Journal of Network and Systems Management, Vol. 13, No. 2, June 2005 (© 2005)
DOI: 10.1007/s10922-005-4439-4

Design and Implementation of a Resource Manager in a Distributed Database System

Norman Bobroff[1] and Lily Mummert[1,2]

This paper describes a system called Trends for managing IT resources in a production server environment. The objective of Trends is to reduce operational costs associated with unplanned outages, unbalanced utilization of resources, and inconsistent service delivery. The Trends resource manager balances utilization of multiple resources such as processor and disk space, manages growth to extend resource lifetimes, and factors in variability to improve temporal stability of balancing solutions. The methodology applies to systems in which workload has a strong affinity to databases, files, or applications that can be selectively placed on one or more nodes in a distributed system. Studies in a production environment demonstrate that balancing solutions remain stable for as long as the 9–12 months covered by our data. This work takes place in the context of the Lotus Notes distributed database system, and is based on analysis and data from a production server farm hosting over 20,000 databases.

KEY WORDS: Resource management; load balancing; autonomic management; capacity planning.

1. INTRODUCTION

In large-scale server installations, operational and maintenance costs dominate equipment costs. A significant source of cost is attributed to unanticipated problems such as poor client response time, server overload, or exhaustion of server resources such as disk space. These problems require attention from administrative personnel, increasing operational costs. They often occur during peak operating periods, as opposed to scheduled maintenance windows during which intervention would be less intrusive. Thus reducing the risk of such problems, and the consequent need for human intervention, is an important part of reducing the cost

[1]Computer Science Department, IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA.
[2]To whom correspondence should be addressed at Computer Science Department., IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA. E-mail: bobroff@us.ibm.com

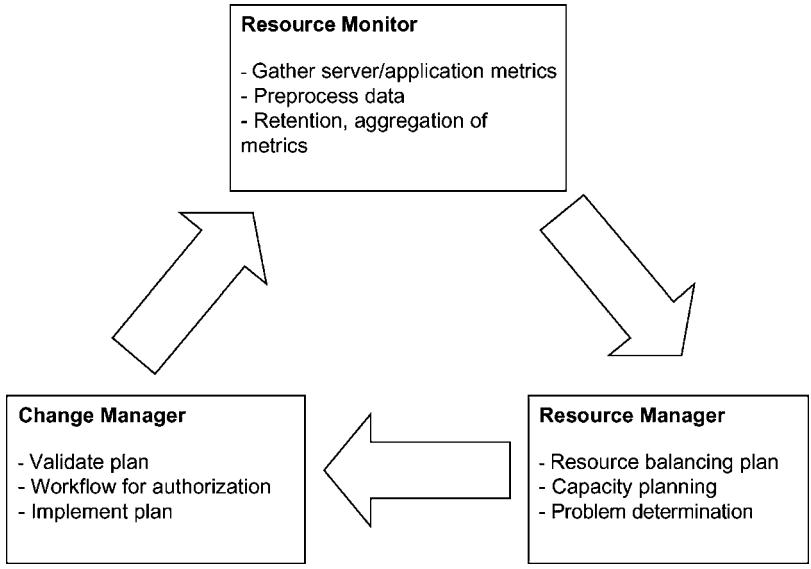

Fig. 1. Primary components of the autonomic manager.

of managing a large server installation. In this paper, we describe an autonomic management system called Trends to address these problems. Figure 1 shows a logical overview of Trends. It consists of three key components.

1. Resource monitor—Gathers capacity and utilization information from the managed system elements, typically servers, applications, network, and storage devices. This component retains historical records of this data to determine trends in the rates of resource utilization. In our implementation this component is distributed, an instance on each server.
2. Resource manager—Generates a plan for how to redistribute workload according to the objectives and methodology appropriate to the system. The resource manager applies administrative constraints to the plan. For example, it may be specified *a priori* that certain applications or databases be grouped together, possibly on specific servers. It also considers trade-offs between accuracy of the plan and cost to implement the plan. This component operates centrally, one instance on each workstation of an active administrator. It retrieves data from the *resource monitors*.
3. Change manager—Accepts the plan submitted by the *resource manager*. The change manager validates the plan (e.g. checks that all servers are available, whether the entire plan can complete off-shift or requires staging, etc), generates a workflow for verification and approval by human

administrators, and automatically implements the plan. The *change manager* resides on one of the servers in the management domain.

The primary subject of this paper is the methodology and implementation of the Resource Manager. The methodology is used to correct resource overloads and prevent outages caused by resource exhaustion (e.g. running out of disk space). It has the following distinguishing characteristics.

- Manages multiple resources. An integrated approach to managing multiple resource dimensions is essential. One system may be underutilized in CPU but running out of disk space, whereas another system is processor constrained. The Resource Manager must generate a plan to redistribute workload among servers to make effective use of both resources on both machines.
- Extends resource lifetimes. Extending the lifetime of the solution is an improvement over point-in-time methodology. Consider a system as it evolves from a condition of satisfactory performance or balanced resource utilization. After some period the system exceeds a threshold, which triggers implementation of a resource balancing plan. When the cost of implementing the resource rebalance plan is low, monitoring and redistribution of workload can occur frequently. (The network dispatcher commonly used in busy web sites to route HTTP requests based on metrics obtained from the backend servers is an example of low cost rebalancing.) However, significant management cost, downtime, or user impact associated with rebalancing demands a methodology that extends the lifetime of the solution. Furthermore, extending the solution lifetime facilitates planning for new resources. Consider a group of servers having the equal quantities of free disk space at a point in time. Applications on each server are consuming free disk space at different rates so that servers will be running out of space at many different times. Management costs are reduced if the resource balance plan places applications so that the lifetime of each server is predictable. This approach is especially attractive in a corporate environment where the lifetimes can mesh with the planning cycles for purchases of resources such as storage and CPU. Thus the methodology should incorporate lifetime forecasting based on the resource capacity and observed rate of consumption.
- Manages variability. It is desirable to minimize and uniformly distribute short-term variability in resource consumption where possible. Consider two servers with equal daily average processor utilization of 80%. One machine has variability of 10%, the other 20%. The server with lower variability provides more consistent response times and can function at a higher average utilization.

The methodology developed here applies to systems in which workload has a strong affinity to databases, files, or applications that can be selectively placed on one or more server nodes in a distributed system. We demonstrate that balancing workload distributions over multiple resources results in stable and predictable resource utilization. A case study in a production environment of Lotus Notes mail servers shows that workload distributions balanced over multiple resources can remain stable for many months. Lotus Notes a distributed database system used extensively in large and medium sized commercial enterprises. The methodology is productized in the server performance component of the Lotus Domino Administrator in version 6.0.

The remainder of this paper is organized as follows. Section 2 provides background on the operation of the Notes database system. Section 3 characterizes the workload and describes our approach to multi-dimensional workload balancing. Section 4 describes aspects of the product implementation of the *resource monitor* and *change manager* and expands on the autonomic approach. It also describes administrative and planning tasks that can be provided by a well architected resource manager. These functions provide considerable value beyond the central design point of correcting resource utilization and distribution problems arising from temporally evolving usage patterns. Section 5 shows the results of applying our methodology to a production environment. Section 6 reviews work on other systems related to the resource management problems confronted in the Lotus Notes environment. Conclusions follow in Section 7.

2. BACKGROUND ON LOTUS DOMINO

The paper emphasizes as much as possible the general aspects of this work. Because the work is implemented in the Lotus Domino application environment we provide a brief background on its architecture, especially features germane to autonomic resource management. Section 2.1 and 2.2 describe the basic architecture and resource balancing options of the Notes environment.

2.1. Domino Server and Database Architecture

Lotus Notes is middleware for messaging and collaborative applications used in a client-server environment [1, 2]. The primary elements are the server component (referred to as Domino) that hosts application databases, and the Lotus Notes client. In a typical Notes application the code and data are bound together within the application database and stored as database elements. This feature facilitates replication, distribution, and synchronization of both data and code. A key element of Notes is the ability to replicate via a peer-to-peer model with weak data consistency [3]. Replication merges divergent database replicas and flags conflicting updates that must be resolved manually. The degree of data

consistency and frequency of updates between replicas is application specific. Replicas of heavily accessed databases are often placed on multiple servers or clients. Replication makes Notes well tailored for low bandwidth, intermittently connected networks or mobile users, and "edge-of-network" servers such as branch offices.

Remote access to server database applications occurs over TCP networks via several protocols. The Notes client uses proprietary remote procedure calls (RPC) and is the preferred method of access. Alternative access is supported for open protocols such as SMTP and POP3 (for mail), although these typically compromise performance and functionality. A significant feature of the Notes client is that it includes the subset of the Domino server code that provides access to application databases. Applications that have been fully replicated to the client can execute when the client is disconnected. This enhances application availability in weakly connected networks. It may also improve performance for connected clients by reducing network latency.

Figure 2 shows a Notes installation with M servers accessed from N clients. Application database "B" is replicated across servers to increase availability and distribute load. Clients create local replicas of key databases to improve performance or work disconnected or with intermittently connected networks.

2.2. Domino Server Resource Balancing Mechanisms and Timeframes

A key aspect of the Notes architecture is the affinity of resource consumption to the database applications. To excellent approximation, the processor utilization, disk space, and network bandwidth used by a Domino server is the sum of the

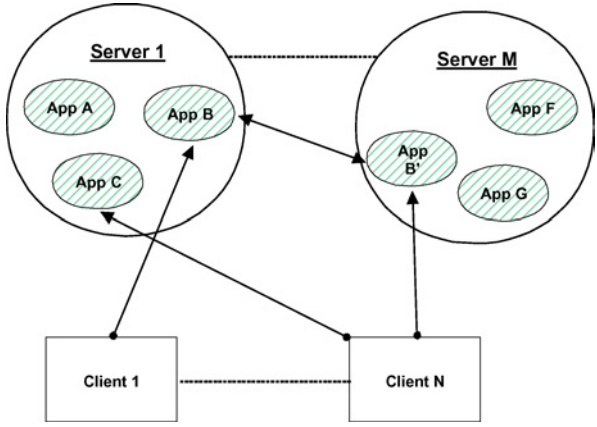

Fig. 2. Overview of notes replicated database architecture.

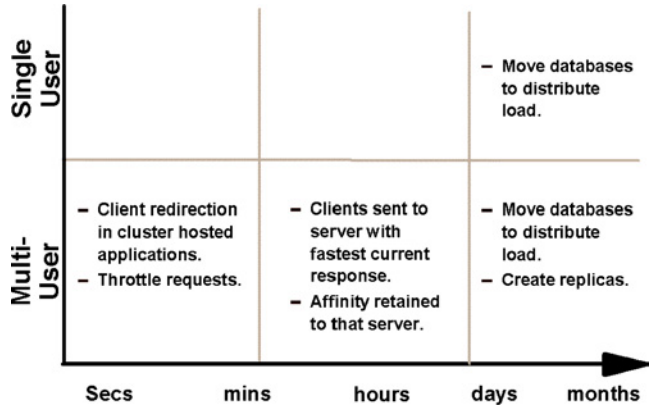

Fig. 3. The load-balancing problem in the time domain.

contributions from the individual database applications on that server. Relocation of an application database from one server to another moves the resource consumption to its destination. Creation of a new replica causes processor and network load to move to the new server in proportion to the fraction of clients (users) that choose to go to the new replica. Thus, moving databases and directing clients are the primary mechanisms of resource reallocation.

It is useful to consider workload imbalance and remedies at different timescales. Figure 3 shows the remedies available to Domino in the time domain. Databases are categorized on the vertical axis according to degree of sharing, single (e.g. e-mail database) or multi-user applications (e.g. collaboration, department financial planning). On a long timescale, databases and replicas are placed to uniformly distribute resource consumption. This is the focus of the *resource manager* in this paper and is especially relevant to single user applications such as email. For short-term balancing, clients can be dynamically redirected to an application replica on a less loaded server. The client maintains an affinity to that replica until the server times its session out. Multi-client applications are often hosted in a server cluster. Client requests to an overloaded server are automatically redirected to another cluster member.

3. RESOURCE MANAGEMENT STRATEGY

The introduction noted that an ideal resource manager balances utilization of one or more resources across servers, the growth or equivalently projected lifetime of the resource, and the variability of resource utilization. This section explains how the concurrent requirements of balancing average resource utilization and variability leads to the concept of equalizing workload profiles across servers.

Equalizing workload profiles also proves beneficial in extending the lifetime and stability of the resource balancing plan as shown in the results of Section 5. Management of variability is also important in achieving efficient utilization of resources.

Section 3.1 characterizes database (user) activity in the Domino server mail environment. Section 3.2 extends the workload characterization to motivate the methodology of balancing workload profiles across servers. Balancing workload distributions is contrasted to "greedy" rebalancing schemes typically used by administrators to reduce overload.

3.1. Characterizing Daily Activity and Variability

Data from a set of 24,000 mail databases is presented with the focus on activity (processor intensive load). This section establishes the linear relation between activity and standard deviation used in the conceptual analysis. Activity is used as the illustrative resource here, but many other metrics are available for resource balancing as described in Section 4.

Data are collected from production Domino release 5 servers running on IBM RS/6000 SP systems with AIX. The servers are dedicated to mail databases accessed from Notes clients. Mail databases in our environment are large and highly active, and mail servers tend to be the most problematic to manage. In Notes, a semantic operation against a database by a Notes client consists of a sequence of RPCs, or *transactions*. Instrumentation on the server logs the type, duration, and I/O activity for each transaction against each database. Operating system instrumentation (such as "vmstat" on AIX) is used to obtain global statistics such as CPU utilization. There is no mechanism in the server code for fine-grained CPU utilization to be accumulated for each transaction against each database. However, it is established that the aggregated transaction count is strongly correlated to system CPU utilization [4]. This correlation applies to the uniform distribution of the types of RPCs within the mail application used in this work. In practice, this is not a restrictive assumption because application deployments in commercial environments avoid mixing application types on the same server. In contrast, the hardware and operating system platforms are heterogeneous. Thus transaction count (also referred to here as activity) is used as a proxy for processor load against each database. All data reported in this paper is collected by post processing Domino server logs on a daily basis. When transactions consume multiple resources they are filtered to more equally compare durations. For example, the duration of a data read transaction depends on both CPU and I/O activity. So response time comparisons within read transactions are made for similar size data transfers.

Figure 4 shows the distribution of mean daily database activity for mail databases on 22 production servers. The servers hosted a total of 24,440 databases occupying 2.51 TB of disk space. We omitted databases created on the server after

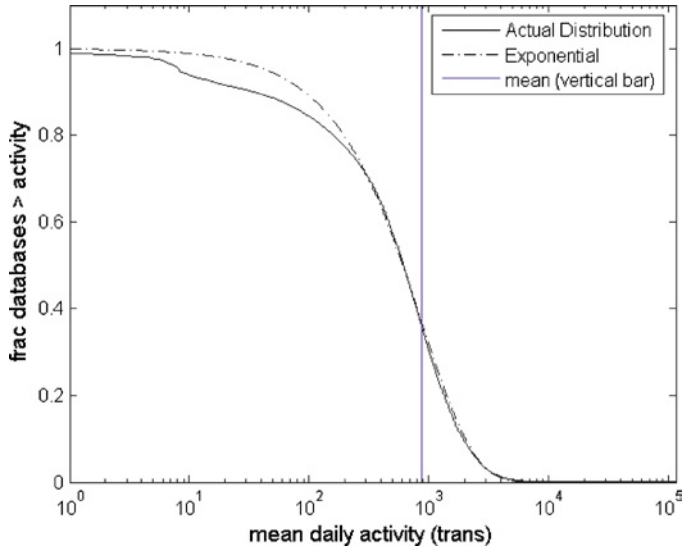

Fig. 4. Distribution of mail databases in mean daily activity.

the start of the data collection period so the figure reflects a population of 21,367 databases. The horizontal axis is the mean activity, while the vertical axis is the number of databases having that mean or greater. For reference, an exponential distribution having the mean of the data is also shown. The actual distribution is very slightly weighted toward high activity compared to the exponential with no free parameters.

The variability in database daily activity is shown in the scatter plot of Fig. 5. Each point on the plot corresponds to a single mail database located by the mean and standard deviation of the daily activity against that database. Databases exhibit variable load because of the stochastic nature of aggregate client accesses, and because of time-dependent accesses from individual clients. The data suggest a power law relates the standard deviation σ of each database to the mean daily activity. In fact, the best-fit slope and normalization of the data are unity so that the standard deviation is equal to the daily activity. This result suggests that the daily activity against each database might be exponentially distributed. However, an examination of individual database access patterns shows that the distribution of daily activity against each individual database is rarely exponential or any other commonly recognizable statistical form. This latter observation indicates broadly varying work patterns of each database's users. However, when data is aggregating from multiple database (about 10 is sufficient), the average usage pattern approaches exponential. Because the number of databases on each server

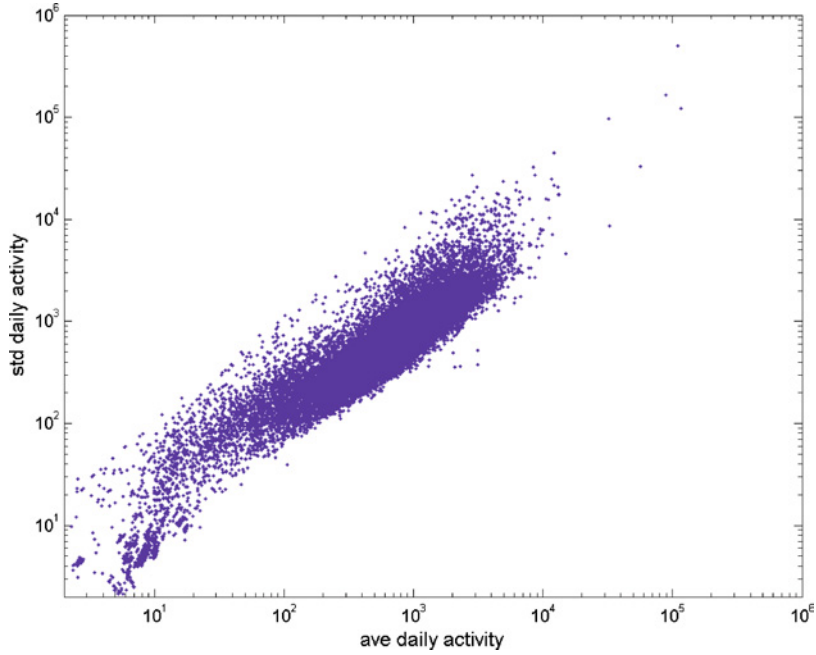

Fig. 5. Daily mean and standard deviation of activity.

ranges from a few hundred to many thousand, server fluctuations in total daily transactions are exponentially distributed in accord with Fig. 5.

A consequence of the linear relation between mean and standard deviation is that a server with an excess of high activity databases has a higher inter-day variability than a server with the same mean but an excess of low activity databases. In this case, managing variability is a distinct goal from managing load.

3.2. Managing Workload Distributions and Variability

The objective is to distribute the database applications across a heterogeneous set of server nodes so that resource loading and variability are distributed uniformly and in proportion to capacity of each server. This result is achieved when the form of the distribution of workload on each server is that of the aggregate of all workload. This solution is independent of the actual distribution, it happens to be exponential for the activity workload of Fig. 4. This conclusion is largely intuitive, but we provide a simple example.

Consider N databases to be placed on two servers with such that the servers have equal mean activity and coefficients of variation (COV). Suppose, as for

exponential distribution, that the standard deviation of activity for each database is equal to the mean. The N total databases consist of two subsets with exponentially distributed mean daily activity; N_1 databases with mean activity A_1, and N_2 databases with mean activity A_2. The total daily activity within each set is $N_1A_1 = N_2A_2$. Because variances of independent variables add, the variances for each set are $N_1A_1^2$ and $N_2A_2^2$, respectively. The COV V_i is given by $V_i^2 = N_iA_i^2/(N_iA_i)^2 = 1/N_i$. If each subset is placed on a separate server, the average load is balanced but the variability is not. On the other hand, equally dividing the subsets among servers to achieve equal workload distributions balances both load and variability. We also emphasize that our approach is to simultaneously balance distributions in multiple workload dimensions.

Figure 6 summarizes this aspect of the resource management methodology. Figure 6a shows two servers in a two dimensional slice of resource space, in this case database activity and free disk space. A target is set for each server by apportioning the total amount of resource according to the processor and disk space capacity of each server. The targets are indicated by the solid lines. The figure shows the pre-balanced location of two servers in the plane of activity and disk space. The goal is to bring the servers to within a specified tolerance of the target T by redistributing the databases. The databases are to be moved so that the condition of equal resource profiles is met, as illustrated in (b). The broken lines indicate the initial distributions of the databases on servers 1 and 2. After balancing, they should have the common distribution indicated by the solid line. An additional goal is to balance the growth rate in resource consumption so as to uniformly distribute the lifetime of each server. Implementation is described in Section 4.

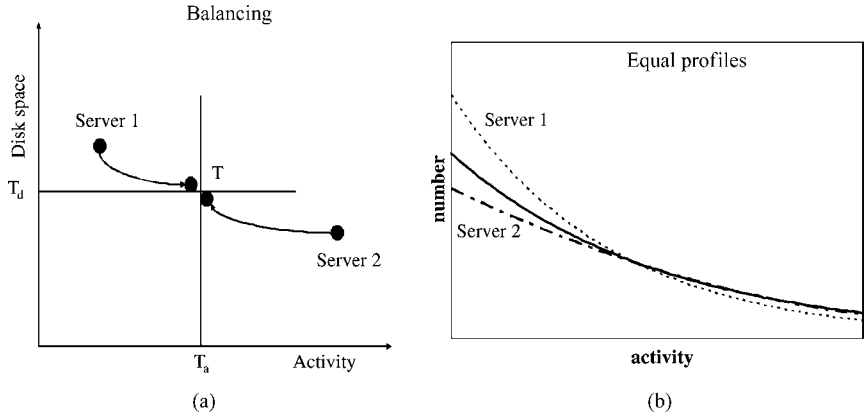

Fig. 6. Balancing total load and disk space (a) while equalizing the database distribution (b).

Prior to introduction of the uniform workload distribution methodology, administrators of large installations typically applied a "greedy" heuristic to balancing servers. When a server runs out of a resource, the most resource intensive applications on that server are moved to the least loaded server. From an algorithmic point of view this is the solution with the fewest moves. However, a consequence is that servers that were lightly loaded become populated with the most active, variable, and growing application workloads. The resource lifetimes of the servers becomes unbalanced, some fail or require rebalancing on a timescale of weeks, whereas others are stable for months. A key benefit of the management strategy presented here is that it provides uniform and predictable resource lifetimes across all servers. This allows capacity upgrades to be planned and increases server availability. Quantitative results from the production environment are presented in Section 5.

4. IMPLEMENTATION

The system consists of three primary interacting software components as described in the Introduction and Fig. 1. These are the *resource monitor*, *resource manager*, and *change manager*. The main functions provided by each component, and their interactions are described in Section 4.1. One interesting aspect is the connection between the *resource manager* and enterprise capacity planning. This connection and the synergy between the administrator and the automated functions are covered in Section 4.2. Section 4.3 describes the rebalancing algorithm used to implement the methodology discussed in the previous section.

4.1. System Overview

A *resource monitor* (introduction and Fig. 1) resides on each server. It is implemented as a Domino database application and collects system and storage configuration, capacity, and individual usage statistics for each application databases hosted on that Domino server. Over 20 metrics are collected; data commonly used for resource balancing includes transactions (activity), disk space, reads/writes, bytes read/written, Domino proprietary RPCs, and HTTP accesses. Data are preprocessed at the server and stored in a local database. Preprocessing includes aggregating the statistics into 5 min summaries. Summaries are also generated for a prime shift and off shift. Additional processing computes the exponentially weighted average for each statistic and a linear estimate of the growth rate. The exponential averaging window defaults to 3 weeks. Raw data is retained for a configurable time period, defaulting to a week, to be used for post-mortem problem determination. A snapshot of the processed data is taken each week and retained indefinitely. Parameters that configure the preprocessing (e.g. averaging times, prime shift hours, holidays, etc.) are set centrally using the administrative

client. They propagate to the *resource monitor* on each administered server via Notes replication.

The *resource manager* accepts as input many constraints on where to place applications. An administrator may define groups of application databases that have attributes such as "keep together," "don't move," "must move," or "assign to a specific server." Another common constraint is to define servers as sources or sinks of applications. A source (sink) server has the property that applications can only be removed (added). The output of the resource manager is itself a Domino database application containing a list of application moves from source to destination server. The list is passed to the *change manager* for administrative approval and automatic implementation.

The *resource manager* presents a graphical view of the state of current resource utilization on each server. The view allows drilldown on each server to show consumption by application. Data is collected on a per user basis, so further drilldown allows the administrator to view the most active users. All these views can be sorted on the resource metric which is extremely useful for rapid problem determination even in systems of hundreds of servers and thousands of users. The administrator is provided measures of how far the system presently deviates from balance (e.g. average and root mean square deviation of the servers from their target utilizations for each resource.) Additional views show the uniformity of workload distribution on each server. Examples of these graphical views are provided in the results Section 5. On the basis of this data, the administrator decides whether a rebalancing is necessary.

The *change manager* receives the plan and checks its validity. It ensures the plan can be implemented automatically by verifying it can contact all servers and has the correct permissions. The *change manager* can decide to stage the plan if there is insufficient time in an allocated maintenance window to fully complete execution. If the plan meets these tests, the *change manager* submits requests for approval to one or more administrators. Upon approval, the plan is automatically implemented during specified maintenance periods.

4.2. Administrator Interaction

In a fully autonomic solution, policies specified by an administrator (perhaps combined with self-learned behaviors) are used to close the resource management loop of Fig. 1. The decision to generate a rebalance plan could be automated based on policies provided to the resource manager. For example, if processor load exceeds a threshold T on any server or the overall misbalance in disk volume lifetime exceeds 30 days, and no disks are to be purchased for 45 days, then rebalance. In practice, there is always a tradeoff between full automation and direct administrator assessment and decision making, and it is difficult to fully automate the loop. In a large-scale production environment, even one that is fairly mature,

each instance of a resource rebalancing action may have unique requirements. For example, servers or users may be added and removed, or resources consolidated or migrated to new equipment. Full automation requires carefully developed and tested policies which have long-term stability. Because the cost of testing is high, the penalty for a mistake large, and the business environment is not stationary, it is cost effective to have human intervention in this task. The majority of cost savings in the production environment is realized by the automated collection of data, the stability and predictability of the system resulting from the rebalancing, and automated implementation of the plan by the *change manager*. Even in a fully automated system, a synergistic approach in which human administrators assist the software elements, particularly in validating plans prior to automatic implementation, is desirable to build trust in the system.

The resource manager exposes its resource balancing function to the administrator so that it can be used as a central element of enterprise capacity planning. This function is serendipitous to the roles usually associated with autonomic management. Resource balancing is closely related to capacity planning. Resource balancing distributes applications on existing servers to provide uniform resource utilization. Capacity planning asks what servers and resources are required to achieve specified utilization levels on the basis of the present or future demands of a set of applications. The *resource monitors* provide current demand and estimate future requirements through linear fits to the resource utilization data. The *resource manager* allows administrators to define a future system configuration by removing existing servers and adding "new" servers. The distribution of applications on servers and predicted resource utilization of is presented to the administrator. This data is used to verify that planned purchases of equipment meet requirements.

4.3. Algorithm

This is an overview of the algorithm used to achieve the multidimensional resource balancing. More details can be found in [5]. The problem is similar in formulation to multidimensional bin packing with the additional constraint that the distribution of items in the bins (servers) be approximately equal. The problem is NP-hard, so a heuristic algorithm is used based on a greedy approach. The algorithm directly balances any two resources selected by the administrator from the approximately twenty available database metrics. The default resources are prime shift RPCs and disk space. Some correlation is expected between metrics. For example, a high daily RPC rate tends to correlate with increased network I/O. Thus, balancing provides benefit to all resources. This behavior is demonstrated in the results of Section 5.1. The administrator also specifies a desired tolerance for the balance solution by specifying the root mean square deviation of all servers from the target in each resource dimension. Decreasing the tolerance increases the number of moves.

The output of the algorithm is a list of databases to move from the source to destination server. This move list meets the objectives of the methodology described in this paper. We now describe the three primary phases of the algorithm.

1. Assign targets for the amount and distribution of resource to place on each server—The total resource used by all applications on all servers is calculated on the basis of data provided by the *resource monitors*. A target is set for each server by distributing the total resource in proportion to the resource capacity of the server.

Workload distributions are equalized by dividing the server level target into sub bins. Each sub bin will be populated with applications having similar resource requirements. To create sub bins, the distribution of resource demand for all applications on all servers is computed for each of the two selected resources. (A typical result would be the distribution of RPC workload illustrated in Fig. 4.) The distributions are used to divide the applications into three bins. The bin boundaries are computed as the values that include the greatest 30%, middle 40%, and lower 30% of applications. These bin sizes are configurable parameters. For two resource dimensions this results in nine bins. Targets are set for each bin on each server using the bin boundaries computed above. The bin targets are simply proportional to the load to be placed on the server, bin size, and server capacity. Section 5 shows examples of the distribution of database applications into bins for the production environment.

On average, some bins on some servers are over target capacity, while others are under. Because we want to minimize the number of applications moved, we don't remove all the databases and redistribute them. Therefore, once targets are computed the algorithm enters a remove phase followed by a place phase.

2. Remove phase, compute which databases to remove from servers—In the remove phase, databases are taken from any of the nine bins that are over target in either of the two resources on each server. These databases are placed on a global move list to be reassigned to another server. The remove phase defines temporary targets for removal that are configurable and below the final targets. This is done to provide a greater set of databases on the place list to improve the balancing. The configurable level of the remove phase target provides another way to trade accuracy against the expense of more moves.

The databases in each bin are sorted in descending order by resource metric. Because balancing is done on two resources, the sort is done on the resource furthest from the target. (The most distant resource is continually reevaluated as the algorithm proceeds.) The algorithm removes databases from this list until the bin is under target. If the algorithm proceeded in sorted order it would be a pure greedy heuristic. This minimizes moves, but may cause systematic imbalances by always choosing the applications with the greatest resource consumption. Some parameters are provided to tune the "greediness" and improve the uniformity of

selection, again at the expense of number of moves (e.g., take every *n*th database). During the subsequent place phase a database may be returned to its original location.

3. Place phase, compute a destination server for each entry on the move list—The move list generated by the prior phase is processed in steps, each step the placing of *N* databases. *N* is configurable defaulting to the maximum of 6 or 5% of the current size of the place list. At each step, the "move" list is sorted in descending order by the resource currently furthest from target, and placement starts from the database at the top of the list. A target server is calculated by scanning the set of servers for the resource sub bin with greatest capacity capable of receiving the database. This is referred to as "worst fit" in contrast to "best fit" and "first fit" heuristics commonly used in bin packing. Empirically, "worst fit" is the best performer. After each of *N* placements, the system computes the standard deviation from target for the servers in each resource dimension. If the system is within tolerance of its final goal, the placement algorithm puts databases from the move list back on their original server when possible. This affinity to the source server continues until the balance goes out of tolerance and the algorithm switches to its place mode. The algorithm terminates when the "move" list is exhausted.

A key element of both the remove and place phase is additional balancing applied at the file system level of the target server. This is done to uniformly distribute the load and disk space growth rate among the file systems on physical volumes so that the lifetime of each volume is equalized. Databases are selectively removed from or placed on the optimal volume to achieve these objectives.

Finally we note that the success of the bin-packing algorithm depends on non-sparse population of the databases in the multidimensional space of resources chosen for balancing. A sparse population may not have sufficient "shapes" to uniformly occupy the bins. This is not a problem for mail databases central to this study (see [5] for a map of the population in a two dimensional resource space). Mail servers host hundreds to many thousands of databases and the distributions follow a continuum as suggested by Figs. 4 and 5. Finding a good heuristic algorithmic is more difficult for servers hosting a smaller number of heavily accessed enterprise applications. The temporal access patterns of large enterprise applications are sporadic and difficult to predict from historical data. In this case, the algorithm suggests the creation and placement of new database replicas to manage the load.

5. RESULTS

The methodology has been developed and deployed in a large production environment. Encouraging results have been obtained in several areas. Among these are the load balance and disk space management results and especially their long-term stability. It is also unreasonable to expect that the methodology

can accommodate every server and application. There are individual servers and databases for which the historical usage patterns of each database do not project into the future. We have observed database applications whose variability greatly exceeds the range shown for their mean in Fig. 5. Therefore, a single database can dominate the solution. However, the instrumentation and methodology of this work are effective in quickly identifying such special circumstances. In practice, outlying behavior is typically a consequence of abnormal activity (e.g. a database indexing task that runs more frequently than required), and is easily remedied.

5.1. Short-term Balancing

Figures 7 and 8 show the workload distributions before and immediately after balancing 22 production servers hosting the databases of Fig. 4. The vertical scale is normalized, corresponding to 1.32 M transactions for Fig. 7, and 217 GB for Fig. 8. The servers are balanced on the basis of transactions and disk space. The distributions are shown as three classes of workload—heavy, medium, and light, which correspond to the top 30%, middle 40% and bottom 30% of all workload. For example, the most heavily loaded servers with respect to disk space contain a disproportionate amount of large (heavy) databases. Transactions are balanced to within 2%. Initially there was over a factor of three between servers in disk

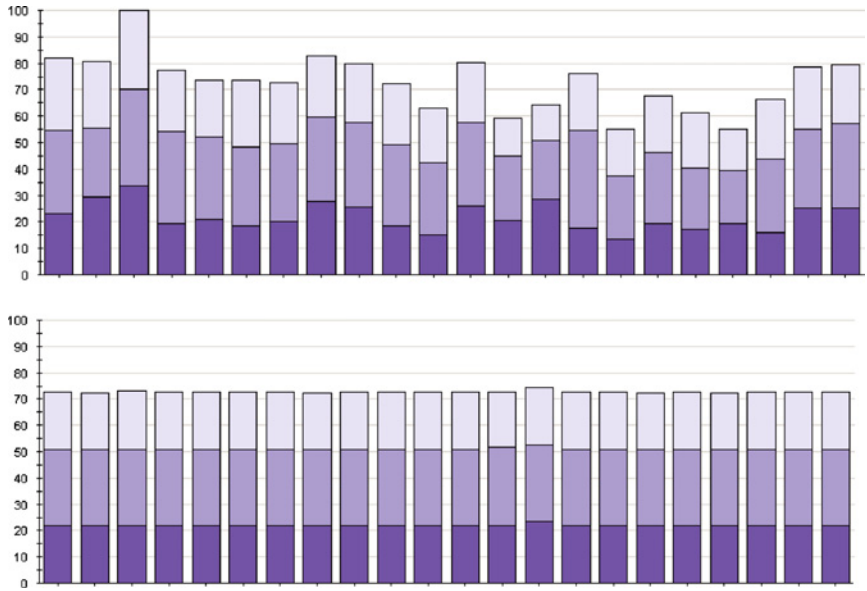

Fig. 7. Transaction totals and distributions before (top) and after balancing (bottom).

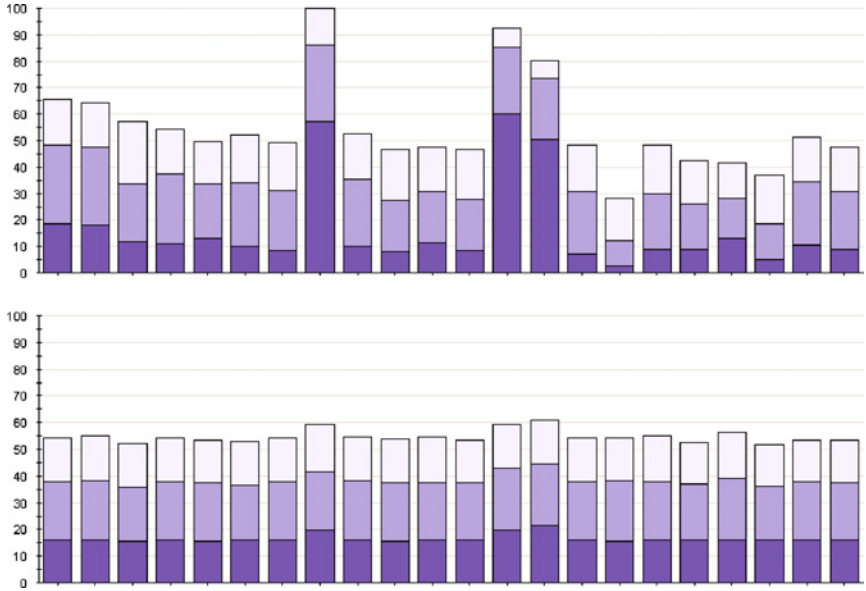

Fig. 8. Disk space totals and distributions before (top) and after balancing (bottom).

space, with distributions on three servers disproportionately heavy. Disk space is balanced to within 12%. To achieve the final configuration, 7228 of the 24,440 databases were moved.

Explicit balancing on just two resources such as activity and space typically results in lowered variance for other resources such as daily network traffic, because activity against a database involves a multitude of resources. This effect is demonstrated in Fig. 9, which shows the COV for several metrics before and after corresponding to the balancing of Figs. 7 and 8. The COV is reduced for metrics such as amount of data transferred to and from the server, documents read and written, and number of databases. The number at the top of each pair of bars is the mean over all servers for each metric.

5.2. Long-term Stability

Balanced server distributions lead to stable and predictable patterns of growth and resource consumption. This is the basis of accurate capacity planning, a key component of cost of ownership in large enterprises. Capacity planning is a complex problem for a distributed system. Resource consumption and growth are planned at multiple hardware levels such as server, network adapter, disk storage, and bandwidth. Failure to upgrade resources leads to poor response at

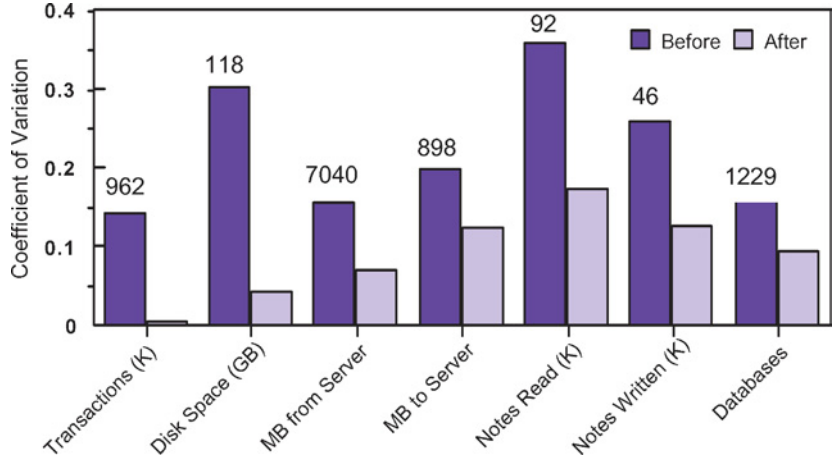

Fig. 9. COV for other metrics before and after balancing.

best, and more typically unplanned outages and costly human intervention. In large data centers, capital resource upgrades are typically planned around business cycles. Unpredictable patterns of resource demand disrupt scheduled upgrades and maintenance.

Figures 10 and 11 show how balanced workload distributions persist over time. Balanced workloads were constructed on 13 production mail servers. The servers were RS/6000 4-way SMPs running AIX, and collectively hosted a total of 19,024 databases occupying 2.86 TB of data. Workloads were migrated in stages to accommodate considerations such as relieving overloaded servers, deployment schedules of production and backup servers, and availability of administrative personnel and maintenance windows. Figure 10 shows the workload distributions for Notes transactions one month after placement (left) and 9.5 months after placement (right). Although the overall totals are somewhat uneven compared

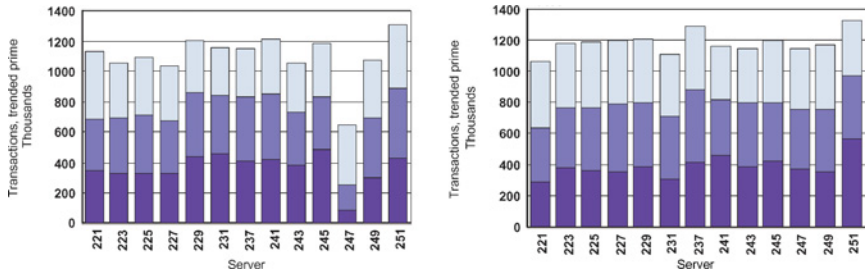

Fig. 10. Transaction totals one month (left) and 9.5 months after placement (right).

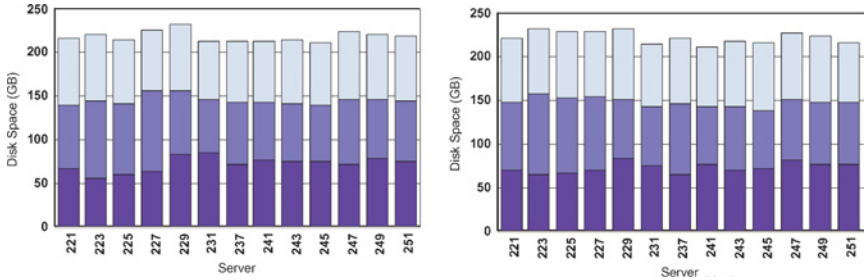

Fig. 11. Disk space totals one month (left) and 9.5 months after placement (right).

to the results in the previous section, the distributions are balanced and both workload and distributions remain stable in the long term. (The data for server 247 one month after placement is an underestimate because data collection was reinitialized during the trending period.) Figure 11 shows the distributions for disk space, with similar stability.

To determine if the COV was stable over time, we examined a set of 6491 databases placed on four servers 1 year after placement. The servers resided on a single IBM eServer zSeries 900 running Linux. Figure 12 shows the server loads in transactions at the beginning and end of that year. The totals did not remain as balanced as in the previous case. The distributions improved slightly (using a distance measure computed as the sum of the differences from the optimal low and high percentages). Figure 13 shows that the COV did not grow during that time, and is roughly equivalent between servers. Each data point in the figure represents the variation over 30 business days of activity for a server. The first point is placed

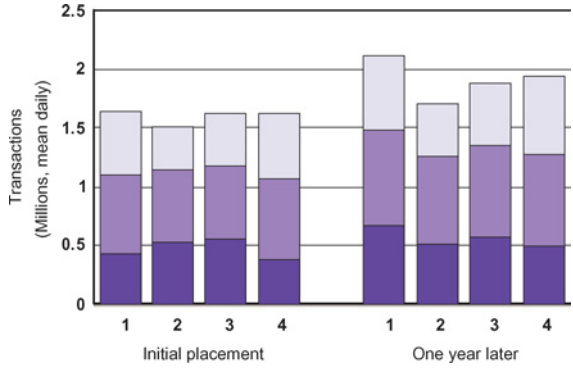

Fig. 12. Transaction loads of 6491 databases placed on four servers.

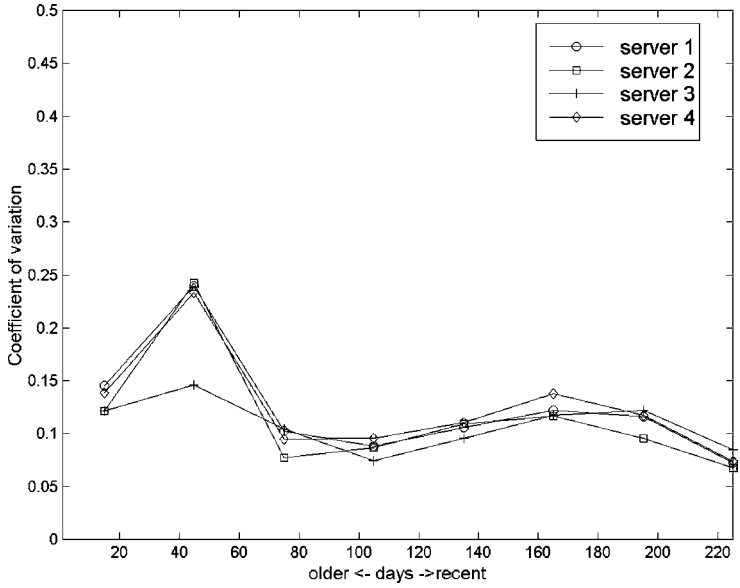

Fig. 13. COV for the four servers in Fig. 12 over 1 year of business days.

in the middle of the 30-day interval. The spike around day 40 is due to lower than normal activity over holidays in December. We were unable to obtain data for server 3 from December 21–January 1 inclusive, the observation for that server reflects the activity with those dates omitted.

6. RELATED WORK

Because of the strong affinity between workload and databases in the Notes environment, resource management as described in this paper is most similar to the problem of assigning data to a group of computer systems in a network for the purposes of query, update, and execution, which has been studied extensively in the form of the File Assignment Problem (see for example the survey in [6]). Dowdy and Foster observe that the file assignment problem is inherently intractable, and proceed to classify and compare the existing work on the problem [7]. All solutions attempt to assign files to computer systems in some "optimal" fashion, where the approaches differ depending on the notion of "optimal." Broadly, the formulations of the problem either optimize for cost, such as storage, query, update, or communication costs, or performance measures, such as response time or throughput. Resource capacity constraints are limited to available storage on a computer system, and in some cases, network channel capacity. Depending on

the formulation, the solutions may or may not be optimal. In practice, heuristics provide good solutions. Further, since the input data characterizing the workload is typically a statistical measure, and the workloads change over time, an optimal solution based on such measures may not be better in practice than a good sub-optimal solution. We describe a few instances of file assignment and data placement problems that are most closely related to the problem described in this paper.

Wolf's formulation of the file assignment problem, which optimizes I/O response time, includes practical considerations such as placement constraints [8]. He observes that the solution is useful in two modes: an "unlimited" mode, which can be used to create an initial placement, and a "limited" mode, which tunes a placement with a limited number of moves. Because of the overhead of moving files, and changes in file workload measures, the predominant usage is "limited" mode. Though the algorithm described in this paper has a different objective, it also incorporates placement constraints (e.g., source and destination servers, pinned databases), and can be used in ways analogous to "limited" and "unlimited" mode.

Hill's method for placing datasets on storage devices to balance device utilization uses a Cartesian representation like that in Fig. 6a, with axes of access time and storage volume [9]. The method takes into account the number of accesses per unit time and the total storage capacity of each device, and the demand of each dataset in terms of the number of accesses per unit time and the volume of data in the dataset. Data is assigned to storage devices on a best-fit basis with the residual capacity of the device.

Pope *et al*. describe a method for assigning workload to servers based on the life expectancy of the resources associated with the servers and used by the workloads [10]. Target life expectancies may be set for specific servers that satisfy administrative constraints such as maintenance windows. The resources define a multidimensional space called a capacity space. The life expectancy of a server in each resource provides a normalized measure over all resources and positions it in capacity space. Workloads are assigned and reassigned to servers based on their effect on the position of source and destination servers in capacity space relative to their targets. The algorithm described in this paper manages life expectancy in a restricted way by balancing growth rates.

Previous work in load balancing recognizes that variability in the metrics of interest should be taken into account [11, 12]. Lee *et al*. studied the assignment of files to disks in a parallel I/O system to balance load and minimize service time variance [13]. They observe that with a multi-class workload, minimizing service time variance is as important as balancing load to minimize mean system response time. Their algorithm assigns files with similar service times to the same disk, in such a way that the average disk load is balanced. This approach minimizes the service time variance for individual disks. One of the underlying assumptions of their algorithm is that service times and access rates are fixed. In contrast, in our

environment, database activity is not fixed, and the standard deviation of database activity is proportional to the mean. For this reason, segregating workloads in this way would not minimize variability of database activity, as discussed in Section 3.2.

Bozman optimized I/O performance in the CMS timesharing system by placing mini-disks (collections of files) on disks to minimize seek distance [14]. Mini-disks were grouped by high or low coefficient of variation (COV) in daily activity, then placed to balance mean daily I/O rate over the disks. One year later, placements were stable with respect to seek distance for the low COV groups, and the overall balance of daily I/O rate was stable for both low and high COV groups. Intuitively, the behavior of low COV groups was predictable, whereas the fluctuations of mini-disks in the high COV groups compensated for each other. An important segment of the mini-disk population was mini-disks with high activity and low COV, which as Fig. 5 shows has no analogue in our study.

In video and multimedia servers, high service time variance can cause jitter. A variety of data placement strategies are used to provide sufficient throughput and bound service times, such as disk striping and data duplexing [15]. Heavily and lightly accessed data may be mixed to balance load across disks and maximize use of both bandwidth and space [16]. Finally, Rommel classified workloads as "ordinary" or "hogs," and observed that the probability of load balancing success is increased with a mixture of ordinary processes and hogs compared to virtually equal processes [12]. Our algorithm's strategy of constructing "clones" of the overall workload distribution is consistent with these approaches.

The work reported in this paper is based on an assumption of balancing across a domain of general purpose servers. Less symmetric balancing may be justified when different servers have been tuned for different types of workload. An example is in the placing of a known workload distribution on a collection of servers. Crovella *et al*. [17] studied optimum partitioning of the long-tailed workload distribution in which all jobs within a certain size range are sent to a dedicated server, though without considering variance. In a Notes environment this approach corresponds to the situation in which dedicated servers are used for mail because mail databases exhibit similar workloads and require similar administrative expertise. These considerations are among the inputs to the complex problem of managing a distributed heterogeneous computer system.

7. CONCLUSIONS

We described some of the key implementations features of a complex and largely automated resource manager. By managing variability, growth, and multiple resource dimensions we achieve cost reductions by reducing outages, management costs and improving the average utilization of capital equipment. The approach scales to large numbers of servers. It also automates and greatly improves

the ability to do capacity planning. Using empirical data from production Lotus Notes servers, we found that balancing workloads over multiple dimensions reduced the COV for resources other than those explicitly balanced. An examination of two sets of production servers 9–12 months after constructing balanced workloads and distributions showed that distributions and COVs were stable. Qualitative observations from the operations staff indicate that the servers were stable, they performed similarly, and that capacity planning was easier over the study period.

ACKNOWLEDGMENTS

Bucky Pope and Bill Tetzlaff provided insightful comments that strengthened this paper. Our reviewers were helpful in improving its presentation. Other members of the Trends project contributed substantially to its success: Kirk Beaty for his work on data collection and analysis, Simon Pope for the change manager, and Ray Mansell for data exploration and visualization. Perhaps the most important contribution was made by members of our data center staff, notably Elsie Ramos, Lorrie Renz, and Dick Williams, for their willingness to deploy our tools in a production environment.

REFERENCES

1. L. Kawell Jr., S. Beckhardt, T. Halvorsen, R. Ozzie, and I. Grief, Replicated document management in a group communication system, in D. Marca, and G. Bock, (eds.), *Groupware: Software For Computer-Supported Cooperative Work,* IEEE Computer Society Press, Los Alamitos, CA, pp. 226–235, 1992.
2. J. Lamb and P. Lew, *Lotus Notes & Domino 5 Scalable Network Design*, McGraw-Hill, 1999.
3. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, Epidemic algorithms for replicated database maintenance, *Proceedings of the Sixth Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada, pp. 10–12, August 1987.
4. W. G. Pope, Planning domino e-mail servers using notes transactions, Proceedings of the 24th International Conference for the Management and Performance Evaluation of Enterprise Computing Systems, Anaheim CA, December 1998.
5. N. Bobroff, K. Beaty, and G. Bozman, A Load Balancing and Resource Management Strategy for a Distributed Database System, IBM Research report RC-21933, October 2000
6. B. Gavish and O. Sheng, Dynamic file migration in distributed computer systems, *Communications of the ACM*, Volume 33, No. 2, February 1990.
7. Lawrence W. Dowdy and Darrell V. Foster, Comparative Models of the File Assignment Problem, *Computing Surveys*, Volume 14, No. 2, June 1982.
8. Joel L. Wolf, The placement optimization program: A practical solution to the disk file assignment problem, *Proceedings of the 1989 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Oakland, California, United States, pp. 1–10, 1989.
9. Reed A. Hill, *System for Managing Data Storage Based on Vector-Summed Size-Frequency Vectors for Data Sets, Devices, and Residual Storage on Devices*, U.S. Patent No. 5,345,584, September 6, 1994.

10. Pope, William G and Mummert, Lily B. Using capacity space methodology for balancing server utilization: description and case studies, *26th International Computer Measurement Group Conference*, December 10–15, 2000, Orlando, FL, USA, Proceedings. Computer Measurement Group 2000.
11. D. Ferrari and S. Zhou, An empirical investigation of load indices for load balancing applications, *Proceedings of Performance '87, the 12th Annual International Symposium on Computer Performance Modeling, Measurement, and Evaluation,* pp. 515–528, 1987.
12. C. G. Rommel, The Probability of Load Balancing Success in a Homoegenous Network, *IEEE Transactions on Software Engineering*, September 1991, pp 922–933.
13. L. Lee, P. Scheuermann, and R. Vingralek, File Assignment in Parallel I/O Systems With Minimal Variance of Service Time, *IEEE Transactions on Computers*, Vol. 49, No. 2, pp. 127–140, February 2000.
14. G. Bozman, *A File Allocation Study in a Large Time-sharing System*, MS thesis, New Jersey Institute of Technology, 1980.
15. R. Flynn and W. Tetzlaff, Disk striping and block replication algorithms for video file servers, *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, 1996.
16. A. Dan and D. Sitaram, An online video placement policy based on Bandwidth to Space Ratio (BSR), *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 22–25, 1995.
17. M. Crovella, M. Harchol-Balter, and C. Murta, Task assignment in a distributed system: Improving performance by unbalancing load, *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, June 1998, Madison, WI.

Norman Bobroff has been at IBM T.J. Watson Research Center since receiving his PhD in Physics in 1983. After working in optical lithography and laser metrology he joined computer science. His current interest is storage area networks and server performance.

Lily Mummert is a researcher at IBM T.J. Watson Research Center in Hawthorne, NY. She received a PhD in Computer Science from Carnegie Mellon University in 1996. Her interests include distributed and mobile systems, self-managing systems, and distributed systems management.