Mining Web Usage and Content structure Data to Improve Web Cache Performance in Content Aggregation Systems

Carlos Guerrero
Department of Mathematics and Computer Science
University of Balearic Island
Palma, E-07122, Spain
Email: carlos.guerrero@uib.es

Carlos Juiz
Department of Mathematics and Computer Science
University of Balearic Island
Palma, E-07122, Spain
Email: cjuiz@uib.es

Ramon Puigjaner
Department of Mathematics and Computer Science
University of Balearic Island
Palma, E-07122, Spain
Email: putxi@uib.cat

***Abstract*—Web cache performance has been reduced in Web 2.0 applications due to the increase of the content update rates and the higher number of personalized web pages. This problem can be minimized by the caching of content fragments instead of complete web pages. We propose a classification algorithm to define the fragment design that experiences the best performance. To create the algorithm, we have mined data of content characterization, user behaviour and performance. We have obtained two classification tree as result of this process. These classification trees are used to determine the fragment design. We have optimized the model of a real web site using both classification trees and we have evaluated the user observed response time. We have obtained significant results which prove that the optimization of the fragment designs can achieve high speedups in the user perceived response time.**

***Keywords*-Web caching; classification trees; web performance engineering; web content aggregation.**

I. INTRODUCTION

Web Caching is a widely used technique to save bandwidth, to reduce server workload and to improve user response time, i.e., Web Caching improves the performance of web architectures. This improvement is based on the *reusability* of web responses between different users and requests. This happens when several users request the same page or when a user requests the same web pages at least twice before its content changes.

In current web architectures, especially in Web 2.0 systems, content changes are more usual and the personalization of web pages is allowed. The behaviour of user generated content and pages created by collaboration are more unpredictable [1]. As a result of this, the web responses that are stored in the web cache become not *reusable*. It is widely accepted that this problem can be solved by reducing the minimum chacheable unit: content fragments instead of whole web pages [2], [3]. Nevertheless, from a performance point of view, there is a dilemma [4]: on one hand, a high level of fragmentation (a big number of content fragments) improves hit ratio, but response time could be increased due to overhead connection and fragment joining times; on the other hand, a low level of fragmentation (a small number of fragments) minimizes overhead times but it makes hit ratio worse. So that, the problem results on determining when it is better to serve two fragments together (joined) and when it is better to do it separately (split).

To deal with this problem, we use a tree-based classification algorithm which optimizes the performance of the system. This algorithm uses the characteristics of the contents of the page (content fragments sizes, update rates, request rates, ...) to obtain a design of the web pages (which elements are served split and which ones are served joined) for an optimal performance.

We have compared the performance of using *joined* and *split* states in a high number of synthetic pages. We have applied data mining algorithms to the data obtained in these exploration phase and we have produced different classification trees. These classification trees have been tested in a web page model extracted from a real system (The New York Times web site, http://www.nytimes.com). The performance results obtained show high speedups of them with the two basic design alternatives: either all the fragments are joined or all the fragments are split.

Our main contribution is the developing of a *classification algorithm* which improves the performance of the systems in where web pages are created by the aggregation of content atomic fragments. This solution can be applied to systems in where these fragments can be served joined or split. We represent these two ways of delivering the content fragments by a state of the aggregation relationship. The performance of the system changes depending on the state of each aggregation relationship. The inputs of the algorithm are the characteristics of the fragments.

In Section III, we give the details about how the content aggregation application works and about the model we use to represent the contents fragments, the web pages, the characterization parameters of the fragments and the states of the aggregation relationships. In Section IV, we explain in which type of applications we can use our propose and how it fits in these applications. We explain the process

 ISBN: 978-1-61208-127-4

and the results of obtaining the classification trees in Section V. Finally, we analyse the performance of a system where we have implemented our solution to optimise the page fragment design (Section VI). Last section is used to explain the conclusion of our work and possible future works (Section VII).

II. Related Work

The application of techniques based on decision trees and data mining to improve the performance of the system, and more concretely, the performance of web architectures is not either new.

Pallis et al. [5] present a clustering-based scheme in order to improve the performance of a prefetching and web caching system. The input of the data mining process is the users' access patterns. Their solution identifies clusters of correlated Web pages. The idea of mining users' access sequence is also present in studies of Huang et al. [6]. They propose a system to mine popular surfing sequences to obtain rules. The rules will be used by a prediction-based manager to achieve better buffer utilization. Kumar et al. [7], use data mining processes to detect dynamic deviations from normal usage patterns of the users. But the problem of improving the fragmentation design of the web pages have not been addressed before by using classification trees. Our work differs form the previous in two main aspects: the scope and the inputs parameters of the data mining process. Our propose improved the performance of the system by determining the fragmentation design of the content in web pages. The other researches improve it by doing more accurately the selection of the pages to be evicted from the cache or the next web pages to be prefetched. Finally, they only use user access patterns as inputs of the data mining process. In the other hand, we also take into account the structure of the web pages.

Other approaches have deal with the same scope than us: to determine the fragmentation design of the content web pages [8]. But they have based the solution on cost-benefit functions. Their studies are in preliminary phases and they have not validated it in real web systems. We are not able to contrast our results with them.

Similar methods have been used in the study presented by Li et al. [9]. Their objective is to predict the performance for services instead of content delivery. Their study also differs in the data analysed. They use object timing information from packet traces. We use data extracted from higher layers (application layer).

III. Content Aggregation Model and Performance

Web systems have evolved from static and general content pages to dynamic generated, personalized and updated content pages. These characteristics are specially present in a group of web applications, for example: content aggregation

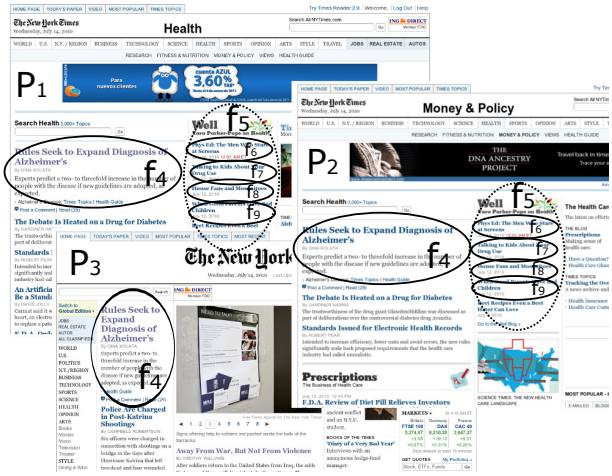


Figure 1: Example of pure content fragments, aggregated fragments and web pages from a real web site (The New York Times, http://www.nytimes.com)

systems, Web 2.0 applications, weblogs, newspaper sites, etc.

Content aggregation technologies are systems that combine content from different sources to create new content elements. These contents could be extracted from server-local databases or from remote systems using RSS, XML, or similar technologies. There are two main types of aggregation systems: mashups and portlets. The differences between them are related to the side where the aggregation takes place: client-side or server-side, respectively. For example, newspaper sites and weblogs are content aggregation systems where contents are stored in server-local database and the contents are aggregated on the server (this aggregation can be done by the application server, the web server or the proxy cache).

So, in these content aggregation systems, pages are created by the aggregation of independent contents (for example, news in a newspaper site or entries in weblogs). These contents are usually included in more than one web page (for example, news included in political and economy section or entries in a blog classified by tags and by sections). But the structure of a web page in these systems can be more complex than aggregations of only one degree of depth. The aggregation of a set of contents can be considered as a new content fragment which could be aggregated by other pages of fragments (for example, the group of highlighted news or the group of the most recent entries in a weblog).

Therefore, we could distinguish between two content element types: pure content elements (which are created/stored by a user/system) and aggregated content elements (which are created from the combination of others content elements). Also we consider another type of elements, the aggregation relationship which defines that a fragment includes the content of other fragment (pure or aggregated). The num-

 ISBN: 978-1-61208-127-4

ber of aggregation relationships is unlimited. In Figure 1, we can observe an example of three web pages extracted from the The New York Times web site (P_1, P_2, P_3). Three of them share the aggregation of a pure content fragment (f_4) and two of them share an aggregated fragment (f_5). This aggregated fragment is created by the aggregation of several fragments ($f_6, f_7, f_8, ...$).

Authors of [10] define an Object Dependence Graph (ODG) to model web pages that are created by aggregation of contents. An ODG is a Directed Acyclic Graph (DAG) where pure and aggregated content elements are represented by vertices (nodes). ODGs have two vertex types: P_i which represents parent vertices (user web pages) and f_i which represents pure or aggregation fragments. The sink vertices correspond to pure elements and source vertices to user web pages. The rest of the vertices correspond to aggregated elements. Finally, the edges of the graph represent aggregation relationships between a pair of elements.

We use ODGs in our system to represent the design of the web pages. Nevertheless, ODGs are not able to model all the information we need. Thus, we have extended the ODG model to include information about the characteristics of the fragments and information about how fragments are delivered by the server: joined or split (Figure 2).

Our main goal is to determine the fragment design of a web pages which experiences a better performance. The fragment design is the information that refers to how the fragments are delivered by the server and stored by the web cache: if a set of contents are delivered and stored independently (split) or together (joined). This information can be represented in a ODG using labelled edges which indicates if a pair of content fragments are joined or split.

In the results presented in [11], we studied the relation between characterization parameters of the content fragments and the performance experienced by the web cache for aggregated content systems. We concluded that, in these type of web sites, there is a strong correlation between compared performance speedups of aggregation states (joined or split) and the characterization parameters for a given pair of related content fragments. The characterization parameters that we took into account in the study were: for both fragments, update rates, request rates and sizes; for the *father* fragment, the number of *children*; and for the *child* fragment, the number of *fathers*. We conclude also that other parameters as fragment service times or ODG structure characteristics (as fragment depth degree in the ODG) are not able to predict the difference of the performance between joining or splitting two fragments.

Our classification algorithm uses these characterization parameters, for a given pair of fragments, to determined if it is better to deliver them joined or split. Some of this information is in the ODG (number of fathers and children of a vertex) but other is not (fragment update rate, content size, ...). We have added some attributes to each vertex to represent these characteristics.

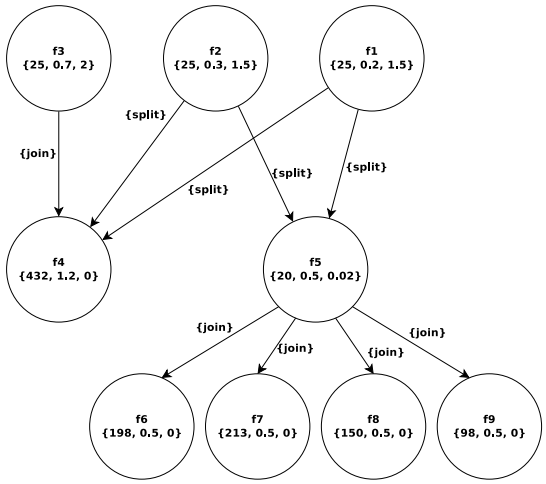


Figure 2: ODG of the example of Figure 1. The text in vertices corresponts to the identificators of the fragments and the characterization attributes vector $\{$size in bytes, request rate in seconds^{-1}, update rate in seconds$^{-1}\}$

To sum up, our web page model is represented by a DAG where the edges represent the aggregation relationships and they are labelled with the state of the aggregation (the contents are joined or not in the server). The vertices have a direct correspondence with the content fragments of the web page. Sink vertices are pure content fragment (user or third-party generated), source vertices are the user web pages and the rest of vertices are groups of aggregated contents. Each vertex has additional information about its request rate, update rate and size (Figure 2).

In Figure 2 we can observe a ODG example. The labels of the edges and the values of the characterization attributes vector have been chosen randomly. We can observed, in one hand, that the web server, using this model, has all the information to determine how to serve the web pages (labels of the edges). In the other hand, our design optimization algorithm also has all the information required to determine the states of the aggregation relationships (characterization vector and number of incoming and outgoing edges).

IV. Applicability of the solution

Our propose is applicable to web architectures that are able to manage fragments of contents. In these types of architecture, web server is able to serve fragments of the web pages instead of only complete HTML document. These fragments must be related using special tags inside the HTML code. Edge Side Includes (ESI) is the *de facto* standard used for that.

If a server responds using fragments of the pages, they must be gathered together in other tier of the system in order to to create the complete web page. This task could be done by client browsers, CDN networks, web proxies, web

 ISBN: 978-1-61208-127-4

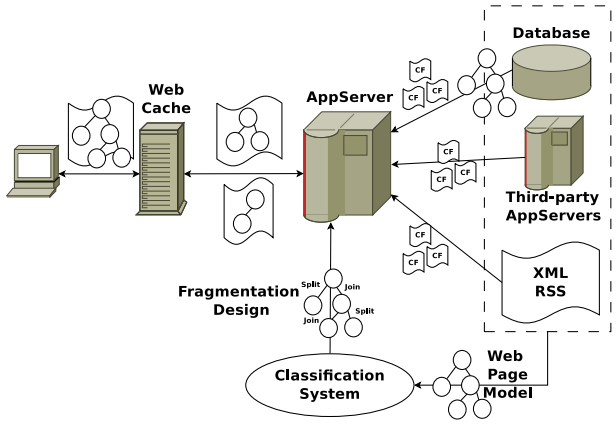


Figure 3: Architecture of content aggregation systems and fragments Web Caching

caches, etc. Nevertheless, if the task is done by elements placed between the user and the web cache, the web cache is able to manage fragments of web pages (it is able to store fragments independently and request them to the web server). As a result of this, the minimum cacheable units are content fragments instead of whole page. Therefore, the performance of the web cache will depend on the groups of fragments, i.e., the content fragments that the web server will join or not.

In Figure 3 we can observe the general architecture of a content aggregation web system. In this model, the cache is the element in charge of create the HTML documents using the fragments responded by the web server. The applications, which are based on this model, are able to manage the contents independently. The contents can be extracted from local databases, third-party applications or by the use of feeds. The application server requests all the contents and gets together them according the page design. The main goal of our research work is that the web server will create the design which makes the web cache experience the best performance. In this paper we present an optimization algorithm to solve the problem of obtaining the best web page design.

This application layout is not applicable to all kind of web applications. The applicability is restricted to web systems where the pages can be easily fragmented. Aggregation content applications satisfy this feature. In these types of applications, the atomic fragments of the page can be associated with the independent contents extracted from data sources. These correspond to pure content elements in our model. There are an important number of applications that are based on the aggregation of contents. Next elements can be considered as pure content elements which are aggregated to create complete web pages: news in a newspaper website; entries in a weblog; user update advices in social networks; users update advices in social networks; gadgets in personalized home page systems; bookmarks in social tagging systems; feeds in online feed reader systems; etc. Therefore, the solution we propose can improve the performance of a wide range of traditional web applications and in the most of Web 2.0 applications.These types of applications usually are developed allowing to the web server to manage fragments of the pages. If not, the additional work to develop this feature is not very difficult because the pure content fragments are managed independently. In both cases, the web application has to be upgraded to manage groups of fragments. It has to join the contents with *joined* state. But this feature is also simple to be implemented.

V. Optimization algorithm

In previous sections, we have presented the problem we have to deal with: adapt the design of the content fragments of a web page to improve the performance of a cache system. To adapt the design, we limit the parameters to use to a subset of the ones available before that the web page is requested. As we have justified in Section III, these parameters are: (a) request rates; (b) update rates; (c) content sizes; (d) number of aggregations of and over a fragment. We name them as characterization parameters . Therefore, these parameters are the inputs of the classification algorithm we want to create.

The output of our algorithm is the fragment design of a web page. The algorithm decides which fragments must be requested grouped (or joined) and which ones must be requested independently (split). So, the outputs of the algorithm are the states of each aggregation relationship.

Once we have decided the inputs and outputs of our algorithm, we need to determine the scope of action of the algorithm. The set of characterization parameters to be used by the algorithm can be done taking into account: all the vertices of the ODG; a subset of vertices; only one vertex; the pair of vertices related by an aggregation. In [11] we study how the characterization parameters of the related pair of vertices are enough to decide the state of relationship (if both vertex are delivered joined or split).

In order to develop the classification algorithm, we have decided to mining the data monitored on an emulation application. The model for the pages and the behaviour of the users is synthetic. We randomly produced the pages of an content aggregation system: we produced enough content fragments to cover a wide range of values for every characterization parameter; these contents are randomly aggregated to create several web pages. We assigned update rates randomly to each content fragment (during the emulation phase, a daemon uses these values to change the content). Finally, we developed a user-emulator that uses random values to request the web pages. All the random values of all the parameters are created using uniform distributions. There are a wide number of studies that determine that these parameters follows specific statistical distributions in real

 ISBN: 978-1-61208-127-4

systems [12], [13]. But in this first phase, we are not only interested in samples from usual scenarios. We also want to cover a wide range of values in order to study a big search space. This search space will give us information either of usual scenarios and extreme ones. This is the reason because we use uniform distributions to create the samples.

Data mining algorithms need a set of data that is used to extract knowledge from it. This is usually called training data set. It usually also needs a second set of data to evaluate the credibility of the knowledge extracted from the first set. This is called test data set. This second set is used to test the outputs of the knowledge model created. This is the reason because we created two different page models. Each of the models had over 5000 pure content fragments and 12000 aggregation relationships creating over 1500 user web pages.

To study the influence of the aggregation states in the performance of the web cache, we randomly assigned *joined* and *split* states to the aggregation relationships and we changed periodically the state of one of the aggregation of each web page. By this way, we could monitor the user observed response time of the web pages for both states of a given aggregation relationship. Both web page models are emulated during the enough time to get confiable meanings. At this point, we were able to create a vector with the characterization parameters of the father fragment and the child fragment of the aggregation and performance information for the aggregation states (joined and split). This performance information is represented by an attribute which indicates if the performance is better for *split* or *joined* state. The vector pattern is:: ((CharacterizationParameters$_{father}$), (CharacterizationParameters$_{child}$), ResponseTime$_{joined}$, ResponseTime$_{split}$) where (CharacterizationParameters$_i$) is (RequestRatio$_i$, UpdateRatio$_i$, Size$_i$, FathersNumber$_i$, ChildrenNumber$_i$).

We preprocessed the data before we applied data mining on it. We transformed both response time attributes in a class one (PerformanceClass = {JoinedClass, SplitClass}). This (ResponseTime$_{joined}$ $>$ ResponseTime$_{split}$) (ResponseTime$_{joined}$ $<$ ResponseTime$_{split}$). Finally, the attributes ChildrenNumber$_{child}$ and FatherNumber$_{father}$ does not influence on the performance; so, they were eliminated from the data set.

At this point we had two data set represented by instances of the vector of characterization parameters which was used as input of the data mining algorithm and a class attribute which is the attributed to infer with the knowledge extracted. In [14], we presented a rule-based system that infers the class attribute. The rules were created manually observing the classes created after apply a clustering process to the input data set. The clustering process was done using WEKA [15]. WEKA is a collection of machine learning algorithms for data mining tasks which could be used in a GUI application or integrated in your own JAVA programs. With the obtained rules we assigned the states of the aggregation to a ODG of a real web site and we emulate the behaviour of the user to obtain performance results (in Section VI we explain the details of the validation process). The results obtained were positive but not very significant: the optimized page model had an average speedup of 1.0488 with the page model where all the aggregation are split (all the fragments are served independently); and it had an average speedup of 1.1970 with the page model where all the aggregation are joined (all the fragments are served together). The reason of these modest results is the low credibility of the data mining process. We obtained a coverage of 40% after testing the rules over the test data set, This low coverage is explained by the way the process to extract the rules was made: manually by the observation of the obtained clusters.

Table I: Summary of the complexity of the classification trees

	Number of leaves	Size of the tree
$CTree_{abs}$	46	91
$CTree_{rat}$	32	63

This previous result helped us to conclude we have to improve the data mining process. This is the reason because we changed the algorithm to assign the states of the aggregations. Instead of a group of rules extracted from a clustering process, we have tried to develop an algorithm based on classification trees. The machine learning scheme used to obtain the classification tree is C4.5 algorithm [16]. We have used WEKA to obtain the classification tree. The implementation of the C4.5 algorithm in Weka is called J48. We used the default set up parameters of the C4.5 algorithm.

We have used two different input data sets. The data was the same in both sets but not the way to express it. In one of them we represented the absolutes values of all the parameters. In the second one, we used the ratios of the characterization parameter shared by both fragments: RequestRate$_{father}$/RequestRate$_{child}$, UpdatetRate$_{father}$/UpdateRate$_{child}$ and Size$_{father}$/Size$_{child}$; while we used the absolute values of the rest of the parameters: ChildrenNumber$_{father}$ and FatherNumber$_{child}$. We have respectively called CTree$_{abs}$ and CTree$_{rat}$ the trees obtained from these data sets.

In this section, we have explained how has been created the classification trees. These models are used to optimize the design of the web pages in order to improve the performance. In the next section (Section VI), we analyse how the web design models obtained with both classification trees affect this performance.

VI. Performance analysis

In order to validate the method we have proposed to improve the performance of web cache systems we need to monitor a real application and compare the results using the traditional fragment designs (all the fragments joined

 ISBN: 978-1-61208-127-4

and all the fragments split) with the results using the fragment design determined by our classification algorithm. To achieve this goal we have to deal with three problems: (a) how to create a model of a real system (b) how to monitor and execute the emulator; (c) and how to compare the results. We explain them in following sections.

A. Modelling a real system

The training data set used to obtain the classification trees was randomly created using uniform distributions because our goal was to cover a wide range of values. Otherwise, we are interested in using results from a real system in order to validate the use of our design optimization algorithm. Due to we are not able to use our solution in a representative web system, we need to emulate it. The models used by the emulator can be created by the use of suitable statistical distributions or mining data from a real system. We have used a combination of the two alternatives.

The model created for our emulator is based on the The New York Times web site. We have mined all the data which a final user can do. This data is related to structure of the pages. Thus, we have mined data about the structure of the aggregation relationships (news and pages created as aggregation of news) and about the size of the contents of each news. This process took place during one week on March 2010, and we got 3082 fragments aggregated in 482 pages using 13979 relationships.

We are not able to obtain information about the behaviour of the users and publishers of the The New York Times web site. This is the reason because we have to use statistical distributions in order to create the parameters of the model referred to update rates and request rates. The number of studies which characterize the behaviour of the users of current web system is very large. But it is commonly accepted that the popularity of web objects follows a power law distribution [13], [1]. These studies also detail the more usual values for the parameters of these statistical distributions. In our case, the usual accepted solution is a power law with $\alpha = 0.83$ for request rates and $\alpha = 0.54$ for update rates, both with $R^2 = 0.99$ [17].

B. Monitoring emulator performance

The emulator has been executed several periods of time to obtain performance results for the four fragment design scenarios. The four emulation scenarios are: (a) *SplitScenario*, fragments are served independently, so the almost 14000 relationships of our model are labelled as *split* (0 joined and 13979 splitted); (b) *JoinedScenario*, fragments are served joined, relationships are labelled as *joined* (13979 joined and 0 splitted); (c) *CTree$_{abs}$OptimizedScenario*, relationships are labelled using the classification tree created with data set of absolute values, CTree$_{abs}$ (12815 joined and 1164 splitted); (d) *CTree$_{rat}$OptimizedScenario*, in this last scenario we use the classification tree created with data set of ratios values, CTree$_{rat}$ (8544 joined and 5435 splitted). Table II is a summary of the states values for each of the scenarios.

Table II: Summary of the states for the four scenarios

	Number of joined states	Number of split states
JoinedScenario	13979	0
SplitScenario	0	13979
CTree$_{abs}$OptimizedScenario	12815	1164
CTree$_{rat}$OptimizedScenario	8544	5435

Table III: Hit ratios and byte hit ratios of the emulation scenarios

	Hit ratio (%)	Byte hit ratio (%)
JoinedScenario	8.4184	10.4679
SplitScenario	91.7699	80.7882
CTree$_{abs}$OptimizedScenario	60.0031	18.2495
CTree$_{rat}$OptimizedScenario	78.9625	18.2986

For each of the scenarios, we have run the emulator the enough time to obtain reliable mean values of the response times. After deleting all the samples of the transient period, we get more than 300,000 requests during 20 hours of execution for each of the scenarios, obtaining a concurrence level of 4 requests/second. The transient period is the initial period of the execution where the elements in the architecture are not in a stationary state. We consider the stationary state is reached when the hit ratio of the cache is stabilized.

C. Results analysis

To show the strength of our new scenarios (*CTree$_{abs}$OptimizedScenario* and *CTree$_{rat}$OptimizedScenario*), we need to compare with other scenarios. We have considered that the traditional scenarios for aggregation applications are good comparison elements. These scenarios are the ones in which : the fragments are all served independently (*SplitScenario*) and the fragments are served as a whole web page (*JoinedScenario*). We have focused in the user response time metric to study the performance of the system, because we are more interested in improve the user performance metrics than in the server performance metrics.

In general, *CTree$_{abs}$OptimizedScenario* shows the best response times (Figure 4). This is explained because the classification tree used to create the fragment design of this scenario is more complex than the one used to create the design of the CTree$_{rat}$OptimizedScenario. Usually, when you have a more complex classification tree, the result of the classification is more accurate. We need to study if the complexity of the tree influences the computational requirements. In this case, we will have to balance the overhead of the execution of the algorithm and the performance improvement obtained by its page model. This work is out of the scope of the presented work, but it is considered as future work.

 ISBN: 978-1-61208-127-4

If we compare the performance of the optimized scenarios with the traditional ones (Figure 4), we can observe the speedup is bigger when we compared with the *JoinedScenario* than with the *SplitScenario*. This is because, as we mentioned in Section I, one of the first improvements for caching in content aggregation applications is to serve the fragments of content independently [18].

Finally, we analyse the hit ratios for the scenarios. In current web systems, where the updates are more usual and the pages are more personalized, invalidation times are shorter and, as a result of this, the system reduces its hit ratios. We can see that in Table III for the *JoinedScenario*, which has lower ratios in comparison with the other scenarios. If we compare the number of joined elements (Table II) and the hit ratios (Table III), we can observed a clear correlation between them. The bigger the number of joined elements is, the worse the hit ratio is.

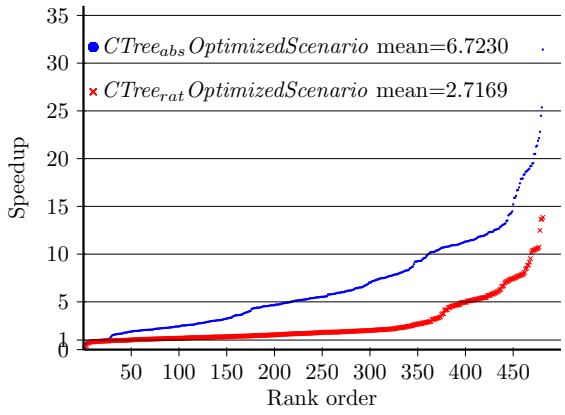
The scenarios with best response times are not the ones which has the best hit ratio. As we explained at the beginning of the document, it is best to sacrifice the hit ratio (by joining fragments) in order to reduce the overhead times of having too many fragments. This is the reason because the $CTree_{abs}OptimizedScenario$ shows the best response time and the worse hit ratio. The $CTree_{abs}OptimizedScenario$ takes profit of reducing overhead times in spite of it gets worse the hit ratio.

It can be very surprising that, between the optimized scenarios, the byte hit ratio is almost equal, but the hit ratios are considerably different. It is also to emphasize the high difference between the hit ratios and the byte hit ratios of both optimized scenarios. This can be explained considering that the mining process was able to inference that it is more important to obtain cache hits in small fragments than in big ones [19], [20].

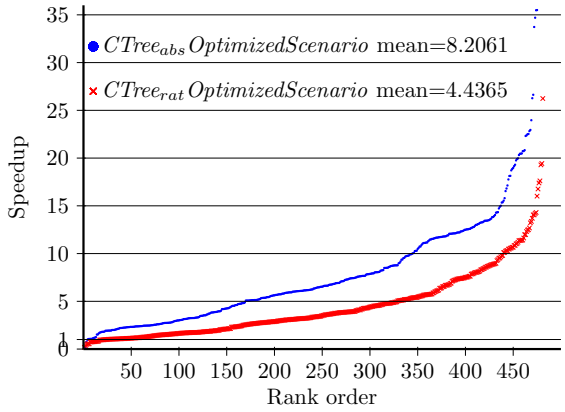
After the analysis of the results, we can conclude that the optimized scenarios show best response times in spite of their worse hit ratios. So it is interesting to adapt this improvement if we focus on improving user performance metrics. To get conclusions about server performance metrics, we would need to analyse how affects these worse hit ratios to the performance of the servers. This study is out of the scope of this work. It is consider as future work.

VII. Conclusion

We have presented an improvement for web caches used in content aggregation systems. These systems have worse cache performance than others because content aggregation systems have high content update rates and high page personalization level. To solve this problem, we have presented a algorithm based on classification trees, which tries to find the best fragment design of the web page in order to minimize the overload times of communication and joining process and to maximize the hit ratios of the web cache.



(a) *SplitScenario* ($Speedup = \frac{RT_{Split}}{RT_{CTreeOpt}}$)



(b) *JoinedScenario* ($Speedup = \frac{RT_{Joined}}{RT_{CTreeOpt}}$)

Figure 4: Rank order chart of the response time speedups

To create the classification trees we have used data mining techniques, more concretely, we have used the J48 algorithm (the Weka's implementation of the C4.5 algorithm). The inputs of the algorithm are a subset of the characterization parameters of the content fragments of the web pages. The output of the algorithm is the state for an aggregation relationship between two content fragments. These states (*joined* and *split*) indicate if the server has to serve the fragments together or independently. We have extended the fragment web pages representation presented in [10] in order to cover all our modelling requirements.

The training and test data sets have been created with the results monitored in an emulation application. The page models and user behaviour models have been created randomly. They cover a wide range of values for the characterization parameters. Depending on the way we express the characterization parameters we can obtain different data sets and different classification trees. We have studied the coverage level of the most usual ways to express the parameters

 ISBN: 978-1-61208-127-4

and we have finally chosen two classification trees: one tree using the absolutes values of the parameters (CTree$_{abs}$) and the other using the ratios of the parameters shared between the two related fragments (CTree$_{rat}$).

In order to validate the use of the classification trees, we have tested them in a emulation application which uses a page model extracted from a real web site, and a user behaviour model created using the most suitable statistical distributions. We have monitored the user response times in the execution of the emulation scenarios (the two corresponding to the classification trees, the scenario with all the fragments joined and the scenario with all the fragments split).

As future work, we need to contrast the results obtained in these experiments with others web pages to validate that the strength of our solution in general terms. Finally, we need to study how influence alternative classification trees.

ACKNOWLEDGMENT

This work is partially financed by the Spanish Ministry of Education and Science through TIN2007-60440 project and TIN2009-11711 project.

REFERENCES

[1] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 1–14.

[2] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, "Automatic fragment detection in dynamic web pages and its impact on caching," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 6, pp. 859–874, 2005.

[3] C. Yuan, Y. Chen, and Z. Zhang, "Evaluation of edge caching/offloading for dynamic content delivery," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 11, pp. 1411–1423, 2004.

[4] C. Guerrero, C. Juiz, and R. Puigjaner, "The applicability of balanced esi for web caching," in *Proceedings of the 3rd International Conference on Web Information Systems and Technologies*, March 2007.

[5] G. Pallis, A. Vakali, and J. Pokorny, "A clustering-based prefetching scheme on a web cache environment," *Comput. Electr. Eng.*, vol. 34, no. 4, pp. 309–323, 2008.

[6] Y.-F. Huang and J.-M. Hsu, "Mining web logs to improve hit ratios of prefetching and caching," *Know.-Based Syst.*, vol. 21, no. 1, pp. 62–69, 2008.

[7] C. Kumar and J. B. Norris, "A new approach for a proxy-level web caching mechanism," *Decis. Support Syst.*, vol. 46, no. 1, pp. 52–60, 2008.

[8] O. A.-H. Hassan, L. Ramaswamy, and J. A. Miller, "Mace: A dynamic caching framework for mashups," in *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 75–82.

[9] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. Greenberg, and Y.-M. Wang, "Webprophet: automating performance prediction for web services," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10. [Online]. Available: http://portal.acm.org/citation.cfm?id=1855711.1855721

[10] J. Challenger, P. Dantzig, A. Iyengar, and K. Witting, "A fragment-based approach for efficiently creating dynamic web content," *ACM Trans. Internet Technol.*, vol. 5, no. 2, pp. 359–389, 2005.

[11] C. Guerrero, C. Juiz, and R. Puigjaner, "Web cache performance correlation with content characterization parameters in content aggregation systems," in *Proceedings of the XXXVIth Latin American Informatics Conference*, 2010.

[12] M. A. Goncalves, J. Almeida, L. Santos, A. H. Laender, and V. Almeida, "On popularity in the blogosphere," *IEEE Internet Computing*, vol. 99, no. PrePrints, 2010.

[13] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao, "Analyzing patterns of user content generation in online social networks," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2009, pp. 369–378.

[14] C. Guerrero, C. Juiz, and R. Puigjaner, "Rule-based system to improve performance on mash-up web applications," in *Distributed and Artificial Intelligence: 7th International Symposium*, 2010.

[15] Weka Machine Learning Project, "Weka," URL http://www.cs.waikato.ac.nz/~ml/weka, October 2010.

[16] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.

[17] F. Duarte, B. Mattos, A. Bestavros, V. Almeida, and J. Almeida, "Traffic characteristics and communication patterns in blogosphere," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2007.

[18] C. Yuan, Y. Chen, and Z. Zhang, "Evaluation of edge caching/offloading for dynamic content delivery," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM, 2003, pp. 461–471.

[19] L. Cherkasova, "Improving www proxies performance with greedy-dual-size-frequency caching policy," HP Technical Report HPL-98-69, Tech. Rep., 1998. [Online]. Available: http://www.hpl.hp.com/techreports/98/HPL-98-69R1.html

[20] L. Cherkasova and G. Ciardo, "Role of aging, frequency, and size in web cache replacement policies," in *In Proceedings of HPCN Europe*. Springer-Verlag, 2001, pp. 114–123.

 ISBN: 978-1-61208-127-4