

Web Service's Runtime Information Collection and Storage Approach and Implementation

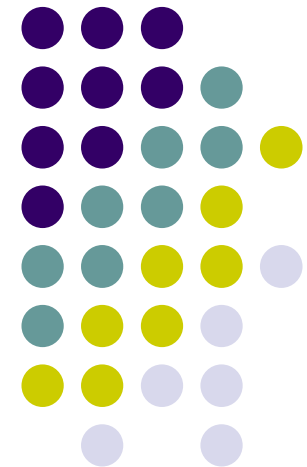
Chengzhiwen, Shaolingshuang,
Fanglu, JinJing, Zhengxiaoxia

QoS Group

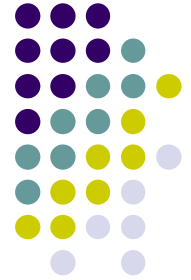
Software Component Lab.

Software Institute

Peking University



Agenda



- **Introduction**
- Our work
 - Requirements
 - Client Side Collect
 - Server Side Collect
 - Information Storage and Retrieve

Program Statistics

- Members: five people
- Time: two months
- Code lines: 8000+ lines



Introduction^{C3}



- The development of Web Services is vigorous
- QoS (Quality of Service) is more important because of the distributed character of Web Services
- QoS is the basement of Web Service based Applications
- Web Services runtime information is a important and direct approach to evaluate web services' QoS

Diapositive 4

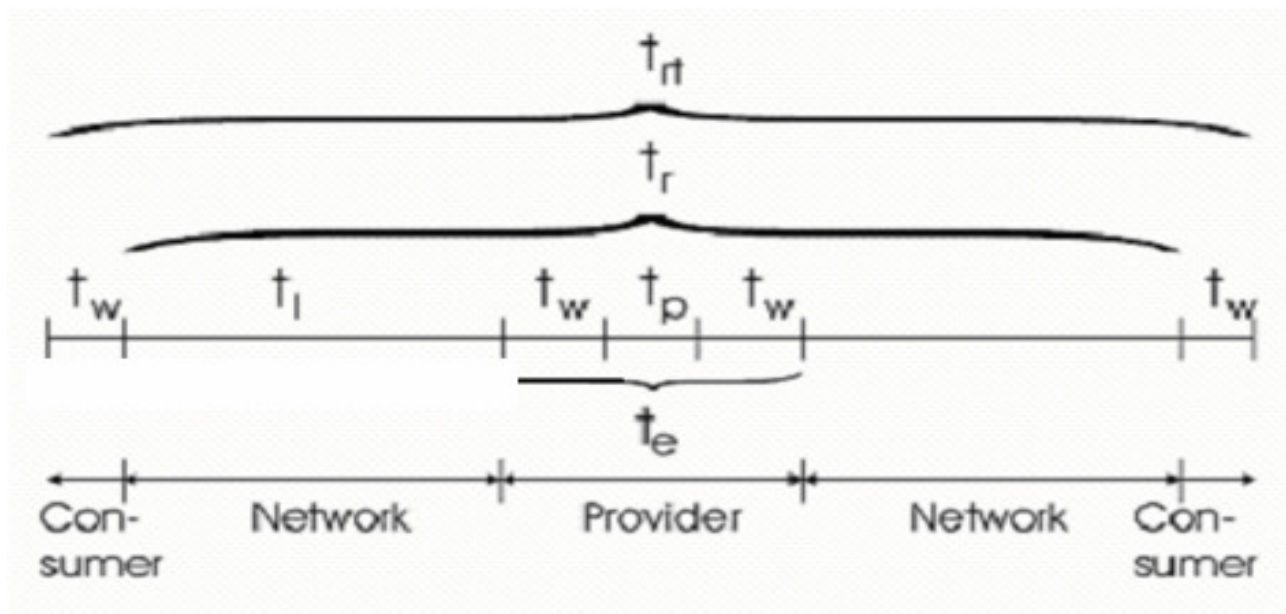
C3

放一些有说明性的数据
Cheng; 16/09/2009

The process of Web Services invocation



- 1 serialize a request to soap message(t_w)
- 2 transform the request to web service server (t_l)
- 3 deserialize the soap message to a request(t_w)
- 4 process the request(t_p)
- 5 serialize the response to soap message(t_w)
- 6 Transform the response soap message to the client(t_l)
- 7 the client deserialize the soap message to response(t_w)



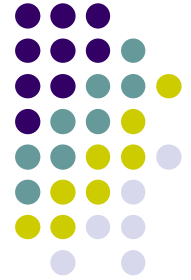
Related Implementations



Approach	Performance Reduce	Accuracy	Sharing	deploy	Complete
APIHOOK	high	Service	Not share	easy	Not
AOP	low	Service	Not share	Not easy	Yes

Agenda

- Introduction
- **Our work**
 - Requirements
 - Client Side Collect
 - Server Side Collect
 - Information storage and retrieve



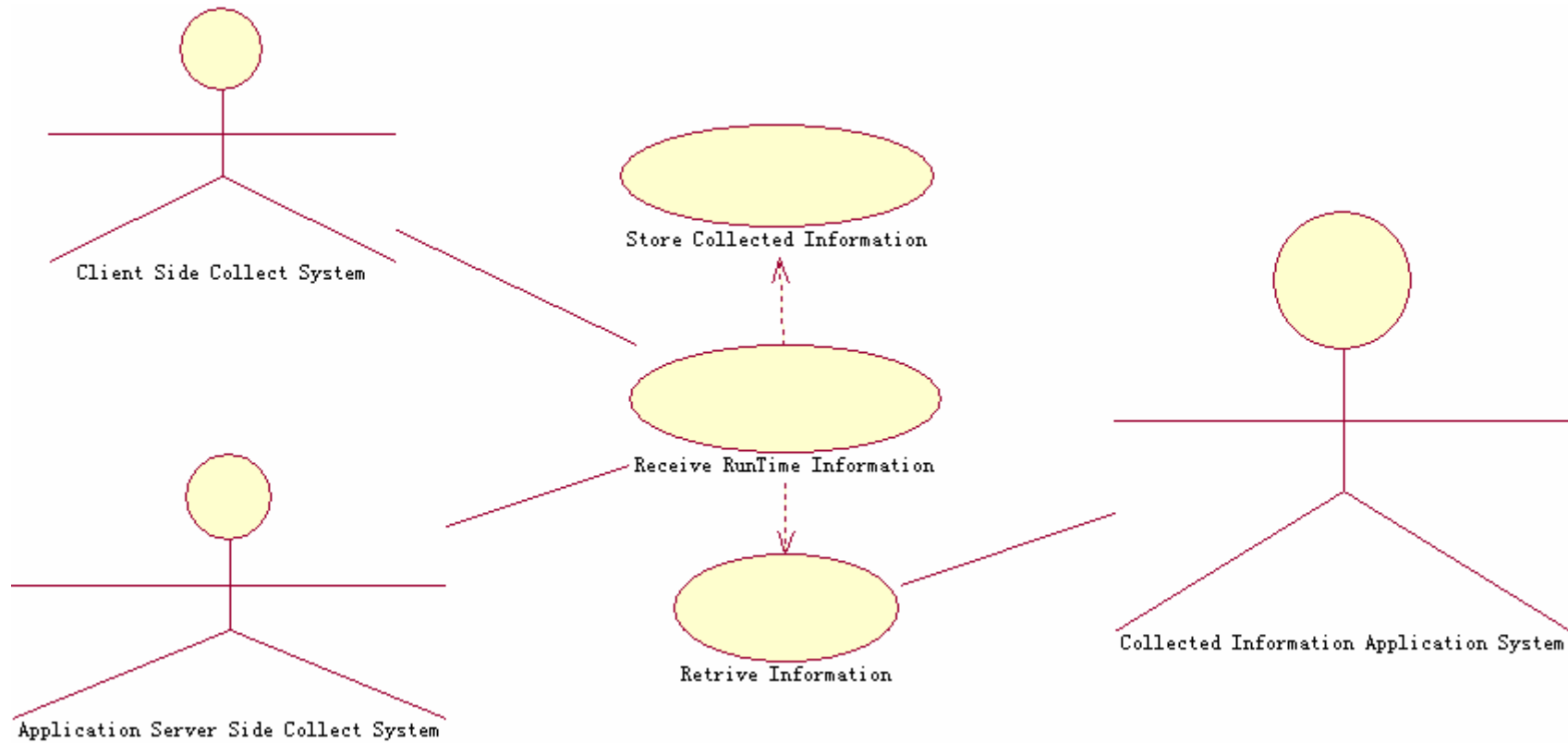


Objects of this program

- Monitoring all the activities of a Web Service's invocation
 - client side monitoring
 - Server side monitoring
- Make it easy to extend the monitoring
- Make it to reduce performance little
- Make it easy to use the collected runtime information

The Usecase of this system

C5



Diapositive 9

C5

去掉

Cheng; 16/09/2009

The Runtime Information Data format

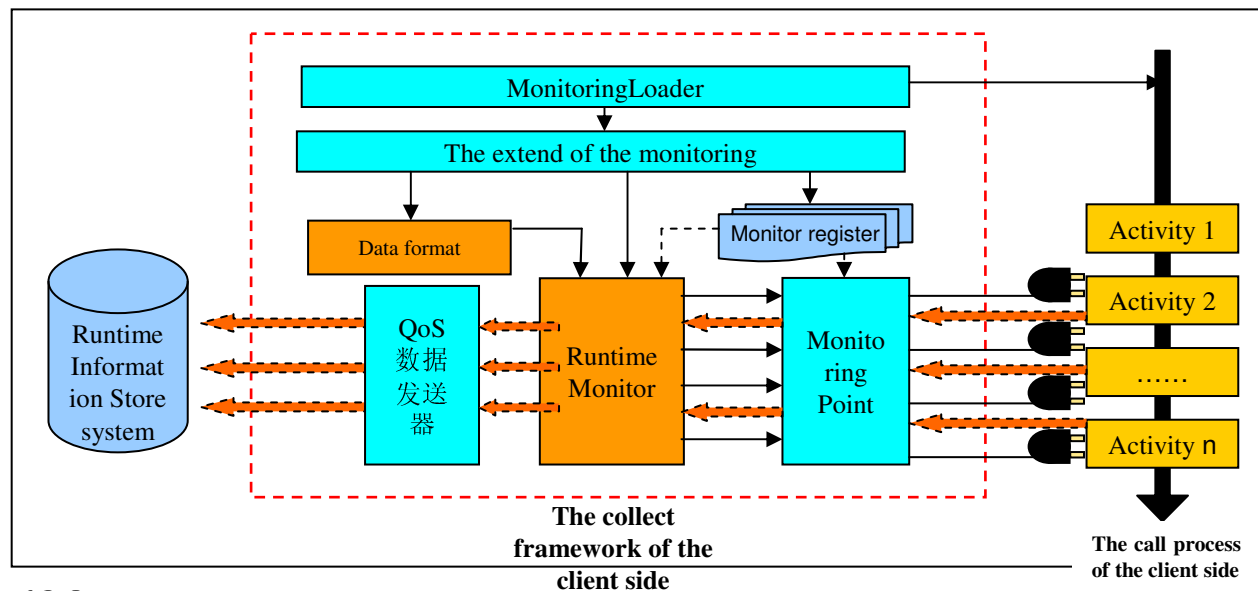


- **The basic data properties**
 - IP the IP of the collect side
 - SOURCE CLIENT side, SERVER side.....
 - SERVICE_URL endpoint of web service
 - OPERATOIN the invocation operation of the web service
 - PROXY the proxy address(if exists)
 - START_TIME request begin time
 - END_TIME response end time
 - REQUEST request soap message
 - EXCEPTION if exception occurred
- **The extended data properties**
 - EXTENSIBLE special application



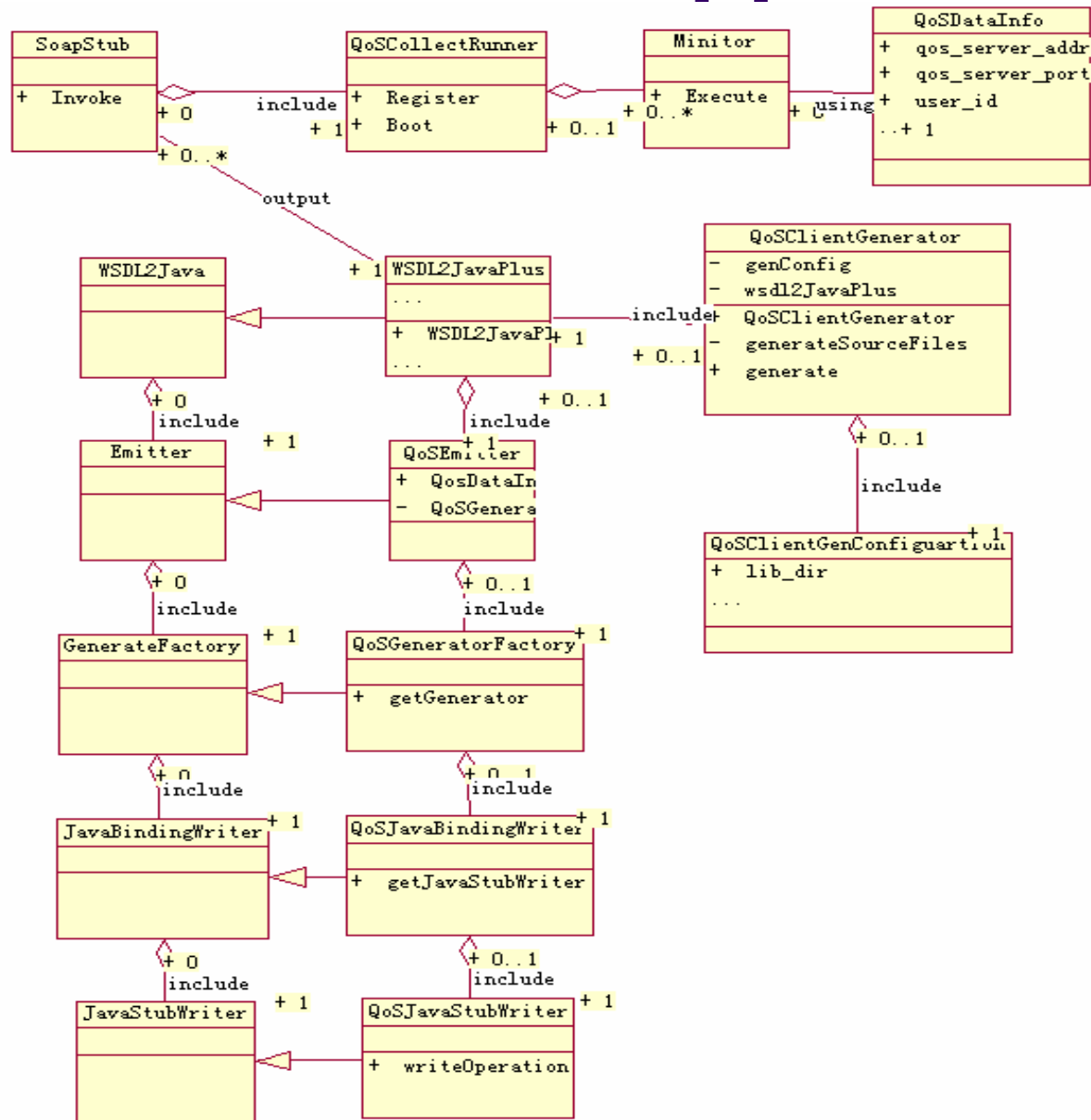
Client side collect approach

- Extending WSDL2JAVA class in Axis to monitor runtime activities



- Features
 - Using popular tool Axis
 - Reduce system performance little
 - Transparent to client users
 - Can monitoring a special operation of a web service

Client side collect approach



Client Side Collection approach



```
public void execute() {
    qosData.setProperty("serviceName",
runtimeContext.getProperty("serviceName"));
;
    qosData.setProperty("operationName",
runtimeContext.getProperty("operationName"
));
    qosData.setProperty("startTime",
System.currentTimeMillis());
}
```

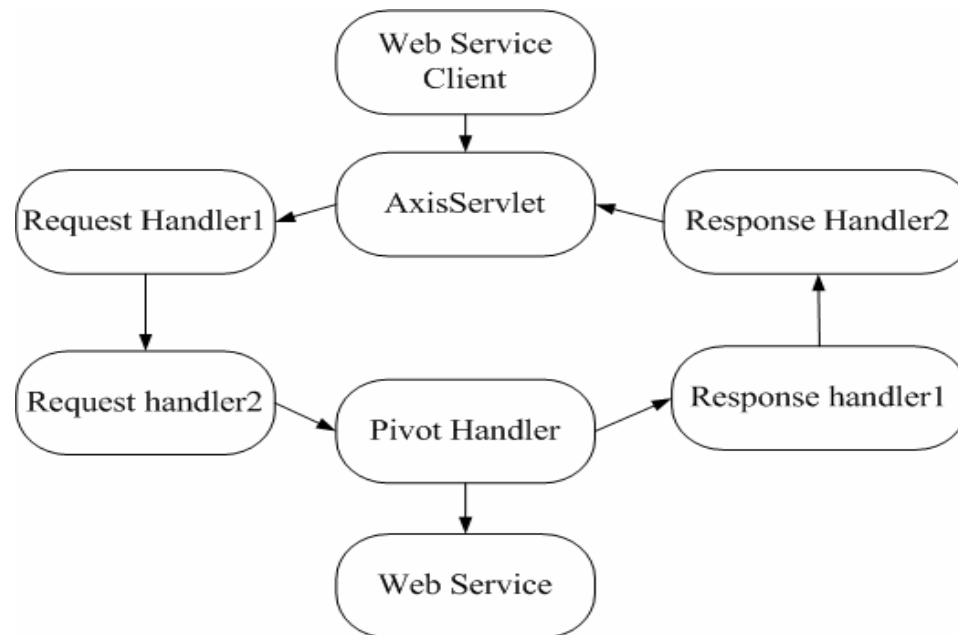


```
public void writeOperation(PrintWriter pw,
BindingOperation operation, Parameters
parms, String soapAction, String opStyle,
    boolean oneway, int opIndex) {
    ...
    pw.write("qosCollectRunner.boot (BeforeInvokeActivity.class, serviceId,
operation.getName());");
    pw.print("java.lang.Object _resp =
_call.invoke(");
    ...
    pw.write("qosCollectRunner.boot (AfterInvokeActivity.class, serviceId,
peration.getName());");
    ...
}
```

Web Service server side collect approach

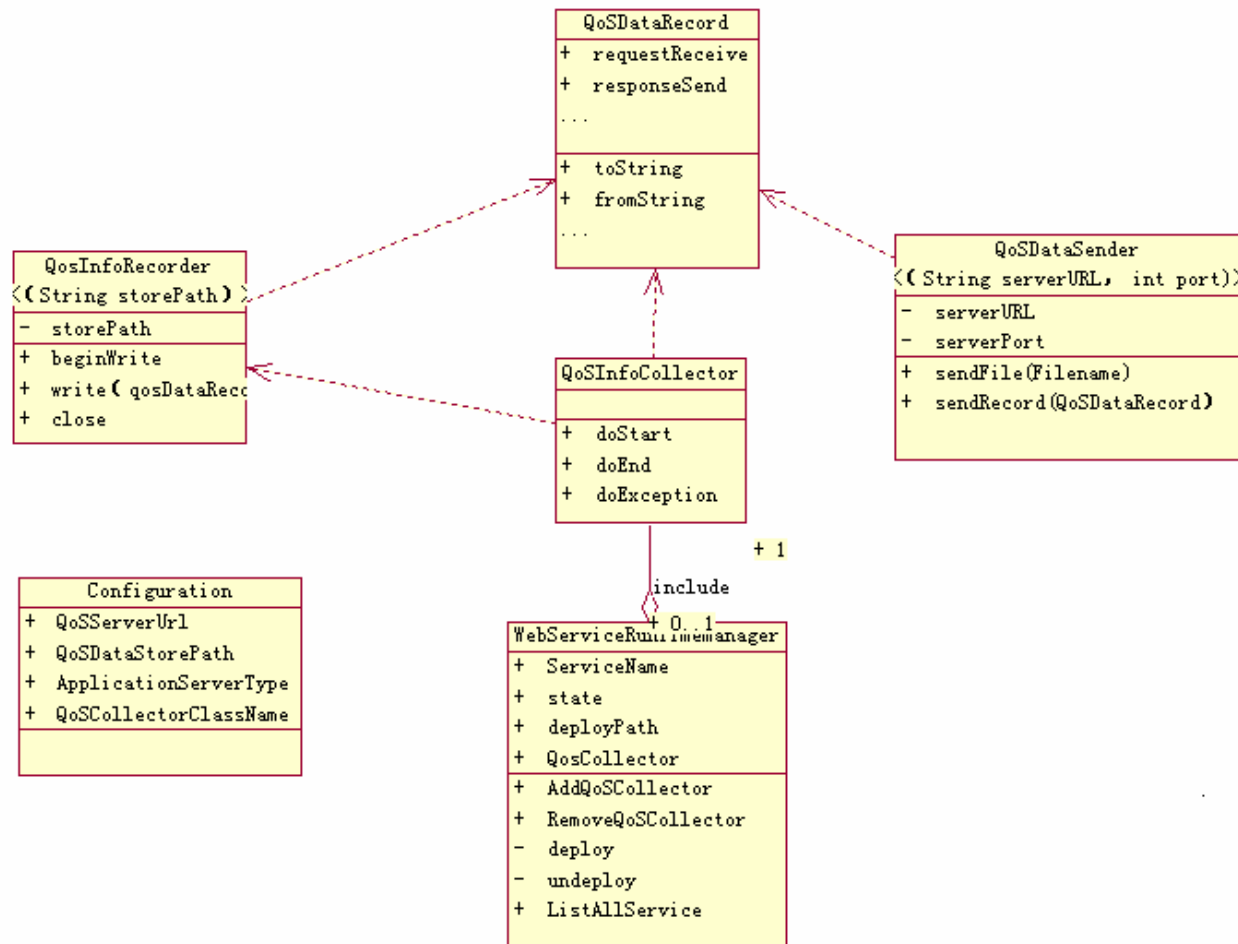


- Approach: Handler-Based runtime information collect

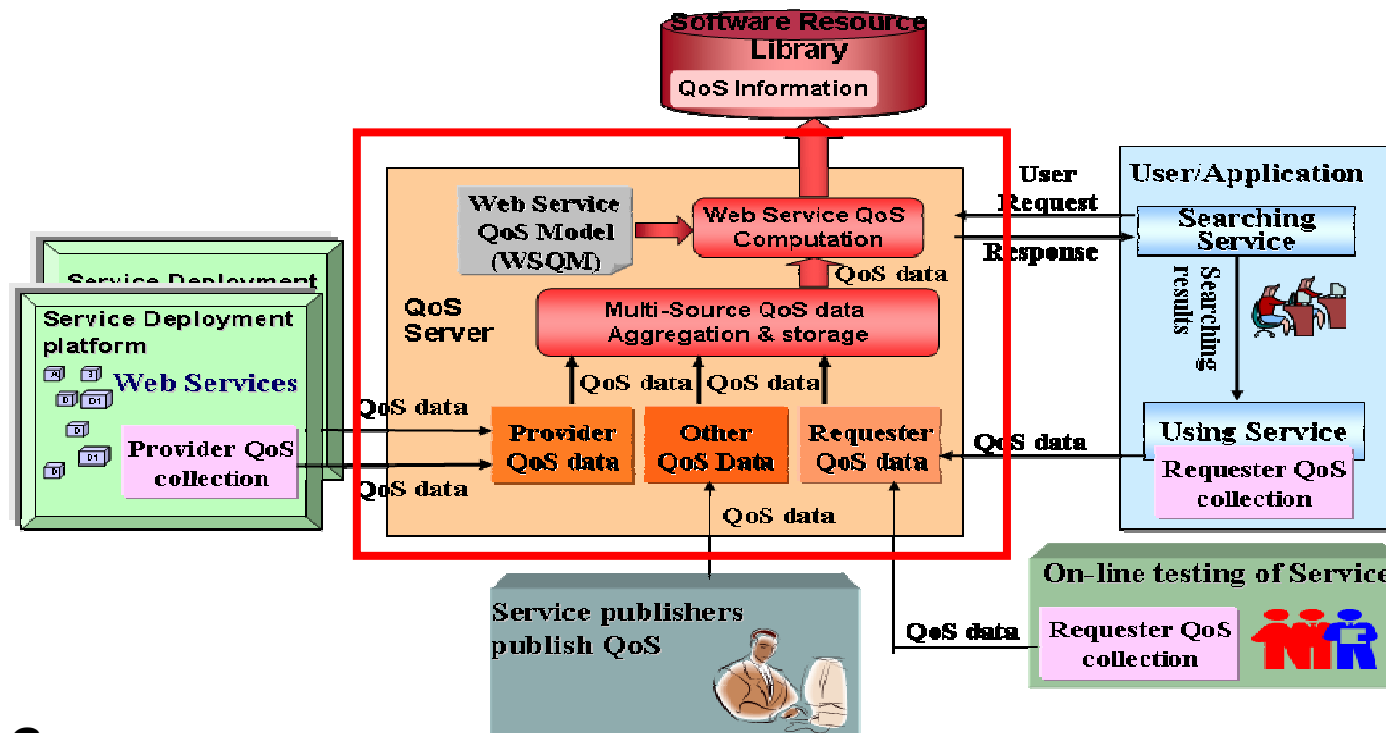


- Features
 - based on JAX-RPC standard
 - Easy to deploy
 - Not obviously cause performance down

Web Service server side collect approach



Approach of Information Storage Server



QoS Server

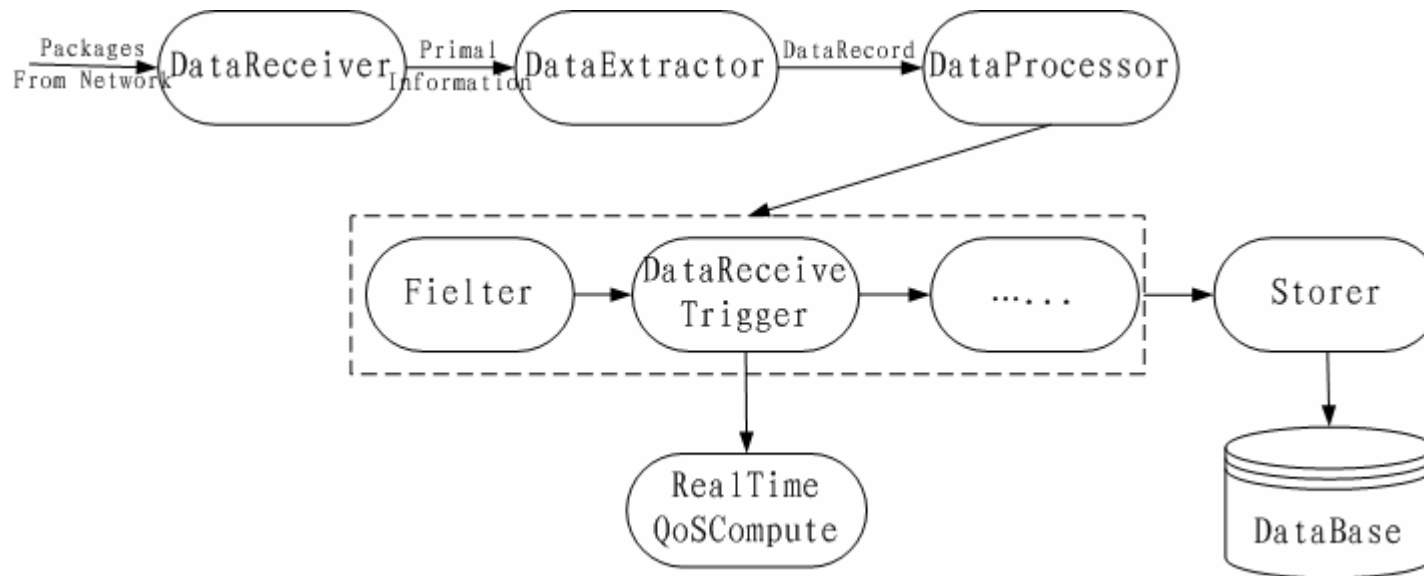
- Receive QoS data collected by all collection sources
- Estimate on QoS level according to specific QoS model
- Provide an efficient data query interface
- Provide an event-based mechanism to support scalable QoS computation

Information Storage Server implementation Technology



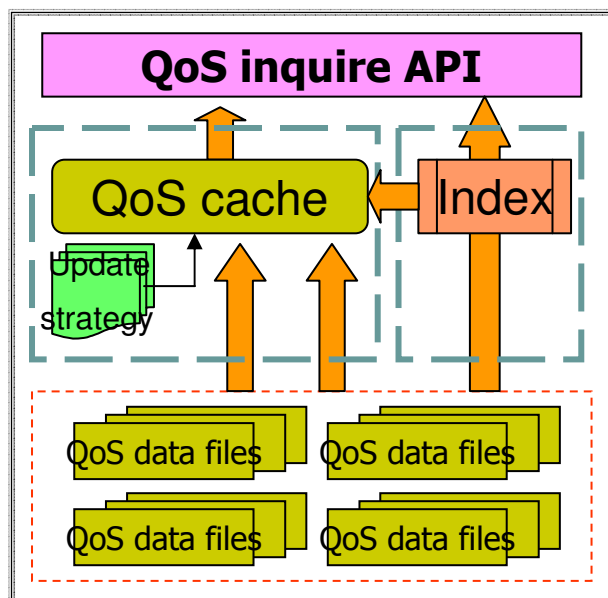
- Java NIO supports direct operation to cache and increase efficiency by realizing non-blocking I/O.
- QoS information gathering uses **XSocket** (open source framework based on NIO) to realize a high throughput receiving interface.

Information Storage



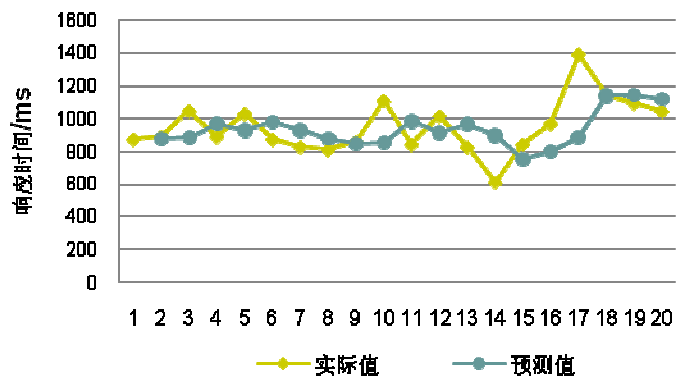


Information Retrieve

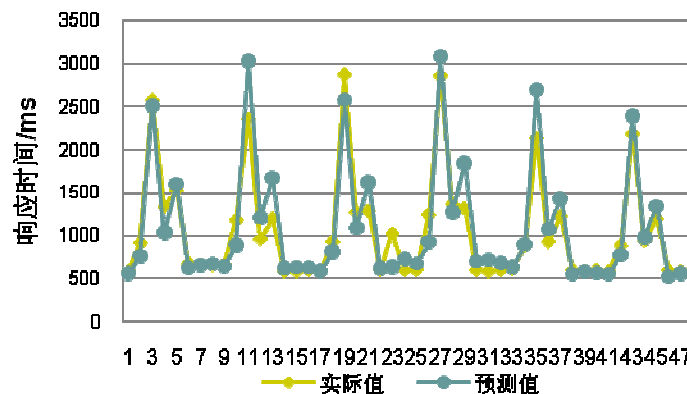


- Organize and store raw QoS data, provide an efficient data query interface
- Use file storage as QoS data storage medium
- Two kinds of index: index based on time and index based on service name
- Buffer mechanism : index and QoS data

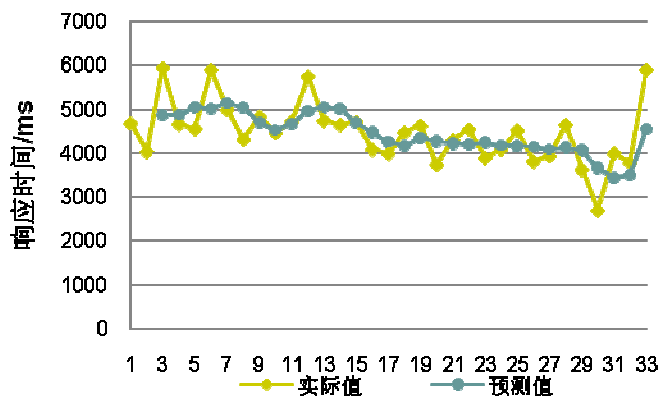
Application



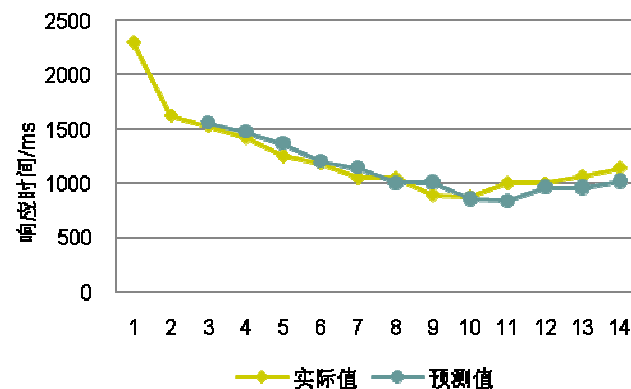
平稳模式 Amazon service 2008年5月13日-14日
采样频率:每隔1小时进行自动发起对该服务的访问



稳定波动模式 Google search service 2009年5月5日-5月7日
采样频率:每隔30分钟进行自动发起对该服务的访问



随机模式 ISBN service 2008年5月13日-5月16日
采样频率:每隔1小时进行自动发起对该服务的访问



趋势模式 Google search service 2009年5月5日21点-24点
采样频率:每隔15分钟进行自动发起对该服务的访问



Thanks !

Diapositive 21

C6

演示跟截图

Cheng; 16/09/2009