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 via 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 via 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 via 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. A user can place his/her take-away order via 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 food to the user.

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?

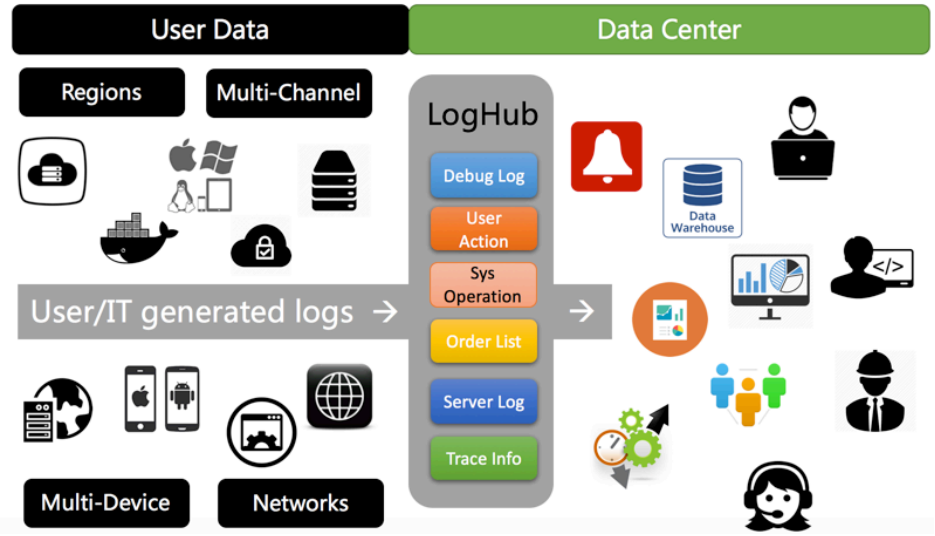
Data Collection Challenges

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 back end 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 via 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 via the Logtail Collection.
- The application writes the logs via 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 via 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 terminals: 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.

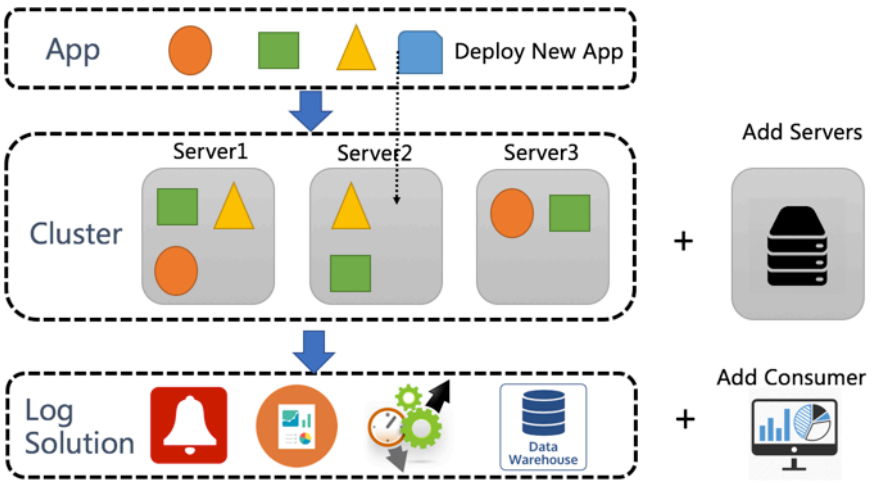
For more information, please refer to the Network Access. You can always find a suitable solution.

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.

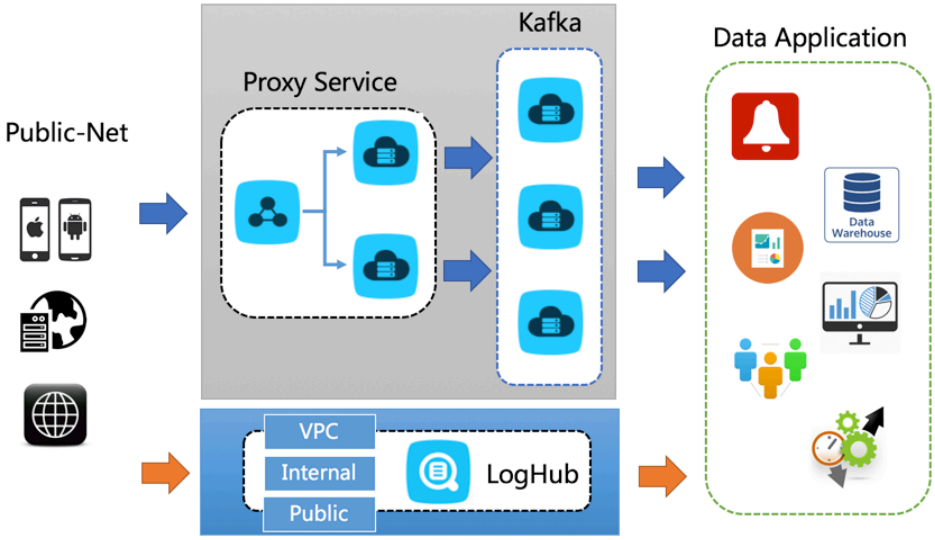
Collect public network data

In some application scenarios, it is required to collect data from a public network (for example, mobile terminal, 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 terminals, 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 via 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 are 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 are 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

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. A user can place his/her take-away orders via 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 user.

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 via 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
```

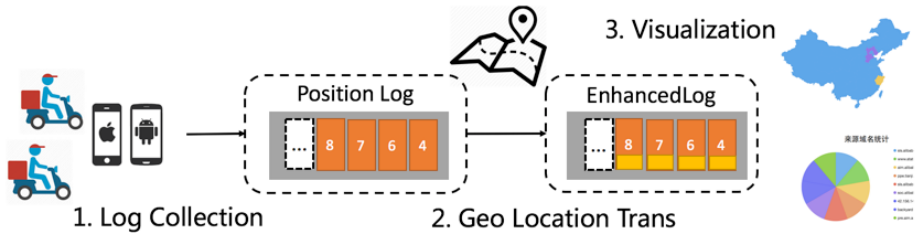
The data feed records the reporting time, courier ID, network in use, device serial number, and coordinates (GPS-X and GPS-Y). The longitude and latitude given by GPS are very accurate, but the operation team actually does not need such accurate data to understand the current status statistics for each 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:

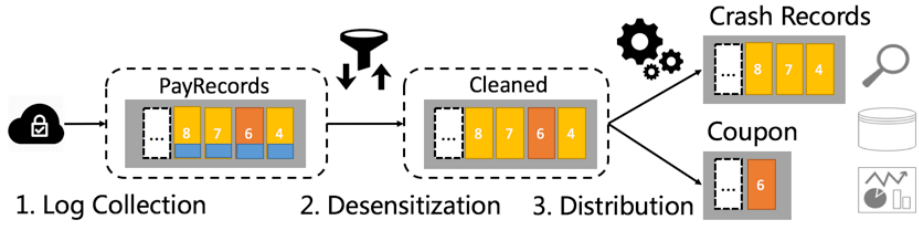
Create four LogStores for raw data (PayRecords), desensitized data (Cleaned), cash order (Cash), and coupon (Coupon) respectively. The application uses Log4J Appender to write the order data into the raw data LogStore (PayRecords).

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, via 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 via 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 OTS, MaxCompute and CNZZ use the Log Service Logtail to collect log data and consume data via 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.

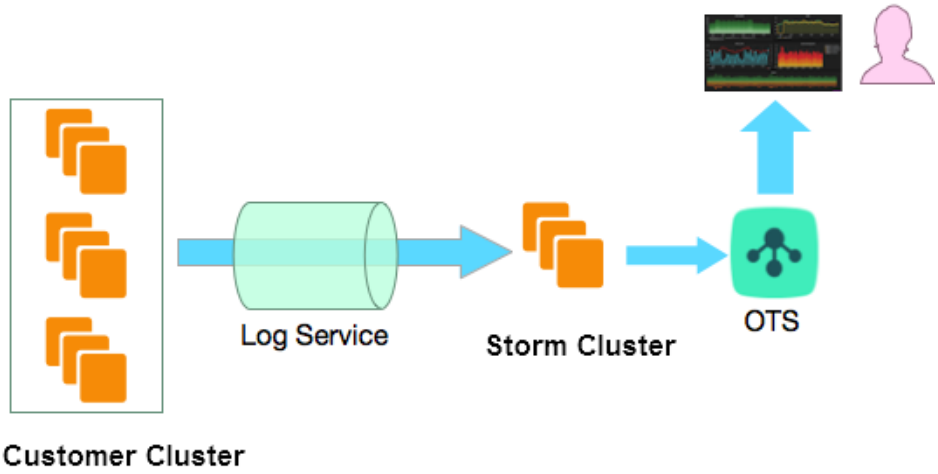
The Log Service helps users collect logs, unify log format and offers APIs for downstream consumption. Downstream systems can be connected to multiple other systems for repeated log consumption, such as using imports from Spark or Storm for real-time computing, or using imports from Elasticsearch for 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 are 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 SLS 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 via 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 OTS.

4. Display the monitoring information

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

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

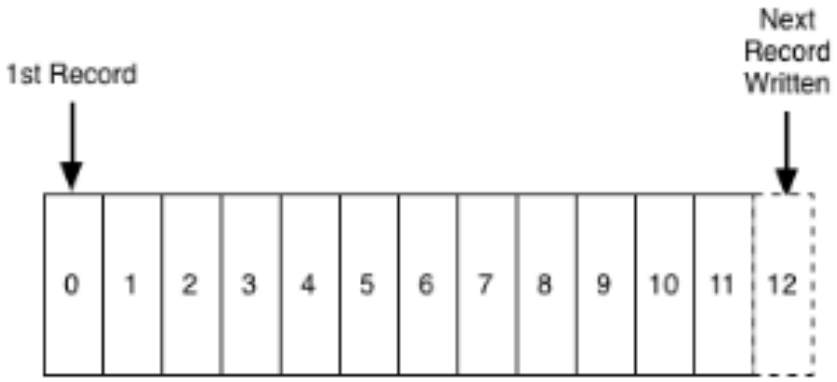
upstream/downstream service system and dramatic fluctuation of service traffic.

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 should 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:

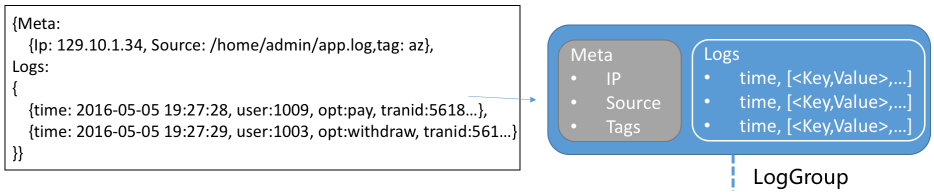
```
2016-06-28 08:00:00 Michael Jacob Deposit US$1,000
2016-06-27 09:00:00 Shane Lee Withdraw US$2,0000
```

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.

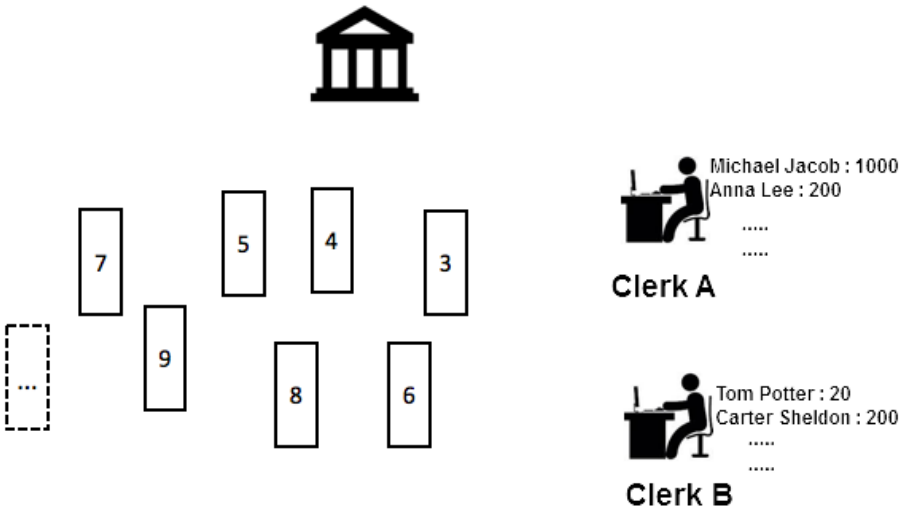
In the world of distributed system, we take a clerk as a single server with fixed memory size and computing capacity. Users are regarded as requests from different data sources, and the bank's lobby as the log database (LogStore) that processes users' access data.

- 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.

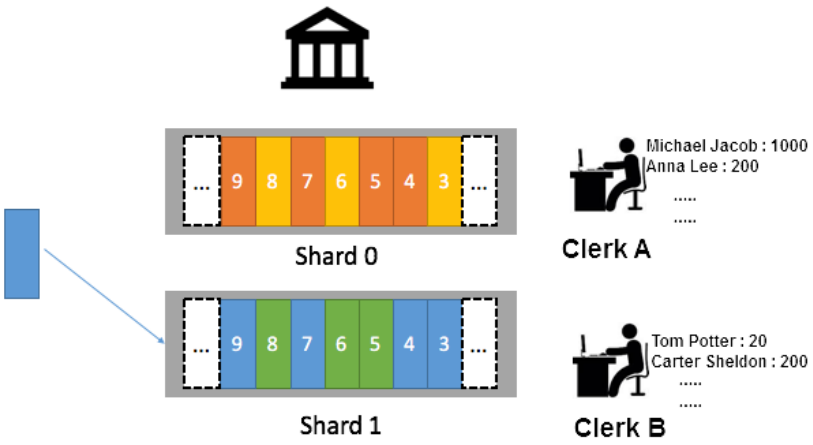


It is easy to achieve ordering: Queue user requests, create a Shard, and assign Clerk A as the only clerk who handles the requests. User requests are handled on a First-In, First-Out (FIFO) basis. Everything works fine except for low efficiency. In the case of 1,000 users, it won't improve the efficiency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



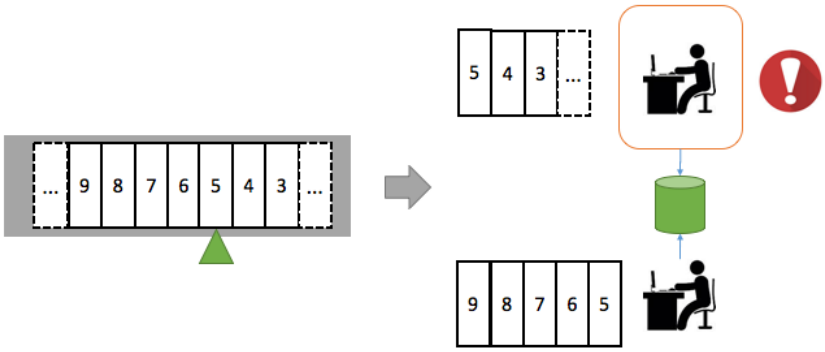
Question 2: At-least once

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

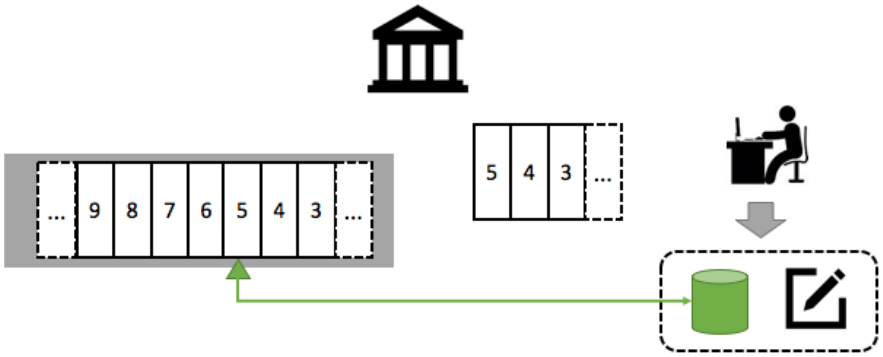
Will repeated entries cause problems? Not necessarily.

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

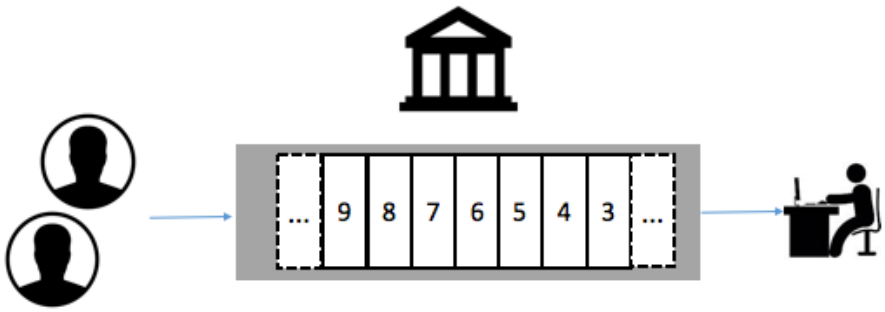
Now we have explained the three questions, which seem easy to resolve. However, in the real world, scale changes and uncertainty may further complicate the three questions above. For example:

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

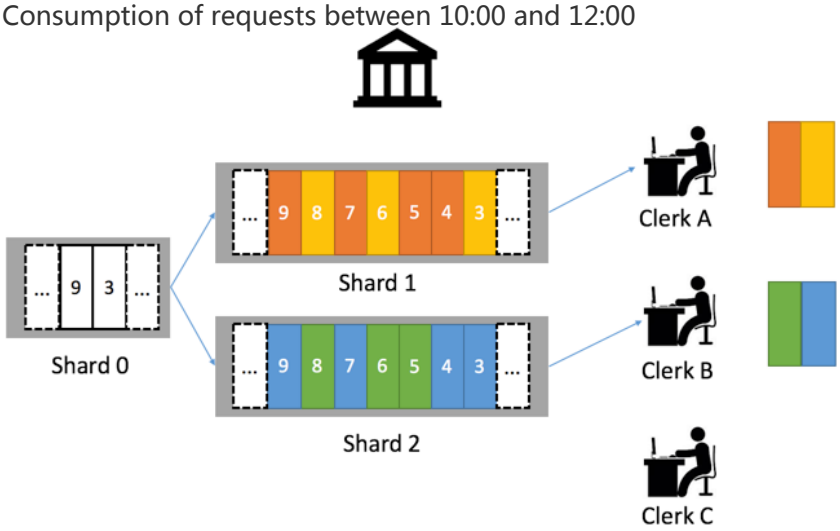
At that point, there was only one Shard, Shard0. All the user requests went to the queue for Shard0, and Clerk A was comfortable to handle the workload alone.



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

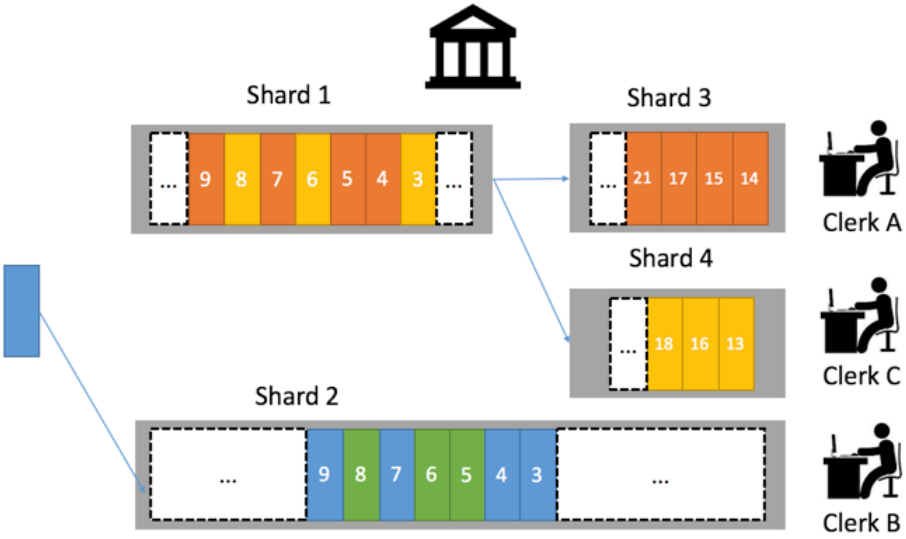


Seeing that Clerk A was difficult to handle two Shards at the time, the bank manager sent Clerks B and C. Since there were only two Shards, Clerk B took over one Shard from Clerk A and Clerk C stood by.

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.

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.

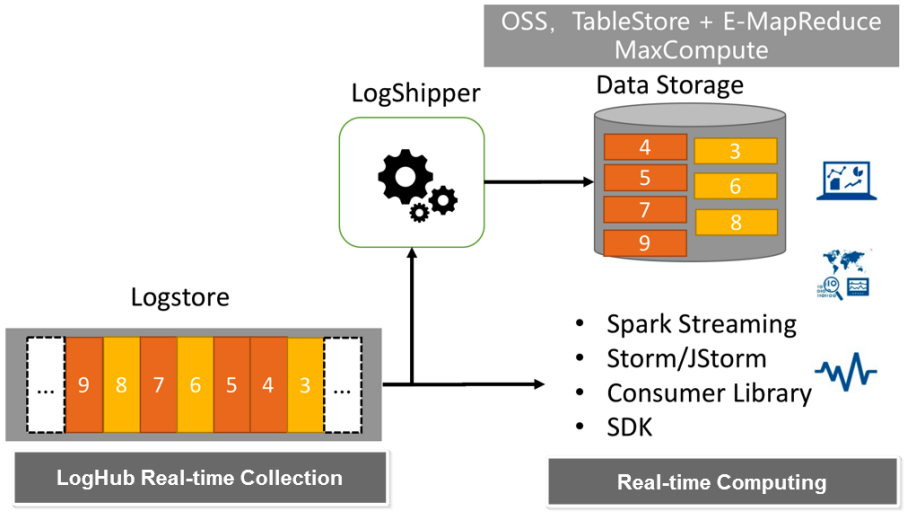
The Log Service provides a delivery mode similar to that of Lamdba Architecture.

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 via 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 via 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 terminal via 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.