

# Optimization of Dynamic INTERNET Portal Content

Audrius Naslenas

Institute of Mathematics and Informatics, Akademijos 4, 2600 Vilnius  
[audrius@vpu.lt](mailto:audrius@vpu.lt)

**Abstract.** The business purpose of any Internet portal is to attract as much advertising as possible. The bigger the number of users, the more advertisers want to reach them. Therefore, every portal tends to aggregate as much content as possible in one place. To ensure repeat visits, the information must be updated as often as resources allow – the portal is highly dynamic. If we consider the fact that hundreds of thousands visitors flip tens of pages every day, while the information is updated several times during the day, it becomes evident that the underlying web server and database engines experience high loads. The paper discusses how to reduce server loads while ensuring dynamic content and ongoing update. The empirical data from no caching and two types of proposed caching systems have been compared. The obtained results allow drawing some conclusions about how to optimize loads of portals. The solution proposed in this paper has been applied in practice (<http://www.banga.lt/>) and is based on the empirical data collected from that project.

**Keywords:** portal, caching, server load.

## 1 Dynamic web pages

The simplest scheme of Internet browsing is illustrated in Figure 1: user requests a page, server sends back requested HTML document, which is rendered by the browser software on the screen of the user.

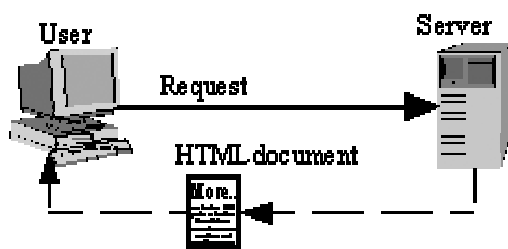


Figure 1. Simple scheme of web browsing

Two types of web pages can be pointed out: static documents, and dynamic documents.

Static documents are stored on the server in HTML format fully prepared for displaying in the browser. The server sends them without taking any other action; this gives static pages their biggest advantage, namely – speed and low server load. However, such documents are very inefficient in terms of maintenance: to change their design or update information, the administrator has to edit every single document. Therefore, static pages are best for small scale web sites.

Dynamic pages are written in specific programming languages (PERL, PHP, ASP etc.). When a dynamic page is requested by the user, the web server runs the programming language interpreter or compiled code which produces the contents of the documents based on the programmed logic and passed parameters. Therefore, a single script can generate thousands of pages. Dynamic pages cost more to produce, but pay back with ease of maintenance and update: change in the central script or template changes the design or layout of every single page produced by that script. When information is stored in a database, then even a layperson can update web sites without changing HTML templates or programming code.

Dynamic pages range from simple to quite complex: simple dynamic pages update just a small section of the document, while complex ones extract information from databases and generate full websites on the fly.

The biggest drawback of complex dynamic pages is that when the page is generated by software code, it results in higher server loads. When the visitor stream is huge, this becomes a critical problem.

## **2 Portals**

### **2.1 Features of a portal**

The goal of a portal is to attract the biggest number of visitors and turn them into repeat users, which allows to earn money from advertising revenues. Thus, portals try to publish loads of information (such as news, articles, forums, links, TV programs, movies), and need to update it as often as possible. Portals also allow visitors to express themselves in forums, opinions, suggestions, which helps in building virtual communities. To sum up, portals have a lot of information, and they also need to be very dynamic [1].

Major portals attract a large number of users. This results in huge server loads.

### **2.2 Solution of server load problem**

Solving the problem of server load is a complex and time-consuming activity; therefore, it is often neglected in the beginning, and becomes critical later. With the portals, availability is critical from the very start.

There are several ways to solve this problem.

*Optimal configuration and tuning.* Seldom the server is configured optimally for maximum load. Thus, there is slack that can be used up by tuning the server and the

operating system [2]. However, there are no universal recipes for this, and the efficiency gains are not that significant.

*Pre-caching* – a dynamic web site is turned into static, i.e. temporary pages are pre-generated and saved in the hard drive. Instead of dynamic files, static files are served to the user<sup>1</sup> <sup>2</sup>. This is the most popular way of solving the problem, but constant information make this method harder to implement.

There are ways to automate this (“Zend Cashe” <sup>3</sup>), but they update the page based on changes in the software code. This does not solve the problem of updating pages when information in the database is changed.

*Better hardware*, or clustering to distribute loads on several machines. This is an expensive method.

*Separating most dynamic parts*. Using this method, the most updated and seldom updated information is not placed on the same web page. The dynamic pages are left as-is, and for the less-updated pages the pre-caching method is used. However, this may cause problems to the usability of the web site<sup>4</sup>.

*Reject most dynamic elements*. To avoid the problem of caching, the highly dynamic information is not used at all.

## 3 Temporary cache

### 3.1 Problem formulation

Right after the launch of “Banga” portal<sup>5</sup> it became apparent that the web site is too dynamic. The server was working almost at 100% of its capacity. Buying a more powerful server was out of question, and the problem was to reduce the server load without sacrificing functionality or user experience. For this reason, separating dynamic information from semi-dynamic or getting rid of dynamic information was not acceptable.

Caching is the obvious solution, but there are some further problems: how to ensure that cached pages with blocks of information with different update cycles is always up to date?

A caching solution has been proposed and presented below. It is based on temporary files and is very related to the structure of the portal. This is the biggest drawback of the solution – changes in the portal structure require to review caching logic. In addition, there are some pages that must not be cached, so there is a need to

---

<sup>1</sup> *Controlling PHP Output: Caching and Compressing Dynamic Pages* (2000), <http://www.phpbuilder.com/>.

<sup>2</sup> *Portal Systems* (2001), [http://www.hotscripts.com/PHP/Scripts\\_and\\_Programs/Portal\\_Systems/](http://www.hotscripts.com/PHP/Scripts_and_Programs/Portal_Systems/).

<sup>3</sup> *Zend Cashe* (2001), <http://www.zend.com/>.

<sup>4</sup> *Controlling PHP Output: Caching and Compressing Dynamic Pages* (2000), <http://www.phpbuilder.com/>.

<sup>5</sup> *Portal “Banga”* (2001), <http://www.banga.lt>.

make sure they are not cached by mistake. However, the resulting server load reduction outweighs the drawbacks of the solution.

### **3.2 Solution description**

The user sees hundreds of different pages in the portal. However, there are only a dozen page templates, which specify which information blocks – objects – should be placed and where. Each object, given the required parameters, returns a piece of HTML code and information.

When the user navigates the portal, he is in effect passing different parameters to the central script, which identify the template number, and the parameters to be passed on to the objects. Based on information in the template, the right objects are invoked. The objects return HTML and information which is glued into a single HTML document, which is then sent to the user and also cached on the hard drive. Next time, the cached page is returned to the user without invoking any objects or database access. However, information in some objects is updated (this happens almost every minute in the forums) and in some it stays current for a week (i.e. in an article). We need to know, which objects have outdated information, and which not.

We separated the page into three parts: the header and footer (least dynamic), and body (most dynamic). The portal has a specific “registration log” which registers which page was cached and when, and also when the bits of information displayed on the page have been updated in the database. When an object is executed, it stores the time of this execution and identifies what information we are talking about in this log. When the user requests information, the server looks-up in the registration log to see if any information in the requested page has expired; if yes, the whole page is re-created and cached again, if not – the cached version is delivered to the user.

In other methods of caching the page has no separate agents that store information about their state; the page as a whole is cached and system deletes the cached pages after a period of time. When the users of the portal can update pages (as in forums), such methods are of no use.

### **3.3 Two types of caching**

Two types of caching system were devised and tested. The first type involves saving the cached pages in a database (A-Database), and the second type – saving as files (A-Files). It was expected that A-Database would be more efficient as databases are more effective than opening, writing, locking (to avoid simultaneous writing), closing files. A-Database could be slower only if the cached pages would significantly increase the database size. In advance, it can be noted that both types were almost identically efficient.

## 4 Efficiency of caching

### 4.1 Data collected during experiment

During the experiment, the time it takes to generate page was logged, and also the number of database access times was logged. Three days were used to collect data: first day was without any caching, second day with A-Files, and third day with A-Database. Efficiency was tested for:

- The whole portal
- The most dynamic part (forums)
- The most complex part (first page with many different information blocks)

### 4.2 Efficiency in the whole portal

Different parts of the portal are updated in different cycles; therefore, cache expires on different dates for different pages. In addition, there are pages not to be cached: such as personalization data, error reports, and the like.

As the Table 1 shows, more than half of requests were fulfilled by taking HTML document from the cache (as mentioned, three parts of the page were cached separately, thus “partial update” means that either header, footer, or body needed to be updated).

Table 1. Use of caching in page creation

	No cache	A-files	A-database
New page	100 %	8.5 %	9 %
Partial update	0 %	27.1 %	36.9 %
Cached	0 %	64.4 %	54.1 %

Figure 2 compares how caching reduces time needed to create a page. Using caching, three times more pages were delivered in the first second than without caching: thus, users get significantly faster response times with caching. The Figure 2 is a quantitative measure of caching efficiency.

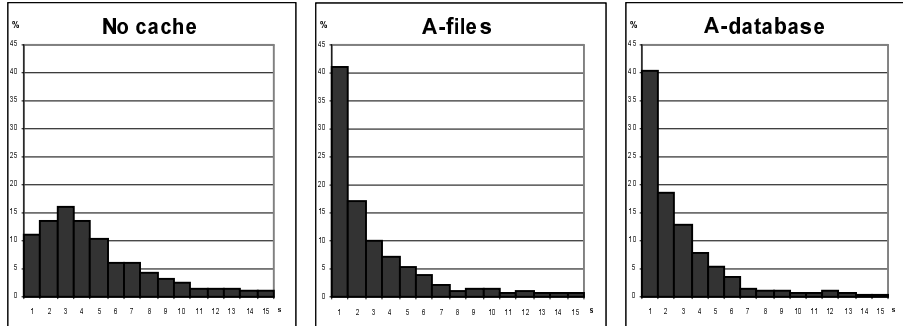


Figure 2. Page creation time

Data also shows that using caching the database was accessed two times less than without caching (see Figure 3), i.e. caching also reduced database load more than twice. Table 1 and Figure 3 shows a direct correlation between caching and reduction of database loads.

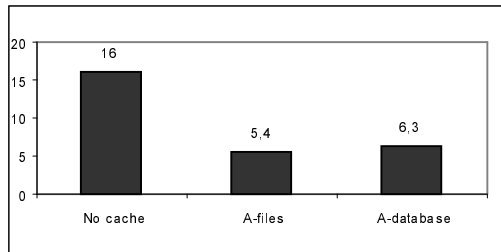


Figure 3. Average number of times database was accessed

### 4.3 Caching in the most dynamic part

It is very interesting to see how caching affects the most dynamic part of the portal – forums. Perhaps this section is so dynamic that caching is of no use? As the Table 2 shows, forums produce 38% of visitor stream, i.e. more than a third of portal load.

Table 2. Dynamic part traffic

No caching	A-files	A-database
41.3 %	35.9 %	38.5 %

Experiment showed that even the most dynamic section received major gains from caching. Table 3 shows that pages are generated every other time and even more seldom.

Table 3. Caching in the most dynamic part

	No caching	A-files	A-database
New page	100 %	0 %	0 %
Partial update	0 %	42.5 %	44.5 %
Cached	0 %	57.5 %	55.5 %

Figure 4 shows that using cache, a third of the pages are delivered in one second, when without caching, a more than third of the pages need more than one second.

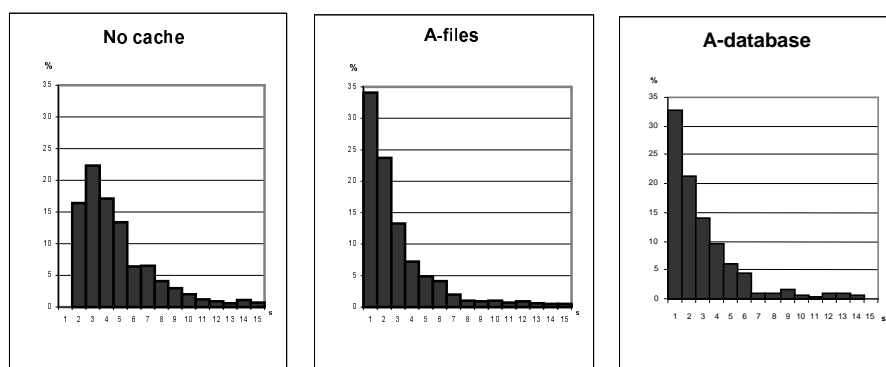


Figure 4. Page creation in the most dynamic part

Using cache, database requests were halved (see Figure 5). As a result, caching in the most dynamic part has reduced load by half.

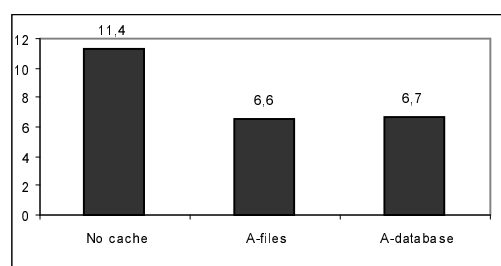


Figure 5. Average number of database requests per page in most dynamic parts

#### 4.4 Caching in the most complex part

The first portal page is the most complex and one of the most visited (Table 4 shows it gets 16% of all traffic); it produces the highest number of database access requests of any pages. How efficient is caching in such complex pages?

Table 4. First page traffic

No cache	A-files	A-database
16.8 %	16.8 %	15.8 %

Table 5 shows that first page used cached file more than 90 times out of a hundred. Due to complexity and lower dynamism caching was a very efficient solution. Users were able to access the page several seconds faster than before (see Figure 6). Due to caching, number of database requests was reduced 10 times (see Figure 7).

Table 5. Caching in the most complex page

	No cache	A-files	A-database
New page	100 %	0 %	0 %
Partial update	0 %	1.2 %	4.9 %
Cached	0 %	98.8 %	95.1 %

The data show that the most important page of the portal benefited significantly from caching: server response times and load were reduced up to 10 times.

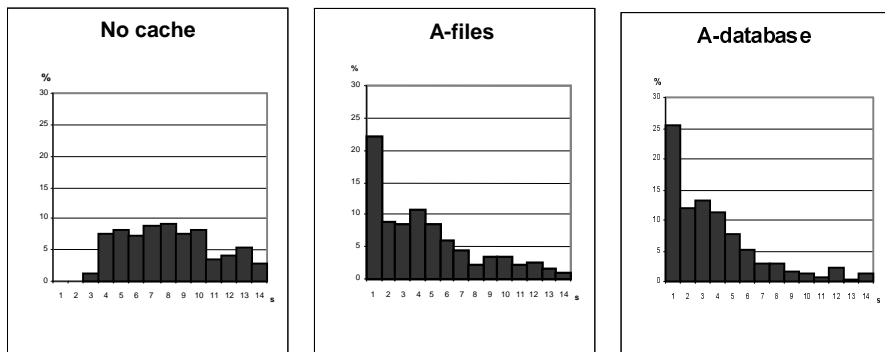


Figure 6. Page creation in the most complex part

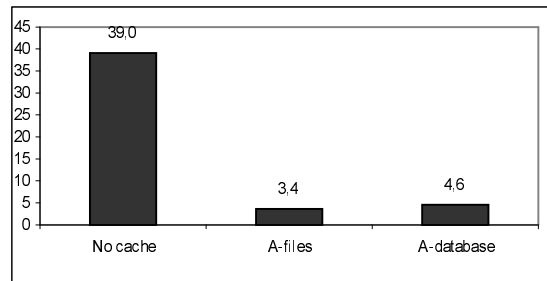


Figure 7. Average number of database requests per page in the most complex part

## 5 Conclusions

In the research, the empirical data from no caching and two types of caching systems have been compared. The obtained results allow drawing some conclusions about how to optimize loads of portals. The most complex and least dynamic pages benefit most from caching. A surprise was that even the most dynamic pages benefit significantly. The proposed caching system can be improved even more, but to balance time invested into such project and gains of efficiency, a simpler system serves quite well. Even if the server load is not that high, the caching system is still worth implementing, as the significant gains in server response times show. The caching depends on the portal structure. Therefore, the drawback of the solution proposed in the research – changes in the portal structure require to review caching logic.

Both A-database and A-file types of caching are equally efficient in our research, thus a choice can be made here on the basis of convenience. The reason is that the current project “Banga” uses a simple database engine “MySQL”. In other systems that use powerful database engines (such as “Oracle”) it is likely that A-database type of caching will be more efficient.

## References

1. L.Rosenfeld, P.Morville. *Information Architecture for the World Wide Web*. O’Reilly, 1998.
2. P.Killelea. *Web Performance Tuning*. O’Reilly, 1998.