

Advanced Techniques for Analyzing Web Server Logs*

Ernst-Georg Haffner, Uwe Roth, Andreas Heuer, Thomas Engel, Christoph Meinel
Institute of Telematics
Bahnhofstraße 30-32
D-54292 Trier, Germany

Abstract *This paper gives an account of the practical experiences made in generating special statistical information of web server logs. It emphasizes the problem of combining different data sources to retrieve acceptable working results. Several critical aspects in analyzing logfiles are discussed and solutions are described. The structural capacities of multilingual online authoring systems, especially DAPHNE, have to be combined with the flexibility of an information server, here the SDS, in order to improve functionality and efficiency of the World Wide Web.*

Keywords: Web Server Logs, DAPHNE, SDS

1. Introduction

The World Wide Web (WWW) is growing very fast and the need for tools to efficiently analyze web server logs is immense. There are many standard tools to analyze the logged information of common web servers. According to the survey of Netcraft, the Apache web server [1] reached a share of more than 50 percent on the web server market in January 2000 [2]. This server produces - as most of the others - access logfiles in the common logfile format [3]. Logging into a database is also possible.

There are lots of tools to analyze, summarize or split logfiles in CLF-format. For instance, Roy Fieldings perl [4] script *wwwstat* [5] is able to read logs and produce access statistics for several categories as summaries of requests *by date*, *by hour* or transfers *by URL*.

Other popular logfile analyzing tools are *Analog* [6], *Wusage* [7] and *Tabulate* [8] with similar functionality (see also [9] for a list of

web analyzing tools).

The statistical information mentioned above can be achieved by analyzing only the web server logs. Things are getting much more complicated, though, if additional requirements come up. Meta information about the documents like the *title*, the *author* or the *creation date* can only be retrieved when access to a data source where all of this data is stored is possible.

Another problem occurs when statistical summaries also have to be done by grouping together documents on the web server side (for instance, departmental pages of a company) or when the grouping of request IP's according to their logical or structural dependencies is required (e.g. requests by companies of the same branch).

There are, undoubtedly, a lot of additional possibilities for statistical analysis, but in this paper we will describe a practical type of work where several different requirements (as they will be described in section 2.1) have to be fulfilled according to the wishes of a customer, a great European bank. Furthermore, it was clear from the very beginning that future requirements should be treated adequately as well.

2. The Problem

2.1 Customer requirements

The starting points for the project are the requirements of a customer to evaluate web server log statistics. The following section not only emphasizes these concrete requirements, but also stresses what (software-) environment is needed to fulfill the requirements.

On the one hand, we had monthly log files from two (virtual) Apache web servers. Some files on

* In Proc. „International Conference on Internet Computing”, CSREA, IC00, Las Vegas, Nevada, 2000, pp. 71-78

both servers were identical, while others differed. One virtual web server allowed public access while the other was dedicated to a closed circle of business customers. Only the public web server provided multilingual pages (e.g. English, German, French, Spanish).

On the other hand, we could use the information of the online authoring system DAPHNE [10] to retrieve the needed meta-data.

The following aspects describe the different information parts (requirements) that have to be extracted from the log files. According to the

data sources needed to retrieve the information, we arranged the requirements in 3 “problem classes” A to C (table 1 to 3):

| A | Simple logfile extraction |
|---|---|
| 1 | Summarizing total transfers by request date |
| 2 | Summarizing total transfers by request hour |
| 3 | Total transfers by toplevel/client domains |
| 4 | Total transfers by reversed subdomains |
| 5 | Summarizing total transfers by filename/URL |

Table 1: Requirements of class A

| B | Complex logfile extraction or online authoring access | Examples |
|---|--|---|
| 1 | Showing meta data of documents together with the number of accesses (hits) | Title, author, original filename, creation date, expiration date, description, language |
| 2 | Analyzing requesting Browsers | Grouping by Browser producer (e.g. Netscape (Mozilla) or Microsoft Internet Explorer) Grouping by Browser version (e.g. 4.0, 4.01, 4.6, 4.7) |

Table 2: Requirements of class B

| C | Complex grouping of hits | Examples for groups |
|---|--|---|
| 1 | Grouping according to the content of a document | Departmental information, language, content similarity (e.g. tables with share values), types (html, gif, jpeg...) |
| 2 | Grouping according to the accessing domain-name (only if available; an important part of IP numbers can not be resolved to domain names) | Grouping different servers of the same Internet service provider (mostly with the same subdomain, but not always, e.g. n.svr.t-online.de, ...) Grouping together according to the branch of the requesting domain (e.g. summaries of scientific institutions www.mit.edu , www.uni-trier.de , www.ti.fhg.de or banks www.bankofamerica.com , www.deutsche-bank.com) |
| 3 | Grouping of long time statistics by requesting month or year | Monthly or yearly trends in access statistics, not only single hits but also session requests |

Table 3: Requirements of class C

Certainly, the class C problems can not fully automatically be determined. Instead, a database stores manual affiliations of domains. Unfortunately, the grouping of “private accesses” and “business accesses” is sometimes hard to fulfill. Small companies and private persons can have the same Internet service providers (ISP), while also private requests can be made by business domains.

In paragraph 2.2 the functionality and the relevant database model that DAPHNE provides is highlighted. To combine the information sources considering the requirements of class C it was necessary to work on the data of a central server. Here, we used the Smart Data Server SDS (2.3) as flexible, modular and scalable framework for distributed functionality. Furthermore, also the requirements of class A

were solved by the SDS. We only checked the results of our program against those of *wwwstat* during the implementation phase to see whether there were some errors in the counting procedures.

2.2 DAPHNE

We used the online authoring system DAPHNE (Distributed Authoring and Publishing in a Hypertext and Network Environments) as described in [10] to produce the Internet web pages and to generate the files provided in the Intranet, their meta tags (description, keywords, etc.) and the securing of web accessibility.

For our project we used the system to get the meta-data for the documents on the web server and thus being able to fulfill the requirements of class B1.

DAPHNE uses a relational database to store meta-

data for documents. For instance, the author, the document title, the original path and name of the source file (on the local computer of the author) are meta-data information that can be retrieved directly from the DAPHNE database. There are also several statuses from “locked” over “free” to “published in the Internet” stored for each file. The access to them happens on a role based access control [11]. For our problem those values were not too important.

Each document belongs to one or more hierarchically structured areas and every area has one document as cover page. Figure 1 shows the relevant part of the database conceptual model concerning information needed to improve statistical analysis of web accesses.

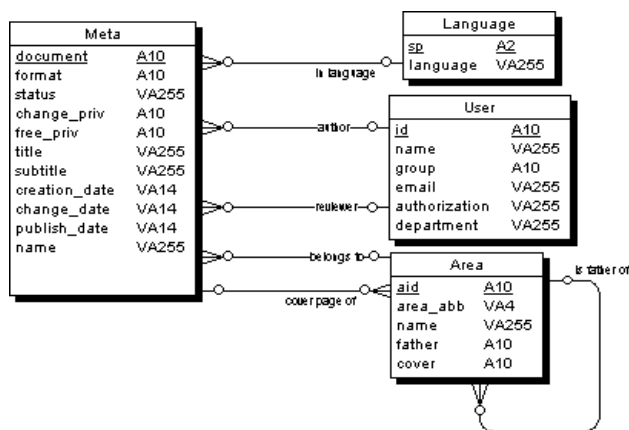


Figure 1: Web-statistic relevant part of the conceptual model of the DAPHNE database

Cover pages can have an alias name to generate links to the according area or to the special document. Every document receives a unique, numeric name (document) when put into the DAPHNE system and every area gets a unique identifier (aid). Cover pages of areas can be referred to by simply using the unique area name and adding the “.html” as suffix¹. Therefore, all accesses to cover pages must be summarized to the according document requests so that the results become correct.

To fulfill class C requirements the area hierarchy represented in table “Area”, that references to itself, must be *flattened*. Each web server access to a document that belongs to an

area A counts also for all areas (in the hierarchy) above A. The sense of this counting gets clear when imaging that one wants to compare exactly the hits for every department of the company.

Our idea was to directly access the DAPHNE-database without making any further data transformation.

2.3 SDS

The requirements C and B2 described in section 2.1 were not easy to achieve. For instance, we could have adapted a perl script as *wwwstat* [5] to make additional summaries for web browser versions but such a solution would not have been very flexible. Instead, to group several classes of client domains together it was necessary and obvious to use a database.

So we decided to install the *Smart Data Server* SDS [12] as a general framework for distributed applications. The SDS as platform for many applications in heterogeneous network environments is able to access several databases at a time [13]. It comes with its own relational database and can access also the DAPHNE-database. To be able to fulfill the requests of class C we added the following database tables to the SDS database (figure 2).

The central table is “Access”. Here we store a data record for every line in a logfile. The tables “OS”, “Browser” and “Versions” serve to fulfill the requirement B2. The host and domain related tables on the right side are necessary for requirement C2. The “WebServers” table helps handling the log information of different web servers.

The mechanism to obtain flexible grouping summaries is explained in the section 3.2. The tables “ComeFrom” and “Search” contain the extracted information of the fully specified web server request. Especially, the parameters given by search engines (with the HTTP-Get Method [14]) are used to evaluate the share of document accesses that are initiated by search engines (and not other, mostly static html pages).

The tables “Statistics” and “ReferenceData” are generated to store the analyzing results of the (monthly) evaluation. Thus, the information needed to create longtime statistical summaries of log data (trends over the year) can be easily derived from that tables instead of calculating all these results again from the “Access”-table. The disadvantage of keeping somewhat redundant data can be overcome by

¹ The export procedure of DAPHNE stores all documents of the same language in the same directory so that no path information has to be considered.

efficiency considerations. If needed, the data records of both tables could be regenerated from

the other tables.

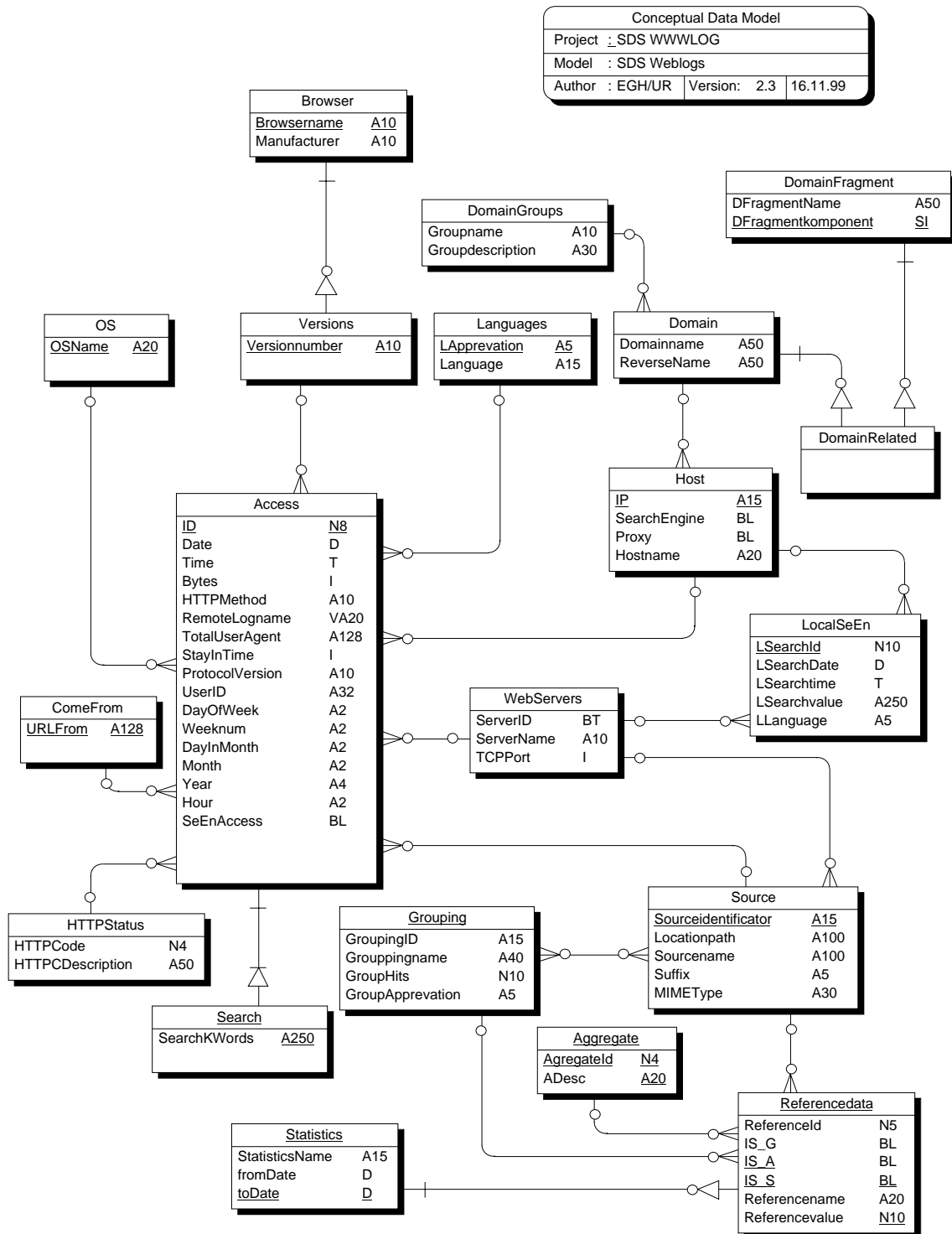


Figure 2: The SDS conceptual database model for analyzing web server logs

3. A solution strategy

3.1 General considerations

To be able to solve all classes of requirements (2.1) we used the logfiles from the Apache web server, the DAPHNE-program, especially its database, and the SDS as central server to handle the different data sources as a useful platform to implement the statistical algorithms. The graphics

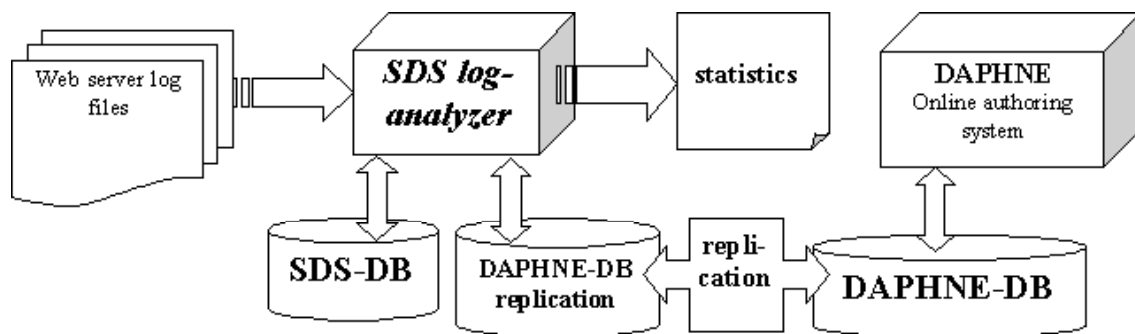


Figure 3: Log-analyzing architecture

of figure 3 show the architecture of our project approach.

The SDS comes with its own relational database and accesses also a replication of the original customer DAPHNE-database. Once a month a data replication of the tables shown in figure 1 must happen.

The results of the log analyzing process are output by the SDS in form of ASCII-tables, pairs of name and values. At the moment, there is work in progress to automate the postprocessing of the results into a graphical representation (e.g. data transfer to Microsoft-Office products as Excel [15] with DDE-interface and/or a Unix based solution with latex [16] and gnuplot [17])

Our idea was, that the architecture must provide much flexibility. The number and volumes of the Apache log file data poses no problem to the SDS, also the database sources are simply registered as part of the SDS configuration. The same model would work for much more databases and several locations of the tables.

3.2 The grouping algorithm

The central algorithm to analyze log information consists of the grouping mechanism. It has relevance especially for the grouping of affiliated

documents and has to work on the self-referencing DAPHNE-table "Area".

The two-phase process to transfer the grouping hierarchy of the online authoring system into the "Grouping" table of the SDS is called *flattening*. In order to describe the algorithm adequately, we need to have a closer look at the physical representation (ANSI level 2) of the SDS "n to m"-grouping table relation. Figure 4 shows the resolution of the relevant part of the model:

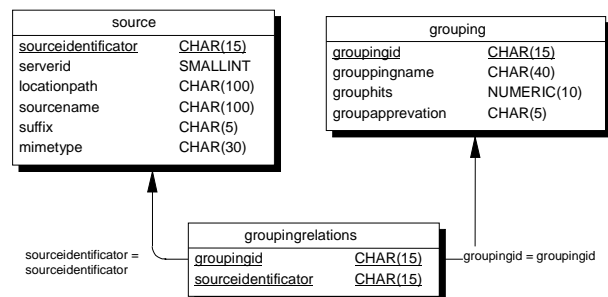


Figure 4: Log-analyzing architecture

The flattening algorithm can thus be described as (figure 5):

Flattening Algorithm

Phase 1: "transferring the table information"

INPUT: set of records "Area" and of "Meta" of the DAPHNE-database
 OUTPUT: RESULT-CODE only
 SIDE-EFFECT: set of records "Grouping" and "Source" of the SDS-database

- transform and copy records from Area to Grouping
- transform and copy records from Meta to Source

```

Phase 2: "generating source-grouping
affiliations"
INPUT: set of records "Area" of the
DAPHNE-database
OUTPUT: RESULT-CODE only
SIDE-EFFECT: insert records into
"GroupingRelations" of the SDS-
database
function retrieveFathers(areaRecord,
    resultSset) {
    if (hasNoFather (areaRecord))
        return resultSet;
    else {
        fatherRecord =
getFather (areaRecord);
        resultSet = resultSet ∪ father;
        return retrieveFathers (father,
            resultSet);
    } // function retrieveFathers
main function ("Area" records) {
    relationsSetOfSets := empty;
    for every record areaRecord in "Area"
        do {
            if (not existsEntry areaRecord in
                relationsSetOfSets) {
                relationsSetOfSets (areaRecord) =
                    retrieveFathers (areaRecord, {
                        areaRecord });
            }
            for every entry relation in
                relationsSetOfSets (areaRecord) do
            {
                insertRecord (relation) into
                    "GroupingRelations";
            }
        } // main function

```

Figure 5: The "Flattening Algorithm"

After the re-generation of the grouping tables of the SDS in phase 1 the flattening-algorithm inserts a data record into the "GroupingRelations" table for every document and every group. The group accesses can easily be retrieved afterwards using simple SQL-statements. The core of the algorithm consists in recursively stepping through the "Area"-table and considering the father relation for every record. In the last step the "flattened" grouping information is stored in the grouping relations table.

4. Practical Hindrances

As usual in the IT-world the main tasks of our project were quickly and stable implemented, but several little problems posed a lot of difficulties.

An important point was the evaluation of the responses to the server requests. In web server statistics forbidden requests (403) or "file not found"-errors (404) are often treated the same way as positive answers [14]. But what would be

the best reaction? If a request for `"/department/filex.html"` can not be fulfilled, this is an other situation as if a request `"/filex.html"` leads to a 404 error. In the first case the path to the department may be correct, so the missing file would belong to this group (for counting grouped error accesses). For this reason, we distinguished between the path information and the filename to evaluate the answer.

Another problem arises when the content of a page changes. When is it appropriate to summarize all hits for that file and when should the requests be regarded as for two different files?

The "Meta"-table of the DAPHNE-database allows it to use a creation and expiration date, but practice shows that especially the expiration feature is seldom used for standard HTML-pages. Another possibility would be to take care of the title: if it changes, we could speak of a fundamental content change and distinguish between hits before and after the change (even though this might not be correct). But the monthly update of the replicated database prevents us from evaluating that point: we only see the results of the change. Alas, we have no other possibility as to summarize all hits of target documents with the same identification (filename).

Much harder is the affiliation of a document to one or more groups. A change in the affiliation has lots of consequences for the group summaries. Unfortunately here again, we retrieve only monthly updates of the accordant records. A single change of affiliation of a document is very seldom and (perhaps) statistically not relevant. But to overcome the problem when a restructuring of the whole page is planed, we arranged to send an additional update of the replicated table records and the logfile information so far. At the end of the month the remainder of the logfiles is added and the monthly statistic works quite fine.

Evaluating the "URL referrer"-part of the HTTP-request, manifold information from the referring page can be retrieved. We have to distinguish between *search engine result pages*, *bookmark requests*, *static HTML link* and *direct input* URL sources. If search engines (or other CGI-scripts [18]) work with the HTTP-GET method (instead of POST), the arguments of the request can be evaluated by logfile analyzing tools [14]. It is relatively easy to find out the keyword of the user search when looking at the request line, but an automatical extraction is

rather difficult, because the search engine providers can change the format every time and there are several possibilities to present the keywords together with several further information.

5. Summary and outlook

In this paper, we described the problem of extracting very specific statistical information from web server log data, sketched a solution and several pitfalls. The topology we worked on consisted of an Apache web server, an online authoring system called DAPHNE and the Smart Data Server (SDS) as central server to implement the calculating requests. The combination of these constituents posed comparatively few problems. Especially the flexibility of the SDS to implement diverse algorithms was persuasive.

There are some problems still unsolved and we plan to find their solution in the future. Particularly the complex of evaluation session traces can be very useful to help improving the quality of web sites.

References

- [1] *Apache*, <http://www.apache.org/>
- [2] *Netcraft*, <http://www.netcraft.com/survey/>
- [3] *CLF*,
<http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format>
- [4] *Perl*, <http://www.perl.org>
- [5] Roy Fielding et. al., *wwwstat*,
<http://www.ics.uci.edu/pub/websoft/wwwstat/wwwstat.html>
- [6] Stephen Turner, *Analog*, <http://www.statslab.cam.ac.uk/~sret1/analog/>
- [7] Boutell.com, Inc., *Wusage*,
<http://www.boutell.com/wusage/>
- [8] *Tabulate*,
<http://www.lib.ncsu.edu/staff/morgan/tabulate.txt>
- [9] *Yahoo*,
http://www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/ServersLog_Analysis_Tools
- [10] A. Heuer, Z. Zhang, T. Engel and C. Meinel. *DAPHNE - Distributed Authoring and Publishing in a Hypertext and Networked Environment*. In Proceedings of the International Conference IuK99 - Dynamic Documents, 1999. Jena
- [11] Z. Zhang, E.-G. Haffner, A. Heuer, T. Engel and C. Meinel. *Role-based Access Control in Online Authoring and Publishing Systems vs. Documentation Hierarchy*. In Proceedings of the SIGDOC '99, 1999. ACM
- [12] Uwe Roth, Ernst-Georg Haffner, Thomas Engel, Christoph Meinel. *The Smart Data Server: A New Kind of Middle-Tier*. Internet and Multimedia Systems and Applications, IASTED IMSA '99, Nassau, Bahamas, 1999
- [13] Uwe Roth, Ernst-Georg Haffner, Thomas Engel, Christoph Meinel. *An Approach to Distributed Functionality - The Smart Data Server*. World Conference on the WWW and Internet, AACE WebNet'99, Honolulu, Hawaii, 1999
- [14] *Hypertext Transfer Protocol*
<http://www.w3.org/Protocols/>
- [15] *Excel*, Microsoft™,
<http://www.microsoft.com>
- [16] *LaTeX*, Information about "LaTeX" can be found e.g. <http://www.cl.cam.ac.uk/TeXdoc/TeXdocs.html>
- [17] *GNUPLOT*, http://www.cs.dartmouth.edu/gnuplot_info.html
- [18] *CGI*, The Common Gateway Interface (e.g. <http://tiger.coe.missouri.edu/~research/cgi/>)