SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

"Hit Me Right" – A Primer on Generating Good Web Statistics

Applies To:

SAP Web Application Server

Summary

Web statistics have always been a source of misunderstanding. I will try to clear things up and attempt to sum up the various possibilities for enabling web statistics.

By: Eddy De Clercq

Company: Katholieke Universiteit Leuven

Date: 01 June 2005

Table of Contents

"Hit Me Right" – A Primer on Generating Good Web Statistics

Introduction

According to Wikipedia, statistics "is the science and practice of developing knowledge through the use of empirical data expressed in quantitative form… one aim of statistics is to produce the 'best' information from available data." I have a love/hate relationship with statistics. Sure, they are necessary and certainly very useful, but I find the interpretation bit always a trifle dangerous. Statistics can always be (mis)used in order to prove that a particular view is correct.

To give a stupid and simplistic example: when a glass is 50% full, is it half full or already half empty? It all depends on the person interpreting the figures. And this is the thing I don't like. Despite the power of statistics, they lose a lot of credibility due to (mis)interpretation.

Hits

Web statistics have always been slightly misunderstood. During my web development career I've seen a lot of people gloat about the amount of hits they have had. People tend to get angry when I try to explain that the amount of hits doesn't indicate a lot and has no value as such. Let me explain by defining what a hit is. A hit is nothing more than a request for a resource (hence the Uniform **Resource** Locator) made to a web server. I deliberately say resource, since it can be more than the usual file. It can be anything: static info in files, or dynamic via CGI for example.

The confusion already starts over here. If a user requests something, it won't automatically translate into just one hit. How come? The user requests something that we usually call a page. That page doesn't stand on its own; it usually contains graphics, pictures, audio files, etc. The user won't notice but the browser will detect this and sends a request for each extra bit it needs. So that means that an HTML page with 10 pictures counts for 11 hits. So a rather complicated page that includes external CSS and JS files will maybe boost your counter or maybe not. It could even be that this same complicated page won't generate any hits at all. Huh? Yes indeed, and the keyword for this is caching. It can happen everywhere between the browser and the web server. The most obvious ones are the browser cache, (reverse) proxy, and server cache.

Caching can also have the reverse effect and generate hits even when the user didn't request anything on your server. That sounds very spooky, but it isn't as such. Maybe you've heard about the new "prefetch" technology Google is using in order to load results faster. There was a lot of fuss about this, but really Google is only using the Mozilla link prefetching technique. One of the criticisms was that it generates unwanted traffic, but I don't see why. If you don't want to allow prefetching on your server or browser, just have a look at this guide.

Speaking of search engines… these can generate hits on your machine too. Everybody wants their site listed in search engines, preferably with a high ranking. There is price to pay, though - they generate unwanted traffic and hits. So the statistical measurement tool must make a distinction between human and robot visitors in order to obtain accurate figures.

Better Metrics?

The above doesn't seem very helpful for people wanting to determine how busy their site is. It can be more difficult though. As previously stated, a hit is a request. A request though isn't the same thing as an actual

useful result for the user. A lot of things can go wrong. The most obvious one is a request for a resource that doesn't exist, a so called 404 (see also HTTP status codes). Unless you want to correct these errors, you aren't really interested in this. That's why some web stat tools like to count the files retrieved. The number of files used can also give an indication of repeating visitors. The higher the differences between the two should imply that there were a lot of files being cached. That is only the case if nothing is going wrong of course.

Now the term "visitors" is also something that's interpreted differently. One agrees that a visit to a site starts with the request of a page for the first time and lasts until the user leaves the site. But when does a user leave the site? If I go to site A followed by site B and return to site A, am I then counted twice? Well it depends on the stats tool you're using. They will set a certain time frame within the current and the previous request. Some applications set 30 minutes as the margin to take; some others let you configure this. Some tools only compare pages; others will allow all type of files. This makes comparing things far from easy. If I set this margin very low, I will have of course more visitors than somebody who can't configure it or who sets a higher margin.

Speaking of pages... what is a page? Is it just HTML or do you allow CGI scripting? In some completely dynamic sites the page as such will remain the same. The parameters will be different though. What about sites (like mine) built in Flash? You only retrieve one Flash object, which contains the whole site. Unless you don't use includes or do some client/server communication, not much traffic will be generated.

Define Standards

As you can see, it isn't all as easy as it looks at first sight. Does this mean that it's impossible to have some good statistics? Yes and no. As long as you define standards well and stick to the same tools for comparing sites, you will be able to have a good insight into the traffic on your website(s). Which tools should you choose then? It all depends on your needs and the available budget. In the following, I'll try to summarize the possibilities in a far from complete overview.

A. Web Log Analyzers

Most of the web server stat tools can be categorized within this section. They use the logs of a web server for making the statistics on traffic (presuming of course that your web server is in effect logging the activity). On the SAP Web AS you need to use transaction RZ10 for this and to define the y_icm_HTTP_logging_0 parameter for this. Besides logging settings you also need to define the format.

%b	Length of the response in bytes
%h	Name of the remote host (the client, such as the browser)
%H	Name of local host
%S	Local port name or service
%a	IP address of the remote host

%l	Specifies the *Remote Logname*. This name is the result of an IDENT query to the client. This only works if the identity check is activated there.
%u	User name of 401 authentication
%t	Time specification in CLF format: [15/Dec/2000:16:18:35 +0100]
%T	Duration of a request in seconds
%L	The duration of a request in milliseconds
%r	1. Row of an HTTP request: such as GET /bc/ping HTTP/1.0
%f	Name of requested object without form fields
%U	Whole URI of a request (with form fields)
%s	OK code of the response
%v	Name of the virtual host (IP address or name of the server with which the client is linked)
%V	Fully-qualified host name (FQHN) of the server (value of parameter icm/host_name_full or FQHN of the operating system).
%{name}i	Name of a request header field, e.g. %{user-agent}i
%{name}o	Name of a response header field, e.g. %{server}o
%{cookie}c	Output of a request cookie
%{cookie}c	Output of a response cookie

A typical setting in RZ10 would look like this:

```
y_icm_HTTP_logging_0

PREFIX=/, LOGFILE=access.log, LOGFORMAT=%h %l %u %t "%r1" %s %b
%{referer}i
```

The way things are logged is rather important. There are a couple of standards for this:

- Common Logfile Format (common/CLF)
- Combined Logfile Format (NCSA combined/XLF/ELF)
- IIS (W3C)
- Others (see also this overview)
- Own configurations, as long as the log analyzer can interpret them

Having set all this, one can then choose the software for analyzing the web log. This isn't an easy task since literary hundreds of tools exist for this. Some are focused on marketing analysis, some just count the hits, some are very expensive, some are free, and so on. I won't go into details about them or otherwise I will end up with a book instead of an article. This slightly outdated list will give you an overview. Here is a short description of three free tools which are usually provided by service providers (as a subcomponent of e.g. Cpanel). These are basic, no-nonsense analyzers that do their work well.

- Analog: probably the most bare-bones analyzer one can think of. It's rather popular though since it's fast, runs on virtually any platform, is highly tweakable, and reports in 32 languages. It's perfect for server administrators who want to know what the traffic on a particular server is like. It's not so suitable for analysis of the content since it shows the hits in a poor graphical representation.

Figure 1: Analog

Luckily, there are 'extensions' like Report Magic, which gives prettier output and enables you to analyze more than Analog does. It lets Analog do the brute force to convert raw web logs into Computer Readable Output format and then does its own thing with it.

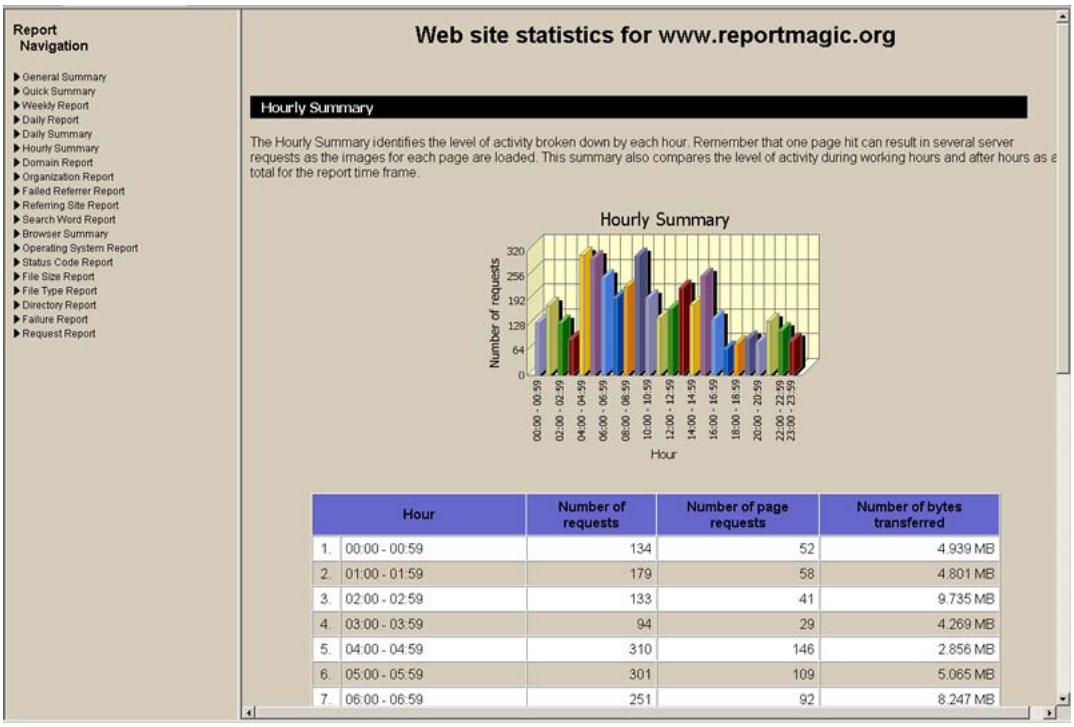


Figure 2: Report Magic

Webalizer: Fairly similar to Analog concerning the basic features. It scores with the extra analytical features like counting the visits, pages, difference between files and hits. My favorite features are the capabilities for hiding, ignoring, and grouping URLs. This comes in rather handy with dynamic sites where you can have a page with several possible parameters. With other tools, the top 10 is then filled with that same page with all different parameters. Webalizer lets you group all this into one URL/page and thus it is mentioned in the Top 10 only once, with all the figures summed for that group. Besides URLs, you can also group referrers, visiting sites, browsers (agents), domains, and users. This is an example on how it is done:

```
# Grouping options

GroupURL    /cgi-bin/*  CGI Scripts
GroupURL    /images/*   Images

GroupSite   *.aol.com
GroupSite   *.compuserve.com

GroupReferrer     yahoo.com/  Yahoo!
GroupReferrer     excite.com/     Excite
```

```
GroupReferrer     infoseek.com/   InfoSeek
GroupReferrer     webcrawler.com/ WebCrawler

GroupUser      root            Admin users
GroupUser      admin           Admin users
GroupUser      wheel           Admin users

# The following is a great way to get an overall total
# for browsers, and not display all the detail records.
# (You should use MangleAgent to refine further...)

GroupAgent  MSIE        Micro$oft Internet Exploder
HideAgent   MSIE
GroupAgent  Mozilla           Netscape
HideAgent   Mozilla
GroupAgent  Lynx*       Lynx
HideAgent   Lynx*
```

The only problem with Webalizer is that there hasn't been a new version released since 2002, which is rather a pity.

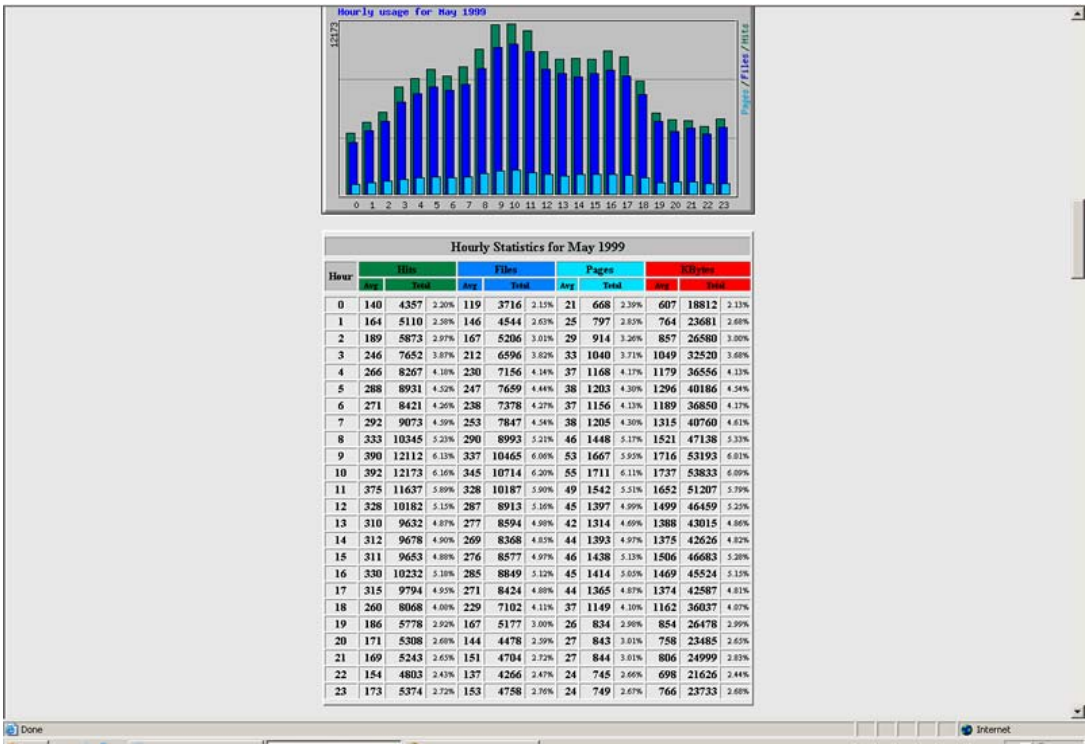


Figure 3: Webalizer

- Awstats: This is the rising star of the free web log analyzers. It has an endless list of features. One that I find most remarkable is the ability to provide statistics in real time on demand (besides the usual batch mode). One must be careful though with high traffic sites since Awstats could demand a lot of your machine, so a dedicated machine is preferable. As such there is nothing in particular of note concerning daily analysis of your web site's traffic. The only minor thing that I can think of is the poor grouping capabilities of URLs. Having the same capabilities as Webalizer would be the nec plus ultra.

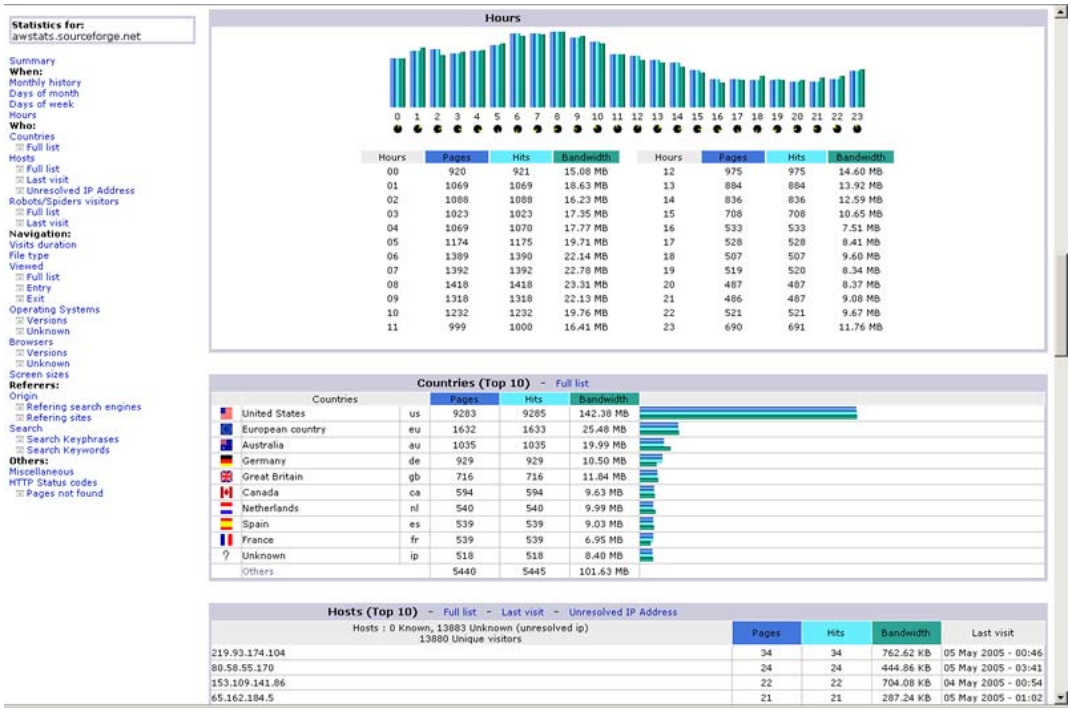


Figure 4: Awstats

B. Real-time/Online Tools

This is a rather different approach. Instead of analyzing logs, you need to include a piece of code on each page you want to analyze in stats. As such they use the old visitor counter principle. You put an image on the site, but instead of referring to a real image, it points to a script, which increments the counter and shows it. But instead of a script of your server, it'll point to a script at the tracking server and depending on how much you pay it'll show a commercial image or not.

An example of the code that ecommStats uses:

```
<script language="JavaScript" src="statscript.js"></script>
<script language="JavaScript">statImage();</script>
<noscript><a href="http://www.ecommstats.com/"><img
src="http://img.ecommstats.com/webstats/statimg.jsp?s=SITEID"></a></noscript>
```

The drawback of this method is that you need to include this code in every page, which can be tedious. Certainly if you use commercial applications (e.g. online shop) where you may not understand the structure of the application or are not even allowed to alter anything in the code. Another problem could be the fact that

your whole website is dynamic and has one page as entry point and all the rest depends on the parameters you provide. The tool must understand and interpret these parameters correctly in order to provide an accurate analysis.

Again, plenty of tools exist. It all depends on your needs and budget.

C. Generating Own Stats in BSP

C.1. Logging It

If all of the aforementioned tools don't cover what you're looking for, or there are technical/other reasons not to choose them, then there is only one thing left to do. Write your own stuff. It really isn't that difficult to do and, if you are confident enough, you can make something that suits your needs 100%. Or as a host in a Flemish regional TV DIY show always says: "What you do yourself is usually done better." Let us see what you need to do in order to achieve this.

First of all you need data on the browser. This can be done via JavaScript and the navigator object:

appCodeName	The code name of browser (i.e.: Mozilla)
appName	The name of the browser (i.e.: Microsoft Internet Explorer)
appVersion	Version information of the browser (ie: 4.75 [en] (Win98; U)
userAgent	String passed by browser as user-agent header. (ie: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Hotbar 3.0))
Platform	The platform of the client's computer. JavaScript 1.2 property. (ie: Win32)

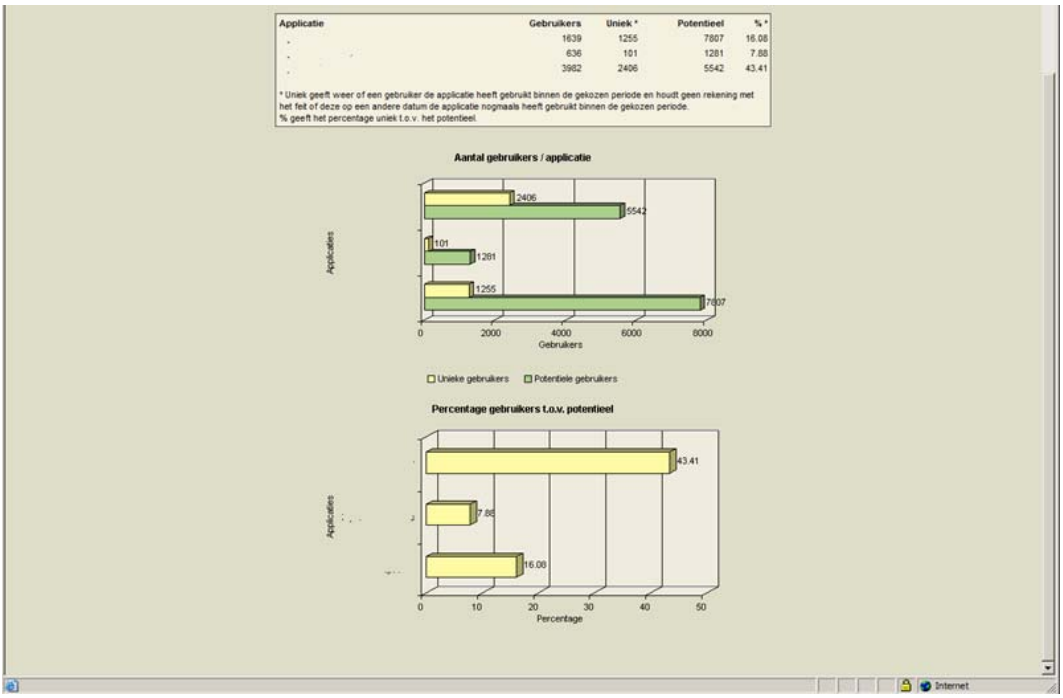
You probably need some extra information like IP number. This can be done via the `request->get_header_field` method.

As such you don't need to retrieve this info every time if you have a single entry point like a login page. Just put it in a server side cookie. Each new page will read this cookie and save this info together with the page name, user id, eventual parameters, etc. into the log. Now I talked earlier about the visitor concept. This is also an ideal place to implement this. If you are only interested in returning visitors with a full day time interval, you can for example determine whether that user has already used that application. If so, you just increment a counter. If not, you have to create a new line. This method will also reduce the size of your table.

C.2 Showing It

The only thing you need to do is to analyze and visualize stuff. The analyzing stuff is completely up to you. The following screen shot is an example where the total number of visitors within a given time frame is shown. It also shows the number of unique visitors in that same time frame compared to the potential number of visitors. You might find this of interest if you want to know how many (internal) users actually use your intranet

application. You can couple the user info to the department info and then you can figure out whether department X or Y is actually using the application.



The graphics are done with the HTMLB chart tag. Have a look at SBSPEXT_HTMLB\chart.bsp example application for detailed info. I would recommend the coupling with the IGS server, which delivers better quality than the standard applet generated charts.

Conclusion

As you can see, there is a lot to say about web statistics. Not everything has been said though. I didn't talk about doing stuff within EP 6 for instance. However, do not despair - Prakash Singh wrote an excellent weblog concerning this matter. I think that all this information could give you a head start in analyzing web traffic on your site(s).

Author Bio

Eddy De Clercq has 20 years experience in computing. He currently works at the Katholieke Universiteit Leuven, the oldest university of the Low Countries and the largest Flemish university. Eddy is part of the E-university team that creates mostly self service (web) applications.