

Optimized Allocation of Distributed Applications Across Local Area Networks

Bao Hua Liu, Sanjay Jha, *Chun Tung Chou, Pradeep Ray
 The University of New South Wales, *The University of Wollongong
 Email: {mliu, sjha}@cse.unsw.edu.au, ctchou@elec.uow.edu.au, p.ray@unsw.edu.au

Abstract

Enterprise-wide distributed computing systems are inherently stochastic and the performance management of distributed applications running on the top of DCS is a complex and computationally hard task. In this paper, we define the enterprise distributed application and provide queuing analysis on the relationship between the application level performance parameters and the network and computer system parameters. We also provide a binary integer programming (BIP) model and a case study on how to allocate application components across an enterprise local area network.

1. Distributed Application and Network Model

We model the network as *nodes* and *connections*. The nodes include computers and other network switching apparatus. A connection links a pair of nodes, and can be direct through one link, or indirect through multiple links and other nodes. Nodes can be used to distribute application components, but only some dedicated server machines are used for this purpose. We call these dedicated server machines *distributing nodes*. For all distributing nodes, we assume that a connection or route exists between every pair of nodes. Two nodes may be connected through one or more switching nodes and links. A link is described by its bandwidth or data rate.

We define following parameters in our model: 1) Node memory size is described as $MEMS_i$ for node i , 2) Average disk I/O time is described as IOT_i for node i , 3) A link l is described by its bandwidth B_l , 4) A router r is described by its router latency RL_r , 5) An application component a requires MEM_a size of node memory at run time, 6) The average CPU time for component a on node i is described as S_{ai} , 7) K_a denotes the average number of visit to disk I/O for component a to execute a transaction job, 8) Current CPU utilization $U'_i(CPU)$ for node i , 9) Current disk utilization $U'_i(disk)$ for node i , 10) Current memory usage MEM'_i for node i , 11) Current load γ'_l in packet per time unit for link l , 12) Current utilization U'_r for router r .

We now discuss the mapping of response time to network and system parameters. We model a distributed application system as an open queueing network. These include node queues and connection queues (note a connection may include multiple links and routers). For simplicity, we model each service provided by nodes and connections as a $M/M/1$ queue, and we call each node or connection a *service facility*. We assume the whole application system to be a *markovian system*, which means the distributions of the inter-arrival times and service times are exponential distributions (Poisson) that exhibit the markov (memoryless) property. Within

the system, service facilities may have inputs from multiple facilities and outputs to multiple facilities. We assume all input and output processes are Poisson processes and that merged or split Poisson processes are still Poisson processes. As an example, we modeled the distributed application (with process flows) described in [1] with the *Jackson Network* in Figure 1.

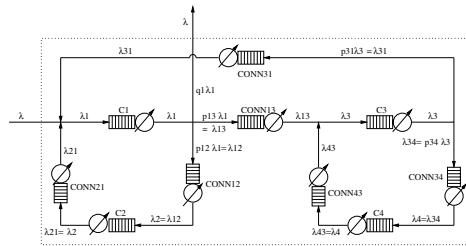


Figure 1. An Application Queueing Network Model

The average response time of a distributed application can be given through Little's Law $\bar{R} = \bar{N}/\lambda$, where λ is the arrival rate of the customer requests and \bar{N} is the average number of customer requests in the application system. We assume λ is a known parameter and we need to resolve the \bar{N} to calculate the average sojourn time for a customer request in the system. Because the series of queues include queues at nodes and connections between nodes, we can write $\bar{N} = \bar{N}_{NODE} + \bar{N}_{CONN}$, where \bar{N}_{NODE} is the average number of customer requests in node queues, and \bar{N}_{CONN} is the average number of customer requests in connection queues. Queueing analysis is then used to calculate \bar{N}_{NODE} and \bar{N}_{CONN} based on the parameters we defined above (detailed queueing analysis are omitted here to save space).

2. The BIP Model and Case Study

A binary integer programming (BIP) model for the deployment of application components over a given local area network is given in this section. Our objective is to minimize the response time of the application. Following constraints must be satisfied by the optimized deployment pattern: 1) A component can only be deployed to one node, 2) Node CPU utilization can not exceed 1, 3) Node disk utilization can not exceed 1, 4) Node memory usage can not exceed node total memory size, 5) Average load on each link can not ex-

ceed the link bandwidth, 6) Router utilization can not exceed 1.

We define $M \times N$ decision variables, where M is the number of components for an application and N is the number of distributing nodes in the network. We define $x_{ai} = 1$ if component a is assigned to node i and $x_{ai} = 0$ otherwise. The completeness constraint specifies that each component is assigned to one and only one node, and that all components in the application are assigned, $\sum_{i=1}^N x_{ai} = 1$. The run time memory usage constraint for each node i can be described as $\sum_{a=1}^M x_{ai} \times MEM_a + MEM'_i \leq MEMS_i$. The run time CPU and disk utilization constraints for each node i can be described as $U_i(CPU) = \sum_{a=1}^M x_{ai} \times U_{ai}(CPU) + U'_i(CPU) \leq 1$ and $U_i(disk) = \sum_{a=1}^M x_{ai} \times U_{ai}(disk) + U'_i(disk) \leq 1$, where $U_{ai}(CPU)$ and $U_{ai}(disk)$ denote the utilization of CPU and disk for component a on node i . At network level, the constraints specify that each link's average load can not exceed the link bandwidth and router utilizations can not exceed 1. We require two auxiliary parameters to describe these constraints. For a link, we define $v_{ij}(l) = 1$ if link l is on the route from node i to node j and $v_{ij}(l) = 0$ otherwise. For a router, we define $w_{ij}(r) = 1$ if router r is on the route from node i to node j and $w_{ij}(r) = 0$ otherwise. if we assume a route can be defined through a routing algorithm or the topology of the network, $v_{ij}(l)$ and $w_{ij}(r)$ are known parameters if i and j are given. The link constraint can be described as below (note $j \neq i$ represents the case when two components are allocated to different nodes): $Load_l = \sum_{i=1}^N \sum_{j=1, (j \neq i)}^N \sum_{a=1}^M \sum_{b=1, (b \neq a)}^M COMM_{ab} \times x_{ai} \times x_{bj} \times v_{ij}(l) + \gamma_l \times \bar{V}_l \leq B_l$, where $COMM_{ab}$ is the communication data rate from component a to component b , γ_l denote the average load in packets per time unit on link l before the deployment of application and \bar{V}_l is the average packet size. The router utilization constraint can be described as below: $U_r = \sum_{i=1}^N \sum_{j=1, (j \neq i)}^N \sum_{a=1}^M \sum_{b=1, (b \neq a)}^M U_r(ab) \times x_{ai} \times x_{bj} \times w_{ij}(r) + U'_r \leq 1$, where $U_r(ab)$ is the utilization of router r for the process flow between component a and b , and U'_r is the utilization of router r before the deployment of application.

Our objective is to minimize the response time of the distributed application. To achieve this, we must find the average number of customers in the whole application system. By using the decision variables, we define the average number of customer requests on nodes as: $\bar{N}_{NODE} = \sum_{i=1}^N \sum_{a=1}^M \bar{N}_{ai} \times x_{ai}$, where \bar{N}_{ai} is the average number of customer requests for component a on node i . The average number of customer requests on connections is defined as: $\bar{N}_{CONN} = \sum_{i=1}^N \sum_{j=1, (j \neq i)}^N \sum_{a=1}^M \sum_{b=1, (b \neq a)}^M \bar{N}_{ajib} \times x_{ai} \times x_{bj}$ where \bar{N}_{ajib} is the average number of customer requests from component a to component b if component a is assigned to node i and component b is assigned to node j . The objective function is written as below:

$$\text{Minimize } Z = \frac{\bar{N}}{\lambda} = \frac{1}{\lambda} \left(\sum_{i=1}^N \sum_{a=1}^M \bar{N}_{ai} \times x_{ai} + \right.$$

$$\left. \sum_{i=1}^N \sum_{j=1, (j \neq i)}^N \sum_{a=1}^M \sum_{b=1, (b \neq a)}^M \bar{N}_{ajib} \times x_{ai} \times x_{bj} \right)$$

We provide a case study to illustrate our model with the Ethernet LAN shown in Figure 2. There are four distributing nodes N1-N4 and we assume that there are three application components. We assume that the arrival rate of customer requests (λ) is 2 requests per second. Other correspondent parameters are omitted here to save space.

We are using the GAMS (General Algebraic Modeling System) [3] package to solve this problem. As our objective function is a binary integer non-linear function, the associated solver is SBB/CONOPT2. SBB (Simple Branch and Bound) is based on a combination of the standard branch and bound method known from mixed integer-linear programming and nonlinear programming solvers (e.g., CONOPT2). GAMS solved this problem with 641 iterations and 18 B&B nodes. According to the GAMS output, the optimized response time is 1.673s with $x(C1, N1)$, $x(C2, N4)$, $x(C3, N3)$ equal to 1. This tells us that for the given scenario, we can achieve a response time of 1.673 seconds if we deploy component C1 on node N1, C2 on N4 and C3 on N3. Interestingly, if we set λ to 0.5, all components are allocated to node N4 with achievable response time 0.9412s; if we set λ to 1, the allocation pattern is C1 on N1, C2 on N4 and C3 on N4 with achievable response time 1.2182s; if we set λ to 4, there is no feasible solution. This tells us two things: 1) the allocation pattern is de-

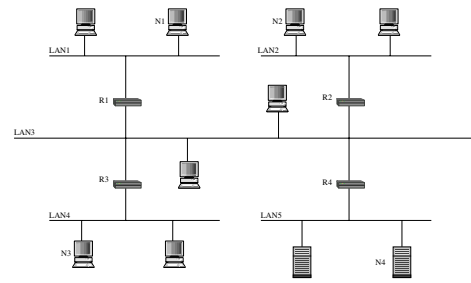


Figure 2. An Example Network

pendent on the arrival rate of customer request; 2) the current network and application configuration can not support a heavy customer load (e.g., 4 requests per second). Some solutions for the second issue include upgrades of hardware to increase the service rate and duplicate application components to balance the customer load.

3. Conclusions

We formulated a stochastic analysis of the mapping of distributed application response time to network and computer parameters. We also provided a non-linear binary integer programming model for allocating application components over a LAN to minimize response time. The allocation pattern is dependent on the arrival rate of customer requests.

References

- [1] B.H. Liu, P. Ray, S. Jha, "Mapping Distributed Application SLA to Network QoS Parameters", Proc. of 10th IEEE ICT 2003
- [2] Geoff Bullen, "Managing e-Business Applications - On the Future of Application Management", Borland Software Corporation. 2001
- [3] <http://www.gams.com>