THE ADAPTIVE WAREHOUSE CONCEPT FOR THE RESOURCE MANAGEMENT IN THE WOSTM

HERWIG UNGER
FB Informatik, Universität Rostock, D-18051 Rostock, Germany
Phone: +49 381-4983403, Fax: +49 381-4983366,
Email: hunger@informatik.uni-rostock.de

Abstract

A new execution paradigm for large distributed systems will be introduced with the concept of local resource warehouses and effective mechanisms for their use. It avoids the use of any kind of centralized instances and supports easy possibilities to adapt the resource management strategies of the system to the profile of the respective users of the system.

Keywords: Resource Management, Warehouses, Internet, Web Operating System, Data Access, Distributed Computing

1. EXECUTION PARADIGMS IN LARGE DISTRIBUTED SYSTEMS

Most Resource Management Systems were built for the use in small or middle large distributed computing environments like workstation clusters [2][7]. The resource management activities of these systems ensure the fastest possible execution of a distributed application within the given architecture as well as an evenly distributed workload of all computers [3][4].

However, the execution of any distributed application is only a special case in a large distributed and maybe worldwide acting operating system of the future [9][11], like the WOS [8]. As described in [8] and [12] the more important tasks of such a system is
• to find a suitable machine to execute the requested service
• within a given time frame (including an execution in the future).

In such a manner, every service will be considered to be an own application which is not connected with any other one.
With the specialization of the hard- and software the client server relation become the most used computing principle in distributed architectures. But this paradigm requires an explicit knowledge about the location (address) of the respective server. Furthermore, no (automatic) load balancing can be supported. This can result in long waiting times, if too much clients send request to a given server process (see Figure 1).
The broker architecture as known from [15] or [16] avoids some of the problems of the client server regime. In this case broker controls the activities of a lot of servers. Although all clients coordinate the service execution directly with the assigned server, the server assignment realize a load balancing between a number of servers and long waiting times can be reduced
by the parallel working servers. Nevertheless, the broker remains as a central instance and failures, breakdowns or low communication and processing bandwidth will disturb the service execution in such a system, too. In addition there is still no method implemented to propagate free service opportunities and there is also no automated search for brokers offering a special service. This can be a mentionable disadvantage, if more than one broker exists in a (large) system.

Figure 1: Client Server and Broker Relation in Distributed Systems

2. SERVICE EXECUTION IN THE WOS

The WOS-Concept [8] elicits that each computer in the WOSNet has two tasks: to be a WOS-client which requests the service execution for the user sitting in front of the machine and to be a WOS-server which accepts or rejects remote service calls. Therefore, the structure shown in Fig. 2 for the effective work of the system, including resource scheduling and communication was suggested in [1].

The left side of this figure shows the server, and the right side represents the client part of each WOS node. The functionality is the following. Each user command is processed by the client side’s execution control unit. It decides whether a process must be locally executed (e.g. a ps-command) or not. While local executable jobs will directly proceed to the user interface of the respective operating system, all other service requests are sent to the user resource control unit. A decision will be made, whether the requested service can be fulfilled in a prereserved standard user space or not. If it is so, a common load sharing mechanism will determine on which machine this will be done. Otherwise, a search over the WOSNet must be started as described in [12]. This is the task of the Search Evaluation unit which also evaluates the results of that request. For that search, former search results from a local data storage, e.g. a so called warehouse, can give a satisfiable answer. Otherwise, a WOSNet wide search must be organized. The results will be used by the user resource control unit to contact potential hosts. A successful service

168
execution requires, that at least one remote resource control unit of a machine must obtain
from its local warehouse a valid list entry that the service can be executed on that machine
for the requesting user and must be able to allocate the necessary resources for that job. The
job can then be started there under the responsibility of the remote job control unit which also
contacts the user execution control to transmit the results. Therefore, the use of a Network
File System, like it will be used in WebOS [5], was suggested in [6]. However, all search and
execution information will be used to update the respective user profile and search ware-
house entries.

![Diagram of a Node in the WOSNet](image)

Figure 2: Structure of a Node in the WOSNet

3. WAREHOUSES

3.1 Basics

In the above section the execution of any service request in the WOS was specified. In addi-
tion it was described in which manner information must be kept and accessed in each com-
putation node of such a system. Furthermore, it was figured out that each node has to fulfill
the tasks of a server as well as of a client while executing remote service request or requests
of its local owner/user. Nevertheless, the amount of data which can be kept on a local node
is very limited. That is the reason why not all available information can be kept. The only
decision criterion, which information are worth to be kept, is their usefulness for the local
user.

In such a manner a first approach a locale warehouse could be considered to be something
like a normal database with a limited storage amount. But warehouses of a WOS-node have to fulfill a broader functionality than described in [13] or [10]. Each warehouse in the WOS must have the ability to decide without any additional user activity, which information should be

- kept on which place in the warehouse
- replaced or deleted
- obtained from which other warehouse
- checked or replaced by new information or a-priory collected information (a decision about a-priori activities will be made depending on user profiles and access statistics).

Warehouses having the first two properties are very similar to obvious (database) warehouses in the literature, they will be also called passive warehouses. Basing on the use of WOSP/WOSRP [1] warehouses in the WOS, which are able to fulfill the 3rd requirement, are called active warehouses. In higher WOS-Versions all four data management activities will be provided by the warehouses. These are called adaptive warehouses. Their concepts will be considered more detailed in this paper.

### 3.2 Data Transactions of Adaptive Warehouses

![Diagram](attachment:image.png)

**Figure 3: Introducing the Structure of Adaptive Warehouses**

From the above said it is clear that a local node with its adaptive warehouse has to fulfill much more tasks than answering requests from its own or remote users and keeping information in its warehouse (Figure 3). Concerning the warehouse activities there are two oppositional tasks:
1. **Acquisition of new informations.**  
   in order to have all informations requested by the local user available without accessing  
   other warehouses and causing hereby an additional communication overhead;

2. **Control and update of existing data entries.**  
   to delete obsolete information and to limit the size of the warehouse as well as avoiding  
   the contact to useless machines.

Both tasks require communication activities of the warehouse which might be done in low  
network traffic times. However, a warehouse management facility must determine, with  
which volume these activities shall be done.

In addition, for further use each warehouse propagate its state (empty, standard, collector) for  
a free access of other warehouses.

The following local data can be used to define the respective input values for the warehouse  
management unit (see Figure 4).

---

**Figure 4:** Control Mechanism for Adaptive Warehouses

- the amount of data already kept in the warehouse (Size)
- the number of successful (Hit) or non-successful (Fail) data requests of the local or  
  remote machines the warehouse had to answer during a determined interval in the past  
- the success of data search activities, e.g. how fast requested information could obtained  
  from other warehouses and how often data control activities has confirmed the actuality  
  of warehouse data (Success)
The above pictures shows that from that data 3 different functional units can determine,

- which activities the search unit can perform per time unit (α, Max),
- which part of these activities yields to new data entries (β, New) and
- how much or how often data entries shall be checked (γ, Check).

The partition in the three functional units simplifies their specification. The chosen time constants and the respective functional characteristics (PT₁, PT₂, delay-units) reflect the characteristic of the appropriate user and were be defined during the configuration of the system in an interactive process.

Nevertheless, the general dependencies are defined by the following basic definitions.

- a high rate of access failures causes a more intensive search for new service locations (β-unit);
- after long periods the service search will be activated to find new versions and installations of software (β-unit);
- a high percentage of access hits decrease control activities (γ-unit);
- large data amounts in the warehouse cause higher activities to check their actuality (γ-unit) and decrease the need for other (general) system activities (α-unit);
- high success rates of the search and control unit also reduce (general) system activities (α-unit).

There are also interdependencies between the different functional units.

- low system activities (Max) also decrease the search of new service locations (β-unit); however, after some times the control activities for the data entries of a warehouse will be increased (γ-unit);
- high data control activities (Check) reduce the possibility of the system to supply new service locations (β-unit) with its search unit (because the system seems to be already very busy with the management of its current data);
- high data control activities (Check) decrease general system activities, because these activities are not directly a part of the general task of the system (α-unit).

3.3 Analysis of the Behavior

After the configuration of the system and its start a stable initial state of the three functional units can be expected, which will be changed depending on the users activities.

The dynamic behavior of the given system can be easily analysed using results from cybernetics. Therefore, the above introduced in- and output signals must be considered to be time-dependent signals, while the functional units are the respective transfer elements. In such a manner each unit consists of a number of sub-transfer-elements processing the influence of each input signal (the transfer functions are $G_{α,1}...G_{α,3}$, $G_{β,1}...G_{β,4}$ and $G_{γ,1}...G_{γ,3}$) and a summation element combining their output signals in a unit to the respective output signal of the unit (New, Check and Max).

Using the discrete Z-Transformation for this analyses, we get corresponding to figure 4
\[ \text{Max}(z) = \text{Check}(z) \cdot G_{\alpha, 1}(z) + \text{Size}(z) \cdot G_{\alpha, 2}(z) + \text{Success}(z) \cdot G_{\alpha, 3}(z) \]

\[ \text{Check}(z) = \text{Max}(z) \cdot G_{\gamma, 1}(z) + \text{Size}(z) \cdot G_{\gamma, 2}(z) + \text{Hit}(z) \cdot G_{\gamma, 3}(z) \]

and finally

\[ \text{New}(z) = \text{Fail}(z) \cdot G_{\beta, 1}(z) + \text{Check}(z) \cdot G_{\beta, 2}(z) + \text{Max}(z) \cdot G_{\beta, 3}(z) + \text{Timer}(z) \cdot G_{\beta, 4}(z) \]

From this system of equations the behavior of each output signals of the units can be derived. Because of the cycle between the \( \alpha \)- and \( \gamma \)-unit

\[ 1 - G_{\alpha, 1}(z) \cdot G_{\gamma, 1}(z) \neq 0 \]

gives a criterion for the stability of the whole system.

4. FUTURE WORKS

During some experiments the usability of the described warehouse approach was already been proven. The configuration for a single machines is easily to be done depending on the results of an interactive dialog.

Nevertheless, a lot of open questions arise from the fact that a huge amount of machines will collaborate in the WOSNet. From the experience it can be expected that clusters of machines will be automatically generated through mutual entries in the warehouses. The reason therefore can be found in a similar resource structure and a similar request profile of different machines. In such a manner it can be expected that the machines of such a cluster often exchange data and service requests. This results in interactions of the warehouses, too.

Until now the mechanisms building clusters and algorithms for an optimization of this clusters are almost unknown and must be considered in future works to ensure an optimal system behavior.
REFERENCES


Herwig Unger was born in 1966 in Halle/S. He got its MSc graduation from the Technical University of Ilmenau in 1991. In 1994 he received the PhD from the TU Ilmenau with a work about transformation and implementation methods of Petri-Nets used as models of distributed programs. After a tenure-track position at the rank of an Assistant Professor at CS-Department of the University of Rostock he got a permanent position there in 1996. His research interests include resource management strategies, extensions of operating systems and adaptive methods for the use in very large distributed systems as well as respective methods for their analysis and simulation. He is an active member of the WOS-Project-Group since 1997.