Large site architecture of distributed message queue

http://www.finalshares.com/read-6764?f=gk-61

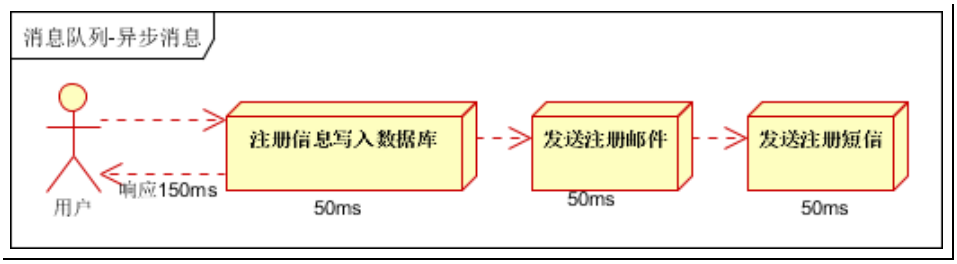
Message Queuing Middleware is a distributed system is an important component, mainly to solve the coupled applications, asynchronous messaging, traffic cut front and other issues.High performance, high availability, scalable architecture and eventual consistency. Large-scale distributed systems is indispensable for the middleware.

Currently in production, the use of message queues have more ActiveMQ, RabbitMQ, ZeroMQ, Kafka, MetaMQ, RocketMQ like.

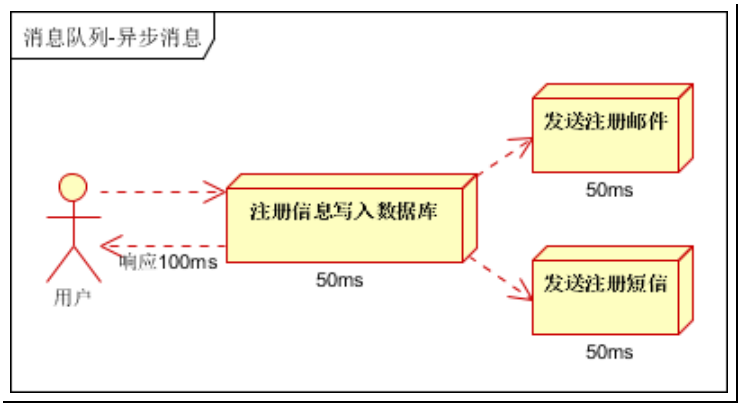
2.1 asynchronous processing

Scenario: After user registration, we need to send registered mail and SMS registration. 1. The traditional approach has two serial fashion; 2. parallel.

(1) Serial method: after the success of the registration information into the database, send registered mail and then send the registration message. After more than three tasks completed, returned to the client. (Architecture KKQ: 466097527, welcome to join)



(2) parallel: After successful registration information into the database, send registered mail at the same time, send a registration SMS. After more than three tasks completed, returned to the client. Is the difference between serial and parallel way to improve the processing time.



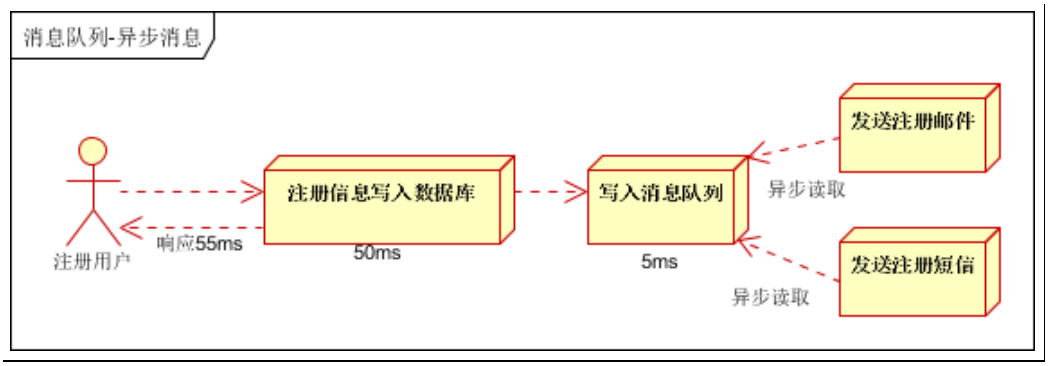
Suppose three service node for each 50 millisecond clock, without considering other overhead networks, the time is 150 milliseconds, serial, parallel, time may be 100 milliseconds.

Because the number of requests processed per unit of time the CPU is certain, it is assumed inside CPU1 sec throughput is 100 times. The serial volume of requests can be processed

within one second CPU is seven times (1000/150). The amount of parallel processing of the request is 10 times (1000/100).

Summary: As the case described above, the traditional performance in a systematic way (concurrency, throughput, response time) there will be a bottleneck. How to solve this problem?

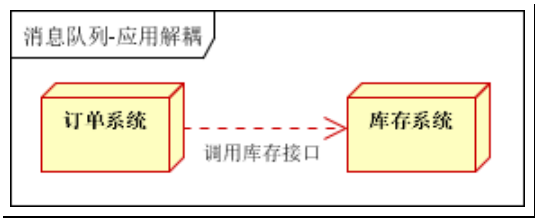
Introducing the message queue, you will not have business logic, asynchronous processing.After transformation, the structure is as follows:



According to the above agreement, the user's response time is equivalent to the registration information is written to the database time is 50 milliseconds. After registered mail, send text messages written in the message queue, return directly, writing the message queue is fast, can be ignored, so the user response time may be 50 milliseconds. So after the organizational changes, the system throughput of up to 20 per second, QPS. It increased 3 times than serial, parallel than tripled.

2.2 Application of Decoupling

Scene Description: The user orders, the order system to notify inventory system. The traditional approach is to order the system call interface inventory system. As shown :(architecture KKQ: 466097527, welcome to join)

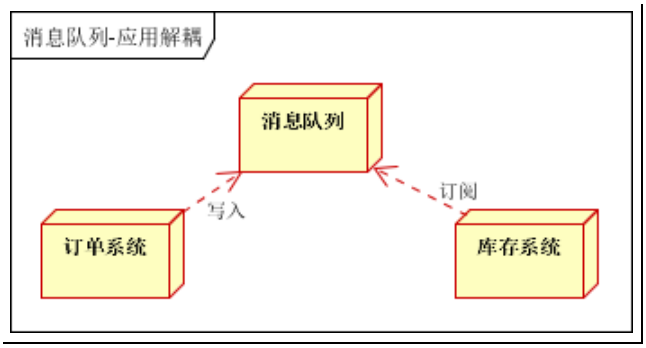


The traditional model of disadvantages:

1) If the inventory system is not accessible, orders minus inventories will fail, resulting in the order failed;

2) ordering system and inventory system coupling;

How to solve the above problems? After the introduction of the application message queue of the program, as follows:



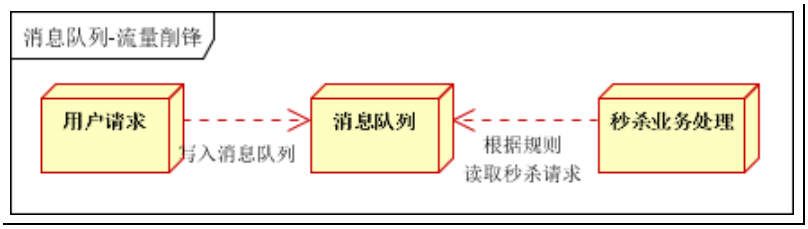
- Order System: User an order, the order processing system to complete persistence, writes messages to the message queue and returns the user orders a single success.
- Inventory System: Subscribe single message, using the pull / push mode, get the next order information, inventory information system according to orders, inventory operations.
- If: the next one when the inventory system is not working. It does not affect the normal order, because an order, the order system writes message queue is no longer the concern of other subsequent operations. Decoupling order to achieve application systems and inventory systems.

2.3 Traffic cut front

Feng also cut traffic message queue common scenarios is generally used in groups or spike rush activities widely.

Scenario: spike activity, usually because of too much traffic, leading to traffic surge, application hang. To solve this problem, generally you need to apply to join the front end of the message queue.

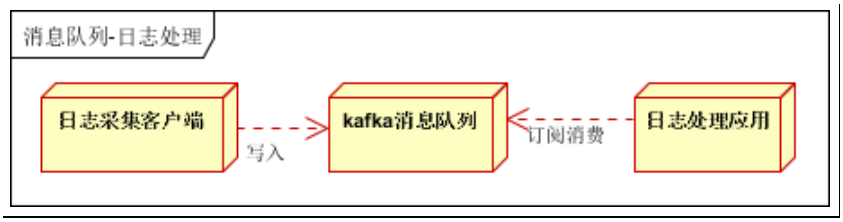
1. You can control the number of activities;
2. You can ease the high traffic in a short time crushed application;



1. The user's request, the server receives the first written message queue. If the message queue length exceeds the maximum number of user requests or discarded directly jump to the error page;
2. Spike business information upon request message queue, then a follow-up treatment.

2.4 log processing

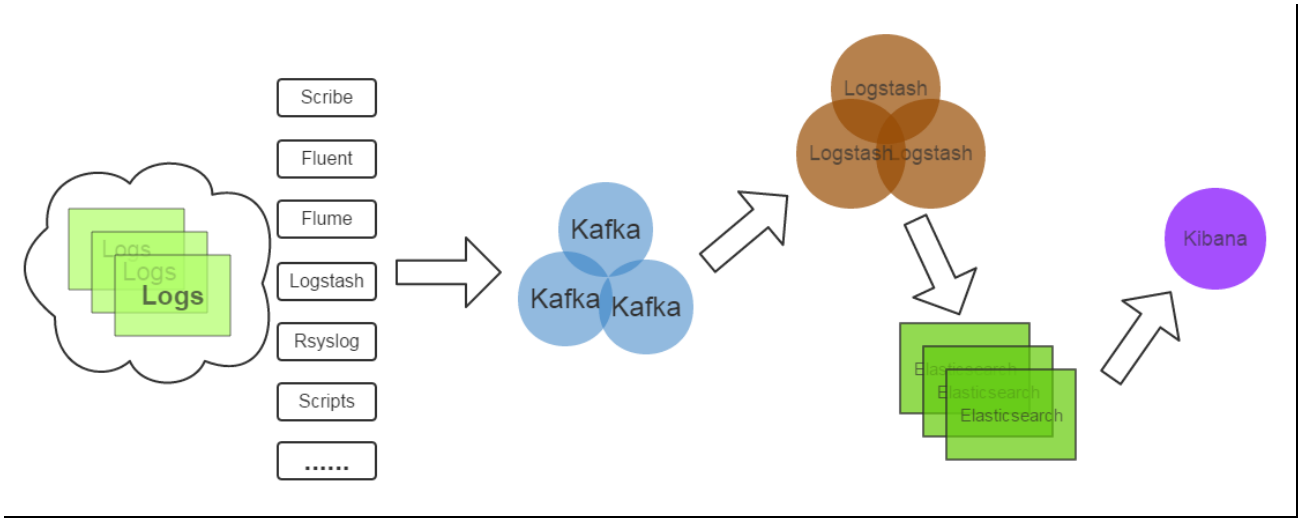
Log processing refers to the message queue is used in log processing, such as Kafka's application to solve a number of issues log transmission. Architecture simplifies as follows:



- Log collection client, the log data collection, written by Kafka timed write queue;
- Kafka message queues, is responsible for receiving log data, store and forward;
- Log Processing Applications: subscription and consumption of kafka queue log data;

The following is Sina kafka log processing Applications:

Transfers (http://cloud.51cto.com/art/201507/484338.htm)



(1) Kafka used to live : receiving a user log message queue.

(2) Logstash : do log analysis, unified into JSON output to Elasticsearch.

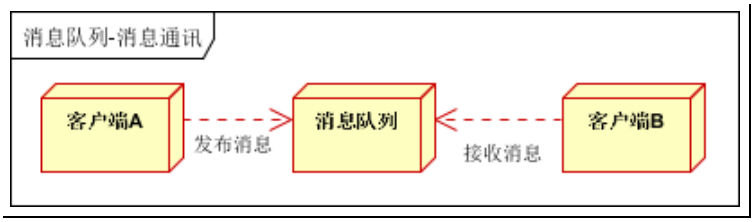
(3) elasticsearch : Real-time log analysis service of the core technology, a schemaless, real-time data storage services through the organization index data, both powerful search and statistical functions.

(4) Kibana : Elasticsearch based data visualization components, powerful data visualization capabilities of many companies choose ELK stack of important reasons.

2.5 Message Communication

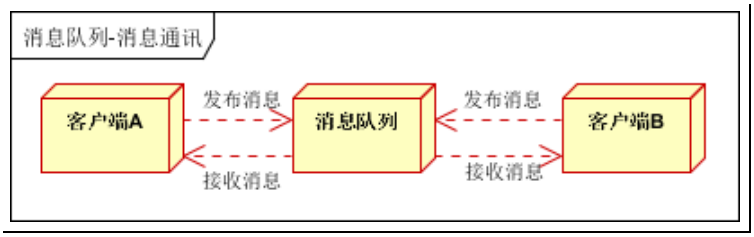
Message communications means, message queues are generally built an efficient communication mechanism, and therefore can be used in pure message communications.For example, point to point message queue, or chat rooms.

Point to point communication:



Client A and Client B use the same queues for message communications.

Chat Room Communication:

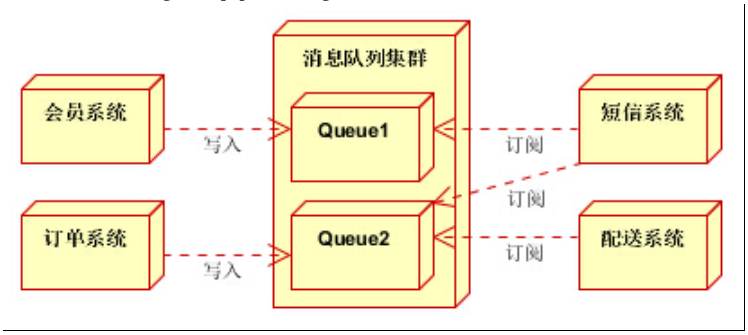


Client A, Client B, N clients subscribing to the same theme, message distribution and reception. Chat room to achieve a similar effect.

Are two or more actual message queue message mode, point to point or publish subscription model. Model diagram for reference.

Third, messaging middleware examples

3.1 electricity supplier system

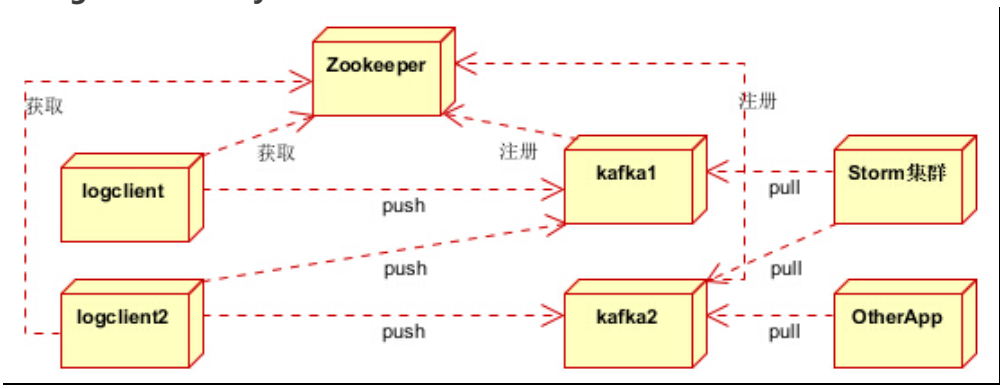


Message queues are highly available persistent messaging middleware. For example, Active MQ, Rabbit MQ, Rocket Mq. (1) Application of the main logic process is completed, write the message queue. Whether the message has been sent successfully mode can be turned on confirmation message. (Message Queuing returns the message after receiving a successful state, and then return to the application, so protect the integrity of the message)

(2) extension process (texting, distribution processing) subscription queue messages. Using push or pull mode to acquire and process the message.

(3) message decoupling applications while bringing data consistency, consistency can finally be resolved. For example, data is written to the primary database, expanding the application according to the message queue, combined with database ways subsequent processing based on the message queue.

3.2 log collection system



Zookeeper into the registry, the client log collection, consisting of four parts Kafka clusters and cluster Storm (OtherApp).

- Zookeeper registration center proposed load balancing, and address lookup service;
- Log collection client for acquisition applications system logs and data push kafka queue;
- Kafka cluster: receiving, routing, store and forward messaging;

Storm Cluster: OtherApp and at the same level, the use of pull mode consumption data queue;

Four, JMS message service

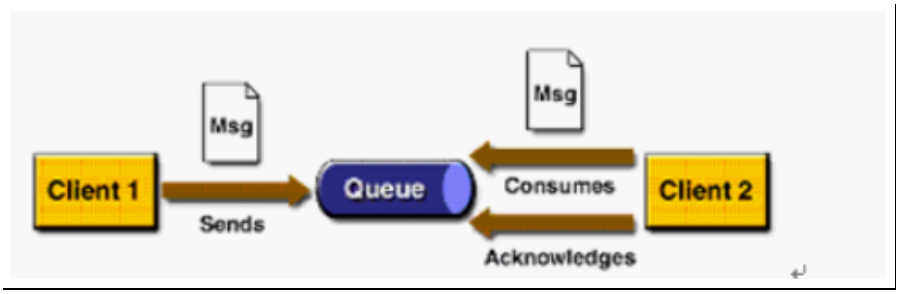
Speak the message queue have to mention JMS. JMS (JAVA Message Service, java Message Service) API is a standard messaging service / specification that allows application components based on JavaEE platform to create, send, receive and read messages. It enables distributed communication coupling lower messaging service more reliable and asynchronous.

In EJB architecture, there is a message bean can be seamlessly integrated with JM message service. In J2EE architecture model, the news service provider model for achieving direct message decoupling applications.

4.1 messaging model

In the JMS standard, there are two types of messaging models P2P (Point to Point), Publish / Subscribe (Pub / Sub).

4.1.1 P2P mode



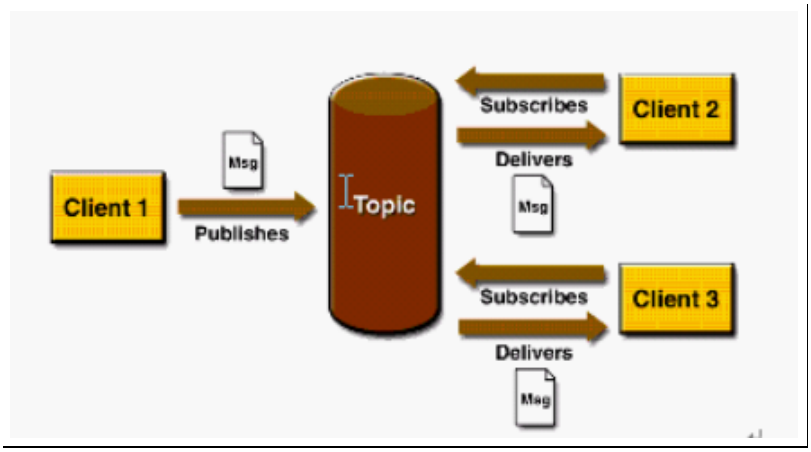
P2P mode contains three roles: the message queue (Queue), the sender (Sender), receiver (Receiver). Each message is sent to a particular queue, recipients get messages from the queue. It retains the message queue until they are consumed or timeout.

P2P features

- Each message is only a consumer (Consumer) (that is, once consumed, the message is no longer in the message queue)
- Between the sender and receiver do not depend on time, that is to say when the sender sends the message, whether the recipient has no running, it will not affect the message is sent to the queue
- After successfully recipient receives the message queue for an answer to success

If you want to send each message will be successfully processed, then the need for P2P mode. (Architecture KKQ: 466097527, welcome to join)

4.1.2 Pub / sub model



It contains three roles topics (Topic), Publisher (Publisher), subscribers (Subscriber). Multiple publishers to send a message to the Topic, the system will deliver these messages to multiple subscribers.

Pub / Sub features

- Each message may have a plurality of consumers
- Dependent on the time between publishers and subscribers. After for a theme (Topic) subscribers, it must create a subscriber to the publisher of news consumption.
- In order to consume the message, the subscriber must keep the state running.

In order to alleviate such a strict time-dependent, JMS allows subscribers to create a persistent subscription. Thus, even if the subscriber is not activated (run), it can also receive the publisher's message.

If you want to send a message can not be done any treatment, or by only one person handling the message, or may be processed multiple consumers, then the Pub / Sub model can be used.

4.2 Consumer News

In JMS, produce and consume messages are asynchronous. For the consumer who, JMS message can be consume messages in two ways.

(1) Synchronous

Subscribe or recipient to receive messages receive method, receive method (or time-out before) will remain blocked until the message is received;

(2) Asynchronous

Subscribers or receivers can be registered as a message listener. When a message arrives, the system automatically calls the listener's onMessage method.

JNDI: Java Naming and Directory Interface, is a standard Java naming system interface. You can find and access services on the network. By specifying a resource name that corresponds to a record in a database or naming service, and return the resource information necessary to establish the connection.

JNDI serve to find and access the delivery destination or source role in JMS. (Architecture KKQ: 466097527, welcome to join)

4.3JMS programming model

(1) ConnectionFactory

Create a Connection object factory for two different jms messaging models, respectively QueueConnectionFactory and TopicConnectionFactory two kinds. ConnectionFactory via JNDI to locate the object.

(2) Destination

Destination means the target message producers send messages or news sources consumers. For news producer, its Destination is a queue (Queue) or a theme (Topic); for news consumers, its Destination is a queue or topic (ie, source).

So, Destination actually two types of objects: Queue, Topic by JNDI to find Destination.

(3) Connection

Connection between the client representation and JMS systems link established (packaging for TCP / IP socket's). Connection can generate one or more Session. Like ConnectionFactory, Connection, there are two types: QueueConnection and TopicConnection.

(4) Session

Session interface operation is the message. You can create producers, consumers, and other messages through the session. Session provides the functionality affairs. When you need to use the session sending / receiving multiple messages, these can transmit / receive operation into a transaction. Similarly, sub-QueueSession and TopicSession.

Producer (5) of the message

A message producer is created by Session, and used to send messages to a Destination.Similarly, the message producer of two types: QueueSender and TopicPublisher. The method (send or publish method) can be called message producers to send messages.

(6) Consumer News

Message consumers created by the Session, for receiving a message is sent to the Destination. Two types: QueueReceiver and TopicSubscriber. Respectively created by the session createReceiver (Queue) or createSubscriber (Topic). Of course, it can also creatDurableSubscriber method to create a session persistence subscribers.

(7) MessageListener

Message listener. If you register a message listener, once the message arrives, it will automatically call the listener's onMessage method. EJB in the MDB (Message-Driven Bean) is a kind of MessageListener.

Depth study and mastery of JAVA architecture JMS, EJB architecture has a very good help, messaging middleware large-scale distributed systems is also necessary components. The main job of sharing overall, specific depth we need to learn, practice, summarize, comprehend.

Fifth, the common message queue

General commercial vessels, such as WebLogic, JBoss, JMS support standard, easily developed. But free such as Tomcat, Jetty and so you need to use a third-party messaging middleware. This section describes the common messaging middleware (Active MQ, Rabbit MQ, Zero MQ, Kafka) and their characteristics.

5.1 ActiveMQ

Apache ActiveMQ is produced, the most popular, a strong ability to open source message bus. ActiveMQ is a fully supported JMS1.1 and J2EE 1.4 JMS Provider Implementation Specification, though JMS specification has been introduced a long time ago, but in the middle of today's J2EE JMS application still plays a special status.

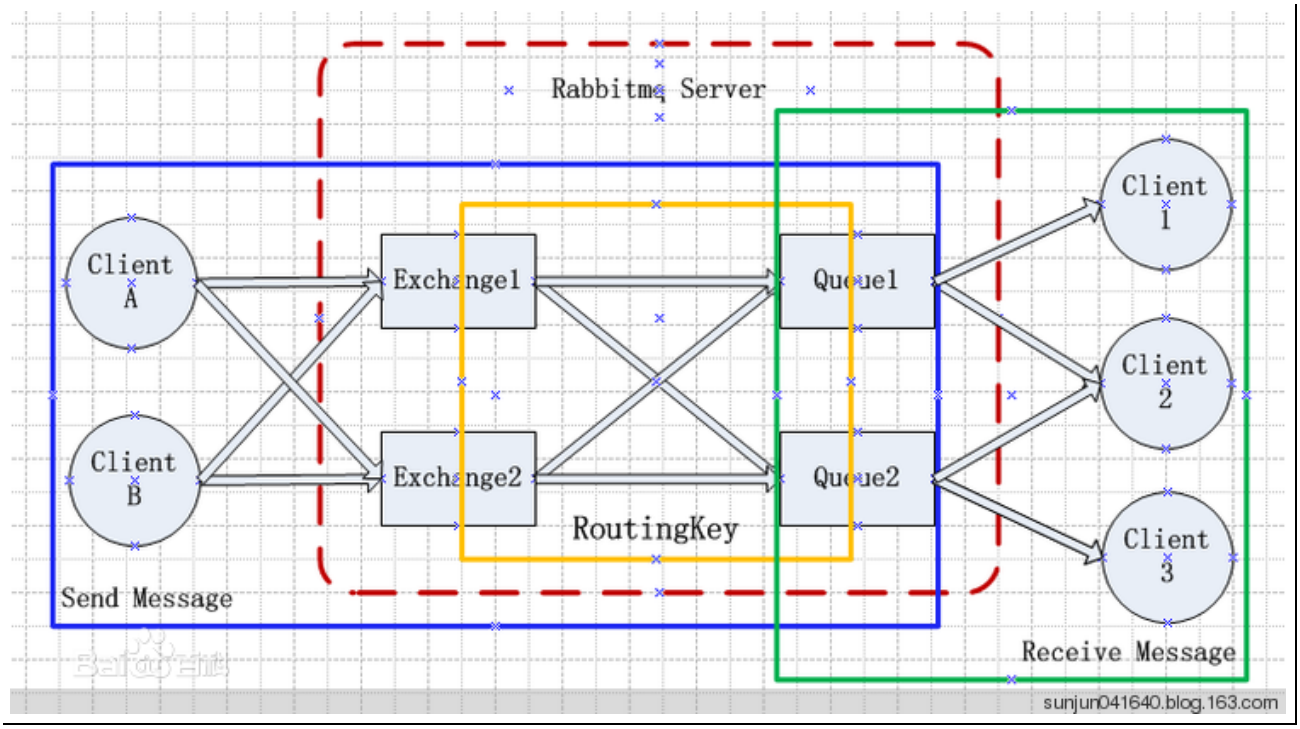
ActiveMQ characteristics are as follows:

1. languages and protocols prepared by the client. Languages: Java, C, C ++, C #, Ruby, Perl, Python, PHP. Application protocol: OpenWire, Stomp REST, WS Notification, XMPP, AMQP
2. JMS1.1 and fully supports the J2EE 1.4 specification (persistent, XA messages, transactions)
3. support for Spring, ActiveMQ can be easily embedded into the system using Spring to go inside, but also supports the feature Spring2.0
4. through common J2EE servers (such as Geronimo, JBoss 4, GlassFish, WebLogic) test, which by JCA 1.5 resource adaptors configuration allows ActiveMQ can automatically be deployed on any J2EE 1.4 compatible server business
5. supports multiple transport protocols: in-VM, TCP, SSL, NIO, UDP, JGroups, JXTA
6. JDBC support is provided by high-speed and journal message persistence
7. from the design to ensure high performance clusters, client - server, peer
8. support Ajax
9. support integration with Axis
10. can easily get to call the embedded JMS provider, tested

5.2 RabbitMQ

RabbitMQ is a popular open source message queuing systems, develop with erlang language. RabbitMQ is AMQP (Advanced Message Queuing Protocol) standard implementation. It supports a variety of clients, such as: Python, Ruby, .NET, Java, JMS, C, PHP, ActionScript, XMPP, STOMP, and support for AJAX, persistence. In a distributed system for message store and forward, in terms of ease of use, scalability, high availability, and so doing well.

Structure is as follows :(architecture KKQ: 466097527, welcome to join)



Several important concepts:

Broker: Message Queue Server is simply an entity.

Exchange: messaging switch that specifies the message according to what rules, which are routed to a queue.

Queue: Message Queuing carrier, each message will be put into one or more queues.

Binding: binding, its role is to exchange and queue bound up in accordance with the routing rules.

Routing Key: routing keywords, exchange of messages delivered in accordance with this keyword.

vhost: web hosting, where a broker can open multiple vhost, used for different users privilege separation.

producer: news producer, is a program delivery message.

consumer: the message to consumers, it is to accept the message of the program.

channel: message channels, each connected client, the can create multiple channel, each channel representing a conversation task.

Message queue use, as follows:

(1) client to connect to a message queue server, open a channel.

(2) a declaration client exchange, and sets the relevant properties.

(3) the client declares a queue, and set the associated property.

(4) The client uses routing key, to establish a good relationship between the exchange and the binding queue.

(5) client posting messages to the exchange.

After the exchange receives the message, it is based on key messages and binding has been set, route messages, the message delivered to one or more queue.

5.3 ZeroMQ

Known as the fastest in the history of the message queue, it actually resembles a series of Socket Interface, his difference with Socket is: ordinary socket end to end (1: 1 relationship), but it is ZMQ N: M relationship of people are more understanding of the BSD socket connection is point to point, point to point connection needs to explicitly establish a connection, to destroy a connection, select the protocol (TCP / UDP) and processing errors, and ZMQ shield these details, so you network programming easier. ZMQ used for communication between the node and the node, node can be a host or a process.

Cited official statement:. "ZMQ (hereinafter referred ZeroMQ ZMQ) is a simple and easy transport layer, the same framework as a socket library, he makes Socket programming easier, more simplicity and performance is a message queue processing library, movable between a plurality of threads, the kernel and the host cartridge elastically stretchable .ZMQ explicit goal is to "become part of the standard network protocol stack, after entering the Linux kernel." we have not yet seen their success. However, it is undoubtedly a very with prospects, and that people needed more "traditional" BSD socket layer on top of the package .ZMQ make writing high-performance network applications extremely simple and fun. "

feature is:

- High-performance, non-persistent;
- Cross-platform: support for Linux, Windows, OS X and the like.
- Multi-language support; C, C ++, Java, .NET, Python and other 30 kinds of development language.
- It can be deployed or integrated into applications used alone;
- Socket can be used as a communication library.

Compared with RabbitMQ, ZMQ Message Queue Server does not look like a traditional sense, in fact, it is not a server, like a low-level network communication library, made on the Socket API layer package will network communications, process and thread communication communications abstract unified API interface. Supports the "Request-Reply", "Publisher-Subscriber", "Parallel Pipeline" three basic model and the extended model.

ZeroMQ high-performance design features:

1, lock-free queue model

For data exchange between the cross-thread (client and session) exchange between the channel pipe, lock-free queue algorithm CAS; the pipe ends have registered asynchronous event, when reading or writing messages to the pipe, it will automatically trigger a read write event.

2, batch processing algorithms

For conventional message processing, each message sent and the time of reception, the system needs to call, so for a large number of messages, the system overhead is relatively large, zeroMQ for bulk messages, optimized adaptation, you can batch receive and send messages.

3, under multi-core thread binding, without CPU switch

Different from the traditional multi-threaded mode, semaphore or critical section, zeroMQ full use of the advantages of multi-core, each core is bound to run a worker thread to avoid the CPU overhead of switching between multiple threads.

5.4 Kafka

Kafka is a high-throughput distributed publish-subscribe messaging system that can handle the size of the consumer action website all stream data. This action (web browsing, search and other user's action) is a key factor in many social functions in modern networks. These data are usually due to the throughput requirements be resolved through logs and log aggregation. Like for like Hadoop logs and offline data analysis systems, but requires real-time processing limitations, this is a viable solution. Kafka's purpose is Hadoop parallel loading mechanism to unify both online and offline messaging, but also to the cluster through the machine to provide real-time consumption.

Kafka is a high-throughput distributed publish-subscribe messaging system, has the following characteristics:

- It provides message via O (1) disk data structures persistent, even if such a structure for hundreds of TB of the message store can be maintained long-term stability performance. (File additional ways to write data, delete obsolete data on a regular basis)
- High throughput: even a very ordinary hardware Kafka can support millions of messages per second.
- To support the partition and consumption messages through Kafka server machine cluster.
- Support Hadoop parallel data loading.

Kafka related concepts

- Broker

Kafka cluster consists of one or more servers, such servers are called broker [5]

- Topic

Each message posted to Kafka cluster has a category, this category is called Topic. (Topic of different messages on a separate physical storage, logical though a Topic message stored on one or more of the broker but the user need only specify a message Topic to production or consumption data without having to be concerned about where the data is stored in)

- Partition

Parition concept physically, each containing one or more Topic Partition.

- Producer

Responsible for dissemination of information to Kafka broker

- Consumer

News consumers to read Kafka broker message client.

- Consumer Group

Each belongs to a particular Consumer Consumer Group (Consumer can specify for each group name, without specifying a group name belong to the default group is).

Generally used in large data log processing or real-time (a small amount of latency), reliability (a small amount of lost data) less demanding usage scenarios.

VI. References

The following is a reference to share information and recommended reference data.

References (references available):

(1) Jms

http://blog.sina.com.cn/s/blog_3fba24680100r777.html

http://blog.csdn.net/jiuqiyuliang/article/details/46701559 (layman JMS (a) - JMS basic concepts)

(2) RabbitMQ

http://baike.baidu.com/link?url=s2cU-QgOsXan7j0AM5qxxlmruz6WEeBQXX-Bbk0O3F5jt9Qts2uYQARxQxl7CBT2SO2NF2VkzX_XZLqU-CTaPa

http://blog.csdn.net/sun305355024sun/article/details/41913105

(3) the Zero the MQ

http://www.searchtb.com/2012/08/zeromq-primer.html

http://blog.csdn.net/yangbutao/article/details/8498790

http://wenku.baidu.com/link?url=yYoiZ_pYPCuUxEsGQvMMleY08bcptZvwF3IMHo2W1i-ti66YXXPpLLJBGXboddwgGBnOehHiUdslFhtz7RGZYkrtMQQ02DV5sv9JFF4LZnK

(4) Kafka used to live

http://baike.baidu.com/link?url=qQXyqvPQ1MVrw9WkOGSGEfSX1NHy4unsgc4ezzJwU94SrPuVnrKf2tbm4SllVaN3ArGGxV_N5hw8JTT2-lw4QK

http://www.infoq.com/cn/articles/apache-kafka/

http://www.mincoder.com/article/3942.shtml