Performance of an Infrastructure for Worldwide Parallel Computing

Thomas T. Kwan* Daniel A. Reed†
Department of Computer Science
University of Illinois
Urbana, Illinois 61801
kwan@cs.uiuc.edu, reed@cs.uiuc.edu

Abstract

The millions of Java-capable computers on the Internet provide the hardware needed for a national and international computing infrastructure, a virtual parallel computer that can be tapped for many uses. To date, however, little is known about the cost and feasibility of building and maintaining such global, large scale structures. In this work, we evaluate the performance of a Java/WWW-based infrastructure to gain insights into the feasibility and projected cost of initializing and maintaining wide-area hierarchies that contain up to one million nodes.

1. Introduction

With advances in high-speed networking, exploding interest in the World Wide Web (WWW), and widespread availability of Java, millions of Java-capable computers are now connected to the Internet. These heterogeneous laptops, personal computers, and workstations are emerging as a pool of distributed, platform-independent, Java virtual machines. The large scale deployment of these systems provides the substrate for a possible national and international computing infrastructure — a virtual parallel computer that can be tapped for many uses.

Although such an infrastructure could catalyze new software for distributed information mining and wide area computing, developing the middleware needed to support such wide area computing poses a host of thorny problems. Wide fluctuations in achievable network bandwidth, changing system availability, and significant differences in system capabilities (i.e., ranging from high-performance servers to inexpensive laptops) all make exploiting the large number of computing cycles available in Java-capable, WWW clients and servers extraordinarily difficult. One can effectively exploit these distributed computing cycles only when an infrastructure exists that can adapt to these variations, while providing reasonable performance and acceptable reliability. Simply put, the unreliable nature of wide area networks suggests the need to evaluate the reliability and feasibility of a globally distributed, parallel computing infrastructure.

The remainder of this paper is organized as follows. In §2, we discuss related work, and in §3, we describe the design of a large scale computing infrastructure. We evaluate the reliability and performability of the infrastructure in §4 and §5, respectively. In §6 we analyze the feasibility of wide area hierarchy formation, and in §7, we evaluate the feasibility of world view maintenance and task scheduling, We explore the impact of technology on the feasible hierarchy size in §8. Finally, we conclude with a brief summary in §9.

2. Related Work

Among the many distributed computing systems described in the literature, our work shares with AppLeS [2], Legion [6], Globus [4], Condor [10], Prospero [12], and ATLAS [1] the goal of harnessing distributed resources for large scale computing. However, our work differs from these projects in leveraging Java-capable, desktop, WWW clients to explore scalability and related performance issues. Whereas WebOS [13] aims to provide general operating system services, we leverage extant WWW browsers as the basic software substrate.

Jini [14], DOGMA [7], Javelin [3], and WebVM [5] typify recent Java/WWW-based distributed computing systems. Our work differs in emphasizing wide-area scalability and evaluation of the cost and feasibility of

*This work was supported in part by a National Semiconductor Graduate Fellowship and a NASA Graduate Student Researcher Program (GSRP) Fellowship.

†Supported in part by the Defense Advanced Research Projects Agency under DARPA contracts DABT63-94-C0049 (SIO Initiative), F30602-96-C-0161 and DABT63-96-C-0027, by the National Science Foundation under grants NSF CDA 94-01124 and ASC 97-20202, and by the Department of Enegry under contracts DOE B-34194, W-7405-ENG-48 and 1-B-333164.

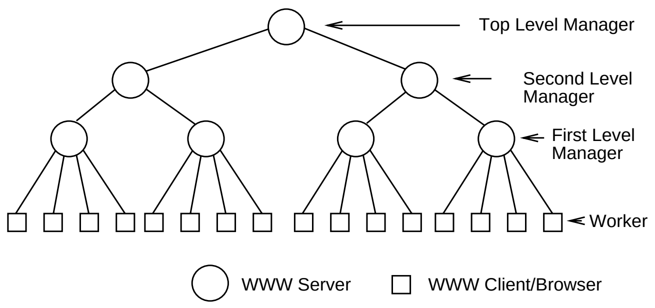


Figure 1. Example Client/Server Hierarchy

building large, wide-area structures.

3. Design Decisions

To evaluate the feasibility of large scale computing, we designed and prototyped a Java/WWW-based infrastructure that uses WWW browsers as computation workers and WWW servers as hierarchy managers. To connect a large number of distributed machines, we first exploit locality and link the machines in a tree, where the leaves are WWW browsers, the internal nodes are WWW servers, and the edges are logical network links. The servers cooperatively manage the hierarchy of browsers executing the computation.[1]

Figure 1 shows an example of a hierarchy. In this example, the manager branching factor, b_m, is two, and the client branching factor, b_c, is four. That is, the maximum number of children of each non-first level manager is two and the maximum number of children of each first level manager is four. As we will see in §4-§5, these branching factors are important design parameters that directly affect the reliability of the hierarchy.

In our design, we use a greedy algorithm to insert clients, attempting to cluster machines that are close in time to each other into the same subtree. (Servers join the hierarchy via a node splitting algorithm similar to that of B-trees.) To join the hierarchy, a new client connects to the root via servers specified in a configuration file. The new client then contacts each of the child servers of the root, determines the child server with the lowest latency, and establishes connection with this lowest-latency child server. The new client then continues to descend the tree by successively following the lowest-latency child server at each level until the new client reaches a first level manager and attaches to it.

To discover the arrival of new resources and the departure of current resources, we must periodically acquire information about resource availability in the infrastructure. In our design, the root maintains a global view of resource availability by periodically issuing requests for updated system state data. Information acquisition consists of a down sweep where the request is propagated from the root to the leaves, and an up sweep where information (e.g., network and machine loads) is gathered, summarized (to reduce data transmission cost), and percolated to the root. To support loosely coupled applications, we use a master-worker programming model — the master distributes tasks to the workers, and the workers perform the computation.

To determine the feasibility and cost of operating millions of nodes for global computing, we must understand the cost of the operations described in this section – that is, hierarchy formation, world view maintenance, and wide-area task scheduling. To evaluate these costs, we prototyped our infrastructure design, measured the cost of the above operations in small to modest configurations, and projected the cost of these operations in large configurations to gain insights into the possibility of operating a million node hierarchy.

[1]We use the terms clients, browsers, and workers as synonyms. Similarly, we use the terms servers and managers interchangeably.

Notation	Explanation
b_m	Branching factor of managers
b_c	Branching factor of clients
n_m	Total number of managers
n_c	Total number of clients
H_c	Height of the tree, including the clients
M_1	Number of first level managers
p_f	Probability of node or link failure
E_{nc}	Expected number of surviving clients
r	Number of redundant servers per node
l_r	Level of redundancy

Table 1. Notation Used in the Models

4. Hierarchy Reliability

Because workers in our loosely coupled, distributed computing infrastructure join and leave the hierarchy at unpredictable times and because network connections to remote sites can fail, quantifying the reliability of a geographically distributed computing hierarchy is critical. To understand infrastructure reliability, we developed and validated a probability model that projects the reliability of large structures and estimates the machine availability under different configurations.

4.1. Probability Model

To gauge the reliability and performance of large hierarchies, we use the expected number of available

Source	Expected Client Count	
	50 Client Experiment	100 Client Experiment
Measurement	47.97	71.18
Probability Model	47.98	68.99

Table 2. Probability Model Validation

clients – the number that remain connected to the root server. For a client to remain connected to the root (i.e., remain visible in the hierarchy), all the nodes on the path from the client to the root must be available. Therefore, the expected number of surviving clients is the product of the total number of clients and the probability that each client is visible to the root.

To compute the probability that a client is visible to the root, let p_f be the probability of failure of each node, and H_c be the tree height. (See Table 1 for notation.) Thus, $(1-p_f)$ denotes the probability that a node is available. If we assume that node failures are independent, then $(1-p_f)^{H_c+1}$ is the probability that all the nodes from the client to the root are available.

Thus, the expected total number of surviving clients, E_{nc}, is given by

$$E_{nc} = n_c(1-p_f)^{H_c+1} \tag{1}$$

where n_c is the total number of clients in the hierarchy.

Before using this simple probability model to project the behavior of large hierarchies, one must validate it using measurements of hierarchy behavior. Using a prototype of our infrastructure, we initialized hierarchies with 50–100 clients, observed them for 24–48 hours, measured the mean-time-to-failure (MTTF) and mean-time-to-repair (MTTR) of the machines, and computed the probability of failure (i.e., the input to the probability model) using the equation

$$p_f = \frac{\text{total repair time}}{\text{total observation time}}. \tag{2}$$

Table 2 shows that the model closely matches measurement, providing confidence that the model yields reasonable predictions of infrastructure behavior.[2]

Using the validated probability model, one can estimate reliability as a function of the hierarchy's organization, expressing the fraction of available clients as a function of the manager branching factor. For a full tree, we have

$$n_c = {b_m}^{H_c-1} * b_c$$

[2]We also developed a sophisticated Markov model. Because our results show that both models yield acceptable predictions, we use the simpler probablity model for tractability.

$$\Rightarrow H_c = 1 + \log_B(\frac{n_c}{b_c})/\log_B b_m \tag{3}$$

where B is the base of the logarithm. Substituting (3) in (1) and rearranging, we have

$$\frac{E_{nc}}{n_c} = (1-p_f)^{2+k/\log_B b_m} \tag{4}$$

where $k = \log_B(\frac{n_c}{b_c})$ is a constant for a fixed number of clients and a fixed client branching factor.

As shown by (4), the reliability of the infrastructure (i.e., the percentage of available clients) is bounded by $(1-p_f)^2$. In other words, the hierarchy is only as reliable as the combined availability of a client and the root. Any reduction in the reliability of the machines/networks quickly decreases the overall reliability of the infrastructure.

Clearly, to project the reliability of the infrastructure under typical network conditions, we need an accurate estimate of the average probability of node and link failure (i.e., p_f in (4)). Below, we summarize measurements of node and link failure probabilities and discuss their implications.

4.2. Reachability Measurement

To assess the machine and network failure rate, we measured the reachability of seventeen international World Wide Web (WWW) sites using the Unix `ping` utility. We selected these sites to cover all continents and all U.S. regions. Once every fifteen minutes for a three month period (from April 1,1997 to June 30, 1997), we recorded each site's response to the `ping` message. We classified the site as unreachable if the `ping` timed out before receiving a response.

Based on the measurements, we used (2) to compute failure probabilities, concluding that across seventeen WWW sites, the average probability of failure is 0.022. On average, there is a 2.2 percent chance that a particular machine or network link is unavailable.

To verify the accuracy of the computed probability of failure, we compared our results to those obtained by Kalyanakrishman *et al* in their HTTP performance study [8]. During the same month (April 1997) we conducted our `ping` study, Kalyanakrishman *et al* independently measured the availability of WWW hosts by fetching HTML pages from the one hundred most popular WWW servers in the United States. Their measured average probability of failure is 0.007, lower than that from our `ping` measurements. Because the majority of the HTTP requests were to the most popular sites in the United States, these hosts tend to have good network connectivity and low machine failure rates. Therefore, we believe that our measurement

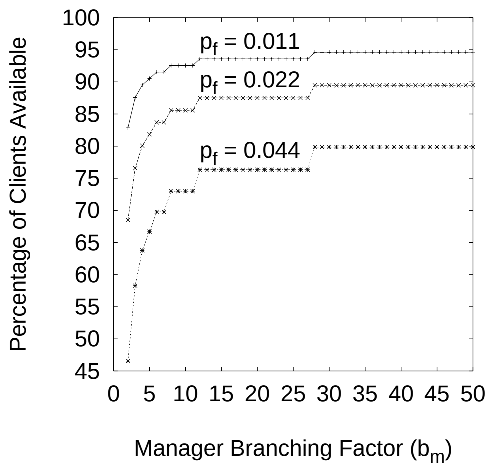
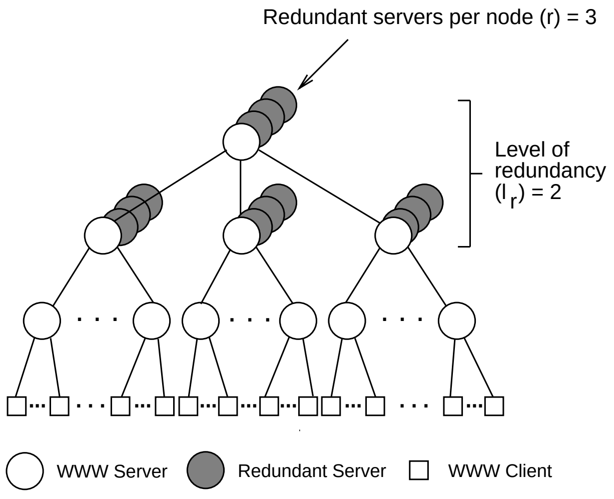


Figure 2. Projected Availability (10^6 Clients)



Redundant servers per node (r) = 3
Level of redundancy (l_r) = 2
WWW Server
Redundant Server
WWW Client

Figure 3. Hierarchy with Redundancy

is a reasonable, conservative estimate of the current reliability of nodes on a wide area network, whereas the estimate from the HTTP study represents an improvement over the average reliability.

4.3. Reliability Projection

Based on (4), Figure 2 shows reliability projections when the probability of failure is 0.022 (as measured), 0.011 (a 2X improvement), and 0.044 (a 2X degradation). In these projections, we used a client branching factor of fifty. However, results are qualitatively similar for other large branching factors.

As shown in Figure 2, a small increase in the manager branching factor greatly enhances the reliability of the infrastructure when the branching factor is small. The rapid decrease in tree height significantly reduces the number of nodes between the clients and the root, concomitantly increasing the reliability.

Figure 2 also shows that the percentage of available clients in a one million client configuration lies between 65 and 90 percent for measured link/node probability of failure. However, even a two-fold decrease in p_f (e.g., as might occur in a highly congested or bandwidth constrained environment) substantially reduces client availability, suggesting the need for redundancy techniques (e.g., scheduling redundant tasks). Below, we discuss the balance between server redundancy and computation performance.

5. Performability

Given fixed resources, one can allocate servers to maximize reliability or performance, but not both. Hence, one must balance the need for reliability against the desire for high performance, judiciously allocating resources to maximize performability [11].

Figure 3 illustrates a hierarchy with three redundant servers per node at the top two levels of the hierarchy. To accommodate redundancy, one can modify server insertion algorithms to insert new servers at key points. When a server later fails, one of the redundant servers can assume the function of the failed one. To quantify the performability of the hybrid infrastructure, we derive the number of available clients as a function of redundant servers per node and levels with redundancy.

5.1. Performability Metrics

To derive the number of available clients, let r be the number of redundant servers per node, and let l_r be the number of layers of managers (numbering from the root) with redundant servers. Hence, on the path from a client to the root, there are l_r nodes with redundant servers. The probability that all these nodes are available is given by

$$\left(1 - p_f^{(1+r)}\right)^{l_r} \qquad (5)$$

where p_f is the probability of node or link failure.

For the remaining nodes on the path that have no redundancy, the probability that they are available is given by

$$(1 - p_f)^{H_c+1-l_r} \qquad (6)$$

where H_c is the height of the hierarchy. Hence, the expected number of available clients E_{nc} is obtained

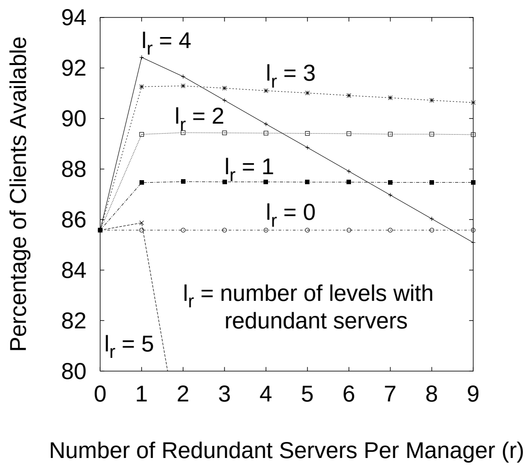


Figure 4. Performability of 10^6 Clients

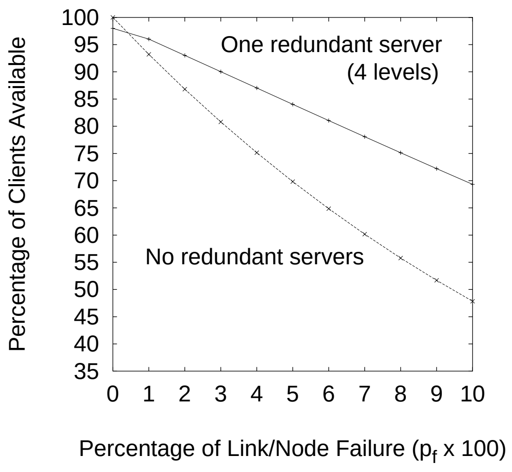


Figure 5. Availability of 10^6 Hierarchy

by multiplying the number of clients by the product of (5) and (6):

$$M_1(l_r, r)b_c\left[(1-p_f)^{H_c+1-l_r}\left(1-p_f^{(1+r)}\right)^{l_r}\right] \qquad (7)$$

where $M_1(l_r, r)$ is the number of first level managers, and b_c is the client branching factor. Because the number of first level managers now depends on the number of managers removed from the original hierarchy for redundancy, M_1 is a function of the level of redundancy and the number of redundant servers.

To compute M_1, we remove nodes from the lower levels of the tree until we have secured enough nodes for redundancy use; M_1 is the number of first level managers remaining. During the removal, however, we seek to preserve as many first level managers as possible by removing internal managers that have no descendants so the first level managers can continue to anchor clients, the computing engines of the hierarchy.

In sum, (7) gives the performability metric of the infrastructure, allowing us to explore performability as a function of the level of redundancy (l_r), the number of redundant servers per node (r), the probability of link or node failure (p_f), and the hierarchy structure (b_c, n_c, and H_c). Below, we evaluate performability by systematically varying these parameters.

5.2. Hierarchy Structure Implications

Using a link/node failure rate of 0.022 as analyzed in §4.2, Figure 4 shows the performability of a one million client hierarchy as a function of the level of redundancy and the number of redundant server per node. The figure shows that the percentage of available clients increases with redundancy up to level four. Beyond that point, the percentage of available clients decreases because the redundant structure requires the removal of too many managers from the lower levels, significantly reducing the number of first level managers that can anchor clients for computation. Figure 4 also shows that the number of redundant servers per manager has little effect on the availability of the clients — one redundant server per manager is sufficient to increase the performability of the infrastructure.

In brief, our detailed analysis [9] shows that varying the number of clients, the client branching factor, and the tree height yield results qualitatively similar to those in Figure 4. Hence, one can achieve reasonable performability by providing one redundant server per node for the first few levels of the hierarchy.

5.3. Performability Projection

To evaluate the effects of technology evolution and network congestion on the availability of the infrastructure with redundant servers, Figure 5 shows the result of computing the percentage of available clients as a function of the link or node failure rate.

With perfect link/node reliability (i.e., p_f=0), redundancy actually reduces client availability because WWW servers are reserved as backups rather than being deployed for production use. However, as the link/node failure rate increases, hierarchies with redundant servers outperform those without redundant structures. The larger the link/node failure rate, the larger the gap between the performance of hierarchies with and without redundancy. With increasing Internet popularity, the surge in networking traffic is likely to result in more network congestion, making redun-

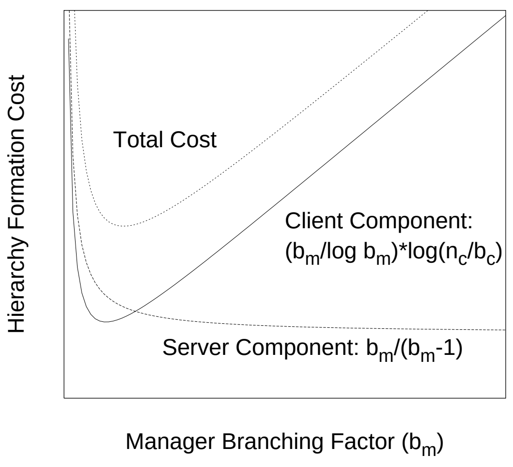


Figure 6. Total Hierarchy Formation Cost

dant structures critical for large scale computing.

6. Hierarchy Formation Costs

To determine whether it is feasible and practical to form large scale computing structures on the Internet given current wide-area latencies, we must quantify the cost of hierarchy construction. Below, we investigate the maximum feasible hierarchy size (i.e., without redundant structures) given current Internet performance and reliability attributes.

6.1. Node Insertion Time Complexity

To estimate the largest scale hierarchy currently feasible, we began by enumerating the number of component operations required to build the hierarchy as a function of the manager branching factor; see [9] for the detailed derivation. Our derivation shows that the time complexity of hierarchy formation is roughly

$$O\Big(\frac{b_m}{b_m-1}\Big)+O\Big(\frac{b_m}{\log b_m}\log(\frac{n_c}{b_c})\Big). \tag{8}$$

The first term in (8) represents the cost of inserting internal nodes; the second term is associated with the cost of client tree descent during client insertion. Based on (8), Figure 6 summarizes the insights we gained from complexity analysis of hierarchy formation.

The figure shows that as one increases the manager branching factor, the cost of inserting internal nodes decreases rapidly due to the exponential decrease in the number of internal managers. However, as one increases the manager branching factor beyond the inflection point, the cost of client tree descent increases due

Operations	Average	Δt
Follow child link	3.51	0.56
Create First Root	10.10	2.84
Make New Root	11.94	3.99
Add internal node	49.59	22.85
Add leaf	9.01	1.49

Table 3. Hierarchy Formation Costs (seconds)

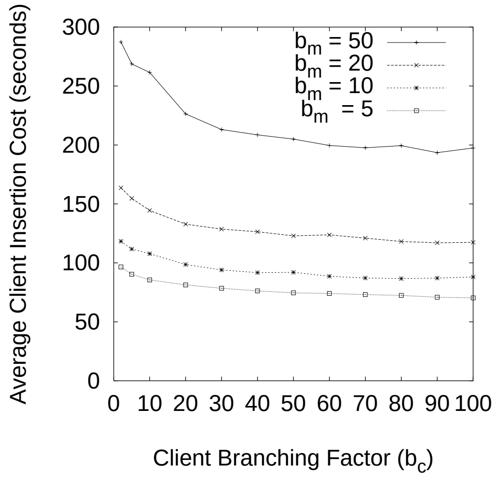


Figure 7. Client Insertion Cost ($n_c = 10^6$)

to the increasing cost of measuring latencies to more servers at each level. That is, despite the decrease in tree height with increasing manager branching factor, the large fanout at each level of the tree increases the cost of client tree descent. The increase in the cost of client tree descent increases the total cost of hierarchy formation, making it infeasible to use large manager branching factors for added reliability.

6.2. Node Insertion Time Scale

To quantify the total cost of wide area hierarchy formation, we measured the component costs of hierarchy formation by constructing wide area hierarchies using machines from Brazil, Chile, Italy, and the United States. We initialized hierarchies with different tree heights and branching factors, and measured the cost of each component operation on an international scale. Table 3 shows the average cost of various component operations and the width of half of the 95 percent confidence interval (i.e., Δt).

Using the data in Table 3 and the results from enumerating the execution frequency of each component operation during hierarchy formation, one can project time time scale for inserting servers and clients in a

Number of Clients	Number of Managers	Total Cost (hours)
20,000	501	16.72
25,000	625	21.51
30,000	750	26.38

Table 4. Feasible Hierarchy Construction Cost

wide area hierarchy. The server insertion cost is about two minutes for hierarchies with over 100,000 clients, suggesting that one should use a manager branching factor between five to ten to lower server insertion cost.

Similarly, Figure 7 shows the effect of client branching factor on client insertion time. One can reduce client insertion time by using large client branching factors. This decreases the number of managers required at the first level, reducing the total cost of insertion. As Figure 7 shows, the added savings from a client branching factor beyond twenty is not significant, suggesting that a reasonably large client branching factor is adequate for reducing the cost of hierarchy formation.

In short, based on the cost and time complexity analysis of hierarchy formation and measurements of current technologies (e.g., network speed), one should build hierarchies using a modest manager branching factor and a large client branching factor to maximize reliability and minimize hierarchy formation time.

6.3. Large-Scale Hierarchy Formation

To project the total cost of forming large hierarchies and to determine the largest, feasible hierarchy size given current technologies (e.g., network latency and processor speed), we combined the cost of server and client insertions to compute the projected total cost.

In this analysis, we made three assumptions. First, we assumed a manager branching factor of ten; second, we used a client branching factor of fifty; and third, we assumed that servers are inserted serially, and clients are inserted in parallel. We allowed up to fifty concurrent client insertions because we believe geographically distributed users will likely insert their clients at different branches of a wide area hierarchy.

Table 4 shows the configurations and costs of building wide area hierarchies that our analysis suggests are feasible given current hardware speeds and a 24 hour hierarchy initialization time.[3] As shown in Table 4, given today's wide area network latencies, the largest hierarchy one can assemble within 24 hours would have approximately 25,000 clients and 600 servers.

[3]These hierarchies have low per server and per client insertion costs because their hierarchy sizes are small compared to a one million node configuration.

7. Scheduling Costs

Because maintaining a global view of the resources available in a hierarchy is essential for efficient job scheduling, and assessing the scheduling overhead is critical for determining task granularity, we also performed cost and feasibility analysis of world view maintenance and task scheduling. The analysis is similar to that described in §6.1-6.2. That is, we enumerated the execution frequency of the component operations for world view maintenance and task scheduling, measured their component costs in a wide area context, and used the measurements to project their costs in a large scale, wide area environment.

Our results show that, for a one million client hierarchy, the time scales of world view maintenance and wide-area task scheduling are both near ten minutes [9]. Thus, unlike hierarchy formation, the short time scales for view maintenance and scheduling make it feasible to maintain a global view of the system and to perform large scale computation.

8 Network Evolution

To place the results in a larger context, we used our models to project the possible hierarchy sizes as a function of changes in network latencies. In this projection, we assumed that the average latencies for international, national, vBNS, and local area networks are, respectively, 300, 150, 60, and 5 milliseconds.

Figure 8 suggests that one can potentially scale to over one million nodes if wide-area network latencies approach those of local area networks, subject of course to time of flight signal delays. At this scale, the number of clients will be two orders of magnitude larger than that of hierarchies feasible with current technologies. As a result, we can view the massive computing power available in the hierarchy as a utility that can be tapped for worldwide dynamic load sharing and distributed information processing.

Given the possible hierarchy sizes shown in Figure 8, we illustrate what applications these hierarchies can enable by estimating the execution time to decrypt a 56-bit RSA Data Encryption Standard (DES) code under these configurations. In this computation, we assume that, with the use of redundant servers, 92 percent of the clients will be available. Furthermore, we assume that all the clients have roughly the computing power of an Intel Pentium 166 MHz processor.

Based on statistics from `distributed.net`'s DES-II-1 project (see `www.distributed.net` for details), we

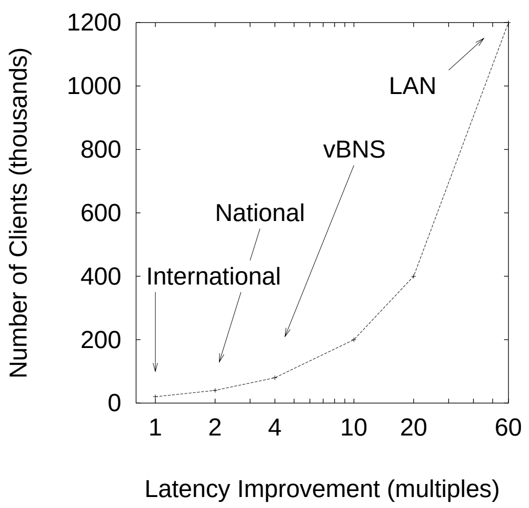


Figure 8. Reduced Latency Hierarchy Size

assume that each system can test 1.2×10^{11} keys per day (5.5×10^6 key blocks tested per day multiplied by 1.1×10^9 keys per key block, and divided by the equivalent of 50,000 Pentium processors used in the project). Finally, we assume one must test all 72 quadrillion (72×10^{15}) keys to break the code.

Using these assumptions, decryption time decreases dramatically from thirty days to under a day with the increase in hierarchy size. Thus, as network and computing technologies evolve, they will enable new kinds of applications that require massive computation power (e.g., knowledge discovery and data mining in distributed databases), solving problems in a small fraction of today's execution time.

9. Conclusions

Until recently, worldwide computing lacks a universal, standard platform for cooperatively harnessing geographically distributed machines for large scale computing. The recent emergence of Java and the World Wide Web (WWW), however, created a common denominator for global computing, enabling us to tap into any machine that has a WWW browser.

Our analysis of a Java/WWW-based infrastructure showed that, given current technologies, it is feasible to build wide area hierarchies with about 25,000 clients and 600 servers. In addition, the analysis suggests using a modest manager branching factor to increase reliability, and a large client branching factor to reduce the cost of hierarchy formation.

We also found it feasible to maintain a global view and perform task scheduling atop an infrastructure that contains up to one million node. As network and computing technologies evolve, large hierarchies will enable new applications to leverage a globally distributed, parallel computer to solve large problems.

Acknowledgments. We thank Doug West for discussing the probability model, and Celso Mendes, Mario Medina, and Giuseppe Serazzi and Rosella Favino, respectively, for access to computers in Brazil, Chile, and Italy.

References

[1] J. E. Baldeschwieler, R. D. Blumofe, and E. A. Brewer. ATLAS: An Infrastructure for Global Computing. In *Proc. of the 7th ACM SIGOPS Workshop on Systems Support for Worldwide Appplication*, Sept. 1996.

[2] F. Berman and R. Wolski. Scheduling From the Perspective of the Application. In *Proceedings of the 5th IEEE Symp. on HPDC*, Aug 1996.

[3] P. Cappello. Javelin: Internet-Based Parallel Computing Using Java. In *Proceedings of the ACM Workshop on Java for Science and Engr. Comp.*, June 1997.

[4] I. Foster and C. Kesselman. The Globus Project: A Status Report. In *Proc. of IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, Mar. 1998.

[5] G. Fox and W. Furmanski. Towards Web/Java Based High Performance Distributed Computing – an Evolving Virtual Machine. In *Proceedings of the 5th IEEE Symp. on HPDC*, pages 308–317, Aug 1996.

[6] A. S. Grimshaw, W. A. Wulf, and the Legion Team. The Legion Vision of a Worldwide Virtual Computer. *Comm. of the ACM*, 40(1):39–45, Jan. 1997.

[7] G. Judd, M. Clement, and Q. Snell. The DOGMA Approach to High-Utilization Supercomputing. In *Proc. HPDC 7*, pages 64–70, July 1998.

[8] M. Kalyanakrishman, R. K. Iyer, and J. U. Patel. Reliability of Internet Hosts: A Case Study from the End User's Perspective. In *Proc. of the 6th Int'l Conf. on Comp. Comm. and Networks*, pages 418–423, Sept. 1997.

[9] T. T. Kwan. *An Infrastructure for Worldwide Parallel Computing*. PhD thesis, Univ. of Illinois at Urbana–Champaign, Comp. Sci. Dept., 1998.

[10] M. J. Litzkow, M. Livny, and M. W. Muka. Condor – A Hunter of Idle Workstations. In *Proc. of the Int'l Conf. on Dist. Comp. Systems*, pages 104–111, 1988.

[11] J. F. Meyer. On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computers*, C-29:720–731, Aug 1980.

[12] B. C. Neuman and S. Rao. The Prospero Resource Manager: A Scalable Framework for Processor Allocation in Distributed Systems. *Concurrency: Practice and Experience*, 6(4):339–355, June 1994.

[13] A. Vahdat. WebOS: Operating System Services For Wide Area Applications. In *Proc. HPDC 7*, pages 52–63, July 1998.

[14] J. Waldo. *Jini Architecture Overview*. Sun Microsystems, Inc., Palo Alto, California, July 1998.