Monitoring of the Web Servers

Date	Version	Author	Comments
5th April 2006	0.1	Julian Harty	New document.
10th April 2006	0.2	Julian Harty	Added more content
21st April 2006	0.3	Julian Harty	Made more generic and added a bit more content

Introduction

Here are a set of three related programs is used to monitor the performance web servers. This document describes the underlying concepts, the design and implementation, and contains details of how to configure and monitor the software in operation.

Overview

The web servers are configured to write log records for each requests received. One useful measure of performance is the time-taken by the web server to service each web request. Provided the web server can be configured to record this data, these log analysis programs should be able to process logs from these web servers. (This value is available in the extended W3C format logs created by Microsoft's IIS Web Server provided with Windows NT4, 2000, XP and 2003 products. Other web servers, including Apache, can record the time taken.)

Each server creates a daily file, and each record includes a consistent set of fields, including the date and time of the request, the time taken and details of which web page was requested. Early each morning the web server closes the

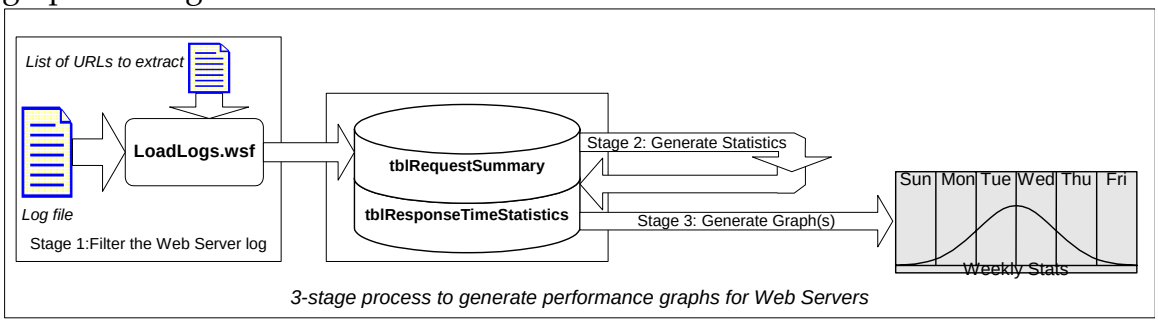
current log file and starts a fresh one. The filename for each day includes the year, month and day, which makes the file for that web server unique. Later that morning the now previous day's file is loaded by the first of 'our' programs.

The first program applies a filter to select a subset of entries from a complete log file, and writes the matching entries to a database. The program writes to a database with the same name as the web server e.g. for a web server called webserver1 the database needs to be called webserver1 as well. The database table for the entries, or records as they are called when stored in a database, is called tblRequestSummary, and contains all the records stored to date. The program is intended to run on the web server, which keeps the process scalable. This program relies on a free utility from Microsoft called LogParser, which needs to be installed on each server.

The second program is run once per web server and generates statistics for a given day's set of records. This program can run on the individual server or on a central server. If the program runs on the same server as the databases, the processing faster as no network calls are required. This program uses a copy of Excel to generate the statistics, so Excel needs to be installed wherever it runs (another reason why the second program is configured to run on the central server, as we only need a single licence for Excel, rather than one per server).

The third program is also run once per web server and generates a graph of the performance that week for the web server. It also uses Excel and runs on the central server, for the reasons outlined above.

The following diagram shows how the 3 stages work together to enable the graphs to be generated.



3-stage process to generate performance graphs for Web Servers

The programs all write summary records to the local computer's Application Event Log. See appendix C for examples of records, both when the program appears to work correctly, and when a failure is detected e.g. if the database is not available.

Detailed description of the 3 stages

The following section provides more information on how each of the 3 stages works. You may choose to skip this on first reading.

Stage 1: Load the log file into the database

The LoadLogs.wsf script file uses a list of URL entries (names of web pages) to select the entries to be written to the database. The names can contain wildcards, wildcards are specified using the syntax of Microsoft SQL Server, e.g. % represents zero or more characters, such as %company.asp will match /this/company.asp, /that/this/company.asp, or xxxxcompany.asp, but will not match /company.aspx

LoadLogs.wsf is a structured script file. It relies on Microsoft Scripting to run, and is written in 'Visual Basic Script' language. The program takes a number of optional parameters. A current list of options is displayed by the program, and by all the 3 programs, if you append /h to the end of the program (/h represents /help). Some sample invocations are provided later in this document.

The program can be called from a script file, or used interactively. It writes messages to the standard output (the screen / console, unless the output is redirected to a file).

The program expects a list of URLs, each on a separate line of the configuration file. The program searches for these URLs and writes the matching records to a database. Owing to limitations in the behaviour of the script, the connection parameters to the database need to be specified within the script, rather than being configured via a Database Source Name (DSN).

Stage 2: Generate statistics

GenerateIISStatistics.wsf script file performs the second stage of the 3-stage process. It reads records for a given date from one database table and writes statistical data to a second database table in the same database. It uses Microsoft Excel to perform the statistical processing. Owing to some limitations in early versions of Excel 2000, the program requires Excel 2000 to have service-packs dated 2004 and earlier to be applied to Excel (or Microsoft Office 2000). Later versions of Excel should also work correctly e.g. Excel 2003, however these have not yet been tested with the scripts.

For simplicity, GenerateIISStatistics.wsf only processes records for a single database, i.e. a single web server in our scenario. However by passing the server's name as one of the command-line parameters, the program can easily be run serially for each web server of interest.

Stage 3: Generate weekly graph

GenerateIISGraphs.wsf is the third and final stage in the process. It reads the statistical data produced by the second stage and generates a cumulative weekly graph for the week to date. The resulting graph is also stored as an image file in a directory specified by a command-line parameter. The image file can be displayed via a web server, or simply using standard Windows or Unix utilities e.g. Windows Explorer's Image Preview, or thumbnails are sufficient to display the graphs.

The program is configured to generate graphs for a single server.

The first day of the week is based on Excel's rules.

General comments

The three programs were originally developed as a set of prototypes, which were completed in January 2005. These were installed on a client site and remained in operation for 14 months without being checked. While the programs worked without any faults being reported they are still relatively immature and may benefit from improvements to their robustness and reliability. Changes to the programs are made from time to time. Generally these changes are backwards-compatible, and can be applied without too much effort once you have completed any acceptance testing ☺

Appendix A: Sample invocations

Test

```
cscript c:\statscripts\loadlogs.wsf /d:2006-03-01 /f:"D:\logs\iis\w3svc1\ex060301.log"
/l:c:\statscripts\urls.txt > c:livabe02-ex060301.txt

cscript generateiisstatistics.wsf /list:onyxurls.txt /s:livabe04 /d:2006-04-04

cscript generateiisgraphs.wsf /s:livabe02
/i:"e:\iismonitoringscripts\webserverreports" /list:onyxurls.txt
```

Test

Command-line syntax for each program

Command-line syntax for LoadLogs.wsf

TBD

Command-line syntax for GenerateIISStatistics.wsf

TBD

Command-line syntax for GenerateIISGraphs.wsf

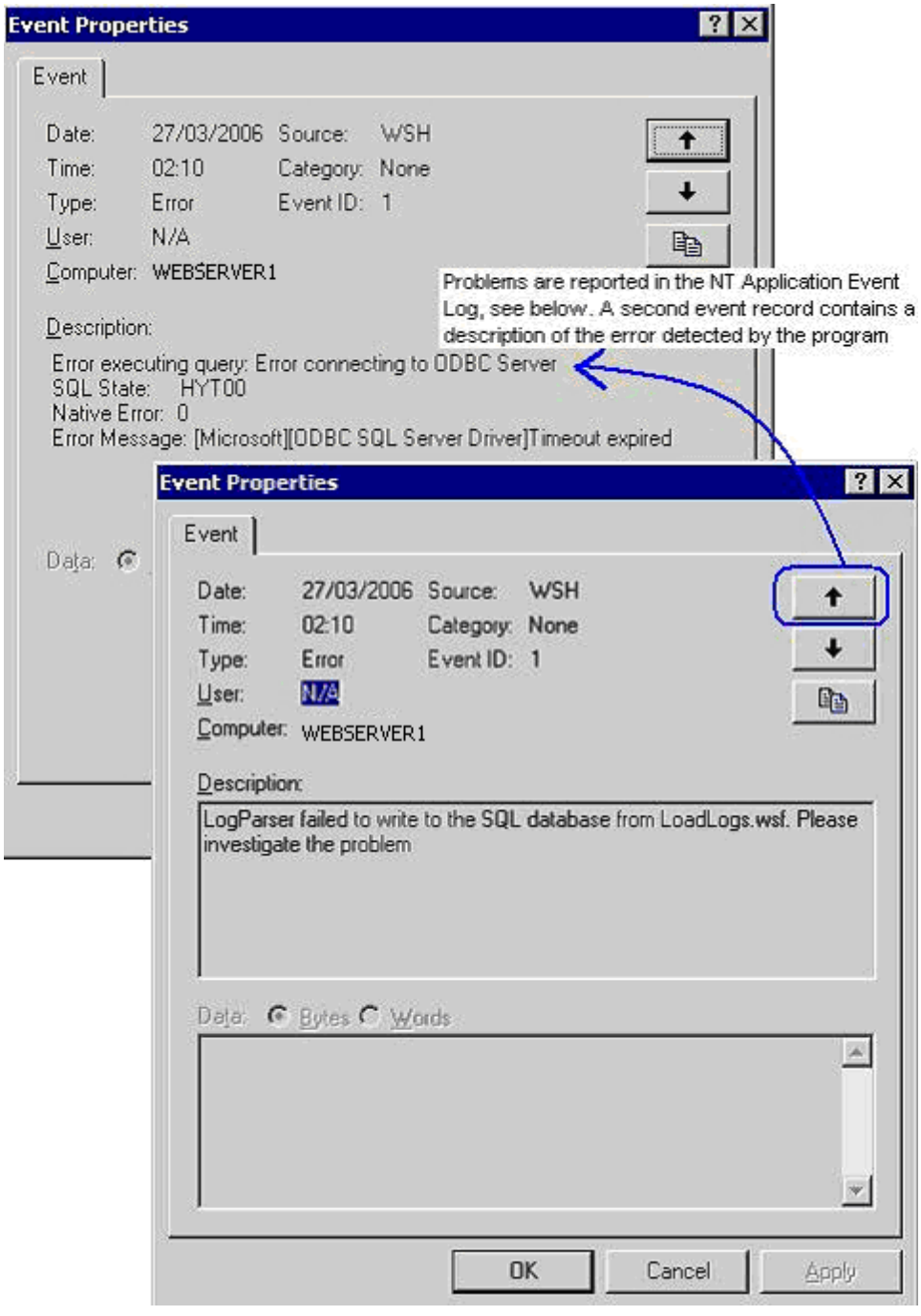
TBD

Appendix B: Housekeeping

None of the programs perform any housekeeping; so you are responsible for any archiving or deletion of web server logs, database records, or weekly graphs.

Appendix C: Examples of entries written to the event log

All three programs write to the Application Event Log. They record the success, or failure, of the program. If the program does not detect a failure it assumes the program has worked correctly and writes a single summary record. However if a failure is detected an error record is written, together with a record that contains information related to the underlying problem e.g. the following pair of event records record a database timeout.



Appendix D: Extensions and enhancements

The programs are written to process records for a single server at a time. However if you wish to aggregate data, e.g. from a set of similar servers that form a load-balanced 'web-farm', you can, with a bit of effort ☺ For instance, if you create a modified version of Stage 2, that reads the data from several source databases at once e.g. using a database view that contains a database union of the data from each of the source databases, then the results simply need to be written to a single, common database, in order for the third stage to generate suitable graphs. Such work is outside the scope of the current document.