



# Search engine software management

by

Yang Guang, Zhu Lida

Supervisor: Morten Goodwin Olsen

## Project report for Distributed System in Spring 2007

based on report template version 3.0 (2006)

Agder University College  
Faculty of Engineering and Science

Grimstad, 13th May 2007

Status: Final

**Keywords:** Accessibility,Evaluation,Crawler,Monitoring,indicators

### Abstract:

*The goal of this project is to explore ways to monitor the production both for tuning the performance and for stabilizing the production. The main tasks to be solved include survey of the related literature, identification of a set of relevant indicators probably based on a set server logs both from the operating system and from the application, some automatic analysis of the log context, outline of a prototype cockpit for driving the crawler.*

This work is licensed under the Creative Commons Attribution-ShareAlike License (<http://creativecommons.org/licenses/by-sa/2.5/>).

## Version Control

<b>Version<sup>1</sup></b>	<b>Status<sup>2</sup></b>	<b>Date<sup>3</sup></b>	<b>Change<sup>4</sup></b>	<b>Author<sup>5</sup></b>
0.1	draft	2007-05-10	Introduction,Problem, background	Zhu Lida
0.2	draft	2007-05-10	Solution,discussion, conclusion	Yang Guang
0.3	draft	2007-05-11	Solution, discussion	Yang Guang
0.4	draft	2001-05-12	Introduction,Problem, Background	Zhu Lida

---

**1 Version** indicates the version number starting at 0.1 for the first draft and 1.0 for the first review version.

**2 Status** is DRAFT, REVIEW or FINAL

**3 Date** is given in ISO format: yyyy-mm-dd

**4 Change** describes the changes carried out since the previous version

**5 Author** is the one who did the change

## Table of Contents

<u>1 Introduction.....</u>	<u>4</u>
1.1 Report outline.....	4
<u>2 Problem description.....</u>	<u>5</u>
<u>3 Background.....</u>	<u>7</u>
3.1 Other monitoring tools_McAfee.....	7
3.2 RPC(Remote procedure call).....	8
3.3 XML(Extensible Mark-up Language).....	8
3.4 Python .....	9
3.5 parser.....	9
3.6 Web monitor.....	10
3.7 Crawler.....	10
3.8 ETLs(Extract, transform, and load).....	11
3.9 Linux shell.....	11
<u>4 Solution.....</u>	<u>12</u>
4.1 Requirements .....	12
4.2 Design Specification .....	13
4.3 Implementation.....	15
4.4 Validation and Testing.....	16
<u>5 Discussion.....</u>	<u>18</u>
5.1 Flexible:.....	18
5.2 Compare with McAfee siteAdvisor.....	19
5.3 Further Implementation:.....	19
<u>6 Conclusion.....</u>	<u>20</u>
<u>7 Appendices.....</u>	<u>21</u>

# 1 Introduction

Software used for monitoring the web, such as EIAO observatory is becoming common. In order to make sure the quality of software, we have to monitor and evaluate the security and stability of software as long as we can. Mission critical software systems need to be monitored in different ways to secure availability and stable production. As long as the world wide web is growing. The amount of information available reaches new heights every day. Mission critical systems not only have to deal with the software, but also have to evaluate the website in the network.

The object of this pilot report is to find out different ways to monitor the observatory and to present a working plan on how we are going to work, creating a simple monitor, that can shows the information of the server in details.

## 1.1 Report outline

The rest of this report is organized as follows: In chapter 2 we present the problem description followed by a review of background in Chapter 3. There we research relevant existing literature about the technologies we think might be useful to extend and make use of in our main report. In chapter 4, we outline the solution for our main project, and we also explain how we design and implement the project. Chapter 5 includes what we have discussed in this project. And at last is the whole conclusion of the report in Chapter 6. In addition, we keep the references of this pilot report in Chapter 7.

## 2 Problem description

The goal of this project is to explore ways to monitor the production both for tuning the performance and for stabilizing the production. The basic idea is we need to create a monitor to search the resource of the website. Such as the number of crawlers, ETLs(Extract Transform Load) and server. And then we can evaluate the website according all the information we have searched.

A web crawler (also known as a Web spider or Web robot) is a program or automated script which browses the World Wide Web in a methodical, automated manner. Other less frequently used names for Web crawlers are ants, automatic indexers, bots, and worms [1]. If there are a lot of crawler in the website, it will give CPU too much work to do and lead to lower stability of the website. On the other hand, if there is very little crawler, when we want open another link, it will slow down the speed to reach further website. So when we want to know the stability of the website, the number of crawler is a necessary element to test. So we also have two elements to test as the type of server and information in ETLs.

Not only crawler is a very necessary element to test, but also ETL is a very important one. ETL (Extract, transform, and load ) is a process in data warehousing that involves

- extracting data from outside sources,
- transforming it to fit business needs, and ultimately
- loading it into the data warehouse.[3]

The reason why ETL is very important to test is because when ETL load into the data warehouse, it should be only one ETL loading. Because if there are two ETL loading the data, and both ETLs have the same numbers of record of data, it will collide with each other, and unstable the web server. So it's also necessary to test the number of ETL. When we have one ETL or none, it's fine. When we have two or even more ETLs, it will get warning.

The problem now is to find a way to make the monitor to test the crawler, ETLs and server of the website. We build a environment as distributed system. We put the monitor in the server part, and the test website in the client part. So we can test the website in using RPC (Remote procedure call). So we can test website distributed.

It is possible to expand the monitor with features such as the number of downloads and evaluate the safety of the website. We will look into these features, but they are not part of our main requirements. For further information on our requirements, see chapter 4.

### 3 Background

We list the element of testing in XML, and parser it in using Python.

#### 3.1 Other monitoring tools

##### 3.1.1 McAfee SiteAdvisor

There is a example of other monitoring tools\_McAfee SiteAdvisor. It can test web, and help the user to resist spy-software, garbage email, virus and on-line scams. It very easy for the user to handle. [14]The user just need to type in the website address and start test. Then the website will list the result of safety testing and some informations in details.

It is a good monitor tools, but it is different from the intention of our tool. It monitors a website. But our intention is to monitor the EIAO Observatory. That means we are more concern the stability and accessibility of a server but not the numbers of safety downloads or test whether has spy-software or something else.

The figures below shows how the McAfee SiteAdvisor works.

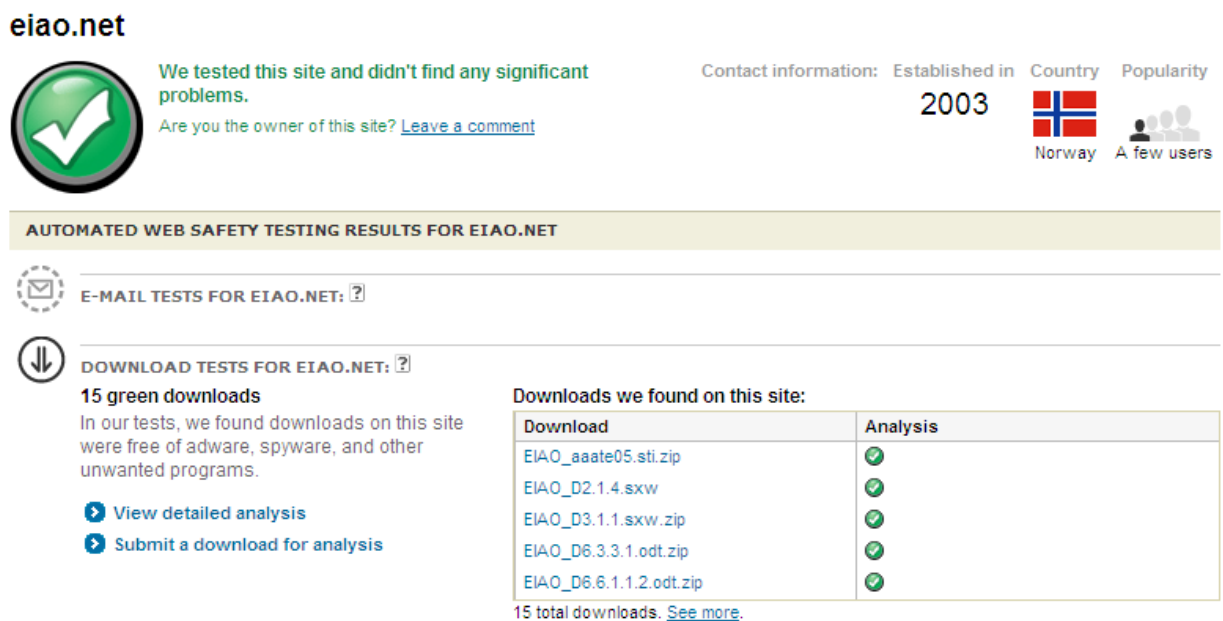


Figure 3.1 evaluate EIAO

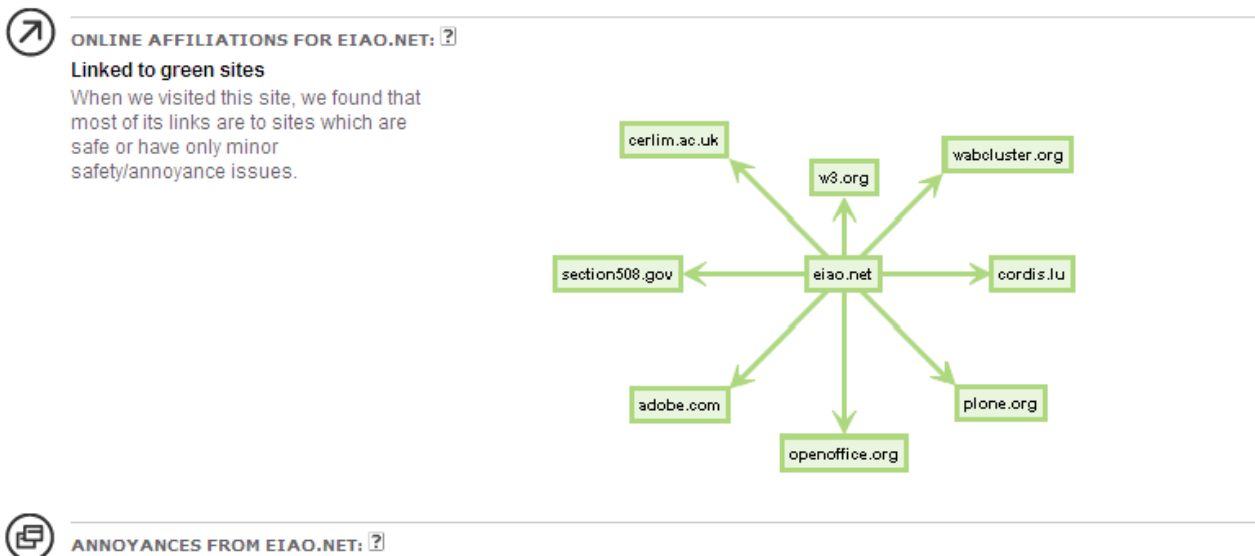


Figure 3.2 EIAO structure [15]

### 3.1.2 ServersCheck Monitoring Software

ServersCheck Monitoring is a tool running on Windows based systems for monitoring, reporting and alerting on network, servers and other IT systems availability. In addition to monitoring regular network devices, the program can also monitor environmental devices like temperature, humidity, flooding. ServersCheck Monitoring Software runs as a local service and is administrated via a browser based interface. Additional features include alerts and graph output for long-term statistics tracking.[17]

It can test over 60 different check types. And when an error is detected, the software can alert users using multiple option: email, SMS (text messages) and much more. The software is accessed using a browser or a mobile device (phone, PDA). It is very powerful. But what we want is to monitor the EIAO Observatory, we can monitor in a Linux Operate System, but should be also work in a Windows based systems in the future.

Here is how ServersCheck Monitor works:

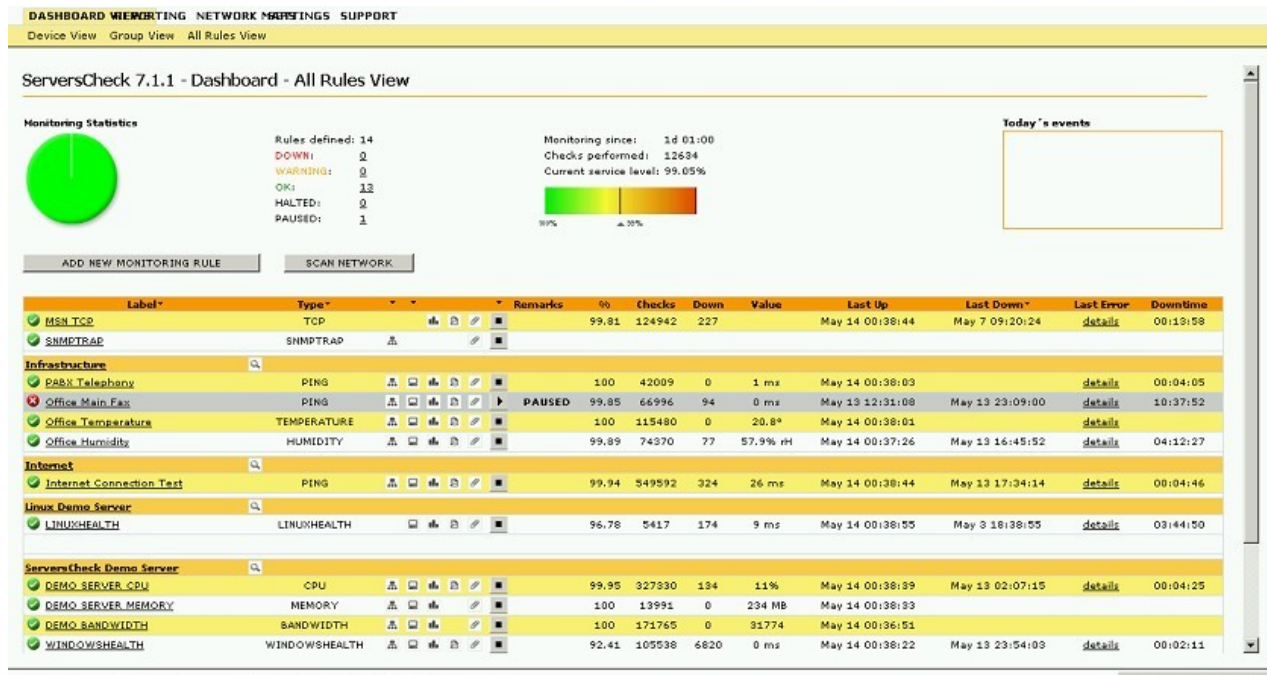


Figure 3.3 ServersCheck Monitor[18]

### 3.2 RPC(Remote procedure call)

RPC(Remote procedure call) is a technology that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction. That is, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question is written using object-oriented principles, RPC may be referred to as remote invocation or remote method invocation.[5]

We use this technology to implement the monitor when it monitoring the observatory. The monitoring part will be in the server, and the client can use the monitor from the server to test the information in the web.

### 3.3 XML(Extensible Mark-up Language)

The XML (Extensible Mark-up Language ) is a general-purpose mark-up language.[6] Its primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet.[7]

XML is recommended by the World Wide Web Consortium. It is a fee-free open standard. The W3C recommendation specifies both the lexical grammar, and the requirements for parsing.[8]

We use it for mark up all the element we test in the monitor.

### **3.4 Python**

Python is a high-level programming language first released by Guido van Rossum in 1991.[9] Python is designed around a philosophy which emphasizes the importance of programmer effort over computer effort, and it rejects more arcane language features, prioritizing readability over speed or expressiveness.[10] Python is often characterized as minimalist, although this only applies to the core language's syntax and semantics; the standard library provides the language with a large number of additional libraries and extensions.[11]

Python is a multi-paradigm programming language which has a fully dynamic type system and uses automatic memory management; it is thus similar to Perl, Ruby, Scheme, Smalltalk, and Tcl.[11]

The language has an open, community-based development model managed by the non-profit Python Software Foundation. While various parts of the language have formal specifications and standards, the language as a whole is not formally specified. The de facto standard for the language is the CPython implementation.[11]

The reason why we choose Python is because we use the language in Linux shell and it can be directly called by Python and runs in Linux. And also Python is a high level human language for machine. It's very easy to handle. So we decide to choose Python to parser the XML document.

### **3.5 parser**

In computer science and linguistics, parsing (more formally syntactical analysis) is the process of analysing a sequence of tokens to determine its grammatical structure with respect to a given formal grammar. A parser is the component of a compiler that carries out this task.[12]

Parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input. Lexical analysis creates tokens from a sequence of input characters and it is these tokens that are processed by a parser to build a data structure such as parse tree or abstract syntax trees.[12]

We want to use Python to parser the XML language and then get what we want in the monitor of web.

### **3.6 Web monitor**

Website monitoring is the process of testing or tracking (monitoring) how end-users interact with a website or web application. Website monitoring is often used by businesses to ensure that their customers are able to access their on-line applications and perform actions such as searching, on-line shopping, checking an account balance, or simply researching.[13]

### **3.7 Crawler**

As we have discuss above, web crawler is a program or automated script which browses the World Wide Web in a methodical, automated manner.[1].

This process is called Web crawling or spidering. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for spam)[2].

A Web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the

page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies[2].

We don't use it. What we want to know is the number of crawler in the server. That is what we need to monitor in the observatory.

### **3.8 ETLs(Extract, transform, and load)**

As discussion before, ETL (Extract, transform, and load ) is a process in data warehousing that involves, extracting data from outside sources, transforming it to fit business needs, and ultimately loading it into the data warehouse.[3]

ETL is important, as it is the way data actually gets loaded into the warehouse. This article assumes that data is always loaded into a data warehouse, whereas the term ETL can in fact refer to a process that loads any database.[3]

We test the ETLs as web crawlers as we have discussed in the problem distribution part.

### **3.9 Linux shell**

The shell was designed as a replacement for the original Thompson shell.[4]

Although it is used as an interactive command interpreter, it was always intended as a scripting language. It gained popularity with the publication of The UNIX Programming Environment by Brian W. Kernighan and Rob Pike. This was the first commercially published book that presented the shell as a programming language in a tutorial form.[4] We use some functions of it to search the information when monitoring the observatory such as the number of crawlers, ETLs and so on.

## 4 Solution

### 4.1 Requirements

In this project,websites monitor could be used under Linux Operation System. Considered that use Unix shell commends in Python programming is practicable and efficient.

Local website monitor:

On local host, the application integrated serial of certain criteria to observe situation locally.

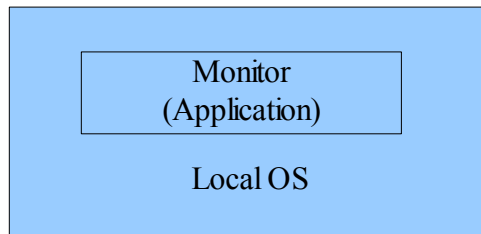


Figure 4.1 Local websites monitor

Remote websites monitor:

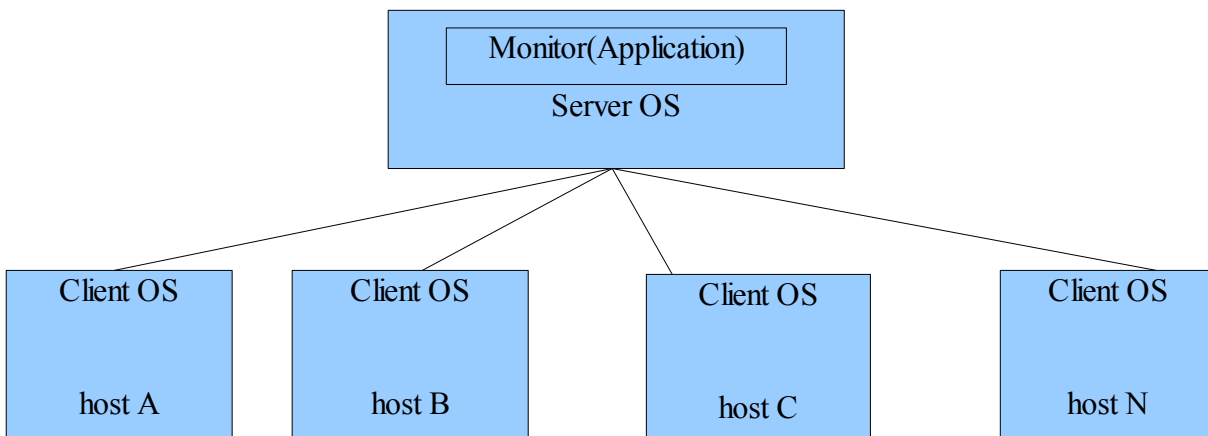


Figure 4.2 Remote websites monitor

Configure the monitor application in web server's operation system, clients call the Monitor Application on server. The execution of the called procedure takes place on server. Information of websites which client want to observe can be transported from the caller to the callee in the parameters and can come back in the procedure result. This is typical way to use Remote Procedure Call to carry out the websites monitor.

As the observatory runs monitors, there are some criteria are used in it to test how it works.

Such as: number of crawler ,number of ETLs,number of sites finished,number of ETLs run ,last time written to log file. Apparently, the application can be expended.

### 4.2 Design Specification

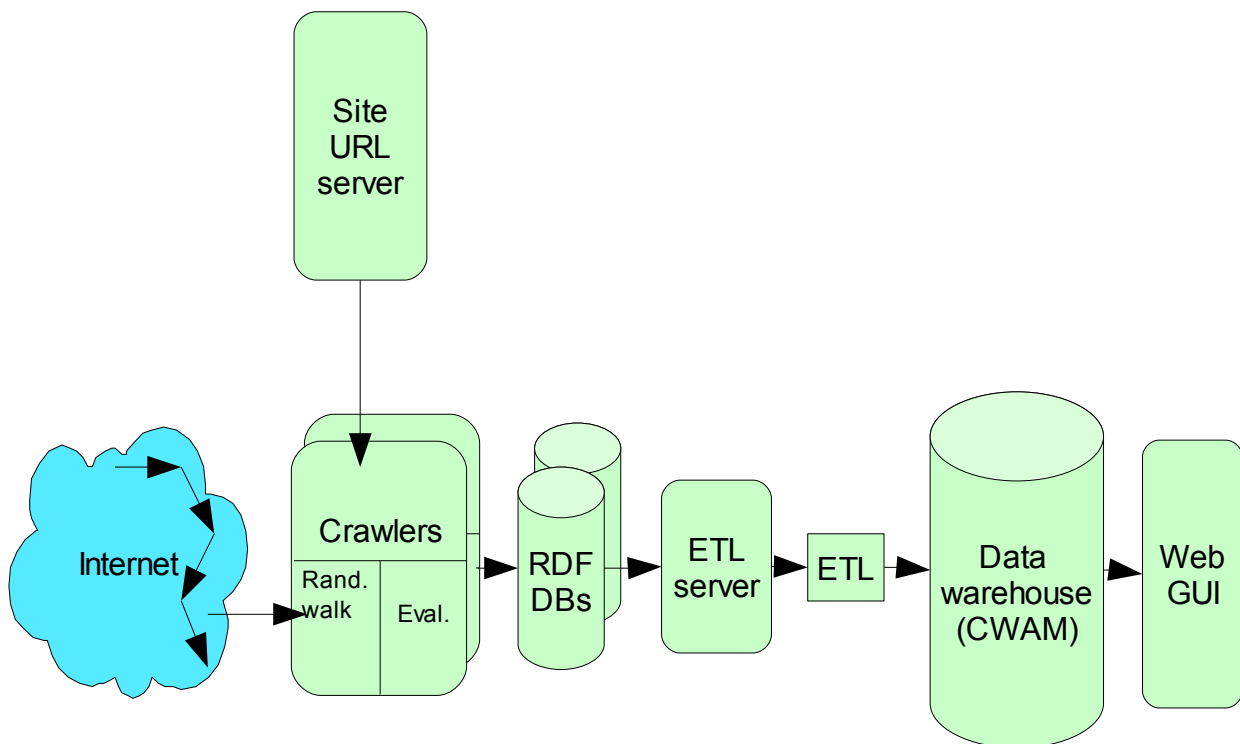


Figure 4.3 Architecture of web works[16]

When it comes to monitoring the observatory, there are many different identifications , which depends on what is aim of monitor. Some monitors aim on security part, some focus on monitor on data transport.

In this project,main purpose is to monitor observatory, which used for further evaluate accessibility of websites. That means,the monitor in this project should be focus on what show in this architecture.

There are lots of useful commands in Linux operator system can be used to observe information of websites:

Number of Crawlers running:

```
ps aux | grep -v grep | grep -v tail |grep -c crawler
```

Get information of apache server(local):

```
ps aux | grep -v grep | grep -v tail |grep -c apache
```

Number of ETLs;

```
ps aux | grep -v grep | grep -c dw20
```

Number of sites finished;

```
grep -c Finished crawler.log
```

Number of ETLs run;

In this mission, in order to achieve those functions which mentioned above, compose commands which can certainly get informations of websites and also other value such as number to limit a certain range of amount in XML elements. Using Python programming to parse it. That means, the application combine with one XML document and one Python parser.

XML documents are hierarchical and structured. Because of it is well-formed, it is a generic framework for storing any amount of text or any data whose structure can be represented as a tree structure. XML can also be used to represent many types of structured data and for transmitting data across a network. It is essential the monitor flexible. When different criteria come out, construct them follow the previous structure.

Python can get import of Linux Operation System command and easily to execute the commend in Python and return results. The application also compare those results with maximum value and minimum value to test whether the results are in limited range. If it is, sent out 'right' message. If it out of the limited range, send out 'warning' message.

### 4.3 Implementation

XML's structure can keep the contents the same. It makes programming simpler by this way. And it makes further extend-development flexible. In this project, the XML document contain different criteria,for example number of crawler:

```
<crawler>
  <query> ps aux | grep -c crawler </query>
  <text> Number of Crawler: </text>
  <minimum>5</minimum>
  <maximum>20</maximum>
</crawler>
```

This is one segment of whole XML document. Four elements hold different contents:

query: Unix shell command to count number of crawler,Python parser would run it and carry out result

text:Python parser can get this text and then print it out

minimum: limit minimum of amount

maximum: limit maximum of amount

When the application gets this limit range of amount, it can compare with what gets from 'query', and then send message about whether this websites has suitable amount of crawler.

This is XML parser:

```
def handle(self,node):
    for child in node.childNodes:           #search every nodes
        if child.childNodes:
            for c in child.childNodes:
                if str(c.nodeName)=='query':
                    queryres = self.query(c)
                if str(c.nodeName)=='text':
                    self.text(c)
                if str(c.nodeName)=='minimum':
                    mini = int(self.mini(c))
                if str(c.nodeName)=='maxmum':
                    maxi = int(self.maxi(c))
```

```

        self.test(child,maxi,mini,queryres)
def query(self,node):                #this method is used to parse 'query'
    commend=node.childNodes[0].nodeValue
    result=os.system(commend)        #execute command which is gotten from XML
'query'
    return result
def text(self,node):                #this method is used to parse 'text'
    print node.childNodes[0].nodeValue    #print out text in 'text'
def maxi(self,node):                #this method is used to parse 'max'
    maxi=node.childNodes[0].nodeValue
    return maxi                    #get value of max for further compare
def mini(self,node):                #this method is used to parse 'mini'
    mini=node.childNodes[0].nodeValue    #get value of mini for further compare
    return mini

def test(self,node,maxi,mini,queryres):    #test part
    if mini<queryres<queryres:
        print "OK"
    else: print "NO"
        etls=os.listdir('.')            #print out all ETLs
doc=minidom.parse('input1.xml')
monitor(doc)

```

Other criteria work in the same way, Python parse programming take advantage of XML document because of it store content in the same structure. What is only need to do , just search every element in XML document and then get all contents.

#### 4.4 Validation and Testing

As we explained in Problem Description part, monitoring crawlers and ETLs is reasonable. Number of crawler is effect on rate of progress of opening a website and only one ETL can guarantee avoidance of collision.

The tests have been run on different machine with different intentions. For example,on local machine, we prefer testing Apache server to crawlers.

On local machine,It is clearly show out that the number of Apache instances , but number of crawlers is zero.

result show out as follows:

Content-type: text/html

2

Number of Crawlers:

0

OK

```
root  4587 0.0 0.4 4644 2112 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4592 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4593 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4594 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4595 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4596 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
root  4610 0.0 0.8 10080 4216 ?   Ss 11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4617 0.0 0.3 9064 1880 ?    S  11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4685 0.0 0.6 231416 3208 ?  Sl 11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4687 0.0 0.6 231416 3212 ?  Sl 11:18  0:00 /usr/sbin/apache2 -k start -DSSL
shane  8978 0.0 0.0 1656 464 pts/0  S+ 13:36  0:00 sh -c ps aux | grep apache
shane  8980 0.0 0.1 2800 768 pts/0  S+ 13:36  0:00 grep apache
```

web server:

256

NO

Number of ETLs:

1

OK

last time wrote to the crawler log:

2007-05-06 02:06:10.000000000 +0200

On EIAO server, it can be carry out number of currently running crawlers. Beside of this,because of this application flexible enough, we can get information of web server which works on the test machine. Coming to explain this in Discussion part.

## 5 Discussion

In test, it works out exact what expect. Actually, every Linux command can works in same way, the application just integrate all of them.

It works well in local machine, and it can be transmitted to a remote machine can works and take that machine locally carry out result. By this way, it can be transmitted to remote web server, it actually works in test but we have done too much of that.

### 5.1 Flexible:

one of most import advantage in this programming is its expandable:

when there is coming something new found we are interesting, what is needed to do ,is construct a new element in XML document. Keep every elements in this element in the same structure as previous elements, but different contents.

for example, add as follows to look into situation of Zope:

```
<webservice2>
  <query>ps aux | grep zope</query>
  <text>web server in zope:</text>
  <minimum>2</minimum>
  <maxmum>5</maxmum>
</webservice2>
```

run it get information of Zope

Content-type: text/html

2

Number of Crawlers:

2

OK

```
root  4587 0.0 0.4 4644 2112 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4592 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4593 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4594 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4595 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
www-data 4596 0.0 0.1 4644 948 ?    S  11:18  0:00 /usr/sbin/apache
```

```

root    4610  0.0  0.8 10080 4216 ?    Ss  11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4617  0.0  0.3  9064 1880 ?    S   11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4685  0.0  0.6 231416 3208 ?   Sl  11:18  0:00 /usr/sbin/apache2 -k start -DSSL
www-data 4687  0.0  0.6 231416 3212 ?   Sl  11:18  0:00 /usr/sbin/apache2 -k start -DSSL
shane   5742  0.0  0.0  1660  464 pts/0  S+  11:41  0:00 sh -c ps aux | grep apache
shane   5744  0.0  0.1  2804  768 pts/0  S+  11:41  0:00 grep apache
web server:
shane   5745  0.0  0.0  1660  468 pts/0  S+  11:41  0:00 sh -c ps aux | grep zope
shane   5747  0.0  0.1  2800  756 pts/0  S+  11:41  0:00 grep zope

```

Number of ETLs:

1

OK

last time wrote to the crawler log:

2007-05-06 02:06:10.000000000 +0200

In this case, Zope server is added in to XML document as element which is needed to monitor. The application carry it out as show in result.

## 5.2 Compare with McAfee siteAdvisor

Compare with McAfee siteAdvisor, which is a on-line software warns users when downloading software or filling out form on a site may make them victims of spyware, spam, viruses and on-line scams.

Obviously, McAfee siteAdvisor focuses on normal web users who use browsers view websites and download from websites. There is automated web safety testing including: E-mail test, download test and on-line affiliation relationship figure. The websites monitor in this project focuses on web server administrators who need to know basic information of websites which they look into. And the most important thing is expansible of this application. It is flexible enough allows others add different parameter what there are interest to check out of websites.

## 5.3 Further Implementation:

1. create a websites to put it in graphics should be more convenient for users.
2. Try programming in Windows
3. monitoring over time, for example: check out how many crawlers have been running the last 24 hours.

## 6 Conclusion

Main aim of this project is to explore ways to monitor the websites both for tuning the performance and for stabilizing the production. The main tasks to be solved include survey of the related literature. Create an application automatically analysis of the log content, count out amount of crawlers and list out situation of web servers.

In this project, XML document and Python programming are used. Taking advantages of XML document can keep the contents in the same structure, in this way, the application can parse it flexible and efficient. Constructed Unix shell commands in XML directly and then run them out used Python. It works well on both local and remote machine. Because of its flexible and expendable,the way which used in this project can be used for further research in monitor field. It is also realizable to develop a web application in Windows Operation System in the same way, just need to adjust commands which are used in Windows.

## 7 Appendices

### 7.1 Glossary & Abbreviations

ETL--Extract, transform, and load

RPC-- Remote procedure call

XML--Extensible Mark-up Language

### 7.2 References

Follow the IEEE guidelines for references

see: [http://www.ece.uiuc.edu/pubs/ref\\_guides/ieee.html](http://www.ece.uiuc.edu/pubs/ref_guides/ieee.html)

- [1] [Kobayashi and Takeda, 2000](#)
- [2] [http://en.wikipedia.org/wiki/Web\\_crawler](http://en.wikipedia.org/wiki/Web_crawler)
- [3] [http://en.wikipedia.org/wiki/Extract%2C\\_transform%2C\\_load/](http://en.wikipedia.org/wiki/Extract%2C_transform%2C_load/)
- [4] [http://en.wikipedia.org/wiki/Bourne\\_shell/](http://en.wikipedia.org/wiki/Bourne_shell/)
- [5] [http://en.wikipedia.org/wiki/Remote\\_procedure\\_call/](http://en.wikipedia.org/wiki/Remote_procedure_call/)
- [6] [More precisely, XML is a general-purpose specification for creating custom markup languages.](#)
- [7] [Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau \(September 2006\). Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\) - Origin and Goals. World Wide Web Consortium. Retrieved on October 29, 2006.](#)
- [8] <http://en.wikipedia.org/wiki/XML>
- [9] [HISTORY. Python source distribution. Python Foundation. Retrieved on 2007-03-21.](#)
- [10] [What is Python? Executive Summary. Python documentation. Python Foundation. Retrieved on 2007-03-21.](#)
- [11] [http://en.wikipedia.org/wiki/Python\\_language](http://en.wikipedia.org/wiki/Python_language)
- [12] <http://en.wikipedia.org/wiki/Parser/>
- [13] [http://en.wikipedia.org/wiki/Website\\_monitoring](http://en.wikipedia.org/wiki/Website_monitoring)
- [14] <http://www.siteadvisor.com/analysis/>
- [15] <http://www.siteadvisor.com/sites/eiao.net>
- [16] [EIAO-review-p3-presentations-v0.2-num-mgo.odp](#)
- [17] <http://www.serverscheck.com/network-monitoring-tool.asp?Country=INTL&LANG=UK>
- [18] [http://en.wikipedia.org/wiki/Image:Serverscheck\\_screenshot.jpg](http://en.wikipedia.org/wiki/Image:Serverscheck_screenshot.jpg)

The references below are the materials we have read but didn't include in the report.

- [19] [HiA, Bruk av kilder i skriftlige arbeider ved Høgskolen i Agder, August 2006, <http://www.hia.no/stud/eksam/kilder.htm>](#)
- [20] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, Computing as a Discipline, Communications of the ACM, Volume 32, Number 1, 1989
- [21] <http://en.wikipedia.org/wiki/XML>
- [22] <http://zope.cn/>

- [23] <http://linux.byexamples.com/archives/242/monitor-custom-programs-with-ps-and-watch/>
- [24] [http://www.w3school.com.cn/xml/xml\\_parser.asp](http://www.w3school.com.cn/xml/xml_parser.asp)
- [25] [http://www.woodpecker.org.cn/diveintopython/xml\\_processing/index.html](http://www.woodpecker.org.cn/diveintopython/xml_processing/index.html)
- [26] <http://fred.webcan.cn/weblog/2006/12/>
- [27] <http://linux.byexamples.com/archives/242/monitor-custom-programs-with-ps-and-watch/>
- [28] <http://my.donews.com/shijun/>
- [29] <http://hi.baidu.com/zhounew>
- [30] <http://www.woodpecker.org.cn/diveintopython/>
- [31] <http://www.eiao.net/>
- [32] <http://www.siteadvisor.com/analysis/#>
- [33] <http://www.openfans.net/user/feed/woodstudio>
- [34] [http://man.chinaunix.net/develop/python/mod\\_python/mod\\_python.html](http://man.chinaunix.net/develop/python/mod_python/mod_python.html)
- [35] <http://www.ibm.com/developerworks/cn/linux/l-django/>
- [36] <http://www.ixpub.net/639973.html>
- [37] Linux administration handbook Nemeth, Evi Hein, Trent R Snyder, Garth
- [38] Moving to Ubuntu Linux Gagné, Marcel
- [39] Ubuntu :Linux for human beings
- [40] Networking for dummies Lowe, Doug
- [41] Python and XML Jones, Christopher A.Drake, Fred L.
- [42] Building embedded Linux systems Yaghmour, Karim
- [43] Web programming : techniques for integrating Python, Linux, Apache, and MySQL Thiruvathukal, George K.Shafae, John P.Christopher, Thomas W.
- [44] Programming Python Lutz, Mark
- [45] Apache cookbook Coar, Ken A. L.Bowen, Rich
- [46] Python Web programming Holden, Steve 1950 Beazley, David M.
- [47] XML processing with Python McGrath, Sean
- [48] Python standard library Lundh, Fredrik
- [49] Beginning XML databases Powell, Gavin
- [50] Beginning regular expressions Watt, Andrew
- [51] Web standards design guide Ruse, Kevin