Holistic Quota Management: The Natural Path to a Better, More Efficient Quota System

Michael Gilfix – Tufts University

ABSTRACT

Disk quota systems exist to protect a limited resource and ensure that users can share it. However, existing quota management systems concentrate on controlling user privileges, rather than protecting resources. This paper suggests a new management model based upon a holistic view of resources and their controls. By acting upon a resource globally rather than upon individual users, the new approach exposes trends, allows for better resource planning, and allows for easy understanding of the impact of changes on a user's ability to accomplish work. A new tool 'Qualm' is the first component of a new system for dynamic resource management that allows for decisions based not upon fixed limits for resource usage, but upon limits that change with usage patterns and demand.

Introduction

Efficient quota management is difficult. As one frustrated admin put it: "I've been making noise for the last couple of years that I think we should increase (at least double, probably more) the default quotas, but the analysis required to figure out where to increase them has been scaring me." This frustration is a product of the limitations of current approaches to quota management; while viewing and modifying quotas for individuals or a segregated sub-section of the user population is relatively simple, it remains difficult to ascertain current file system usage, determine where change is needed, and assess the impact of a change.

This paper presents a new model of quota management designed to address these shortcomings. The model approaches the problem holistically; rather than focusing on privileges for individual users, it focuses on relative resource share and the global effect of change. The result is a paradigm where only changes that assure the integrity of the underlying resource are valid.

Using this paradigm, a new tool, 'Qualm,' was created. Qualm works on top of existing quota systems to provide a simple means of performing global analysis, as well as a framework for making quota changes in a global context. Qualm employs a holistic approach, using multiple graphical formats to display the state of the entire quota system. These displays allow for quick assessment of the state and efficiency of the quota system at a glance, and provide a means for global manipulation of the underlying system.

The Existing World of Quotas

The current most popular, freely available quota management tools, such as the UNIX *quota* [8] utility and the NT quota system [12], solve the problem of quota management with usage limits for individual users. Users are given two kinds of limits on the amount of disk space they may use: hard and soft limits. Hard limits are absolute and can never be exceeded. Soft limits may be exceeded by the user, but the user must subsequently lower his disk usage below the limit within a given time frame or suffer a consequence. These tools also provide limited facilities for the creation of user groupings, where a user inherits the quota limit of his group. However, these group mechanisms offer little benefit beyond the ability to administer the quota level of multiple users in one place.

These tools, however, suffer from other severe limitations. Changes to the quota system are singular, meaning they are made without regard to the global state of the quota system and the underlying storage. Moreover, these tools offer no easy way of assessing the current state or efficiency of the quota system, thus rendering global decisions difficult.

Commercial products targeted at the Fortune 1000 genre, such as Precise Software solution's *StorageCentral SRM* technology [15] (which will soon be appearing in a future version of Windows, thanks to a strategic alliance with Microsoft) offer a step up over their freely available counterparts by bringing the kind of flexibility that one would expect from an enterprise solution. Using Precise's software, the system administrator can set up five *different* kinds of quota limits, better divide users up into service groups, and generate several kinds of reports, from current space usage breakdown by file type, to user or group usage reports. The software even provides a facility for some basic trend analysis: a system administrator can view the usage history for an individual user or a disk.

Nonetheless, this approach still focuses on setting individual user limits. In addition, there is no easy

way to visualize the distribution of the quota system in its entirety, or act upon the quota system in that context; all changes are still delegated at the user-level and those changes are made to user quota limits, regardless of the true state of the underlying disk. Finally, gaining the benefits of Precise's StorageCentral SRM requires a complete and costly switch over to their quota management suite. While another one of Precise's products, *QuotaAdvisor* [14], is much more light-weight, much of the benefits of the complete quota management suite are lost. Much was done during the creation of Qualm to avoid these adoption issues and make the integration of Qualm into the existing quota system relatively simple.

In Search of Inspiration

To overcome the limitations of the existing quota management solutions, a new model of quota management was needed. Recent work by Mark Burgess [3] provided an appealing start. Burgess suggests that a system quota is an inefficient strategy for managing a dynamic resource such as disk space. He then emphasizes the importance of global knowledge when selecting an optimal strategy and concludes, "A quota strategy can never approach the same level of productivity as one which is based on competitive counterforce."

While Burgess' work uses game theory [11] to explore this competition as one between the system administrator and individual users, it ignores an important part of this dynamic interaction: the competition between individual users of the system to meet their own needs above everyone else's. Accordingly, the spirit of Burgess' work was incorporated into the core philosophy of the new quota management model by emphasizing users' relative share of a resource, rather than individual limitations on resource quantities. This fosters an element of competition: since users are alloted a percentage of the resource, an increase in their allotment can only come at the expense of another user.

Burgess' work, unlike some other work in convergent system administration [1, 2, 6, 17], treats policy as a mutable thing that must be changed and tuned for optimal performance. This led to the idea of attempting to model the quota management problem as a problem in control theory, a branch of mathematics that is often used to model electro-mechanical systems in Electrical Engineering [4]. A possible feedback model for a quota system inspired by control theory is shown in Figure 1. In this model, the system administrator defines an operating curve (OC) that describes how the system will respond to disturbances from a user, $d(t)$, to the system. The control function, as defined by the OC curve, then affects the output of the system, which when combined with the continual input of user activity, causes the system to converge to the new desired state.

While the control-theoretic model offered an interesting new way to be able to graphically control how the quota system enforced quotas and responded to aberrant disk usage, it lacked the "global knowledge for decision making" that Burgess' work emphasized. In the end, both these ideas were merged in the formulation of the newly adopted model of quota management.

Finally, there was the challenge of creating an interface that best represented the global state of a quota system. My work in creating *Peep: The Network Auralizer* [9] demonstrated that when digesting a considerable amount of information, the value of the whole can be greater than the sum of its parts; it is more important to convey the general state and trends of the quota system, rather than the individual values that comprise the system.

In addition, Alva Couch's work on visualization of large execution environments when developing *seeplex* [5] and *xscal*. *xscal* provided inspiration for developing Qualm's scalable graphical environment. The algorithms used in *xscal* for visualizing very large numbers of data points proved very relevant in visualizing quotas; a graphical display of an entire quota system needs to be able to plot thousands of data points on a single display.

A New Model for Quota Management

The new model for the quota system combines elements from the competitive model and the control-theoretic model in a novel way. Rather than the traditional approach of viewing quota as assigning a

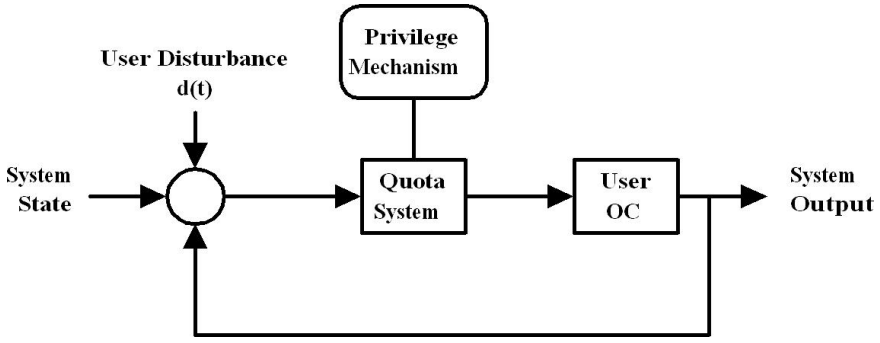

Figure 1: A control theory feedback model for a quota system.

specified amount of disk space to an individual or group of users, all available disk space is treated as a continuous, finite resource, where each user is alloted a fraction of that resource (a simplification which is valid because the smallest unit of measurement, the byte, is very small compared to total the resource size). The system administrator then defines a distribution curve that describes how the administrator thinks the disk resource should be shared, i.e., a certain user population should get more than another user population. This distribution has the constraint that the total area under the distribution curve must be equivalent to the size of the storage resource. Finally, a separate mechanism configured by the system administrator determines user privilege, i.e., where a user gets to sit on the distribution curve.

Figure 2 shows how the model works graphically; this distribution curve allots more space to a sub-section of the population and tapers off for the general population. The X-axis of the graph is then divided up into equal intervals amongst the *n*-user population, where each user is given a portion of the area under the distribution curve. The user's area then translates into a portion of the storage resource. In this model, the sum of disk space allocated to each user is equivalent to the total storage size, or more formally:

$$\int_0^n d(t)\, dt = \sum_{i=1}^{n} \Delta u_i = Total\ \ Storage$$

where n is the total number of users in the quota system, $d(t)$ is the disk space distribution curve, and Δu_i is the amount of disk space allocated to each user. Figure 3 shows a distribution curve that could be used to implement service levels in existing quota systems under the new model. Here, each step on the distribution curve represents a different quota limit, and its relative length indicates the percentage of the user population have that limit.

Just as in traditional quota systems, a user can use any amount of disk space up to the maximum alloted by their position on the distribution curve. Because of the constraint on the area of the distribution curve, increasing the fraction of disk space for a single user reduces the amount of disk space alloted to every other user. Only changes to a certain user's allotment that do not infringe upon the actual usage of other users (barring explicit action from the system administrator) are considered valid.

The new model offers an important advantage: a user's disk space is decoupled from the actual size of the resource. Thus, if the size of the resource changes, the user's alloted space changes while his relative share remains the same. Taxation of the resource is also reflected by this approach: increasing the number of users (in this case n) automatically decreases the amount allocated to each user. Still, the user's relative privilege remains the same throughout both changes.

This approach also helps separate policy from network implementation; the distribution curve describes what the system administrator believes proper space allocation should look like, given the needs of his users and the amount of resources available, while the privilege mechanism determines where specific users fit into the system administrator's master plan.

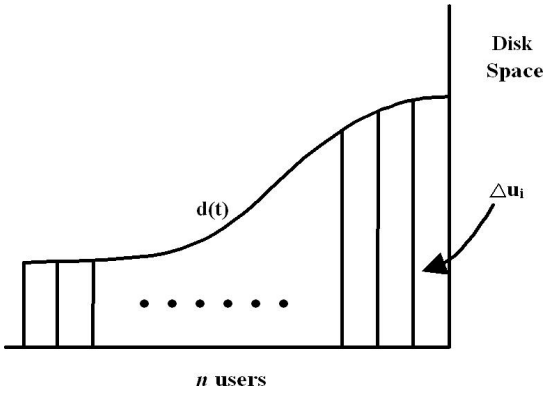

Figure 2: The new model illustrated.

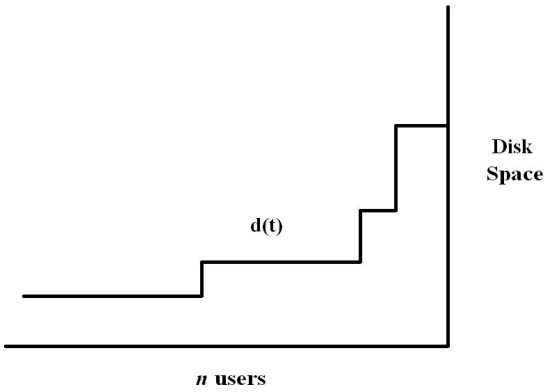

Figure 3: An example of a current quota implementation under the new model.

The model agrees closely with what system administrators currently try to implement as network policy, and incorporates Burgess' idea of competition as a way of maximizing resource efficiency. Here, the distribution curve can be thought of as the system administrator's optimal strategy: a user is given a share of the resource by the system administrator that reflects his given need (as determined by the level of privilege granted by the system administrator) while remaining in accordance with the administrator's general strategy. Giving a single user a larger share of the resource comes at the expense of all other users in the system, which is true of any finite resource, and captures the essence of competition. Consequently, changes are made to a global model, where each change affects the system as a whole, rather than a localized and segregated part.

This has interesting consequences: in the model, the system administrator is no longer trying to limit users but is instead trying to protect and maximize a finite resource. Only changes that maintain the

integrity of the storage and respect other users' use of it are valid.

Towards an Implementation

Some concessions and design constraints were needed to create a successful, adoptable implementation. Above all, the tool needed to be capable of inter-operating with current quota implementations. To meet this requirement, a decision was made to push aside the implementation of the privilege mechanism and focus on the analysis and global manipulation of the quota system. The tool could then use the existing quota system for its back-end operations, while providing the user with an interface that best agreed with the new model.

'Qualm' was created as a compromise between the new quota management model and the inter-operability requirement. The goal in creating Qualm was to create an interface that allowed the system administrator to quickly and easily visualize the state of his quota system, assess problem areas, and make global changes. A flexible interface was also important; if the new interface were to replace the existing quota management interface, the system administrator would need to be able to tailor the displays to reflect his particular needs.

In order to form a scalable view of the state of the quota system that faithfully conveys the global distribution of the data and its interrelationships, several features of *xscal*'s display model were employed. Building upon the graphing techniques used by *xscal* enabled Qualm to avoid many of the performance problems inherent in generating plots for large data sets. Sorting of the data was used to expose trends and groupings inherent in the data set. Qualm moves beyond *xscal*'s abilities in this regard by allowing for multiple data fields to be displayed on a single plot, and allowing for hierarchical sorting. Using hierarchical sorting, secondary fields can be sorted with regard to the result of sorting their parent fields, exposing categories within the data, and trends within those categories.

Qualm's displays use a sum of step functions to depict transitions between values. Because of the large number of data points used in creating the display, a "continuum effect" occurs and these step functions appear to form a smooth distribution curve. Upon closer inspection (using zooming), the steps become more apparent. The step function provides a nice emphasis on the values and transitions between data points without introducing any graphical artifacts. However, the step function is not always ideal when plotting multiple fields on a single graph. In such cases, Qualm provides alternate display mechanisms, such as error bars that extend away from the primary plot, or scattered data plots.

Furthermore, Qualm's capabilities are modular and extensible. Access to the underlying resource, such as quota data or a flat file, is implemented as a module, so Qualm can easily be extended to work on top of other resources. In addition, Qualm's graphing library provides the system administrator with the tools needed to create new graph types or extend existing graph types, so displays can better be tailored towards the administrator's need.

Performing Analysis with Qualm

Running Qualm on the Tufts University EECS network produced some remarkable results. Figure 5 shows a display of block usage for all EECS users. Values of numbers of blocks used appear on the vertical axis and range over all users of the EECS network on the horizontal axis. The axes are sorted in increasing usage values from left to right. The 1-Dimensional frequency plots at the bottom and left side of the graph indicate the frequency of transitions between values in the main plot, where each hash mark represents a transition. The display yields an interesting result: block usage appears to follow a Pareto [13] distribution. A plot of file usage followed the same distribution. Even more interesting was that this distribution remained constant over a period of six months! This strongly agrees with Michael Mitzenmacher's dynamic model of file sizes [10] and reaffirms that file sizes in large networks are indeed statistically predictable.

Next, one can plot quota hard limits against quota soft limits and block usage using a hierarchical sort (as shown in Figure 6). Once again, values appear on the vertical axis and range over all users on the

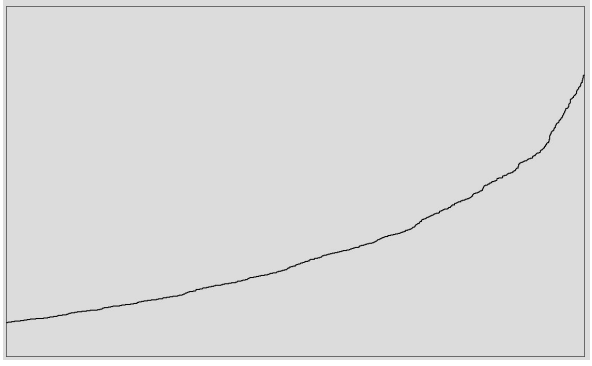

Figure 4a: A Plot at full view.

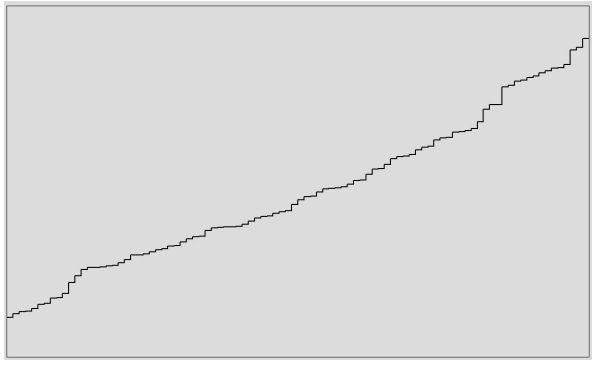

Figure 4b: A zoomed sub-section

horizontal axis. This display allows quick assessment of how quota is distributed across users of the system and how active those users are. The soft quota looked as expected, showing the same distribution as the hard quota settings, except significantly lower. The shape of the quota limits indicate that EECS users tend to fall within four different categories, or four different levels of service. Interestingly, even though Qualm had no knowledge of user groupings, these groupings were implied by the sort order of the graph! Moreover, the block usage yields some very interesting facts: most of our users were given quotas well beyond their needs. At the same time, certain users are over their soft quota limits by a substantial amount and may require additional space.

In order to better assess the effectiveness of the EECS soft quota limits, on-going usage data was accumulated over a period of several months and fed into Qualm. Figure 7 shows a plot of soft limits and block usage with error bars extending from the block usage trace (color coded in reality) indicating the maximum deviation of that usage value over the course of the period. The plot indicates that users who were given larger soft limits tended to deserve it; the space requirements of those users fluctuated more than any other category. Other users have exceeded their soft limit at some point and might be good candidates for an upgrade to a higher service level. Finally, the large number of over-subscribed users suggests that perhaps with a little tweaking, a much more efficient configuration could be achieved, minimizing the cost of future storage upgrades.

Resource Manipulation with Qualm

A holistic approach to quota manipulation was adopted when creating Qualm: when making adjustments, rather than concentrating on the details, the system administrator tries to make the global picture "look right." In the context of Qualm, this translates into making adjustments to the distribution curves on the soft limit and hard limit displays.

An adjustment can be anything from lowering or raising a small sub-section of the curve, to radically changing the entire shape of the curve. In order to adjust the limit of the entire population that presently has a particular limit, adjustments are made by left-clicking on the distribution curve at the level of the existing limit, and dragging the plot up or down. The left-click adjustment affects all users with that same initial limit. Alternatively, the administrator can right-click and

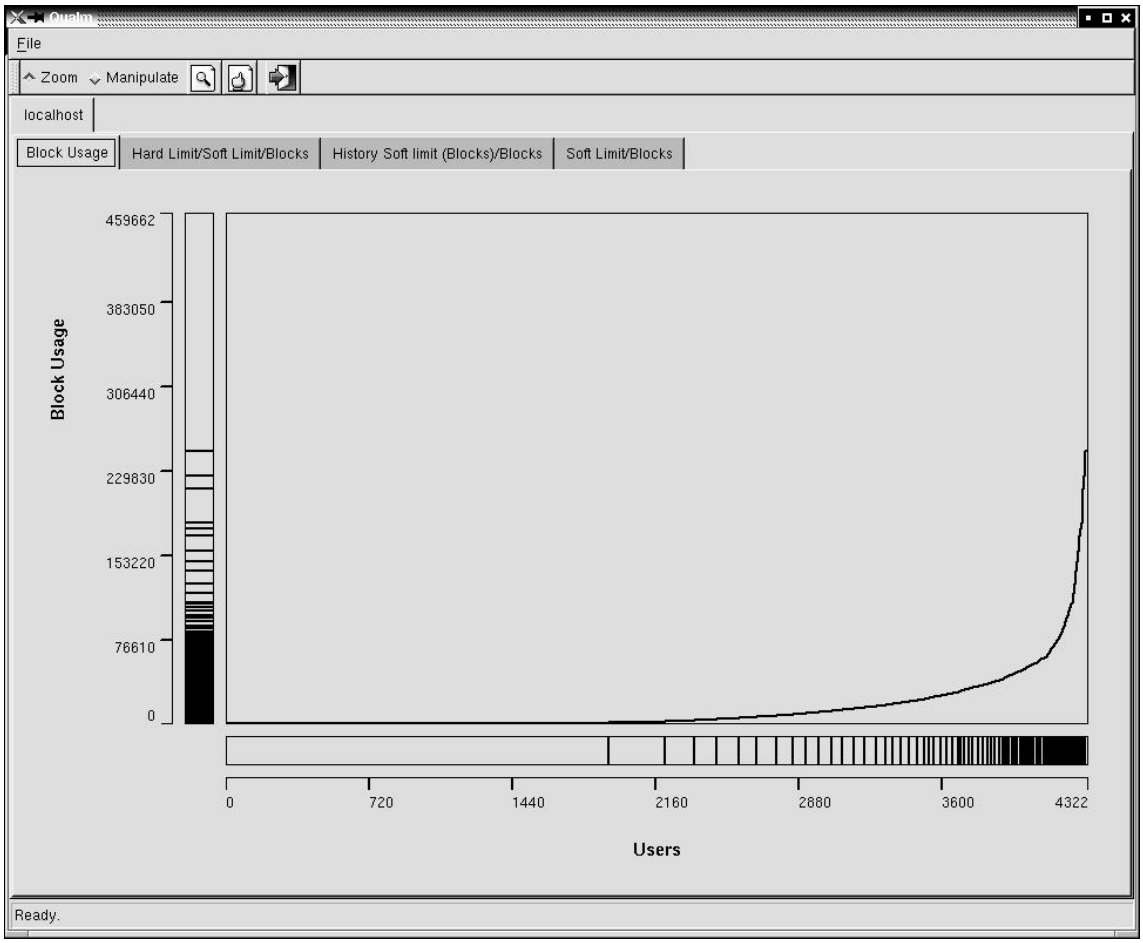

Figure 5: A plot of block usage for all EECS users.

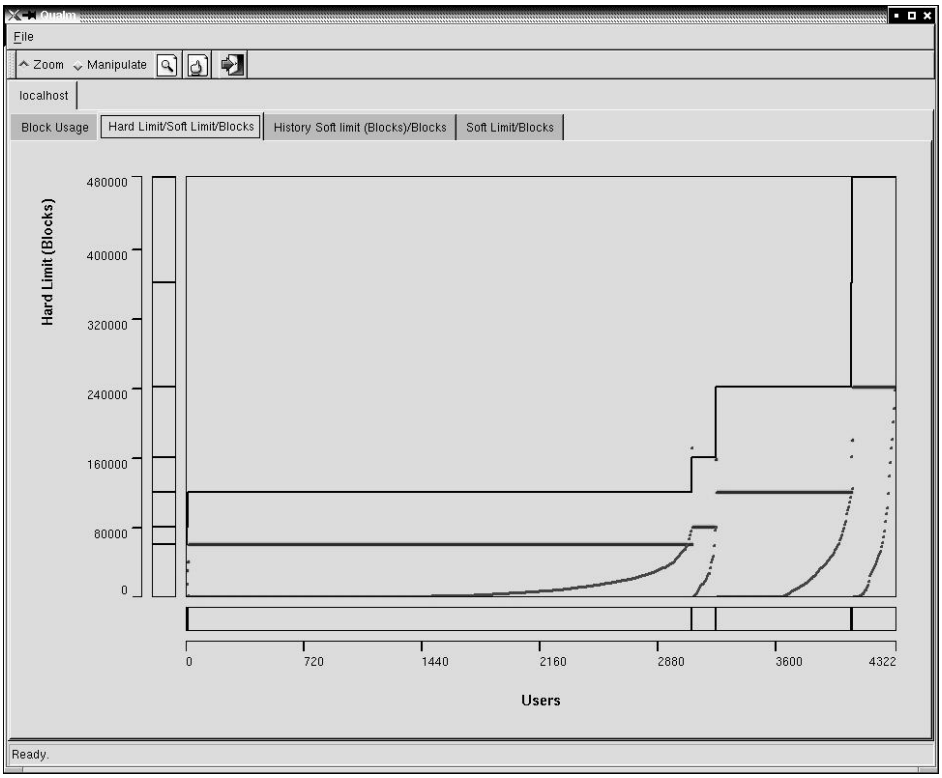

Figure 6: A sorted plot of hard limits (uppermost trace), soft limits (mid-level trace), and block usage (lower most) for all EECS users.

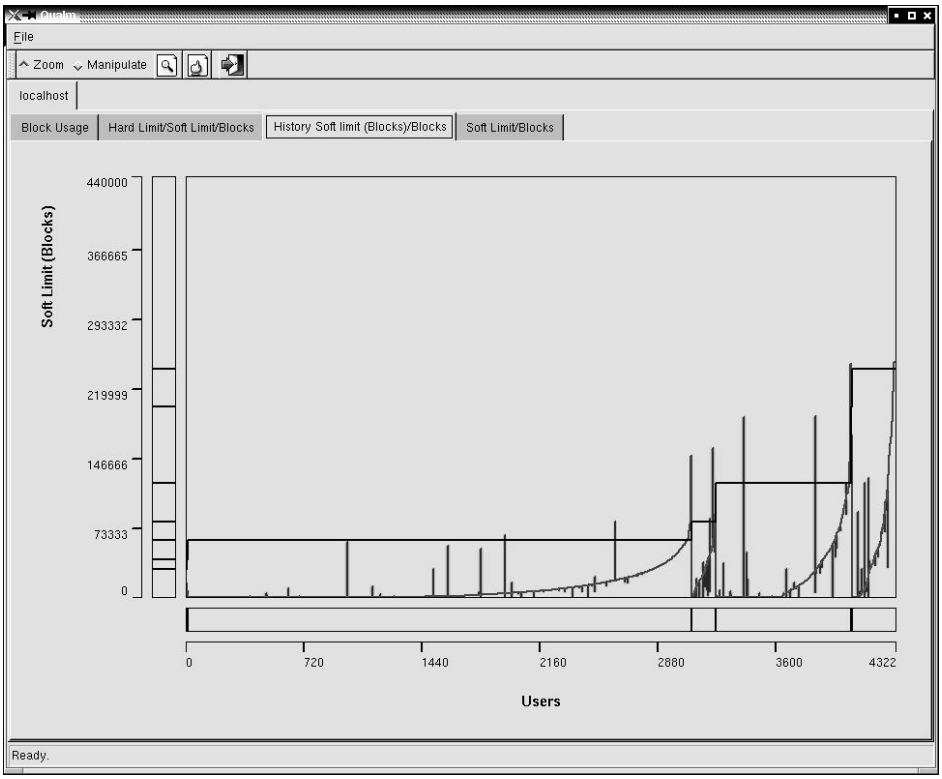

Figure 7: The soft limit (uppermost trace) and block usage (lowermost) trace from Figure 6, with block usage deviations extending from the block usage over time.

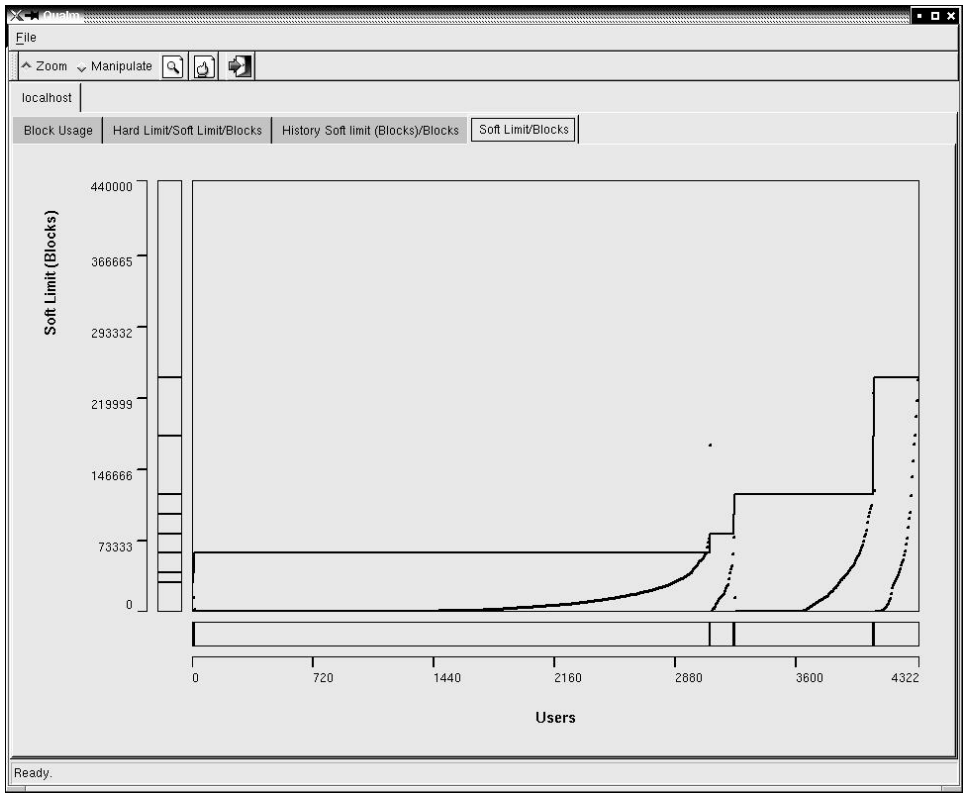

Figure 8a: Manipulation of EECS hard limits to give a higher ceiling to more active users in each usage category.

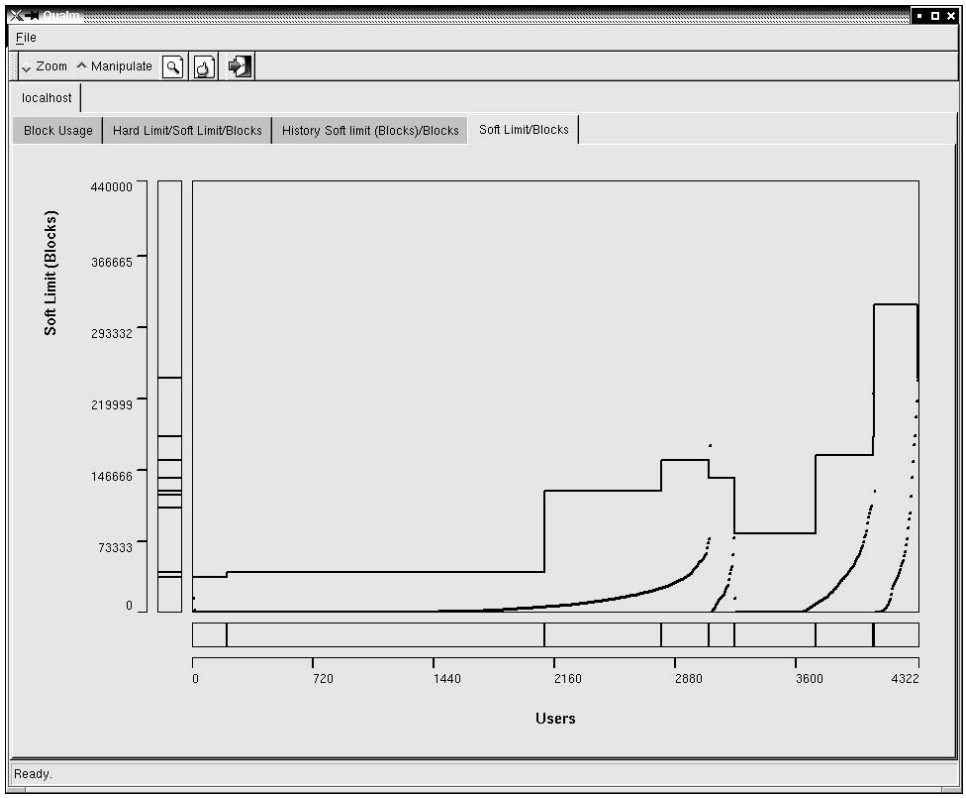

Figure 8b: Manipulation of EECS hard limits to give a higher ceiling to more active users in each usage category, II.

select a sub-segment of the population with a given limit and adjust only the limits of that sub-segment.

Operations in Qualm can only be performed on segments of the distribution curve, never on individual points. However, depending on the zoom-factor, those segments may represent any number of users from a large user population to a single user. A quick rule of thumb is that the smoother the curve, the more users affected by the operation. Here, zooming serves a double purpose: it allows the system administrator to get a closer look at the underlying distribution and to control the granularity of his changes. Additionally, the system administrator can still use the traditional quota mechanism if he deems zooming insufficient.

Finally, changes to the distribution curve are only transfered to the underlying resource when the system administrator explicitly chooses to commit. This allows the administrator to make as many changes as he likes and take some time to examine them fully before letting the changes go live. It also

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<configuration app='qualm'>
  <!-- Configuration file for Qualm -->
  <options>
    <!-- Program options will go here -->
  </options>
  <resources>
    <resource type='quota' name='localhost' host='localhost'>
      <source type='module' module='File'>
        <args>
          <path>history.dat</path>
        </args>
        <headers>
          <header>Login</header>
          <header>Time</header>
          <header>Files</header>
          <header>Hard Limit</header>
          <header>Soft Limit</header>
          <header>Blocks</header>
          <header>Hard Limit (Blocks)</header>
          <header>Soft Limit (Blocks)</header>
        </headers>
        <plot type='FrequencyPlot' name='Block Usage' ylabel='Block Usage'
                                                              xlabel='Users'>
          <field>Blocks</field>
        </plot>
        <plot type='DotFrequencyPlot' name='Hard Limit/Soft Limit/Blocks'
              ylabel='Hard Limit (Blocks)' xlabel='Users' sorted='1'>
          <field>Hard Limit (Blocks)</field>
          <field colour='BLUE'>Soft Limit (Blocks)</field>
          <field colour='DARK GREEN'>Blocks</field>
        </plot>
        <plot type='HistFrequencyPlot' name='History Soft limit (Blocks)/Blocks'
                ylabel='Soft Limit (Blocks)' xlabel='Users' sorted='1'>
          <field>Soft Limit (Blocks)</field>
          <field colour='DARK GREEN'>Blocks</field>
        </plot>
        <plot type='DotFrequencyPlot' name='Soft Limit/Blocks'
              ylabel='Soft Limit (Blocks)' xlabel='Users' sorted='1'>
          <field>Soft Limit (Blocks)</field>
          <field>Blocks</field>
        </plot>
      </source>
    </resource>
  </resources>
</configuration>
```

Figure 9: An example Qualm configuration file.

avoids a rather expensive operation until absolutely necessary; a single change could easily affect thousands of records.

A Generalized Configuration Format

Qualm uses an XML configuration file to determine how data should be retrieved from a resource, labeled, and displayed. Figure 9 shows an example configuration file that fetches quota system data from a flat file and creates the three displays used for analysis in a previous section.

The **source** tag indicates the data source; Qualm uses resource modules as data proxies. Currently, Qualm supports two types of resource modules: a resource module for interacting directly with the quota system and a resource module for reading data from a flat file. The **args** tag contains parameters which are passed to the module during intialization. The **header** tags indicate how to label the data fields, which may be fields delimited by spaces in a flat file or a list of fields in memory.

The **plot** tags tell Qualm which displays to use and how to generate displays when plotting resource data. The **type** attribute indicates what kind of plot to generate and corresponds to an existing plot object within Qualm's graphing library. The **ylabel** and **xlabel** attributes tell Qualm how to label the axes. The **sorted** attribute indicates whether Qualm should use hierarchical sorting when generating the plot. If the **sorted** attribute is set, Qualm uses the listed order of the fields as the hierarchical sort order. In the ''DotFrequencyPlot'' example, the hard limit block usage is sorted first, the soft limit block usage is then sorted with respect to the hard limit, and finally the block usage is sorted with respect to the soft limits.

Using this configuration system, a system administrator can add displays or tweak existing displays to meet his needs by simply adding or modifying existing plot tags. Qualm can also load and display data for multiple resources simultaneously by providing multiple **resource** tags. Adding a new **resource** tag adds another page in Qualm's tabular interface, making it easy to look at two different resources from two different locations at once.

Critique

Differentiating between users who may belong in separate groups, but who still have the same quota requirements, is impossible under the current implementation. In the EECS network, we often have students in multiple classes who are given the same disk quotas. Currently, these students fall under the same category, even though it might be convenient to keep them separated for non-quota reasons.

A common example is having two students in different classes, a student in a data structures class and another in a programming languages class, with the same quota allocations. Under the current implementation, these students become indistinguishable. Both students will lie on the same usage line in a block usage plot, and worse, the relative order of their positions on that line is determined by the data sorting algorithm and thus may fluctuate! The lack of differentiation makes modifying quotas for students in a particular class difficult.

The ideal solution to this problem lies in the implementation of the privilege mechanism described in the new quota management model, and thus a complete implementation of the model. Such a mechanism would need to be able to understand user classes and groupings. This would allow Qualm to better tailor its displays to reflect user grouping (perhaps via color coding or a similar method), keep those groupings together during the sorting process, and allow the administrator to manipulate groupings directly via the display. Such a mechanism would give the administrator the finer grain of control needed to solve this issue. Ultimately, the system administrator can still use the traditional quota system to make changes to specific users.

As with all systems that attempt to increase the efficiency of resource allocation, there is a question of how the system can be defeated. While the analysis with Qualm might suggest a more efficient quota configuration and greatly simplify making those changes, it does punish users who conserve disk space so that they might have that space immediately available when they truly need it. It encourages the mentality to ''hoard now, so that I may hoard later,'' a mentality that is well understood and often employed (sadly, successfully) in the corporate budget world. This problem, however, exists within current quota management systems, and its causes are mostly social factors. Still, because Qualm encourages a more dynamic management model, the influence of these social factors are more significant. The consolation is: while Qualm emboldens change, it also makes it easy to reverse change.

Towards the Future

As Qualm is still in the prototyping stage, the most pressing need in Qualm's current stage of development is user feedback and suggestions. The next phases of Qualm's development will involve working out the kinks and making Qualm a truly usable tool.

Constraining the manipulation of quota levels to keep the area under the distribution curve constant has not yet been implemented at the time of writing. Such a mechanism would play an important role in keeping quota modifications in check; the current interface makes it very easy to lose track of the real scope of a change and make unreasonable changes to quota limits. However, it is crucial for the mechanism to support the concept of over-subscription for inter-operability; existing quota systems greatly over-subscribe storage. Over-subscription may also be desirable for providing users with a bit of breathing room, while keeping a preferable space distribution.

Much future work involves the full implementation of the model; creating an implementation of the privilege mechanism so that users can easily be grouped into classes and different service levels, while keeping the distribution of the resource separate. The privilege mechanism should help solve the most pressing problem with the current Qualm implementation of being able to differentiate classes of users by providing the system administrator with a finer grain of control over how users fit into the quota distribution. Such a mechanism would also make it easier to manage the exceptions that do not fit well into the new quota management model by allowing for a more classic quota management functionality.

Finally, the mechanisms used in Qualm and their implementation are sufficiently flexible that Qualm can be used to analyze any type of network resource with similar properties to disk quota (such as bandwidth, for instance). Future research will explore other areas where the principles used in Qualm may be applied.

Some Lessons Learned

Despite being a fundamental component to modern service networks, disk quota management has changed little since its inception. The traditional focus on controlling user privilege has introduced problems of scalability, making analysis and global manipulation difficult. This paper advocates a new model of quota management that solves these scalability issues; by adopting a graphical approach that emphasizes resource share and distribution, the tasks of determining where change is needed, assessing the impact of that change, and assessing the efficiency of the current quota scheme, are greatly simplified.

Use of Qualm on the Tufts University EECS network has contributed significantly to an understanding of the quota system, and what adjustments are necessary to make it optimally efficient. For the most part, quotas don't affect the usage patterns of the majority of the students; most of the storage allocated to those students will probably never be used and that space might be better allocated to users in higher quota brackets, who tend to place a greater demand on the file system. By graphing patterns of usage over time, it was also easy to determine which users were restricted by their quotas and which users were over-subscribed.

Finally, usage of Qualm has yielded some valuable insight into the nature of our quota systems. It suggests that quota management by category is more effective than artificial user groupings. These categories emerge from similarities found within existing user populations, and provide a better model for controlling global user quotas. Moreover, the strong resemblance of file size distribution to a Pareto distribution suggests that exploiting this knowledge may be the key to building an automated and optimal dynamic quota system in the future.

Availability

Qualm is freely available, open source software. The project is currently under development and I am actively looking for people to get involved with the project, experiment with it, and provide feedback. Qualm is written in pure Python on top of *wxPython* [18] and makes use of the *pyquota* [16] module. To obtain qualm, please visit: http://qualm.sourceforge.net.

Acknowledgments

A special thanks goes to Alva Couch for his guidance during Qualm's initial development and pointers to *xscal*'s scalable graphing algorithms. Additional thanks goes to Collin Starkweather for his insightful comments and Etan Lightstone for providing some good food for thought.

Biography

Michael Gilfix was born in Winnipeg, Canada. At age 4, he moved to Montreal, Canada where he spent his younger years. Strangely enough, he currently resides in Austin, Texas. After attending high school at Lower Canada College in Montreal, he went on to receive his B.Sc in Electrical Engineering from Tufts University in Boston, in 2002. He currently designs and develops software for big blue in IBM's Austin software group. He can best be reached by email at mgilfix@eecs.tufts.edu, or feel free to visit his personal web site at http://www.eecs.tufts.edu/~mgilfix.

References

[1] Anderson, Paul, "Towards a High-Level Machine Configuration System," *Proceedings of the Eighth Systems Administration Conference (LISA)*, SAGE/ USENIX, 1994.

[2] Burgess, Mark, "A Site Configuration Engine," *Computing Systems*, 8, 1995.

[3] Burgess, Mark, "Theoretical System Administration," *Proceedings of the Fourteenth Systems Administration Conference (LISA)*, SAGE/ USENIX, 2000.

[4] *Control Theory and Engineering Links from Theorem.Net*, http://www.theorem.net/control.html .

[5] Couch, Alva, "Categories and Context in Scalable Execution Visualization," *Journal of Parallel and Distributed Computing: Special Issue on Visualization*, Vol 18, 1993.

[6] Couch, Alva, "SLINK: Simple Effective Filesystem Maintenance Abstractions for Community-Based Administration," *Proceedings of the Tenth Systems Administration Conference (LISA)*, SAGE/ USENIX, 1996.

[7] Couch, Alva, "Visualizing Huge Tracefiles with Xscal," *Proceedings of the Tenth Systems Administration Conference (LISA)*, SAGE/ USENIX, 1996.

[8] Elz, Robert, ‘‘Disc Quotas in a UNIX Environment,’’ http://pluto.iis.nsk.su/docs/bsd-4.3/quotas.html .

[9] Gilfix, Michael, ‘‘Peep (The Network Auralizer): Monitoring Your Network with Sound,’’ *Proceedings of the Fourteenth Systems Administration Conference (LISA)*, SAGE/ USENIX, 2000.

[10] Mitzenmacher, Michael. “Dynamic Models for File Sizes and Double Pareto Distributions,” http://www.eecs.harvard.edu/michaelm/NEWWORK/postscripts/filesize.pdf .

[11] Mobius, Markus. “Introduction to Game Theory,” http://icg.fas.harvard.edu/ec1052/lecture/index.html .

[12] *NT Disk Quota System*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fsys_2clf.asp .

[13] *The Pareto Distribution,* http://mathworld.wolfram.com/ParetoDistribution.html .

[14] *Precise Software Solutions and WQuinn, QuotaAdvisor*, http://www.wquinn.com/Products/QuotaAdvisor .

[15] *Precise Software Solutions and WQuinn, Storage Central SRM,* http://www.wquinn.com/Products/StorageCentral .

[16] *pyquota: A python wrapper for manipulating disk quota*, http://www.sourceforge.net/projects/pyquota .

[17] Traugott, Steve and Joel Huddleston, ‘‘Bootstrapping an Infrastructure,’’ *Proceedings of the Twelfth Systems Administration Conference (LISA)*, SAGE/USENIX, 1998.

[18] *wxPython: A Python GUI toolkiT*, http://www.wxpython.org .