Log files management

Katarzyna KAPUSTA

CERN openlab
07 September 2012

oTN-2012-01 openlab Summer Student Report

Log files management

Katarzyna KAPUSTA
Giacomo TENAGLIA
07 September 2012
Version 1

Abstract

The objective of this project is to improve the current log management systems used to maintain access and error logs files for all physical and virtual servers. First aim is to centralize log files in order to create a backup and provide a solution for archiving old log files on servers. The second goal is to use the Splunk software for log monitoring.

Introduction

A log file is a special file, present on every server. Its purpose is to track of what is happening on the server, in order to help the diagnosis of potential problems and provide a way to monitor actions on the machine. It is created and maintained automatically. Various sorts of log files exist, for example access (requests made to the server), application (events logged by programs) or error (all recorded errors) log files. Format and location of the file can vary with the server type.

The CERN IT databases infrastructure includes 18 Weblogic servers with around 80 hosts on them. Each host provides multiple services and each service produces associated log files. These files are a precious source of knowledge that is not only helpful from the point of view of the servers management. Log files are also interesting for developers, because they contain information about the deployment of applications on the servers. Therefore, a solution providing easy access to log files is needed. It can be realised by implementing scripts that will regroup log files into one place. A more complex solution includes the use of special software for monitoring machine-generated data.

1 Technologies and tools used in the project

1.1 Perl programming language

Perl is an acronym for “Practical Extraction and Report Language”. This programming language has been developed for more than 24 years and nowadays its fifth version runs on various platforms, from PC to mainframes. Perl language uses features from the shell programming. It provides powerful text processing facilities without the arbitrary data length limits of Unix tools. Another advantage of this programming language is its flexibility. Thousands of open source modules extensions for different kinds of developments are available. It is used for specific tasks, such as system administration, Web development, network programming etc.

Perl has been used in the project because of it is highly adapted to perform text processing, supports multithreading and can be easily combined with the bash commands.

1.2 Synchronization of files – Rsync

Rsync is a software application and network protocol that copies files and directories from one destination to another. It is similar to the SCP protocol, but provides more features. One of the most important is a possibility of data compression during transfer. The amount of data to be send is minimized by using the delta encoding when appropriate: only differences between files are transmitted.

The Rsync copying tool was used in the project for synchronization of the files between two destinations.

1.3 Monitoring log files with Splunk

Splunk software is a tool for searching, monitoring and analysing machine-generated data coming from different applications, systems and devices. It can be used for monitoring and detecting problems occurring in the IT infrastructure. It allows investigating security incidents.

Splunk collects, indexes and correlates real-time data coming from various sources into one searchable repository from which it can generate graphs, reports, alerts and visualizations. A simplified process schema is shown in the Figure 1.

Figure 1 Monitoring data using Splunk software[1]

While receiving log files, Splunk creates special events associated with the collected data – it is the action of indexing. A Splunk event corresponds to a single record of activity within a log file. Usually it contains a timestamp and provides information about what occurred on the system. Additional facts like the source of the event are also recorded.

The search engine is accessible via a web-style interface. Single or more complex expressions (defining the time or the type of the events to be shown) can be used for searching. To personalize and make the searches more effective, users can add knowledge, for example define new type of events.

In order to monitor the system, there is a possibility to set an alarm, which is activated when predefined, suspicious events, will happen.

The last step consists of exporting and analysing the data.

[1] Image source : www.splunk.com

2 Design

2.1 Log files centralization

The objective of the log files centralization is to collect all of the log files from different servers into one place. This operation cannot generate too much traffic on the network or interrupt the work of the servers. The synchronization of all log files from all servers should proceed in parallel, so an implementation that use multithreading is required. The structure of the backup is supposed to be clear and mirror the file tree directory on the server, so the users can access it in an easy way. Another purpose is the possibility of having various backup modes, like incremental or differential.

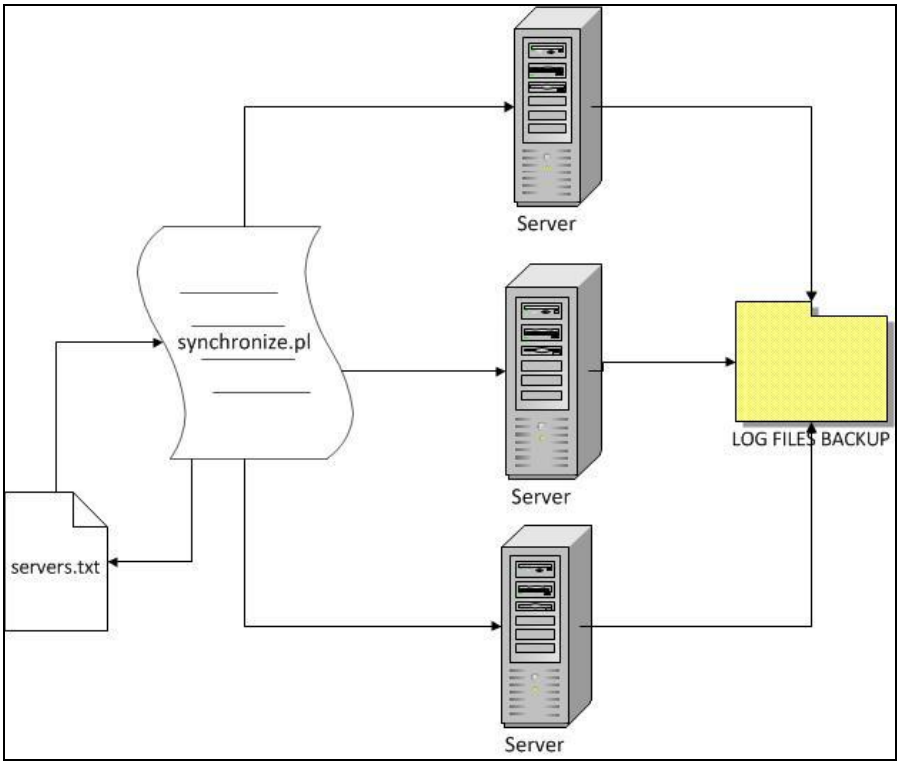

Figure 2 Centralization of log files

The figure 1 presents a design schema of the log files centralization. It contains following elements:

- Perl script synchronize.pl
- servers.txt, a file with the list of servers
- log files backup, a directory that is supposed to contain the backup of the log files
- servers producing log files

The main element is the *synchronize.pl* script. When launched, it does the following actions:

1. Reads the file servers.txt.
2. Connects to each of servers from the list .
3. Creates a directory tree specific for the server in the log files backup directory.
4. Downloads or synchronizes the log files into this directory.

2.2 Saving space on servers

Log files are generated continuously and take space on the devices. To avoid storage problems an archiving solution is needed. The idea is to compress into archives unused files that are not modified anymore. The process is supposed to:

- connect to a server
- find the log files location on it
- sort all files with a “log” or “out” extension by name and date
- compress sorted files into archives

For example, as a final result, all log files on a server with names matching pattern *server_name.log** will be packed into *server_name.tar.gz* archive.

The use of gzip file compression allows saving a considerable amount of space on devices.

2.3 Using Splunk for log files monitoring

The use of Splunk software for monitoring log files requires a solution that includes the installation of Splunk *forwarders* on hosts to be monitored and the creation of a Splunk *indexer*. Each Splunk forwarder contains a configuration file that lists all the files that have to be continuously forwarded to the Splunk indexer. The Splunk indexer processes the files arriving from the forwarders: it reads the information included in the log files and converts it to Splunk events. Users can connect to the indexer’s interface via a web-browser link. The schema in the Figure 3, shows the described structure.

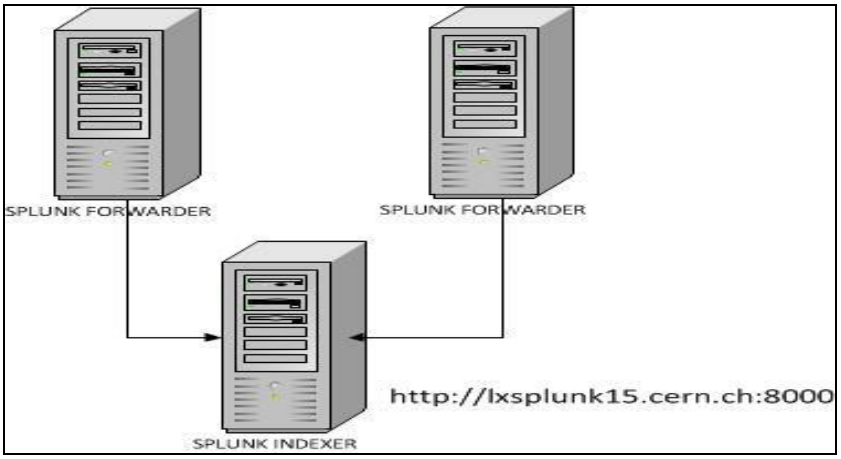

Figure 3 Forwarding data to the Splunk indexer

3 Implementation

3.1 Perl scripts

3.1.1 synchronize.pl

The synchronize.pl script reads the servers.txt file and extracts information about servers, such as hosts associated with the servers, the log locations, and the logins that should be used to connect. After that it creates threads. Each thread is connected with a host and does the following actions:

- creates a tree directory where the log files will be saved
- use the *rsync* protocol to connect to the host with the previously extracted login (there is no need to use password, because a private key authentication has been set before) and synchronize log files
- when the script runs with the "batch" mode parameter: in addition to synchronizing the log files it creates a special file containing differences between source and updated destination
- when the script runs with the "only-batch" mode parameter: it does not synchronize the log files, but only creates a file with differences

3.1.2 archive.pl and archive_on_server.pl

When launched, the *archive.pl* script acts in a similar way than the *synchronize.pl*. It reads the list of servers from the *servers.txt* file and creates threads. Each thread is connected with a host and does only one action: launch on this host another Perl script – *archive_on_server.pl*.
archive_on_server.pl does the following actions:

- finds all directories named "logs" on the host
- puts the log files in a hash structure, depending on the name and extension of the files
- creates compressed archives, containing old log files and having names corresponding with the log files inside

3.2 Splunk implementation

3.2.1 Configuring indexer and forwarders

The Splunk indexer instance has already been running on one of the servers. In order to forward the data, Splunk forwarders has been installed on other devices. The configuration of a single Splunk forwarder included the modification of the *inputs.conf* and *outputs.conf* Splunk configuration files in the directory *$SPLUNK_HOME/etc/system/default*.
Following important parameters have been set:

- the address of the Splunk indexer
- the paths of log files directories containing data to be forwarded to the indexer
- the types of log files

3.2.2 Monitoring logs using Splunk - examples

The Figure 4 and Figure 5 contain examples of Splunk utilisation. The first figure shows the Splunk user interface containing results of a search for :
“FULL THREAD DUMP earliest=08/28/2012:0:0:0 latest=08/29/2012:0:0:0”

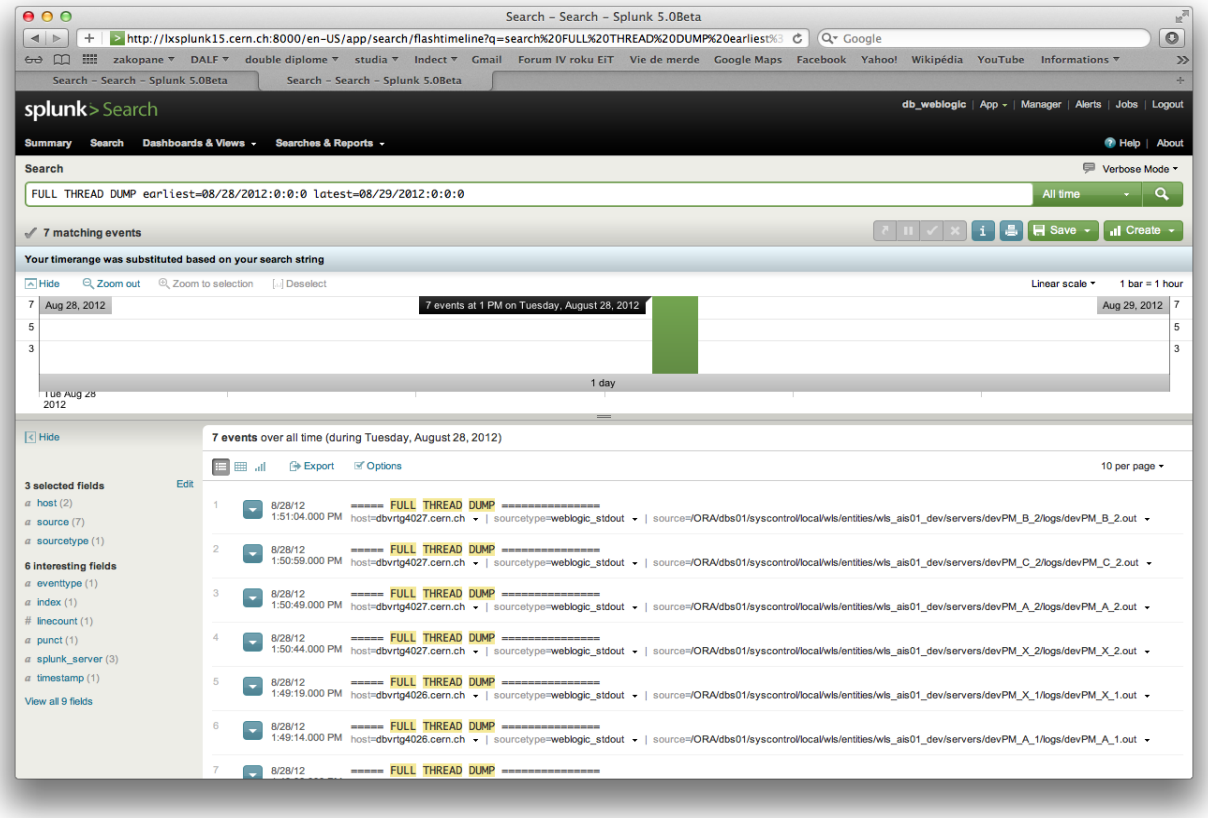

Figure 4 Search for "Full thread dump"

The second figure presents the use of a Splunk *eventtype* . Instead of entering each time the same request, user can save the request as a defined event. It facilitates the search and provides additional options, such as colouring all events matching the declared *eventtype*.

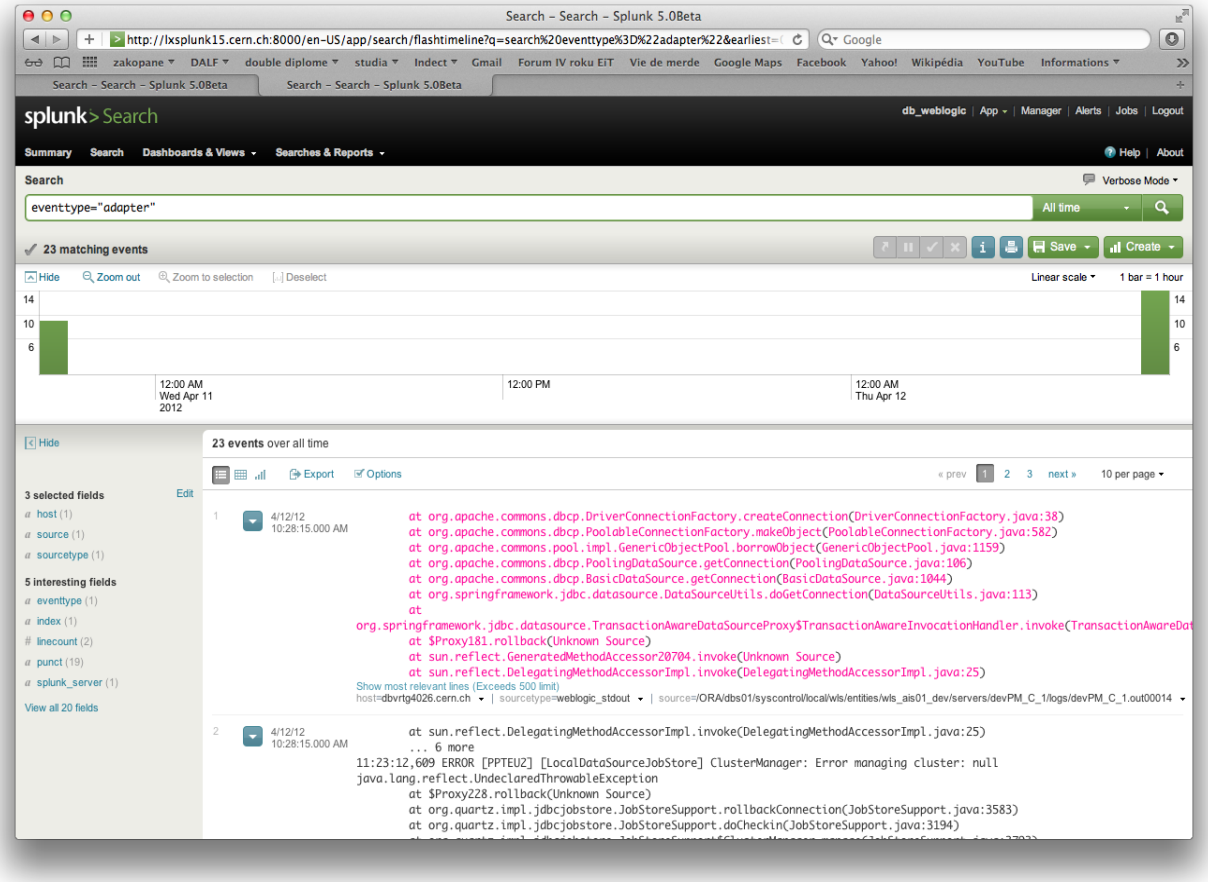

Figure 5 Searching using "eventtype"

4 Summary

All project objectives were realized. Author contributed to the improvement of the IT-DB log files management system. The centralization of log files has been realized and a solution saving space on the devices has also been implemented. The Splunk indexer instance works and processes log files coming from ten destinations. It is ready to integrate more sources.

Appendix A - SCRIPTS DOCUMENTATION

FILES LOCATION

All of the files described in this document are located on the dbsrvd242 server, in the directory:

/ORA/dbs01/distribute_svn/distribute/dist_top_syscontrol/entities/logs_management

SERVERS.TXT

Contains a list of servers with their log files location. It is used by two scripts: synchronize.pl and archive.pl

The list looks like:

Server | Hostname | Login | Logs location

---

wls_itdbims01_dev dbvrts1002 sysctl wls_itdbims01_dev

wls_itdbims01_dev dbvrts1003 sysctl wls_itdbims01_dev

wls_itdbims01_dev dbvrts1001 sysctl wls_itdbims01_dev

ADDING AN ENTRY

When adding an entry: insert into one row (separating with spaces) the name of the server, the name of the host, the login to be used for the connection and the log files location.

The log files location is a directory in the path that points to a log files location on Weblogic server:

/ORA/dbs01/syscontrol/local/wls/entities/<LOGS_LOCATION>/servers/[***]/logs

Information about different Weblogic servers can be find on the Twiki page:

https://twiki.cern.ch/twiki/bin/viewauth/DB/Private/CentralisedLogManagement

SYNCHRONIZE.PL

OPTIONS:

- batch: synchronizes files and creates a batch file with differences
- batch-only: creates only a batch file with differences
- help: provides a short description of the script options
- no option: creates a file structure for each server in the *logs_backup* directory (if does not exist), synchronizes log files. Does not create the batch file.

DESCRIPTION

This script is doing the following actions:

1. reads the file servers.txt with a list of servers
2. connects to each of servers from the list

3. creates a directory tree specific for the server (if it doesn't exist) in the *logs_backup* directory. The path to the backup of logs for the server will look like:
    - *logs_backup* /<*server_name*> / <*host_name*> /<*logs_location*> .
4. downloads or synchronizes (if the logs backup have been already done) the log files
5. if used with:
    a. the " batch" option: in addition to synchronizing the logs, it creates a batch file, that contains differences that were synchronized
    b. the "batch-only" option: creates only the batch file, containing the differences between the backup saved in the *logs_backup* directory and the source (logs on the server). This option is only available when running the synchronize.pl script on a server with the new rsync version installed (not supported on dbsrvd242).

The batch file are being saved in directories with name like "batch_<t*imestamp*>". Timestamp refers to the current date and time.

For example a batch file created the 07 September 2012 at 15:05 will be saved in the directory *logs_backup* / <*server_name*> / <*host_name*> /<*logs_location*> *batch_20120907_1505*.

ARCHIVE.PL

DESCRIPTION:

This script is doing the following actions:

1. reads the file servers.txt with a list of servers
2. connects to each of servers from the list
3. archives files in *logs* folders:
    a. log files starting with the same name will be packed into one archive
    b. the first and the last log will not be compressed

Example:

Files matching devIMS_A_1.log* will be packed into: devIMS_A_1.tar.gz

ARCHIVE_ON_SERVER.PL

A script used by the archive.pl that runs it remotely on each server. Not to be used separately.