Web-Based System Configuration and Performance Evaluation using a Knowledge-Based Methodology

Mara Nikolaidou[1], **Dimosthenis Anagnostopoulos**[2]

[1] Dept. of Informatics, University of Athens,
Panepistimiopolis, 15771 Athens, Greece
mara@di.uoa.gr

[2] Dept. of Geography, Harokopion University of Athens,
70 El. Venizelou Str, 17671, Athens, Greece
dimosthe@hua.gr

Abstract. Since Internet dominated the world, the World Wide Web platform is used as a type of middleware providing a common platform for Intranet-based and Internet-based application development. Web-based applications have become more complex and demanding, in order to fulfil extended user requirements. In this paper, we propose a systematic approach for the configuration, modification and performance evaluation of web-based systems. Its contribution involves the employment of knowledge-based techniques for the design of web-based systems and the description of problems encountered and the solutions proposed. Emphasis is given on the extendable modelling scheme used to depict web-based application functionality and estimate application requirements from the network infrastructure. Web-based system architectures are designed and evaluated using IDIS environment.

1 Introduction

The enormous success of the Internet is mainly based on the World Wide Web (WWW), built to facilitate access to multimedia documents distributed all over the Internet through a common interface, i.e. a Web Browser. Using a Web Browser is possible to download that part of any application that consists of its user interface from anywhere in the world. Such applications are considered as *web-based applications*, and are built based on the multi-tiered client-server model [1, 2]. The first tier, e.g. the user interface or user service, is implemented using the WWW platform, while the other tiers implement the specific application logic that may be based on different architectures, as discussed in [3]. Thus, WWW platform can be viewed as a type of middleware providing a common platform of Intranet-based and Internet-based application development. Many commercial information systems, such as banking and ordering systems, distant learning environments and workflow management systems, fall in this category. Development of standards, such as CORBA, allowing the interaction between heterogeneous, autonomous applications and of programming languages, such as Java, providing native distributed programming support established a well-defined platform for web-based application development. A *web-based system* can be described as a set of web-based applications and the underlying infrastructure. Web-based system configuration is based on the successful combination of interacting components spread over the Internet, also facing the internal complexity of these components [4]. The configuration issue is, thus, a multidisciplinary one, imposing examination of a large number of alternative architectural solutions, exploitation of different replication scenarios and estimation of proposed architectures' performance. Complete and accurate description of web-based application functionality is a critical factor in web-based system design. It ensures the accurate estimation of QoS provided by the network infrastructure and the efficient performance evaluation of the overall system.

Simulation tools usually investigate the behaviour of predefined algorithms for the placement of resources and processes, or estimate the performance characteristics of a given network architecture, performing a "what-if" analysis [5, 6, 7]. Such tools do not make suggestions for the design or redesign of the system architecture. When configuring complex systems, experts rely more on experience than on theory-based calculations [8, 9]. Web-based system configuration issues requires dealing with interrelated problems, such as process and file allocation, which are NP-complete. Dealing with such problems requires methods that are more heuristic than algorithmic in nature. Expert system research has often concentrated on the representation and manipulation of heuristic knowledge and its use in *information system design* and *network configuration* problems [10, 11, 12].

In this paper, we describe a systematic approach for the configuration, modification and performance evaluation of web-based systems. Its contribution involves the employment of knowledge-based techniques for the configuration of web-based systems, the description of problems encountered and the solutions proposed. Emphasis is given on the extendable modelling scheme used to depict web-based application functionality and estimate application requirements. Furthermore, one should have the opportunity to evaluate the performance of the proposed solutions and reconfigure the proposed architecture if user requirements are not fulfilled. Web-based systems are configured using the *Intelligent Distributed System Design* tool *(IDIS)* [13].The rest of the paper is organized as follows: In section 2, web-based system configuration issue is discussed and a systematic approach dealing with it is proposed. In section 3, we present a web-based application representation scheme. Conclusions reside in section 4.

2 Configuring Web-Based Systems

Internet technology can be used in conjunction with middleware technology (message-based or object-based) to produce powerful web-based architectures. The configuration of web-based applications is performed based on the multi-tiered client-server model. The Web client, e.g. the first tier, facilitates a standard user interface allowing the user to retrieve information (in the form of HTML pages) or activate applications (through HTML pages). The Web server, e.g. the second tier, process and redirects user requests, gathers results and sends them to the client in the form of HTML documents. Thus, it provides a middleware platform integrating the desired functionality into HTML documents. An easy way to forward requests to other tiers is to activate CGI programs at the web server site. CGIs are portions of executable code written in scripting languages, as PERL and JavaScript. The Web server does not save any context related to a request coming from the browser, thus every request to a CGI program is handled in isolation (stateless server operation). The concept of a *context file* may be used on the server side, in order to temporary store the results of a CGI program before gradually presenting them to the user through the Web client. The weakness of the CGI approach lies on the fact that the program must be restarted on each request. An alternative solution is the provision of a direct interface allowing the connection to an already active external program using shared object technology. The program is permanently loaded into the server memory and associated with a URL used for its activation. An alternative web-based architecture is the one based on *applets*. Web browsers can be used for program execution (intelligent web clients). In this case, an applet may be downloaded from the Web server and be executed on the client machine to activate other tiers. Web-based applications often employ this technique to communicate with other distributed middleware platforms, as CORBA (Common Object Request Broker Architecture). Old-fashioned applications are incorporated with web environment using wrapping techniques.

Hardware used to support web-based applications is usually described in terms of the workstation-server model. Users have their own workstation (diskless or not) for executing client processes. Server processes are executed on dedicated servers. Replication techniques are employed to increase performance and ensure availability.

Web-based systems are viewed as a combination of web-based applications and the underlying network infrastructure. Both can be described in terms of elementary components [14]. The network infrastructure consists of private intranets and Internet connections. Each intranet consists of interconnected local or even wide area networks supporting TCP/IP protocol stack. Network infrastructure configuration requires the following components to be defined: *processing nodes* used for the execution of client and server processes, *storage devices* and *network connections*. For application description, the necessary components are: *processes* (clients, servers), *messages* and *data*. Users are described through *user profiles.* A typical web-based application architecture described in terms of the aforementioned components is depicted in figure 1.

The *Web client* acts as the user interface for application invocation. Users, depicted as user profiles, access applications through HTML pages in the Web client. Processes communicate through exchanging messages based on the request/reply model. As indicated in the figure, when a *get page* request is sent to the *Web server*, the proper functions are initiated and a *HTML file* is retrieved from the *File Server* through a *read* request. Based upon the HTML page content, the Web server may send the *HTML page*, as a reply, back to the client, or initiate a CGI script or an active program to communicate with any *external application server*. The *get page* request is also used to download an *applet* from the *Web server* and communicate with the *external application server*.

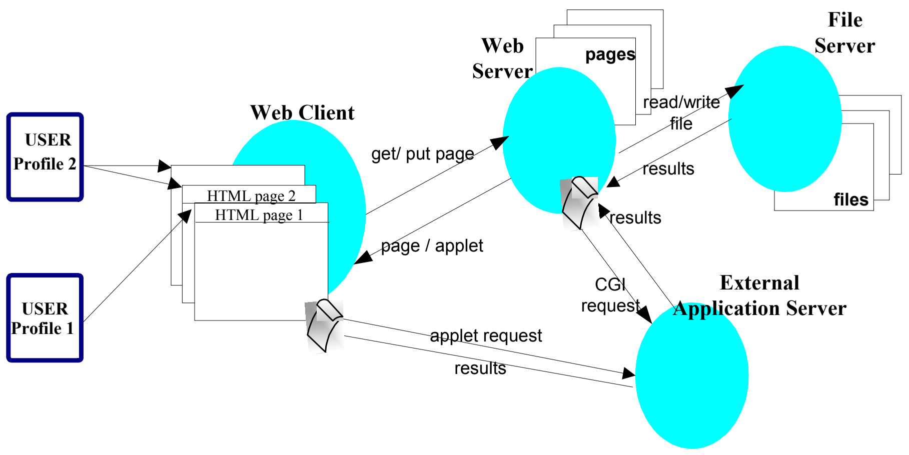

Figure 1. Typical Web-based Application Architecture

In the following, we propose an integrated approach for web-based system design. System design is performed prior system construction and after user requirement gathering, as indicated in figure 2. It includes web-based application functionality description, process and data allocation to minimize Internet traffic and ensure efficient application operation and network configuration (network topology design). The design phase must facilitate the performance evaluation of the proposed solution prior implementation to ensure reliability. It is important to note the significance of a common modelling scheme for the representation of system entities. This enables the detailed description of user requirements while maintaining simplicity in the description of web applications. System design is accomplished in the following steps (figure 2):

1. Functional topology definition
2. Logical topology definition
3. Physical topology definition
4. Performance evaluation

Functional topology definition corresponds to the systematic description of system requirements. Logical and physical topology definition deal with server and data allocation and network configuration respectively. Both are accomplished using heuristics. Resource allocation and network configuration problem cannot be solved independently [11]. Thus, steps (2) and (3) of the proposed approach are invoked interactively until an acceptable solution is reached, as shown in figure 2. Analytical description of each step is presented in the following. IDIS system provides a semi-automated environment guiding the user throughout the aforementioned steps.

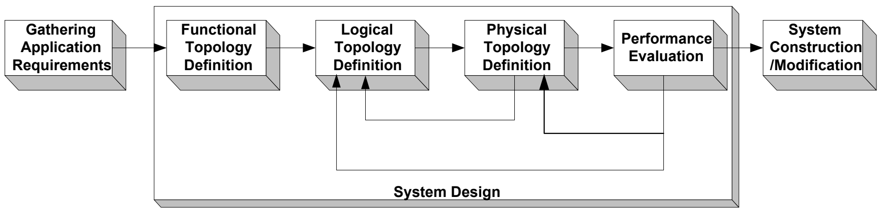
Gathering Application Requirements
Functional Topology Definition
Logical Topology Definition
Physical Topology Definition
Performance Evaluation
System Construction /Modification
System Design

Figure 2. Web-based Configuration Systematic Approach

2.1 Functional Topology Definition

In functional topology definition, applications are described as sets of interacting processes activated by user profiles. The files used by processes are also specified. There are two kinds of files, data files and code files. Application functionality is described in terms of predetermined, high-level operations (or actions), which are customized to conform to the web-based application architecture presented in figure 1. Operations are ultimately expressed in terms of primitive actions used to estimate application requirements. The desired performance characteristics for each application are also defined. More about the application representation scheme is presented in section 3.

Access points of the web-based system, called *locations*, are also specified. Definition of locations as well as specification of their size is performed with respect to the user's view. At the first level of detail, locations are defined as Internet access points. At next levels of detail, the *location* entity can be refined into more elementary ones, allowing the user to adjust the description of the system according to the application scale. Progressive refinement of location concept enables the progressive solution of resource allocation and network configuration problems.

2.2 Logical Topology Definition

Logical topology definition concerns process and file allocation. Allocation of processes and files is performed aiming at a. minimizing Internet traffic, b. fulfilling application requirements and c. minimizing configuration cost. Minimizing Intranet traffic especially on WAN connections and balancing load is also taken into account. Communication cost function C consists of C_U caused by client access, C_P caused by server access and C_F caused by data access. The optimal allocation solution is reached when $C = \min(C_U + C_P + C_F)$, under conditions ensuring access to all process and data replicas. Different replication scenarios can be applied, while locating processes and data [15]. As proved in [16], minimizing C is considered to be NP-complete problem. For the solution of such problems, heuristic methods are introduced, ensuring that, even if the optimal solution is not reached, one very close to it will be found. A variety of allocation algorithms supporting synchronous and asynchronous replication policies can be applied [17, 18, 19].

Adopting the workstation-server model, $C_U=0$ thus $C = \min(C_U + C_P + C_F)$. Since servers may access shared data, there is dependence between C_P and C_F. Even if the optimal solution is not reached, it is assumed that data are allocated before processes. Since network topology is not predefined during process and data allocation, it should be designed concurrently to ensure the efficient support of the solutions adopted. To achieve the required application performance, locality (i.e. keeping servers and data as close as possible to user) is considered as a basic principle. The most popular algorithm for Web server placement is the one based on the avoidance of unnecessary data transfer between WAN connections with asynchronous data replication support. The algorithm does not support optimal performance solutions, as it only focuses on WAN traffic and searches for a "relative good", cost effective and simple solution. Alternative algorithms supporting different data replication schemes and LAN traffic minimization are also supported [20, 21]. Each supported algorithm has an *Activation_Factor,* indicating activation order. Most simple algorithms are first applied. If the proposed solution does not satisfy application requirements, more complex algorithms are tested resulting in more costly solutions. *Activation_Factor* in each algorithm may be altered during system reconfiguration.

2.3 Physical Topology Definition

At the stage of physical topology definition, the network topology is designed. Network topology design is performed progressively. Since a network must be designed for each location, location refinement leads to a more detailed description of the network architecture. At each level of detail, the network architecture is formed by connections between locations of the specific level. For example, the network supporting the *Building* location is formed by network connections between all the *Floors* belonging to that *Building*. IDIS supports physical topology design by providing alternatives for network topology design and network configuration, but does not indicate commercial solutions.

2.4 Performance Evaluation

To evaluate system performance, a discrete event simulation tool was used [22]. Simulation modeling is widely adopted in the computer network domain for performance evaluation purposes. MODSIM simulation language [23] was used for simulation purposes. Object-oriented modelling and pre-constructed model libraries were employed to ensure efficiency of the simulation process [24]. Using simulation, maximum, average and minimum values of all performance measurements can be estimated. If the system requirements are not satisfied, logical and physical topologies must be redesigned. System performance cannot be partially estimated, e.g. even if only a small part of the overall architecture is altered, the entire system must be simulated again to accurately estimate performance measurements. The completion of the simulation phase is the most time consuming part of the overall design phase. Thus, while redesigning an inefficient architecture, all possible changes will be examined before the simulation process is reactivated.

3 Web-based Application Representation Scheme

Web-based application functionality is represented using the modelling scheme presented in [22]. Main features of the modelling scheme are modularity, extendibility and wide applicability. It facilitates accuracy in distributed application description using a multi-layer *action* hierarchy. Actions indicate autonomous operations describing a specific service. The main goals of this modelling method are: a. to facilitate the complete description of application functionality in a simple way and b. to ensure the detailed depiction of application requirements. The modelling framework supports multi-tiered client/server models and can be easily extended to support customized applications.

Applications are modelled as sets of interacting processes. The specific *interfaces*, acting as process activation mechanisms must be defined for each process, along with the *operation scenario* that corresponds to the invocation of each interface. Each operation scenario comprises the actions that occur upon process activation. User behaviour is modelled through *user profiles*. Each profile includes user requests resulting in application invocation through the Web platform. Actions are described by qualitative and quantitative parameters, e.g. the processes being involved and the amount of data sent and received. In most cases, the operation scenario is executed sequentially (each action is performed when the previous one has completed). However, there are cases where actions must be performed concurrently. This is supported through specifying groups of actions that have common sequence number. The basic actions used to define operation scenarios are: *processing* indicating data processing, *request* indicating invocation of a server process, *write/read* indicating data storage/retrieval, *transfer* indicating data transfer between processes and *synchronise* indicating replica synchronization.

Actions can be either elementary or of higher layer. In the latter case, they are decomposed into elementary ones. While *processing* is an elementary action, *write* can be expressed through simpler ones, i.e. a *process* and a *request* sent to a *File Server*. All actions can be ultimately expressed through the three elementary ones, *processing*, *network* and *diskIO*, each indicating invocation of the corresponding infrastructure component. Action decomposition is performed through intermediate stages to simplify the overall process and maintain relative data. Action decomposition hierarchy ensures consistency, reduces complexity and enables following a common predefined decomposition mechanism. The most promising feature of this scheme is that the action hierarchy can be further extended to include new actions, placed at the highest layer. Definition of new actions is based on existing ones to ensure consistency during action decomposition.

In order to support Web-based applications, the action hierarchy presented in [22] was extended to include *Web-related actions.* These actions are used to easily describe operation scenarios corresponding to Web server and Web client functionality in the model depicted in figure 1. They include:

- *Get/put page*: indicating retrieving/storing an HTML/XML page
- *Post:* indicating form/field passing on an HTML/XML page
- *Get applet:* indicating applet download
- *Applet:* indicating applet activation
- *CGI*: indicating a cgi program activation
- *Invoke Program:* indicating active program invocation
- *Handle/Retrieve context file*: indicating context file creation or modification/ retrieval of context file data
- *HTTP request/reply*: indicating send request/reply protocol implemented to support HTTP protocol

The first three are usually used to describe Web client functionality, the following three Web server functionality and the last one HTTP protocol functionality. While all others are used for operation scenario description, the last ones are intermediate actions accurately depicting HTTP internal functionality. The functionality of external application servers can be depicted by further extending action hierarchy to support application specific operations. Web related actions hierarchy is depicted in figure 3.

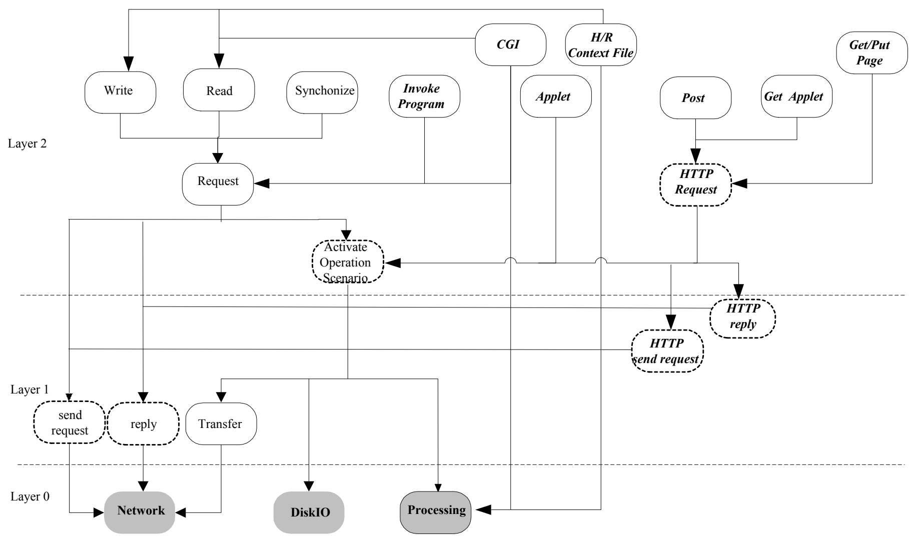

Figure 3. Action Decomposition Hierarchy

Dotted rectangles represent *intermediate* actions, while gray rectangles represent *elementary* ones. Finally, rectangles with black border represent *application* actions used when defining operation scenarios. Web related actions are indicated using italics. The *request* action is used to depict process invocation and is further decomposed *into send request, activate operation scenario* and *reply actions.* Although not indicated in the figure, the *activate operation scenario* action may result in the invocation of any action included in application description. The *http request* action depicts the request functionality as it is implemented by HTTP protocol and it is used in the decomposition of application actions describing *Web client* functionality. Many application actions, as *read/write* or *get/put page*, actually represent the invocation of the corresponding server interface, and are decomposed into a *request* or *http request* action. This type of actions is supported to simplify the description of operation scenarios, since they are described using less parameters that the corresponding request actions. Furthermore they make server invocation transparent to the user, when describing client operation.

The user may further extend action hierarchy to describe external application functionality. When defining a new action, the user must specify its parameters and the actions used to describe it. During action decomposition, all parameters of the invoked action must be defined. In order to avoid knowledge inconsistency, the user ability to add actions is restricted.

4 Conclusions

Web-based systems provide a standard platform for the development of a wide range of applications. They extend to multiple sites and are characterized by internal complexity. Thus, their configuration is not a trivial task. We proposed a systematic approach for the configuration, modification and performance evaluation of web-based systems. During system design, NP-complete problems, such as resource allocation and network configuration must be solved. The proposed approach employs knowledge-based techniques and heuristics for providing solutions. Simulation is the performance evaluation of the proposed solutions. Emphasis is given on the extendable modelling scheme used to depict web-based application functionality and estimate application requirements. Future work focuses on the support of web-based multimedia and real time applications.

References

1. Serain, D.: Middleware. Springer-Verlag, London, Great Britain (1999)
2. Reeser, R., Hariharan, R.: Analytic Model of Web Servers in Distributed Environments. In: Proceedings of the ACM 2000 International Workshop on Software and Performance. ACM Computer Press (2000)
3. Shedletsky, J., Rofrano, J.: Application Reference Designs for Distributed Systems. IBM System Journal **32**(4) (1993)
4. Coulouris, G.F., Dollimore, J., Kindberg, T.: Distributed Systems - Concepts and Design. 3rd edn. Addison Wesley Publishing Company (2000))
5. Arlitt, M.F., Williamson, C.L.: Internet Web Servers: Workload Characterization and Performance Implications. IEEE/ACM Transactions on Networking **5**(5) (1997)
6. Barford, P., Crovella, M.: A Performance Evaluation of Hyper Text Transfer Protocols. In: Proceedings of the ACM 1999 International Conference on Measurement and Modeling of Computer Systems. ACM Computer Press (1999)

7. Khoroshevsky V. D.: Modelling of Large-scale Distributed Computer Systems. In: Proceedings of IMACS World Congress 1999. IMACS **6** (1999)
8. Juengst, W.E, Heinrich, M.: Using Resource Balancing to Configure Modular Systems. IEEE Intelligent Systems **1**(1) (1998)
9. Fleischanderl, G., Friedrich, G.F., et. al.: Configuring Large Systems Using Generative Constraint Satisfaction. IEEE Intelligent Systems **1**(1) (1998)
10. Nezlek, G.S., Hemant, K.J., Nazareth, D.L.: An Integrated Approach to Enterprise Computing Architectures. Communications of the ACM **42**(11) (1999)
11. Lee, S.J., Wu, C.H.: A Knowledged-based approach to the Local-Area Network Design Problem. Applied Intelligence **4**(1) (1994)
12. Dutta, A., Mitra, S.: Integrating Heuristic Knowledge and Optimization Models for Communication-Network Design. IEEE Transactions on Knowledge and Data Engineering **5**(12) (1993)
13. Nikolaidou, M., Lelis, D., et. al: A Discipline Approach towards the Design of Distributed Systems. IEE Distributed System Engineering Journal **2**(2) (1995)
14. Kramer, J.: Configuration Programming - A Framework for the Development of Distributed Systems. In: Proceedings of the Annual IEEE International Conference on Computer Systems and Software Engineering. IEEE Computer Press (1990)
15. Buretta, M.: Data Replication: Tools and Techniques for Managing Distributed Information. Wiley & Sons Inc., US (1997)
16. Morgan, H.L., Levin, K.D.: Optimal Program and Data Locations in Computer Networks. Communications of ACM **20**(5) (1977)
17. Jajodia, S.: Managing Replicated Files in Partitioned Distributed Database Systems. In: Proceedings of IEEE International Conference on Data Engineering. IEEE Computer Press (1987)
18. Awerbuch, B., Bartal, Y., Amos, F.: Competitive Distributed File Allocation. In: Proceedings of ACM Annual Symposium on Theory of Computing. ACM Computer Press (1992)
19. Litoiu, M., Rolia, J.: Object allocation for Distributed Applications with Complex Workloads. In: TOOLS'2000. Lecture Notes on Computer Science, Vol 1786. Springer-Verlag, Berlin Heidelberg New York (2000)
20. Tan, M., Siegel, H.J.: A Stochastic Model for Heterogeneous Computing and Its Application in Data Relocation Scheme Development. IEEE Transactions on Parallel and Distributed Computing **9**(11) (1998)
21. Johnson, G., Singh, A.K.: Stable and Fault/tolerant Object Allocation. In: Proceeding of ACM Annual Symposium on Principles on Distributed Computing. ACM Computer Press (2000)
22. Nikolaidou, M., Anagnostopoulos, D.: An Application-Oriented Approach for Distributed System Modeling. In: Proceedings of 21st IEEE International Conference on Distributed Computing Systems. IEEE Computer Press (2001)
23. MODSIM III The Language of Object-Oriented Programming - Reference Manual. CACI Products Company (1999)
24. Anagnostopoulos D.: An Object-Oriented Modeling Methodology for Dynamic Computer Network Simulation. International Journal of Modeling and Simulation **21**(4) (2001)