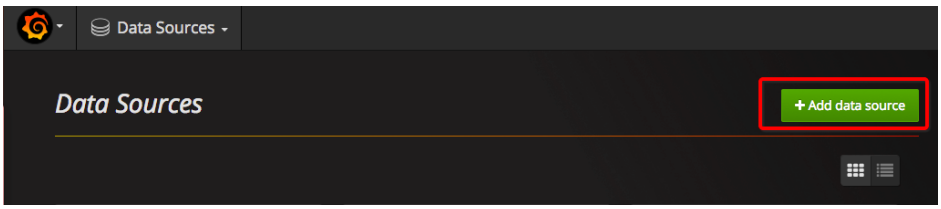
Configure the log data source

For the deployment in a local machine, the default installation location is Port 3000.

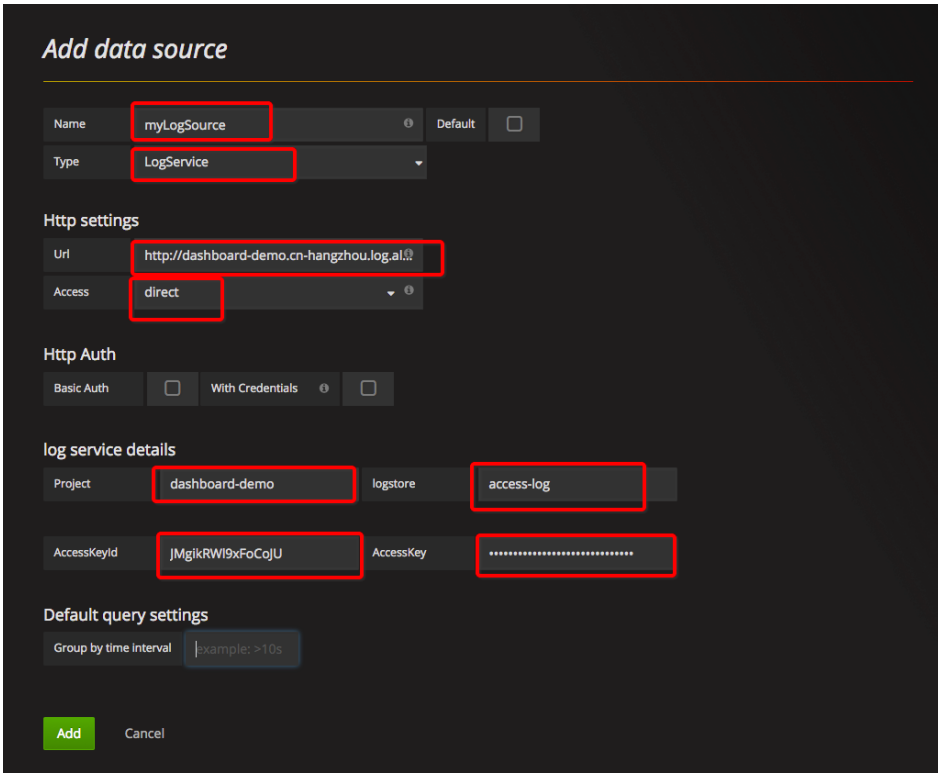
Open the Port 3000 in a browser.



Click on Grafana' s logo in the upper left corner and select Data Sources in the pop-up window.



Click Add data source, and use Grafana and Alibaba Cloud Log Service for log visual analysis.



Each part is configured as follows:

Configuration item	Configuration content
datasource	The name can be customized and the type is **LogService**.
Http Setting	URL input example: http://dashboard-demo.cn-hangzhou.log.aliyuncs.com. The dashboard-demo (project name) and cn-hangzhou.log.aliyuncs.com (endpoint of the project region) must be replaced with your project and region addresses when configuring your data source. You can select Direct or Proxy for Access.
Http Auth	Use the default configuration.
log service details	For detailed configuration of Log Service, enter Project, Logstore, and AccessKey that has the read permission respectively. AccessKey can belong to the primary account or the subaccount.

After the configuration is complete, click Add to add DataSource. Next, add Dashboard.

Add Dashboard

Click to open the menu in the upper left corner, select **Dashboards** and click **New**. Add a new Dashboard to the menu in the upper left corner.

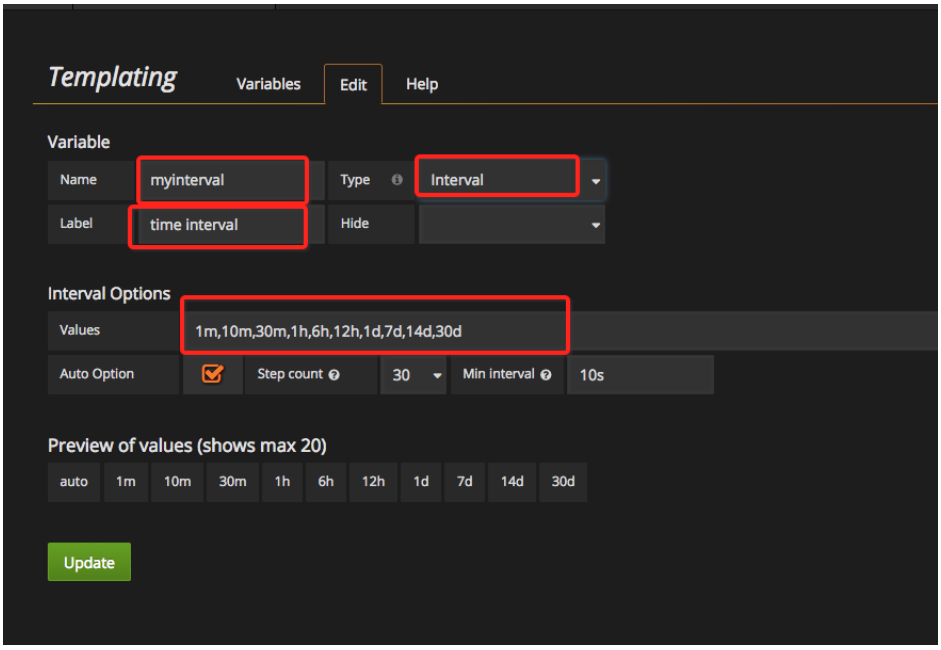
Configure the template variables

You can configure the template variables in Grafana to show different views in the same view by choosing different variable values. This document describes the configuration of each time interval and the access of different domain names.

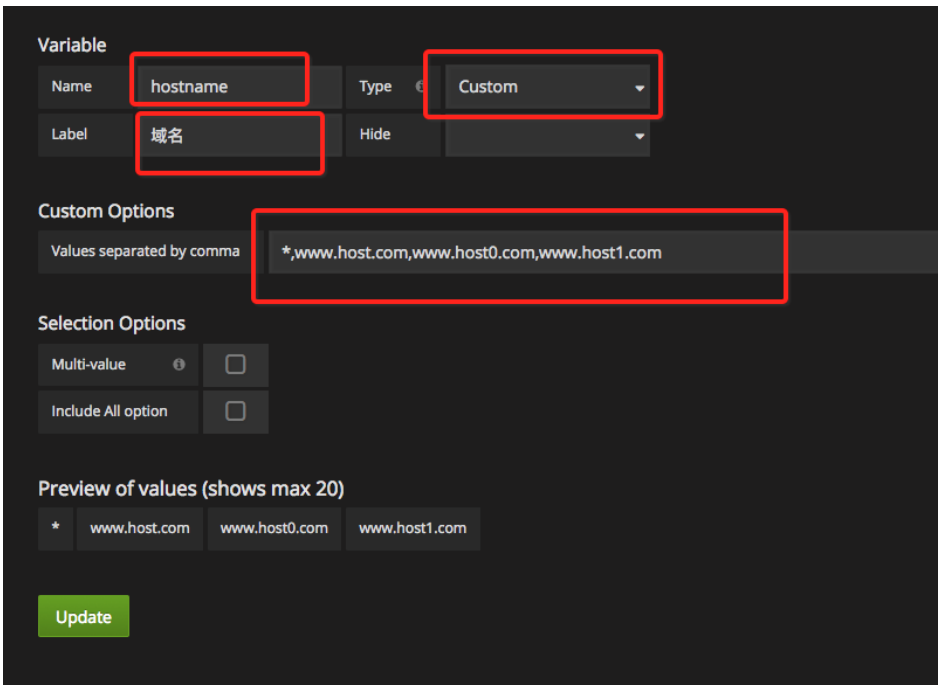
Click the Settings icon at the top of the page, and then click **Templating**.

On the current page, the configured template variables are displayed. Click **New** to create a new template.

First, configure a time interval. The name of the variable is the one you used in the configuration, which is called myinterval here. You must write $myinterval in the query condition. Then, the value of the time interval is automatically replaced with the template value selected on this page.



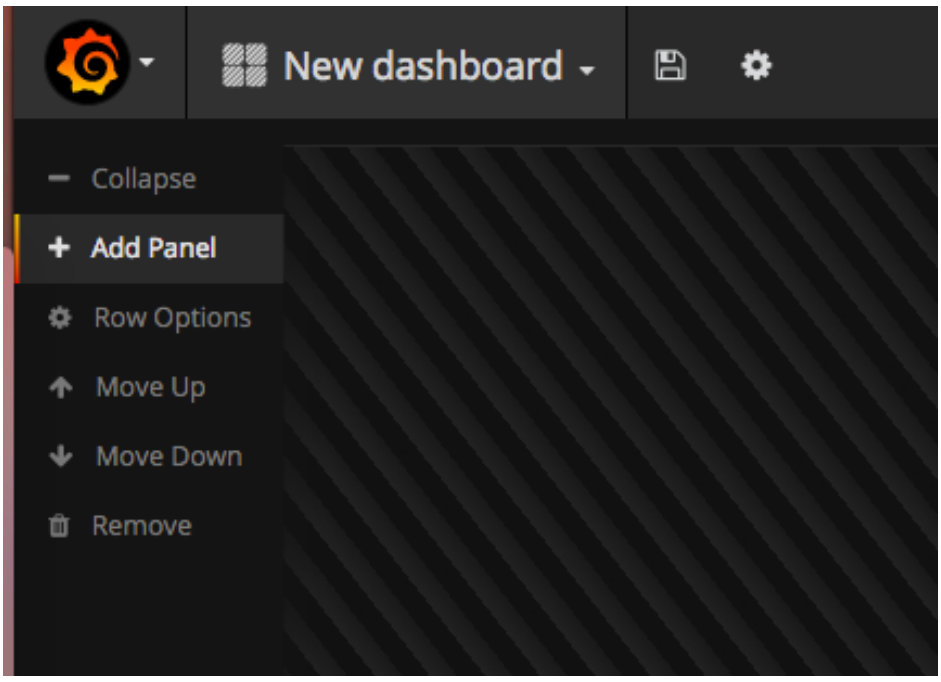
Configure a domain name template. Generally, multiple domain names can be mounted on one vps. You must view the accesses of different domain names. For the template value, enter*,www.host.com, www.host0.com, www.host1.com to view all the domain names, or view the accesses of www.host.com, www.host0.com or www.host1.com respectively.



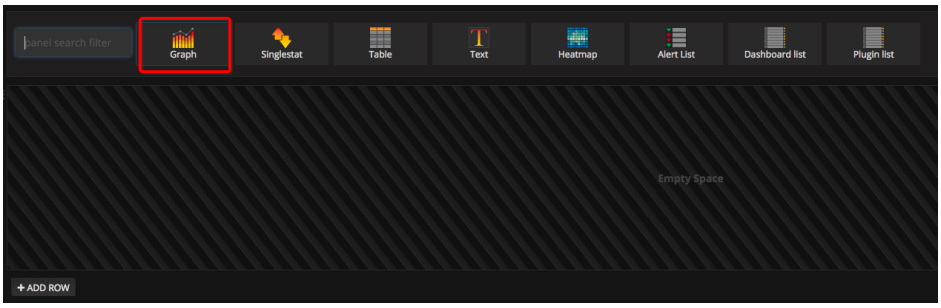
After the configuration is complete, the template variables configured appears on the top of the Dashboard page. You can select any value from the drop-down box. For example, the following values can be selected for time interval:

Configure PV and UV

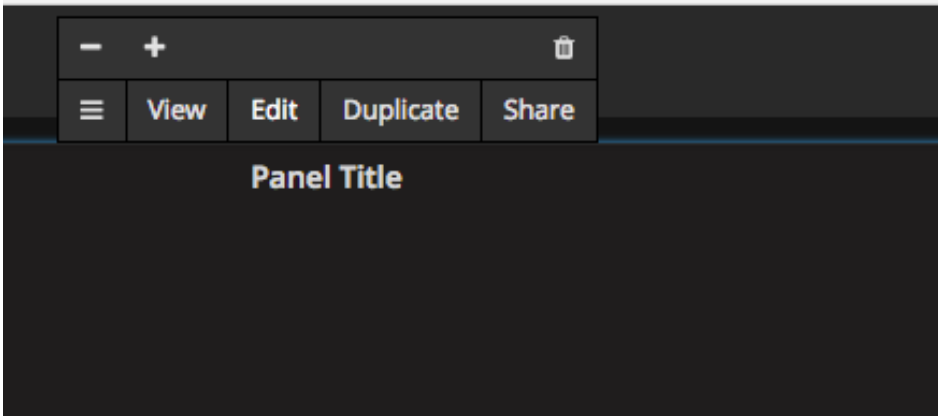
Click ADD ROW on the left to create a new Row. If a Row already exists, you can select Add Panel in the left pop-up menu:



Grafana supports multiple types of views. For PV and UV data, create a Graph view.



Click Panel Title and click Edit in the pop-up window.

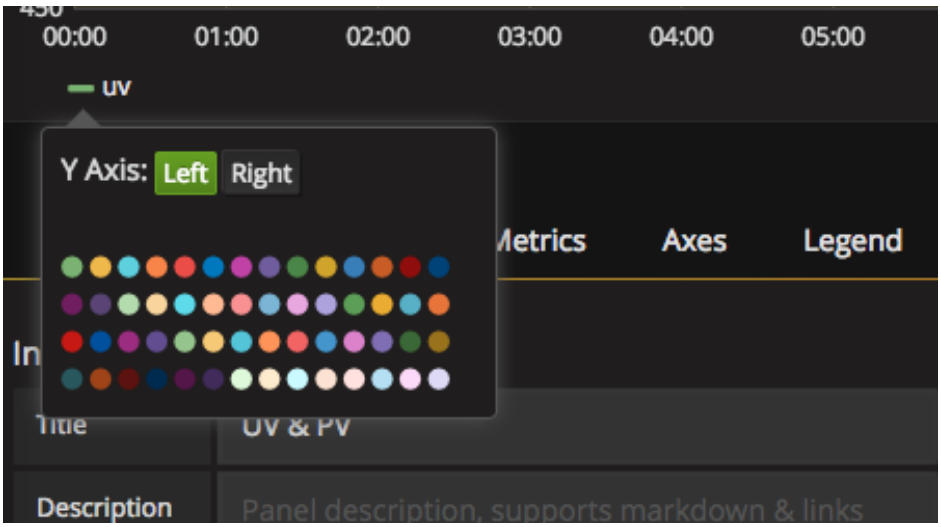


In Metrics configuration, select logservice for datasource and enter Query, Y axis, and X axis:

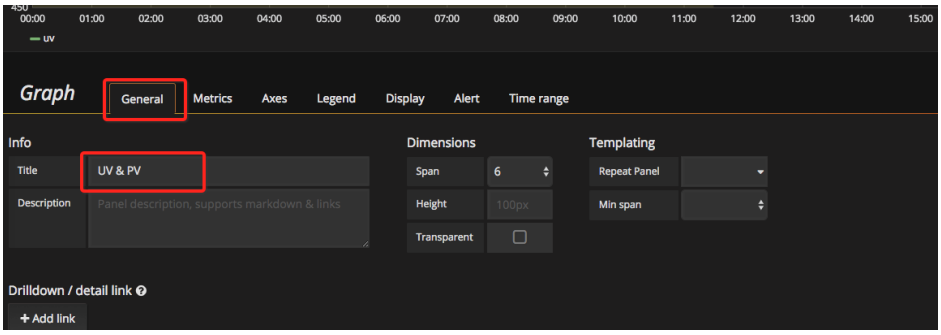
In the dataSource drop-down box, select the configured: logService.

Configuration item	Configuration content
Query	$hostname| select approx_distinct(remote_addr) as uv ,count(1) as pv , __time__ - __time__ % $$myinterval as time group by time order by time limit 1000 The $hostname in the preceding Query is replaced with the domain name selected by the user for actual display. The $$myinterval is replaced with a time interval. Note that the number of $ before myinterval is two and for hostname is one.
X-Column	time
Y-Column	uv,pv

UV and PV values are displayed with two Y axes due to their large difference. Click the colored line on the left of UV below the icon to decide whether the UV is displayed on the left or the right Y axis:



The default view for the title is Panel Title. You can click General to modify the view.



Configure inbound and outbound bandwidth

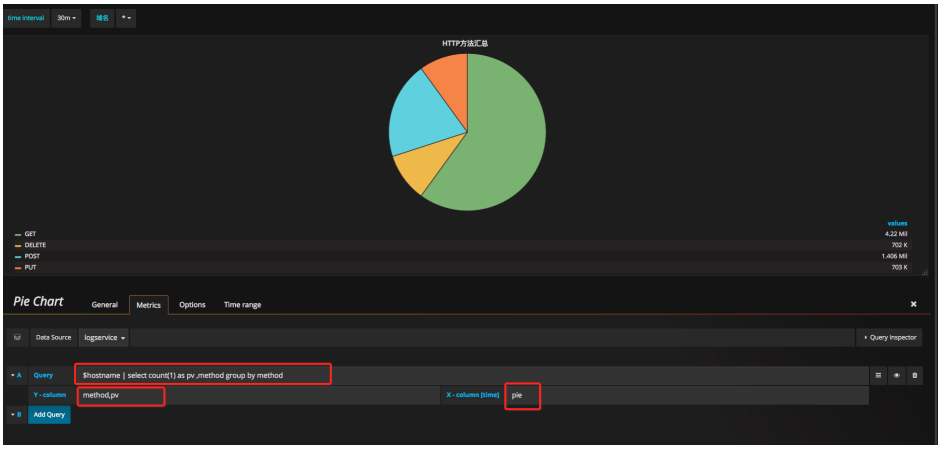
You can add inbound and outbound bandwidth traffic in the same way.

Configuration item	Configuration content
Query	$hostname | select sum(body_byte_sent) as net_out, sum(request_length) as net_in ,__time__ - __time__ % $$myinterval as time group by __time__ - __time__ % $$myinterval limit 10000
X-Column	Time

Y-Column	net_in,net_out

Percentage of HTTP methods

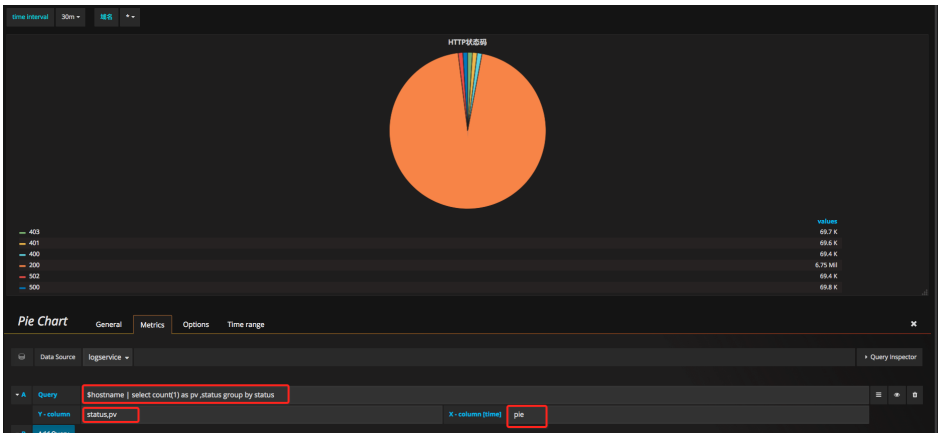
Create a new Row, select Pie Chart, and enter Query, X axis, and Y axis in the configuration.



Configuration item	Configuration content
Query	$hostname | select count(1) as pv ,method group by method
X-Column	pie
Y-Column	method,pv

Percentage of HTTP status codes

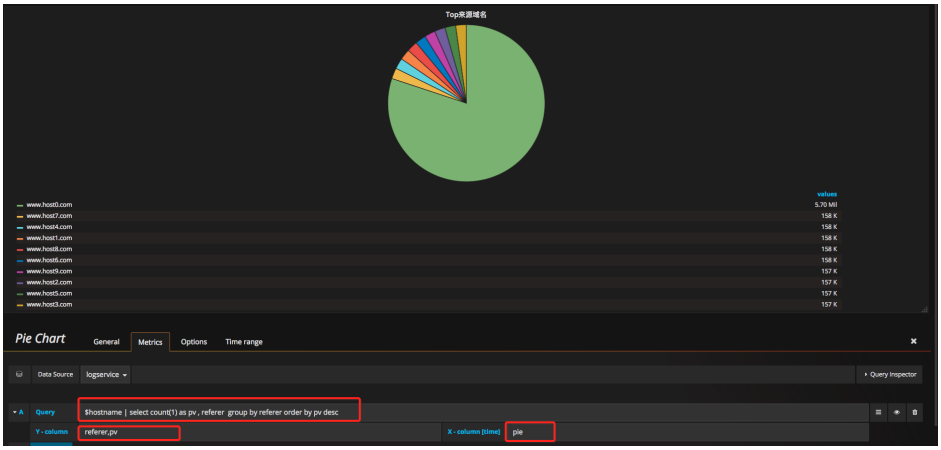
Create a new Row, and select Pie Chart for the view:



Configuration item	Configuration content
Query	$hostname | select count(1) as pv ,status group by status
X-Column	pie
Y-Column	status,pv

Page of top sources

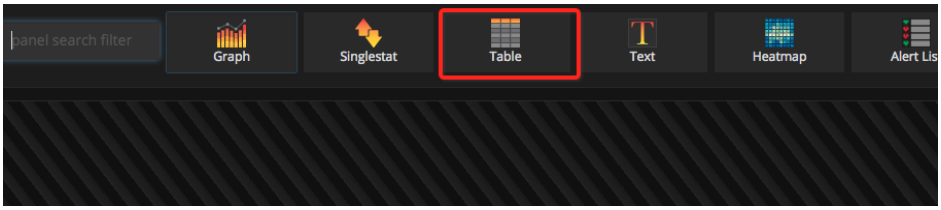
Create a new Row, and select Pie Chart for the view:

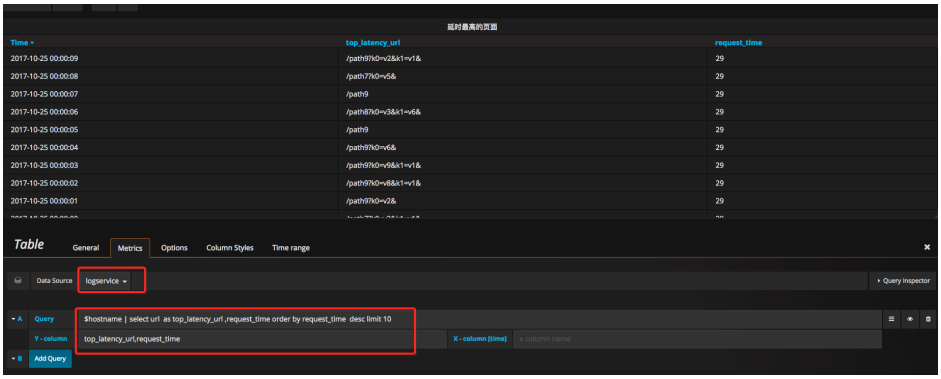


Configuration item	Configuration content
Query	$hostname | select count(1) as pv , referer group by referer order by pv desc
X-Column	pie
Y-Column	referer,pv

Pages of the maximum latency

To show URL and its latency in a table, you must specify the view as Table at the time of creation.



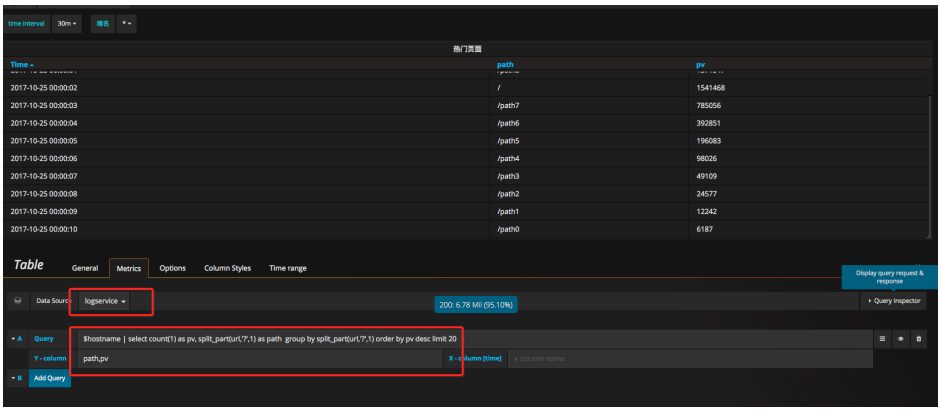


Configuration item	Configuration content
Query	$hostname | select url as top_latency_url ,request_time order by request_time desc

	limit 10
X-Column	X-Column is left blank
Y-Column	top_latency_url,request_time

Top pages

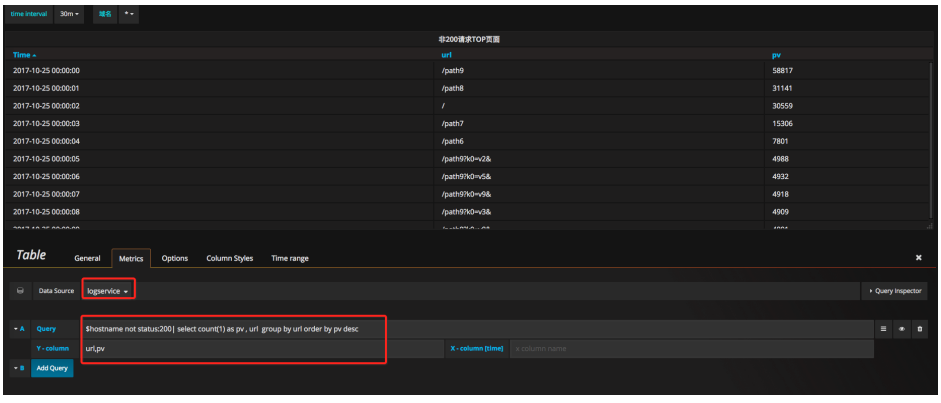
Create a new Table view.



Configuration item	Configuration content
Query	$hostname | select count(1) as pv, split_part(url,'?',1) as path group by split_part(url,'?',1) order by pv desc limit 20
X-Column	X-Column is left blank
Y-Column	path,pv

Top pages of non-200 requests

Create a new Table view.

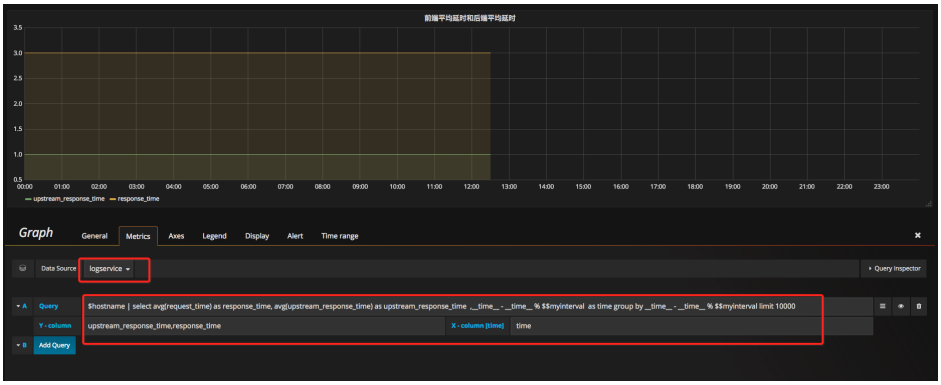


Configuration item	Configuration content
Query	$hostname not status:200| select count(1) as pv , url group by url order by pv desc

X-Column	X-Column is left blank
Y-Column	url,pv

Average frontend and backend latency

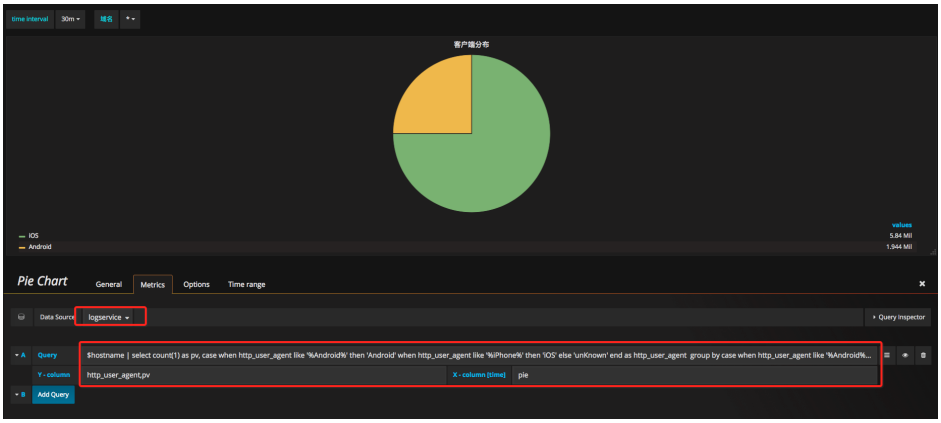
Create a new Graph view:



Configuration item	Configuration content
Query	$hostname | select avg(request_time) as response_time, avg(upstream_response_time) as upstream_response_time ,__time__ - __time__ % $$myinterval as time group by __time__ - __time__ % $$myinterval limit 10000
X-Column	time
Y-Column	upstream_response_time,response_time

Client statistics

Create a new Pie Chart:



Configuration item	Configuration content
Query	$hostname | select count(1) as pv, case

	when http_user_agent like '%Android%' then 'Android' when http_user_agent like '%iPhone%' then 'iOS' else 'unKnown' end as http_user_agent group by case when http_user_agent like '%Android%' then 'Android' when http_user_agent like '%iPhone%' then 'iOS' else 'unKnown' end order by pv desc limit 10
X-Column	pie
Y-Column	http_user_agent,pv

Save and release the dashboard

Click the Save button at the top of the page to release the Dashboard.

View the results

Open the home page of Dashboard to view the results. Demo address.

At the top of the page, you can choose the time range or time granularity of statistics, or you can choose a different domain name.

Now, the configuration of Dashboard for Nginx access statistics is completed, enabling you to mine valuable information from your views.

The Log Service LogShipper function can easily post log data to OSS, Table Store, MaxCompute and other storage services, when used together with e-MapReduce (Spark and Hive) or MaxCompute, for offline computing.

Data Warehousing (offline computing)

Using data warehouse and offline computing together supplements real-time computing. However, the two are used for different purposes:

Mode	Pros	Cons	Scope of Application
Real-time computing	Fast	Simple computing	Mainly used for incremental computation in monitoring and real-time analysis
Offline computing (data warehouse)	Accurate and powerful	Relatively slow	Mainly used for full computation in BI, data statistics and comparison

To satisfy the current data analysis requirements, the same set of data needs to go through both real-time computing and data warehousing (offline computing). In the case of access log:

- Display the real-time market data through StreamCompute: Current PV, UV, and carrier information.
- Conduct detailed analysis of the full data set every night to compare growth, year-on-year/month-on-month growth, and TOP data.

In the world of Internet, there are two classic models under discussion:

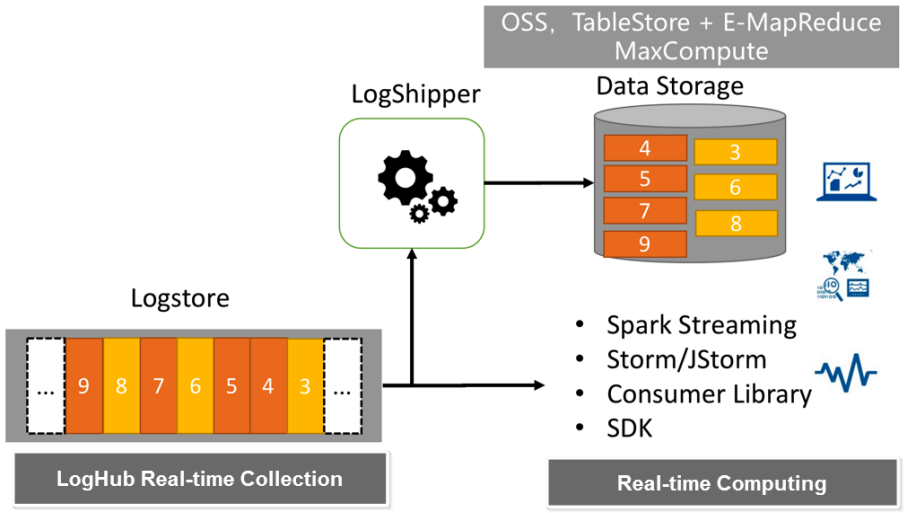
- Lamdba Architecture: When data comes in, the architecture can stream and at the same time save the data into the data warehouse. However, when you initiate a query, the results will be returned from real-time computing and offline-computing based on query conditions and complexity.
- Kappa Architecture: Kafka-based Architecture. With the offline computing feature weakened, all data is stored in Kafka and all queries are fulfilled with real-time computing.

The Log Service provides a delivery mode similar to that of Lamdba Architecture.

LogHub/LogShipper provides a one-stop solution for both real-time and offline scenarios.

Create a LogStore first, and configure LogShipper at the console to enable data warehouse connection. Currently, the following services are supported:

- OSS (Massive Object Storage Service):
    - Instructions
    - Procedures
    - The formats on OSS can be processed using Hive. We recommend the E-MapReduce.
- TableStore (NoSQL Data Storage Service):
    - Procedures



LogShipper provides the following features:

- Quasi-real-time: Data warehousing in minutes
- Enormous data volume No need to worry about concurrency
- Retry-on-error: Automatic retry or API-based manual retry in case of faults
- Task API: Acquire log delivery status for different time frames using API
- Auto compression: Data compression to reduce 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 needs A to capture user visits over the past 180 days and, when necessary, provides the access logs within a given period of time.
- The operation team needs to prepare an access log report on a quarterly basis.

Using the Log Service (LOG) to collect log data from the servers, A turns on the log posting (LogShipper) function, allowing the Log Service to automatically collect, post and compress logs. When an audit is required, the logs within the time frame can be authorized to a third party. To conduct offline analysis, use e-MapReduce to run a 30-minute offline task, getting two jobs done at minimal cost.

Scenario 2: Real-time log + offline analysis

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 using API.

- Conduct real-time log analysis using Spark Streaming and collect statistics on online user visits.
- Use Hive to conduct T +1 offline analysis.
- Grant downstream agencies access to the log data for analysis in other dimensions.

With the LOG+OSS+EMR+RAM combination from today’ s discussion, you can easily fulfill such requirements.