DATA PLACEMENT AND MIGRATION STRATEGIES FOR VIRTUALISED DATA STORAGE SYSTEMS

H.A. Bond N.J. Dingle F. Franciosi P.G. Harrison W.J. Knottenbelt
Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ
Email: {hab06,njd200,ozzy,pgh,wjk}@doc.ic.ac.uk

KEYWORDS
Storage virtualisation, Data placement and migration

ABSTRACT

This paper details a simulation intended to explore the viability of storage systems where data is placed intelligently depending on data-specific quality of service requirements and the characteristics of the underlying storage devices. Our implementation provides a framework for the automatic profiling of a virtualised storage system, specification of data-level quality of service requirements, simulation of data streams acting on the storage system and visualisation of the results. We demonstrate that an intelligent data placement algorithm can achieve a good match between desired and offered storage system quality of service.

INTRODUCTION

The volume of data produced world-wide continues to grow. In the second quarter of 2008, shipped disk storage capacity was 1 777 petabytes, representing growth of 43.7% year on year[1]. Indeed, the International Data Corporation (IDC) forecasts that shipped disk storage capacity will increase at a compound annual rate of 38% for the next three years (Rydning and Reinsel, 2009).
Recently there has been a trend towards centralised storage systems, with the aim of reducing management overhead and lowering total cost of ownership. These systems are usually implemented in the form of a Virtualised Storage System (VSS), i.e. a contiguous logical volume where the underlying storage devices (e.g. solid state disks, RAID arrays, etc.) are abstracted away. Different devices provide very different storage characteristics, however, and their abstraction makes the job of efficiently placing data far more complex.
Performance is key to many users, but it is not the only characteristic to take into account when using VSSs. Reliability is another very important aspect, as is space efficiency. Throughout this paper we refer to these data considerations as Quality of Service (QoS). For example, a bank's transaction records have to be stored in a fashion that guarantees high reliability. A temporary file, by contrast, probably needs only high performance. Data placement is a very important aspect of this problem, but the QoS attributes and the patterns in which data is accessed are not static and might change over time for different data streams. This implies that not only initial allocation must be considered but also migration, to ensure delivered QoS remains as close to that required by the user as possible on a sustained basis.
A number of other projects have investigated similar ideas in terms of intelligent data placement. BORG (Block-reORGanization and Self-optimization in Storage Systems) (Bhadkamkar et al., 2009) is a module for the Linux kernel which sits between the file system layer (e.g. ext3, JFS, NTFS etc) and the I/O scheduler. It constantly evaluates process access patterns based on temporal, process-level, and block-level attributes and constructs access pattern graphs.
In Franciosi and Knottenbelt (2009), the authors present a modification of the well-known ext3 file system to include QoS attributes, which exploits redundant bits in the individual inodes to mark QoS attributes for sets of data. The system will use these attributes along with a performance profile of the underlying storage devices to intelligently place data.
This paper describes a simulation of a system which allocates and migrates data depending on a set of user-specified QoS requirements and the characteristics of the underlying VSS. We begin with a brief overview of modern storage systems, before detailing the design and implementation of our simulator. We describe how our simulation automatically profiles a VSS's logical address space to create a multi-attribute QoS model of the device, and then outline how the user specifies their QoS requirements and the characteristics of the data streams that will be used to drive the simulation. We further present the allocation and migration techniques that we have implemented, and show how the simulator visualises the resulting data layout and performance for the user. Finally, we present a number of case studies of the use of the simulator before concluding.

BACKGROUND

Today the most common storage medium is the hard disk drive. This device consists of a mechanical read/write head which moves over a spinning disk made of fer-

[1]Source: IDC press release, 5 September 2008 (http://www.idc.com/getdoc.jsp?containerId=prUS21411908)

romagnetic material. Multiple hard disks can be used to improve performance and reliability, typically using one of the Redundant Array of Inexpensive Disks (RAID) levels. In this work we focus on the most commonly used configurations: RAID 0 (striping without redundancy), RAID 1 (mirroring), RAID 5 (distributed parity) and RAID 10 (stripes of mirrors). We note that our work could easily be extended to model additional schemes such as RAID 6 (double distributed parity).

RAID Level	Performance		Data Reliability	Space Efficiency
	Read	Write		
0	H	H	L	H
1	H	L	H	L
5	M	L/M	M	M
10	M/H	L/M	H	L

Table 1: Summary of RAID characteristics.

Each RAID level offers different trade-offs between performance, reliability and space efficiency, as summarised in Table 1. For example, RAID 0 offers very high performance as disk accesses can be performed in parallel and its space efficiency is also very good since it stores no redundant data. Its reliability is, however, poor as if one drive fails then data will be lost. On the other hand, RAID 1 offers very good reliability as it maintains a copy of all data so that if one drive is lost, all data on it can be recovered from the mirror. It also offers very good read performance as either disk can be accessed for a particular piece of data. Write performance is degraded, however, as a request must be serviced by both disks, and it is not particularly space-efficient as it requires twice the number of drives to store the same volume of data as RAID 0.

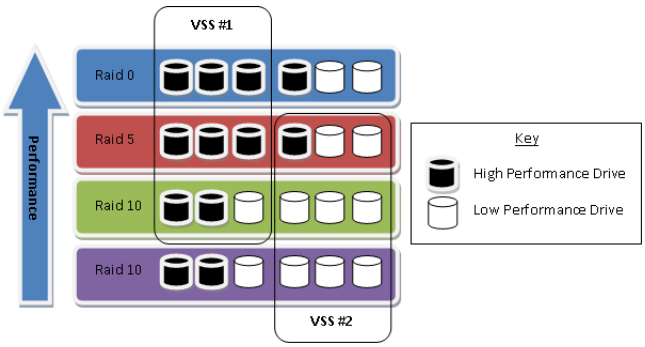

Figure 1: Two VSSs spread over four different storage tiers ordered by performance

It is not as easy to characterise VSSs in these terms, however. Figure 1 shows two VSSs spread over an infrastructure consisting of four different tiers, and that consequently provide different QoS attributes. VSS #1 will have a very good performance profile due to the majority of its underlying storage devices being of high performance and in fast RAID configurations. VSS #2, by contrast, does not provide such good performance but does provide good data reliability, as every single tier keeps redundancy data.

Data Stream Characteristics

Data Stream A	Data Stream B
System critical	Non-critical
> 50% read requests	100% write requests
100% sequential	100% non-sequential
Requests typically small	Requests typically large

Table 2: Example QoS requirements for 2 data streams.

The central concept of this work is that if we know the characteristics of the requests made to a VSS on a per-application basis (which we term a *data stream*) and also the user's QoS requirements for that stream, we can exploit intelligent data placement strategies to deliver better QoS. Consider the situation where Stream A and Stream B both want to access data on the same VSS. The QoS requirements for the two streams are given in Table 2. The most suitable place on the logical partition for data from Stream A is very likely to be different from the most suitable place for the data from Stream B.
We consider three QoS requirements:

- Performance
- Reliability
- Space efficiency.

It is important to note that the QoS attributes for data are not static. We therefore consider not just the initial placement of data according to its requirements and the attributes of the underlying storage system, but also its migration within the storage system in response to changing access patterns and QoS requirements. Ideally this migration should be done automatically to minimise the management overhead of the VSS.

PROFILING

In order to simulate a VSS faithfully, it is necessary to quantify the performance, reliability and space efficiency of every one of its (logical) block groups.
We characterise performance by throughput, which we measure in KB/s. We have implemented an automatic profiler which runs on a real VSS and issues block group-level requests and records the time that they take to complete. As input, we consider sequential and random accesses of two different request sizes (large and small) for both reads and writes, for a total of 8 possible combinations. The resulting times are then processed to yield the average throughput for each block group under each of these I/O patterns.

It is then necessary to classify the performance of areas of the underlying storage as either Low, Medium or High. These classifications should be relative to the underlying storage and not absolute since the profiler should not be dependent on the system it runs. We first find the throughput which lies on the 33^{rd} percentile and the 66^{th} percentile. Any throughput which lies below the 33^{rd} percentile is classified as Low, between the 33^{rd} percentile and the 66^{th} percentile is classified as Medium and anything greater than the 66^{th} percentile is classified as High.

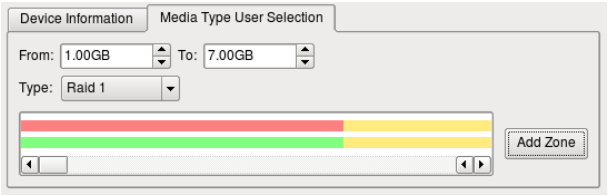

Figure 2: The GUI for selecting device types within logical address ranges.

When profiling reliability and space efficiency, the abstraction of devices within a VSS is a problem since the profiler cannot automatically gather the required information. Instead we must rely on the user's knowledge of the makeup of the VSS, and we thus provide a tool to enable the user to manually choose the types of devices comprising the VSS and to specify the logical address range to which they correspond. Figure 2 shows an example of a VSS where the range 1GB-7GB describes a RAID 1 device and 7GB onwards describes a RAID 5 device. The coloured strips represent space efficiency (upper) and reliability (lower), where red corresponds to Low, yellow to Medium and green to High. From this and the RAID characteristics in Table 1, we automatically create a per block group classification of reliability and space efficiency in terms of High, Medium and Low.

SIMULATION

In real systems a 'sea' of requests will be made to the storage from a variety of sources. For example, the system may be a file server containing users' home directories, while at the same time it may also house a database server storing experimental data. In order to simulate such situations we introduce the concept of data streams, and for each of these in a simulation the user must identify the following characteristics:

- Arrival Rate: the average number of I/O operations per unit time (assumed to arrive according to a Poisson process).
- Performance Required: High, Medium or Low
- Reliability Required: High, Medium or Low
- Space Efficiency Required: High, Medium or Low
- Data Size: The expected proportion of Small:Large data accesses
- Order of Access: The expected proportion of Sequential:Random data accesses
- I/O Type: The expected proportion of Read:Write data accesses

Events and Event Generation

In order to achieve realistic simulation results there are three main points to consider:

- Which stream triggers the event
- What type of event (read or write) is triggered
- Amount of time until the next event

The likelihood of a particular stream being chosen depends on its arrival rate, and is calculated as $\frac{rate(stream)}{\sum_i rate(i)}$. After the stream has been selected, the type of I/O operation issued by that stream needs to be generated. These can be either reads or writes; we further distinguish between three types of write event: data creation, data deletion and data modification.
To drive the utilised capacity of the file system towards a target capacity, t, given space usage c, we require:

$$\begin{aligned} c > t &\longrightarrow P(delete) > P(create) \\ c = t &\longrightarrow P(delete) = P(create) \\ c < t &\longrightarrow P(delete) < P(create) \end{aligned}$$

We therefore use the following probabilities to calculate if a write operation is a modify, a create or a delete:

$$\begin{aligned} P(modify) &= k_{modify} \\ P(create) &= 1 - P(delete) \\ P(delete) &= \begin{cases} \frac{x k_{delete}}{2t} & c \leq t \\ \frac{(x-t)k_{delete}}{2(1-t)} + \frac{k_{delete}}{2} & otherwise \end{cases} \end{aligned}$$

The constants k_{modify}, k_{create} and k_{delete} are set to $\frac{2}{3}$, $\frac{1}{6}$ and $\frac{1}{6}$ respectively in the implementation.
It is assumed that events occur with exponentially-distributed inter-arrival times, with rate $\lambda = \sum_i rate(i)$.

Intelligent Data Placement and Migration

The result of the profiling step is that we have a characterisation of every block group on the VSS in terms of performance, reliability and space efficiency. We also have matching QoS requirements for each data stream under the same three headings. The key challenge now is to place the data from the input streams on the VSS in such a way that these requirements are met as far as is possible. To achieve this we define and use a *Request-Receive* matrix.

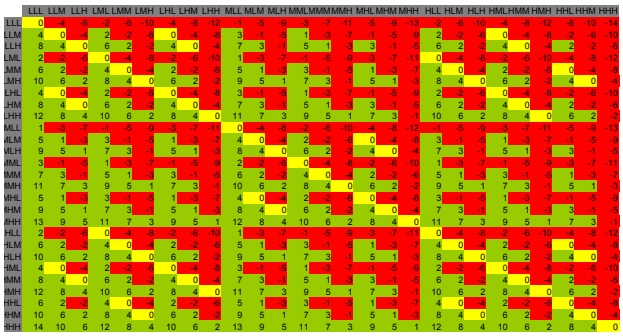

Figure 3: *Request-Receive* matrix displaying scores for requested and delivered QoS.

Figure 3 is an example of a *Request-Receive* matrix. The columns represent the possible QoS requirements of a data stream (high, medium or low in each of the three categories of performance, reliability and space efficiency), and the rows represent the possible attributes of the block groups of the VSS. The values in the matrix encode the match between the QoS required by the data stream and offered by locations on the VSS, where zero implies perfect match, positive numbers imply delivered QoS is better than requested QoS, and negative scores that delivered QoS is worse than requested QoS.
To calculate this matrix the user is asked to specify how much they value performance over reliability over space efficiency by choosing three integer constants k_P, k_R and k_S respectively. The rating for a particular combination of QoS requirements against those offered by the VSS is then calculated as:

$$k_P \times (P_{recv} - P_{req}) + k_R \times (R_{recv} - R_{req}) + k_S \times (S_{rece} - S_{req})$$

where P_x, R_x and S_x are 3 for high, 2 for medium and 1 for low. The constants used in Figure 3 were $k_P = 1$, $k_R = 2$ and $k_S = 4$.

Figure 4: Matrix showing the ordering of suitable storage locations against data stream QoS requirements.

We can then use this to form an ordering for matching the service levels of blocks on a VSS to a data stream's QoS requirements. For example, if a particular stream has the QoS requirements Low, Medium and High for performance, reliability and space efficiency (abbreviated to LMH in Figures 3 and 4), which VSS service levels would satisfy these and in what order? Figure 4 suggests an ordering of [HHH, MHH, HMH, LHH, MMH, HLH, HHM,....] in descending order of suitability.
This ordering is used by our simulator to intelligently place data. When a request is made by a stream to write data, the simulator traverses the address space of the VSS in the order specified by the column of the *Request-Receive* matrix which corresponds to the QoS requirements of the stream making that request until it finds sufficient free space to accommodate the data.
We also consider the migration of data from its initially allocated location. Such migration can occur in response to two situations: newly available capacity and changing stream characteristics.

Newly Available Capacity If the system gets to a high space utilisation, new data will most likely be placed in less satisfactory locations. However, should more satisfactory locations become available due to the deletion of data, data currently residing in less suitable locations can be migrated there.

Changing Stream Characteristics Migration is also required because streams are not static entities and their characteristics can change with time. Data may have been placed in one location according to the stream's initial access patterns or QoS requirements, but if these change it is necessary to move this data to areas which more closely match the new requirements.

VISUALISATION

We have created a visualisation engine in Java for the simulation to show how the well the current data placement meets the user's QoS requirements. The visualisation consists of 3 main views: Stream View, Storage View and Graph View.

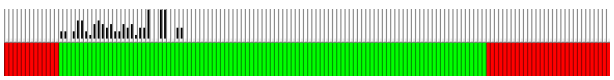

Figure 5: Stream View

Figure 5 shows how Stream View displays the placement and QoS requirement satisfaction of an individual data stream. The upper strip indicates where data currently resides, with a fully white bar meaning that a location is empty and a fully black bar that it is full. The lower strip represents how each area of the storage rates relative to the stream's requirements (Green – Better than required, Yellow – As required, Red – Underperforms).

Figure 6: Storage View

An additional panel known as the Storage View is available which visualises the storage as a whole (stream independent). Storage View is similar to an aggregated Stream View with the exception that is does not have the lower strip representing the storage system's offered QoS. This view is demonstrated in Figure 6.

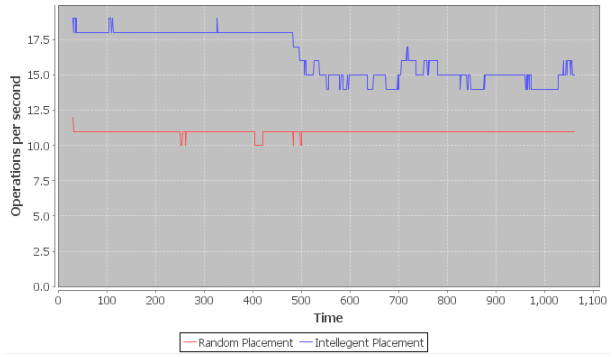

Figure 7: Graph View

Graph View is used to display the performance benefit of the intelligent placement technique, as shown in Figure 7. It displays in real-time a graph of the I/O operations per second achieved under the intelligent placement scheme, and also provides for comparison a graph of the performance of a random placement technique.

EVALUATION

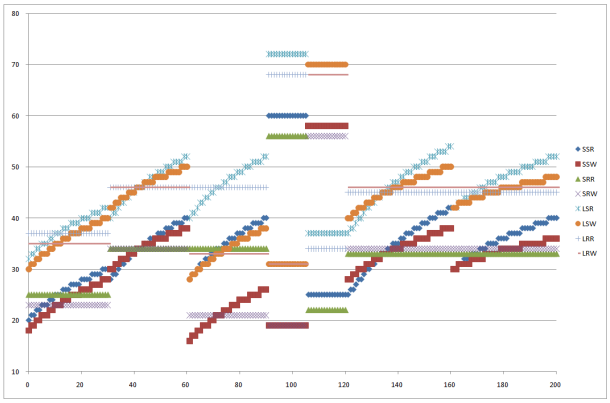

Figure 8: Graphical representation of the 8 profiles used for the case studies. The graph plots logical block address against throughput in bytes/second.

We demonstrate the applicability of our simulation through a number of case studies. Since the simulation requires an underlying profile of a VSS, we use a fabricated profile in order show some interesting characteristics of our techniques. This is shown in Figure 8, which plots logical address (x-axis) against throughput (y-axis), and is intended to simulate a VSS consisting of the following components:

1. Standard Magnetic Hard Disk
2. RAID 0 System
3. RAID 1 System
4. Read-Biased Flash Memory
5. Write-Biased Flash Memory
6. RAID 5 System
7. RAID 10 System

We will describe the characteristics of the streams used to drive the case studies using a 7-tuple of the form $\{[1,H,M,L,50\%,50\%,50\%]\}$. This example has an Arrival Rate of 1 I/O operation per second, High Performance requirements, Medium Reliability requirements and Low Space Efficiency requirements, with equal parts Small Sequential Reads to Large Random Writes. We use the QoS constants $k_P = 2$, $k_R = 3$ and $k_S = 1$ for all case studies. Videos of the case studies are at `http://www.doc.ic.ac.uk/~njd200/studies.html`.

Case Study: Order of placement

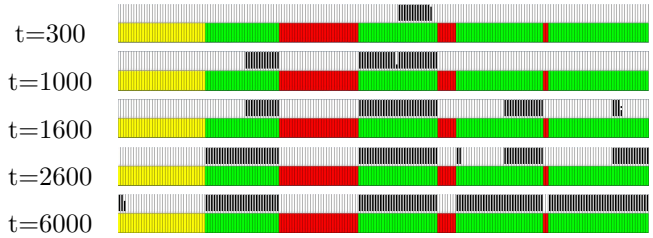

Figure 9: Sequential animation time-line showing correct placement ordering.

In order to verify the order in which the simulation places data, we executed the simulator with a single input data stream with the characteristics $\{[1,M,M,M,50\%,50\%,50\%]\}$. As shown in Figure 9, the green (better than required) portions of the storage are the first to be filled. Next (at $t = 6\,000$) the yellow (as required) portions of the storage start to be filled since there are no more green areas. In this example, since the simulation was only asked to fill 80% of the space, the red portions of the storage will never be used since no more space is needed.

Case Study: Performance Comparison

The next experiment is to measure the performance of the placement method in terms of rate of I/O operations. In order to provide a basis for comparison we will also investigate the case where a random placement algorithm is employed. We explore two scenarios, both with a single input stream with characteristics {*[1,M,M,M,50%,50%,50%]*}:

- Scenario A: 80% free space aim
- Scenario B: 20% free space aim

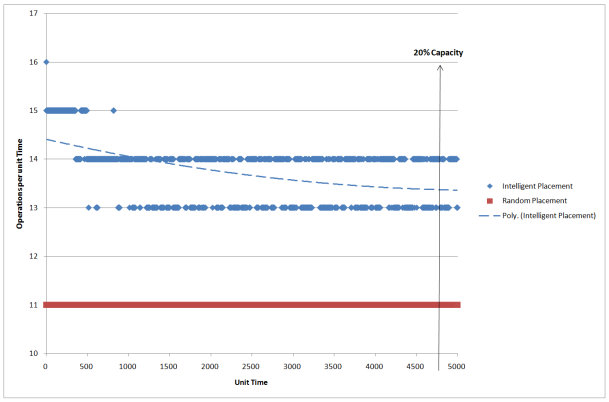

Figure 10: Performance of the VSS in Scenario A.

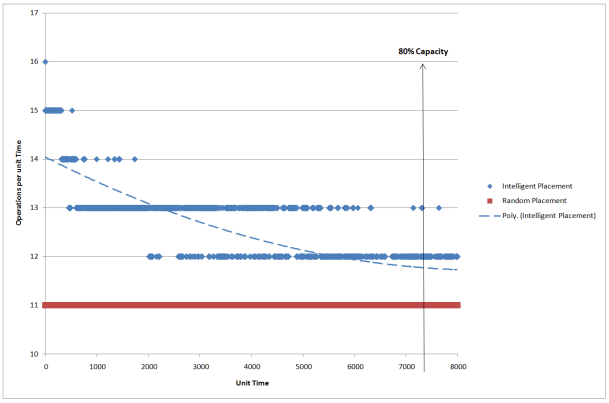

Figure 11: Performance of the VSS in Scenario B.

The graph for Scenario A (Figure 10) demonstrates that by using the intelligent placement technique over the random placement technique there is a performance increase of nearly 2.5 disk operation per unit time. Scenario B (Figure 11) only yields a performance increase of 0.75 disk operations per unit time. This is because when the system is kept at 20% of storage space in use, the probability of finding high performance areas of storage is high since they would not yet have been filled. This means that the majority of the data will be placed in these areas yielding very good performance for the system as a whole. When the percentage of storage space in use is set at 80%, the chance of finding a high performance area is far lower.

Case Study: Multiple Streams

Since in reality many data streams will be competing with each other to allocate space, we now explore the case of multiple concurrent streams. Figure 12 shows the animation of a run with 3 streams: {*[1,H,M,L,17%,18%,84%], [3,L,M,H,78%,81%,13%], [1,M,M,M,50%,50%,50%]*}.

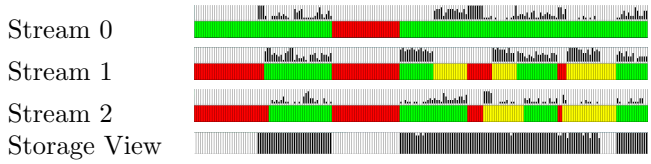

Figure 12: Multiple streams acting on one VSS.

Figure 12 demonstrates the multiple stream simulation. An interesting thing to notice about this run of the simulation is that, because each stream was given very different characteristics, the way each values its underlying storage is very different. There are certain areas of the storage that are preferred by all streams; this can lead to conflict between the streams.

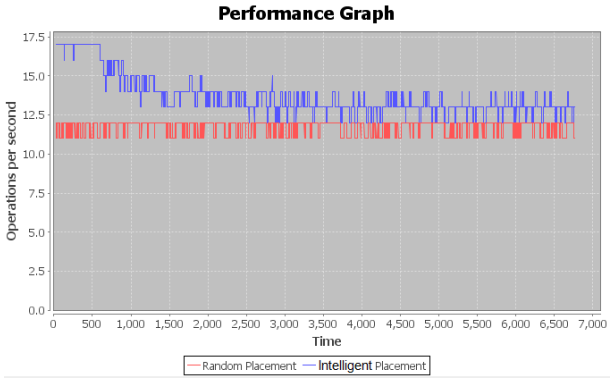

Figure 13: Performance profile when multiple streams are acting on the same storage.

In Figure 13 we observe that our intelligent placement scheme succeeds in allocating storage space such that performance is improved over results from a random allocation scheme.

Case Study: Changing Stream Characteristics

We demonstrate the extent to which our simulation is capable of responding to changing access pattern characteristics by migrating data in a case study with a single stream with parameters {*[1,L,M,H,78%,81%,13%]*}.

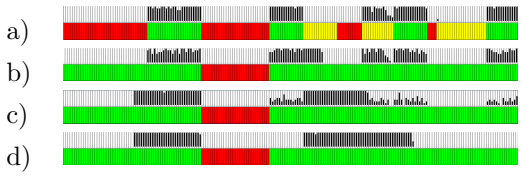

Figure 14: Changing QoS stream characteristics.

Figure 14 shows an example of a simulation where there is a change in the stream's characteristics mid run. The four snapshots correspond to the follow situations:

a): Shortly before stream characteristics change.

b): The point at which the stream's characteristics change. Note that portions of the storage that were unsuitable are now classified as good.

c): The simulation in mid-migration.

d): The final resting place of the data post-migration.

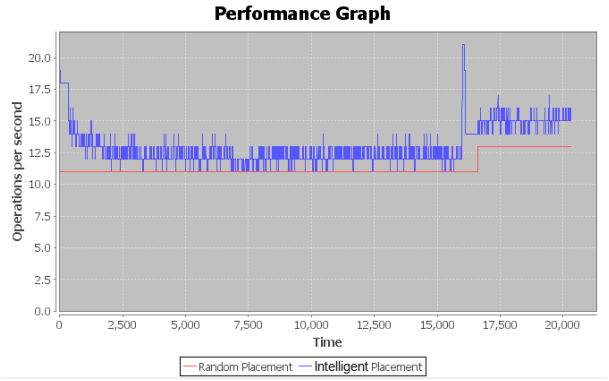

Figure 15: Performance where a stream's characteristics have been changed in mid-simulation.

Figure 15 shows a performance graph of this run of the simulation. The most noticeable aspect of the graph is the spike at $t = 16\,000$, caused by the migration process writing the poorly placed data to the best available space, which is very high performance. Also note the system's performance is higher post migration.

Case Study: Changing Space Consumption

It is to be expected that with time a system's space utilisation will grow and shrink. In this context we investigate the extent to which migration can be applied successfully. Figure 16 shows an example of the simulation with a single stream with settings {*[1,M,M,M,50%,50%,50%]*}.

a): The file system reaches its highest space utilisation.

b): The file system at its lowest space utilisation.

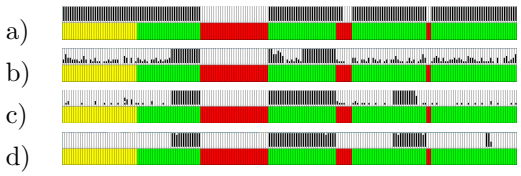

Figure 16: Example of a simulation when the file system grows and then shrinks.

c): Migration is 50% complete.

d): The post-migration layout of the data.

Once again, performance is higher post migration.

CONCLUSION

This paper has described methods of profiling and characterising a storage device under different conditions to achieve an in-depth picture of how the device will perform. We have then discussed the placement and migration of data on such a storage device to best meet users' performance, reliability and space efficiency requirements. This has been implemented in a simulation where data streams can be specified to operate on the storage, and we have demonstrated this in operation through a number of case studies.

There are many opportunities for further work. One attractive feature of VSSs is the ability to add new devices dynamically. This requires a mechanism for detecting when such an event has occurred and automatically reprofiling the VSS. We have also not modelled the failure of hard drives and the operation of RAID systems in degraded mode, but these would obviously have a major effect on performance and would complicate placement and migration. Finally, real-life processes often anticipate the amount of storage they will require and provisionally pre-allocate space to ensure that their data is not fragmented, but this is not currently supported.

REFERENCES

M. Bhadkamkar, J. Guerra, L. Useche, S. Burnett, J. Liptak, R. Rangaswami, and V. Hristidis. BORG: Block-reORGanization for Self-Optimizing Storage Systems. In *Proc. 7th USENIX Conference on File and Storage Technologies (FAST'09)*, pages 183–196, San Fransisco, CA, February 2009.

F. Franciosi and W.J. Knottenbelt. Towards a QoS-aware virtualised storage system. In *Proc. 25th UK Performance Engineering Workshop (UKPEW'09)*, pages 49–60, Leeds, July 2009.

J. Rydning and D. Reinsel. Worldwide hard disk drive 2009–2012 forecast: Navigating the transitions for enterprise applications. IDC Market Analysis, Document 216394, February 2009.