Load Sharing in a Heterogeneous Distributed Object-based System

Josef Matulka
Dept. of Applied Computer Science, University of Economics and Business Adm.
Augasse 2–6, 1090 Vienna, Austria

Networks are quite commonplace nowadays; physically combining heterogeneous computers in a local area network is not a problem any longer. Distributed software that makes efficient use of this hardware, on the contrary, is much more difficult to design. The object based approach is a promising paradigm for designing and implementing distributed software; unfortunately it is notorious for its performance problems.

Given an efficient network communication mechanism, one can use load sharing algorithms to increase performance. A huge number of such algorithms have been developed in the last decade (see Casavant and Kuhl (1988) for a survey) which can be broadly classified in performance-oriented and cost-oriented approaches. The former use a queueing system to model the computer hardware, typically only consider jobs that do not communicate with each other and focus on obtaining performance improvements rather than determining an optimal solution. The latter allow for interaction between different jobs but lack a realistic hardware model; integer programming is used to derive an exact solution of the load balancing problem.

Both approaches have their merits, but are inadequate for object based systems. In this paper we derive an algorithm that integrates these approaches by using several phases that build upon each other. In principle, we answer the following question: Given a set of heterogeneous objects with known workload characteristics and a set of hosts consisting of resources (CPU, disks, etc.) with known performance, what is the optimal allocation of objects to hosts?

In the first phase (structural analysis) we analyse the structure of a distributed object-based system. The performance analysis is an application of the theory of separable queueing networks. We determine the input parameters needed to compute different performance metrics (utilization, response time) for each fixed assignment of objects to hosts. During optimization analysis we specify a cost function as a combination of performance metrics and formulate an assignment problem. In the last phase (coordination analysis), the assignment problem is interpreted as coordination problem between hosts for which a team-theoretically inspired solution algorithm is given.

In the following we only sum up the model and the algorithm; a more detailed derivation together with some examples can be found in Matulka (1992); an application to client-server systems is described in Matulka (1991).

Structural Analysis

A distributed object-based system consists of a set of interacting objects running on a set of hosts connected by a (local area) network. Each host $h$ consists of a number of resources $k$ (such as CPU, disk, and network interface). Each object $o$ consists of a number of methods $m$. The object is the unit of allocation (all methods of one object have to be allocated to the same host) whereas the method is the unit of computation (all workload of the system is generated by some method).
The allocation $U$ is a matrix that links the software (objects) to the hardware (hosts) by specifying which host each object is allocated to. Finding the optimal allocation $U^*$ is the goal of our algorithm.

Performance Analysis

In order to be able to compute performance indices, we specify a queueing network model of the distributed object based system. Each resource is modeled as a service center, for each method the services required are specified. The resources of the hosts provide the services demanded by the methods. In our view, two activities are characteristic of hosts as well as methods: computation and communication. We assume that only remote communication requires service, thus neglecting the comparatively small amount of service required by local communication.

For each method $m$ we have to know:

- $d_{m,k}^{comp}$, the mean service required by one invocation of method $m$ at resource $k$.
- The mean number $d_{m}^{comm}$ of bytes transmitted in case of a remote invocation of method $m$.

The interaction between methods is specified by the interaction matrix $\Lambda$

$$
\Lambda = \begin{pmatrix}
\lambda_{m_1,m_1} & \cdots & \lambda_{m_1,m_{|M|}} \\
\vdots & \ddots & \vdots \\
\lambda_{m_{|M|},m_1} & \cdots & \lambda_{m_{|M|},m_{|M|}}
\end{pmatrix}
$$

(1)

where $\lambda_{m_i,m_j}$ denotes the rate of invocations of method $m_j$ by method $m_i$, and $|M|$ denotes the number of methods.

While this workload model has extended the description of the object based system, we also need some additional parameters to get the configuration model of the distributed system:

- A factor of proportionality $\beta_{k,h}^{comp}$ for each resource $k$ of each host $h$ that defines the performance of this resource relative to some defined standard resource. A factor $0 < \beta_{k,h}^{comp} < 1$ means that the resource $k$ of host $h$ is faster than the defined standard resource. $\beta_{k,h}^{comp}$ is set to $\infty$ if the resource $k$ does not exist at host $h$.
- The service needed at each resource $k$ of each host $h$ to transmit (send and receive) a message of known length.

Given the workload and the configuration model and assuming a fixed and known allocation $U$ the BCMP-algorithm (cf. Basket et al. (1975)) can be used to compute performance indices such as

- the mean number $Q_{m,h}^{comp}$ of concurrently active invocations of each method $m$ at host $h$, and
- the mean number $Q_{m,h}^{comm}$ of concurrently active protocol conversion processes caused by remote invocations of method $m$ at host $h$.

Optimization Analysis

In order to compute an optimal allocation, we have to specify a cost function $\gamma(U)$. The following function to be minimized (see Matulka (1992) for alternative specifications) denotes the sum of active method invocations and protocol conversions (using the Law of Little this can be shown to be equivalent to a weighted sum of response times):

$$
\gamma(U) = \sum_h \sum_m \left( Q_{m,h}^{comp} + Q_{m,h}^{comm} \right)
$$

(2)