Sithole *et al. Journal of Cloud Computing: Advances, Systems and Applications* 2013, **2**:1
http://www.journalofcloudcomputing.com/content/2/1/1

Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH **Open Access**

# Cache performance models for quality of service compliance in storage clouds

Ernest Sithole[1*], Aaron McConnell[1], Sally McClean[2], Gerard Parr[2], Bryan Scotney[2], Adrian Moore[2] and Dave Bustard[2]

**Abstract**

With the growing popularity of cloud-based data centres as the enterprise IT platform of choice, there is a need for effective management strategies capable of maintaining performance within SLA and QoS parameters when responding to dynamic conditions such as increasing demand. Since current management approaches in the cloud infrastructure, particularly for data-intensive applications, lack the ability to systematically quantify performance trends, static approaches are largely employed in the allocations of resources when dealing with volatile demand in the infrastructure. We present analytical models for characterising cache performance trends at storage cache nodes. Practical validations of cache performance for derived theoretical trends show close approximations between modelled characterisations and measurement results for user request patterns involving private datasets and publicly available datasets. The models are extended to encompass hybrid scenarios based on concurrent requests of both private and public content. Our models have potential for guiding (a) efficient resource allocations during initial deployments of the storage cloud infrastructure and (b) timely interventions during operation in order to achieve scalable and resilient service delivery.

**Keywords:** Storage cloud, Enterprise applications, Cache performance, Optimisation

## Introduction

The cloud computing paradigm is emerging as a mainstream approach in the design and implementation of enterprise computing solutions [1-3]. The principal factors favouring the adoption of cloud-based technologies in business computing environments are: (a) the ease with which IT infrastructure deployments and expansions can be achieved when bringing together multiple and heterogeneous computing resources, typically scattered across wide geophysical locations, and (b) the simplified mechanisms by which users can access and utilise hosted IT services.

Based on the specific needs of target user environments, which cloud based technologies are intended to serve, business IT service solutions can be crafted and made available in a variety of offerings, which can be Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) or Infrastructure-as-a-Service (IaaS) computing capabilities. The adoptions of the SaaS solutions [1,2] present hosted applications to user environments customers as usable service entities for business computing needs; the PaaS-based solutions [4-7] present for use by application routines executing at the SaaS level, service capabilities that are derived from the integration of Operating System and virtualisation functionalities; the IaaS solutions [3,8-10] bring together the operational hardware elements such as data centre equipment, processor and storage servers, and networking devices into functional capabilities that can be plugged into and utilised by user routines executing at both the SaaS and PaaS levels of the cloud stack.

Depending on the nature of the affinity groups that are served by cloud-based IT environments, there are four main categories of clouds - Private, Community, Public or Hybrid, which can serve user environments according to their access entitlements. Private clouds are exclusively for intra-organisational needs; Community clouds result from federations of resources that serve the interests of select user groups with common objectives; Public clouds offer on-demand services to anyone with

*Correspondence: esithole@infc.ulst.ac.uk
[1] Networking and Computing Technologies Laboratory, University of Ulster at Coleraine, Coleraine - BT52 1SA Northern Ireland, United Kingdom.
Full list of author information is available at the end of the article

service-provider-authenticated web access over the standard Internet connection; Hybrid clouds fulfill requirements that can only met through services derived from combinations of in-house and off-premises resources. The delivery of cloud computing service according to above mentioned implementations enables reduced setup times for the deployments of outsourced business computing solutions, with little or no requirement being imposed on the customers to understand and manage the underlying technologies operating in the infrastructure.

Despite the enormous strides that have been achieved in developing functional capabilities for cloud computing systems, challenges still remain that present formidable barriers to the reliable performance and therefore effective use of cloud-based IT infrastructures. Performance related issues in the cloud domain encompass a range of considerations such as how to maintain Key Performance Indicators (made up of throughput and response time metrics), on-demand resource and service availability, continuity and scalability of IT services at competitive SLA and QoS levels that will enable business customers to meet performance goals. In order for the management-driven strategies and interventions to meet required levels of service reliability and availability and, thus maintain infrastructure operations within specified SLA and QoS targets, in-depth knowledge is required for establishing quantitative trends that are associated with principal performance indicators such as throughput rates, response times and load-scalability as the levels of user requests being sent to the cloud infrastructure vary. Thus, in quantitative terms, the ability to provide SLA and QoS-capable resource management in cloud-based IT environments requires accurate characterisations of the load response patterns, based on the interplay of factors such as the resource infrastructure's service capacity, the levels of applied user workloads and their resource consumption needs in the cloud service environment.

Given the lack of intimate knowledge that can lead to mature capabilities for establishing the quantitative relationships between user requests and performance in precise detail, the allocations of resources in cloud-based IT infrastructures are largely conducted in a reactive manner, with the assignment of resource entities being carried out statically in response to any changes in user demand. As a result, the fulfilment of SLAs and QoS contracts for outsourced IT services essentially relies on either excess provisioning that leads to inefficiencies or, limited allocation of infrastructure resources that has carries the risk of SLA violations. Hence, instead of having in place proactive technical interventions that can immediately respond by reassigning resources to keep performance levels within acceptable thresholds, any QoS breaches that occur in the infrastructure are handled largely by follow up admin negotiations with a view to settling any business losses that may result from service interruptions through compensation. It is therefore worth exploring approaches by which performance trends in the cloud infrastructure can be determined with sufficient accuracy to enable proactive resource management for QoS compliant delivery of IT services. In our work, we isolate for study the resource utilisation trends of the storage subsystems on server hardware, an aspect that has not been given sufficient consideration in the context of supporting SLA-capable resource management mechanisms in storage clouds.

### Proposed strategy for QoS compliance support in storage clouds

As has been highlighted, the challenge of achieving SLA-awareness in cloud environments is a multifaceted research issue with a number of dimensions stemming from it such as determining the levels of resource availability, service continuity rates and scalability trends of performance that will be able to satisfy QoS constraints. As a starting point towards addressing the vital issues pertinent to QoS maintenance in storage cloud infrastructures, this paper focuses on developing characterisations of cache performance trends, an aspect which we consider to have potential for serving as an important source of guidance for informed decisions on the provisioning of scalable data storage services in enterprise IT environments. Our approach, thus aims to support QoS readiness in resource allocation management strategies for storage clouds through accurate modelling of content availability levels at individual cache entities in the infrastructure, and the modelled scalability trends can serve as a feed into the management strategies for storage space provisioning.

In order to establish the validity of the modelled cache performance, a data centre facility with cloud-based storage elements is used. As a key contribution of this paper, we present and validate scalability trends of cache performance at individual nodes, and the derived theoretical models can be a foundation upon which the considerations for infrastructure sizing can proceed and decisions are made in accord with the applications' resource needs and the service capabilities of the resources in enterprise computing environments on the following: (a) initial sizes of the storage deployments for cloud-based services, (b) re-calibration of the scale of storage resource integrations an ongoing basis in order to preempt SLA violations that could arise from short-term increases in demand and, (c) storage capacity upgrades based on anticipated margins of permanent increases in user demand.

Given that the accuracy of cache performance characterisations is the critical component underpinning the ability of our proposed strategy to quantify the scale of resource allocations required to fulfil performance goals in storage clouds, the next section proceeds with a detailed consideration of cache performance trends, with

the focus primarily on the analytical derivations of the scalability response patterns of content retention rates at individual storage server nodes in the cloud domain. In Section "Experimental facility for validations of cache performance Trends" we describe the key components of the practical facility used for conducting experimental validations of the models. Sections "Validations of cache performance trends for user requests of private data" and "Validations of cache performance trends for user requests of public data" respectively feature the sets of experiments conducted to establish the validity of the modelled cache performance for private and public data requests. Section "Performance characterisations for concurrent accesses to private and public content" extends the analytical models to scenarios that are based on concurrent requests going to private and public data. A brief evaluation of our cache performance characterisations is provided in Section "Discussion" through the consideration of the implications of the results on the ability to provide support for service continuity, scalability and SLA compliance in the management of storage resources. Section "Related research" provides a summary review of other research initiatives that are aimed at developing SLA support strategies for clouds by addressing aspects that are adjacent to our area of focus. The ninth section concludes the paper by highlighting further issues to be investigated in future work so that viable techniques are developed for QoS-ready deployments in the storage cloud infrastructure.

## Derivations of cache performance models

We begin our consideration of cache performance trends in storage clouds by developing theoretical models of data availability levels based on the scalability response to rising numbers user of requests for content. In developing quantitative estimates of cache hit ratios at storage caches, four principal factors and their impact on cache performance are considered. These factors are (a) the respective sizes of the storage capacities of the local cache and source storage devices, $C_L$, and $C_S$ (b) the user loading levels in terms of the average number and average sizes of input files to satisfy each received request, $N_F$ and $S_F$, (c) the mean service time period for the execution of the requests, $\frac{1}{\lambda_{App}}$, during which a cached file is used by a runtime process at the CPU and (d) the affinities of user patterns to the individual files that they request in the cache. Table 1 provides a complete list of the basic input and output parameters that are used in the derivations of the cache models and practical experiments.

**Table 1 Input and output parameters for the local cache node**

| Parameter | Description | Value |
|---|---|---|
| $N_{Users}$ | Number Active Users | 10 - 250 |
| $(\frac{1}{\lambda_{App}})$ | Application Request Inter-repetition Time (sec) | exp (2.5) |
| $J_{Limit}$ | Transaction Limit for Received Jobs | Infinite |
| $J_{Policy}$ | Job Instance Limit Policy | Queue |
| $J_{Priority}$ | Priority assigned for | Regular 5 |
| $(\frac{1}{\mu_{CPU}})$ | Average CPU Service Time (sec) | exp (1.5) |
| $N_F$ | Average I/O File Read Count | Constant (1) |
| $S_F$ | Average Read File Size (MB) | Constant (1) |
| $S_{CPUMem}$ | Average Size of Memory (MB) | Uniform (0- 10) |
| $C_L$ | Capacity of Local Cache (GB) | 40 - 200 |
| $C_S$ | Capacity of Remote Storage (GB) | 1000 |
| $L$ | Local Cache Capacity (users) | 10 - 50 |
| $S$ | Remote Storage Capacity (users) | 400 |
| $P_L$ | Overall Local Cache Hit Ratio | 0 - 1 |

### Cache performance analysis of user requests for private data

Based on the interplay of these factors, the overall cache performance in terms of the average cache hit ratio, $P_L$, at each local storage can thus be summarized by the basic expression, $P_L = f(C_L, C_S, S_F, N_F, \frac{1}{\lambda_{App}})$. We make the assumption that the interplay of these input parameters impacts the local cache performance by predominantly generating capacity misses.

The analysis of cache performance that we consider first applies to application routines that have rigid affinities between user requests and target files i.e. cases where $User_1$ will always request $File_1$ with $User_2$ requesting $File_2$ etc. The criterion of rigid affinity to content is pertinent to situations where each customer using the business computing infrastructure accesses his own master data [11], which he can view and edit. Thus, ignoring the impact of conflict and compulsory misses on cache performance trends and assuming uniform file sizes, $S_F$, for cached content, then the cache hit ratio or the probability, $P_L$, of satisfying data requests in the local cache, follows the relationship:

$$P_L = \begin{cases} 1 & \text{if } S_F N_F N_{Users} \leq C_L \\ \dfrac{C_L}{S_F N_F N_{Users}} & \text{if } S_F N_F N_{Users} > C_L. \end{cases} \quad (1)$$

Assuming that the theoretical analysis applies to those cases where the data request cycles have gone beyond the point of start up misses, the model derivation shows that the hit cache ratio remains at 100% before and upon matching the storage capacity, $C_L$. Whenever the applied user load given by storage space requirements of the generated, $S_F N_F N_{Users}$, exceeds the storage capacity of the local cache, $C_L$, the cache hits begin falling asymptotically

toward zero. Conversely, the local cache capacity miss ratio, $M_L$, is described by the relationship:

$$M_L = \begin{cases} 0 & \text{if } S_F N_F N_{UUsers} \leq C_L \\ 1 - (\frac{C_L}{S_F N_F N_{UUsers}}) & \text{if } S_F N_F N_{UUsers} > C_L. \end{cases} \quad (2)$$

Next, we consider cache performance in scenarios where users generate dispersed requests i.e. unrestricted access is allowed to all the publicly available files that are kept on the remote storage device of capacity, $C_S$.

#### Cache performance analysis of user requests for public data

Unrestricted data access patterns apply to publicly hosted content, which many users will likely have an interest in obtaining, whether from the public internet or in-house data sources [11,12]. We make two assumptions for dispersed file requests: one that the master storage is at least equal or greater to the local cache space i.e. $C_S \geq C_L$, and the other that the time period for considering the cache performance is sufficiently long for the users to cycle their requests over the entire collection of files kept on the master storage device, i.e. $S_F N_F N_{UUsers} \geq C_S$. Since the access to all the content on the remote storage is unrestricted, each of the $N_{UUsers}$ can thus request any of the files at master storage device with equal chance so that the probability of requesting one of the stored $\frac{C_S}{S_F}$ files becomes $\frac{S_F}{C_S}$. If the start up cache misses are disregarded by taking into account the $\frac{C_L}{S_F}$ files that are already in the local cache node, then the cache hit ratio, $P_L$, which is equivalent to the probability that a requested file can be found in the cache node, is equal to $\frac{C_L}{C_S}$. Thus, regardless of the actual number of user requests coming onto the IT infrastructure, the cache performance is given by the relationship:

$$P_L = \begin{cases} \left(\frac{C_L}{C_S}\right) & \text{if } C_L < C_S \\ 1 & \text{if otherwise.} \end{cases} \quad (3)$$

Apart from the fact that load levels of input user requests are irrelevant to cache performance, it also follows that for scenarios where user requests are uniformly scattered over the remotely stored files, the trends for local node cache miss ratios in such cases are given by the expression:

$$M_L = \begin{cases} 1 - \left(\frac{C_L}{C_S}\right) & \text{if } C_L < C_S \\ 0 & \text{if otherwise.} \end{cases} \quad (4)$$

The following section provides an introduction to the setup for the experiments that were conducted to establish the practical validity of the modelled scalability trends of the cache hit rates derived for the two cases of data access patterns considered above.

## Experimental facility for validations of cache performance Trends

As shown in Figure 1, the practical setup for our experiments employs four Virtual Machines: the User Load Generator, Application Server, File Manager and Remote Storage Manager VMs, all of which comprise the software elements for the experiments.

The Load Generator program initiates the operations in the Application Server, File Manager and Remote Storage VMs in readiness for the start of measurements and results collection. As a preparatory step, an initial start signal is sent by the generator to both the File Manager and Remote Storage VMs so that the cache optimisation algorithm and storage partitions are provisioned with target files to be requested by users are set up. After a delay of appropriate duration, a second start signal is generated to initiate first user request, $Request_1$, which will be followed by a train of arrivals, $Request_2 \ldots Request_N$, at the Application Server according to the predefined arrival process rate, $\lambda_{App}$. The File Manager VM responds to the initial start signal by setting aside the required storage space in the local cache and activating the algorithm selected for optimising cached content.

In addition to retaining cached content according to the selected optimisation criterion, the cache algorithm functionality in the File Manager VM responds directly to the data requests by compiling a record of the Request, Hit and Miss events occurring in the local cache. The Remote Storage VM ensures that permanent copies of all the file objects to be requested by the users are kept on its storage partition, ready to be copied across to the local cache partition should any misses occur at the latter. Upon the lapse of the predefined duration of the experimental measurements, a stop signal is emitted by Load Generator to the Application Server, File Manager and Remote Storage VMs.

The number of users, $N_{UUsers}$, that generate data requests at runtime is varied from 0 to 250, and the user sessions execute concurrently in the form of thread instances spawned off from the invoked work process instance. Thus, for each active user session, the runtime execution is in the form a VM instance at the application server. The consumptions of processor hardware resources (CPU and memory), $\frac{1}{\mu_{CPU}}$ and $S_{CPUMem}$, are according to the requests generated by the application routine generated inside the VM instance. The average number of input data files, $N_F$, requested by each user routine, is fixed to 1, with each file being 1000 KB in size and, the reserved cache space on the storage node, $C_L$, is varied from 40 GB to 200 GB for the two cases for data request patterns. The accessed files are indexed as database objects in the MySQL backend database entries, and the file retrievals into cache space are handled as block data transfers of

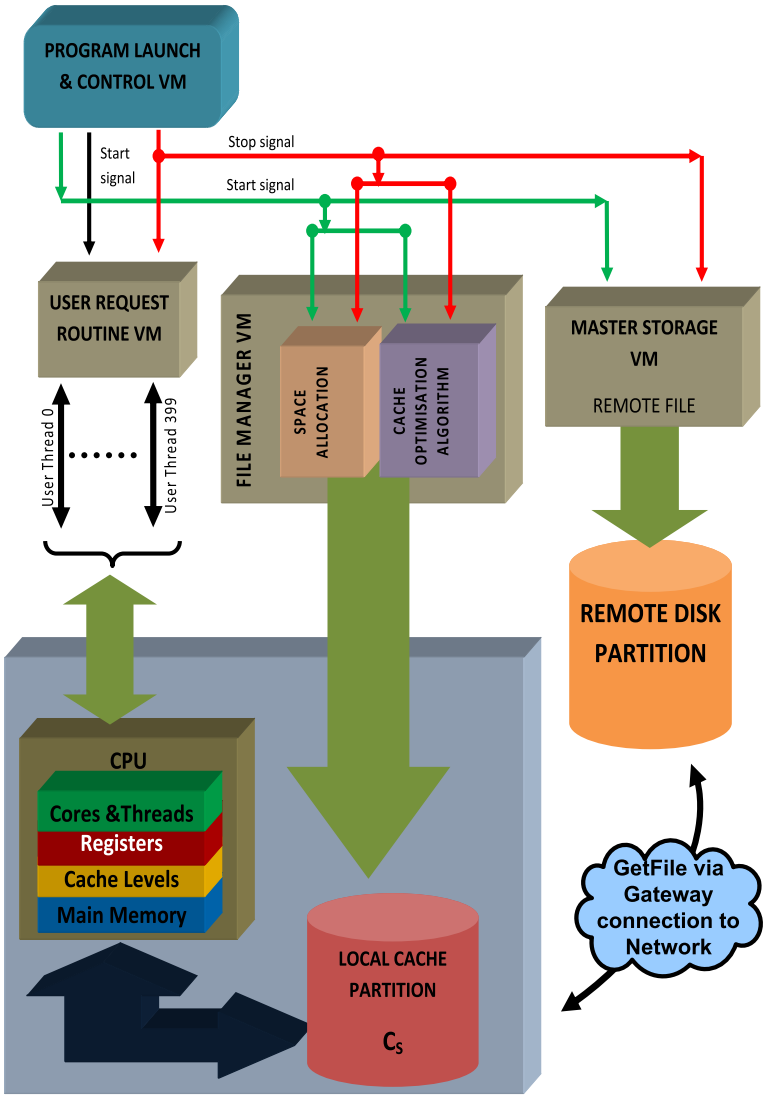


**Figure 1** Functional setup for the experiments.

the database table entries. The cached content is hosted on a local Dell R515 node, whose hardware is made up of an AMD 12-core 4170 HE, 2.1 GHz processor, 128 GB Memory and 25 TB storage. The functional configuration for Dell server is based on the Tier 2 settings so that both the application's CPU executions and the data access services are colocated in the same machine. The inter-arrival time, $\frac{1}{\lambda_{App}}$, for the data requests occurring inside each cycle of user operations is assumed to follow the exponential distribution with a mean of 2.5 seconds. The mean service durations of computation operations when interacting with cached files is set to 1.5 seconds also following the exponential distribution, and the duration of the experiments is 10 minutes. We base the parameter values and the distribution patterns for service times on the workload scenarios described in [11]. The run-in period before the experiment begins recording results data lasts for 2 minutes from the instant at which the experimental run is launched. For each data point that is presented by the graphs in the experimental scenarios that were featured in our studies, the result value was obtained from computing a running average of ten output readings as shown in the screenshot in Figure 2. Before the ten experimental runs for each result are conducted, the input parameters are fed into both the File Manager (which enforces the cache policies) and the User Emulator (which generates requests for files). Reference can be made to Table 1 for a complete list of the input parameters that were used in setting up the practical experiments.

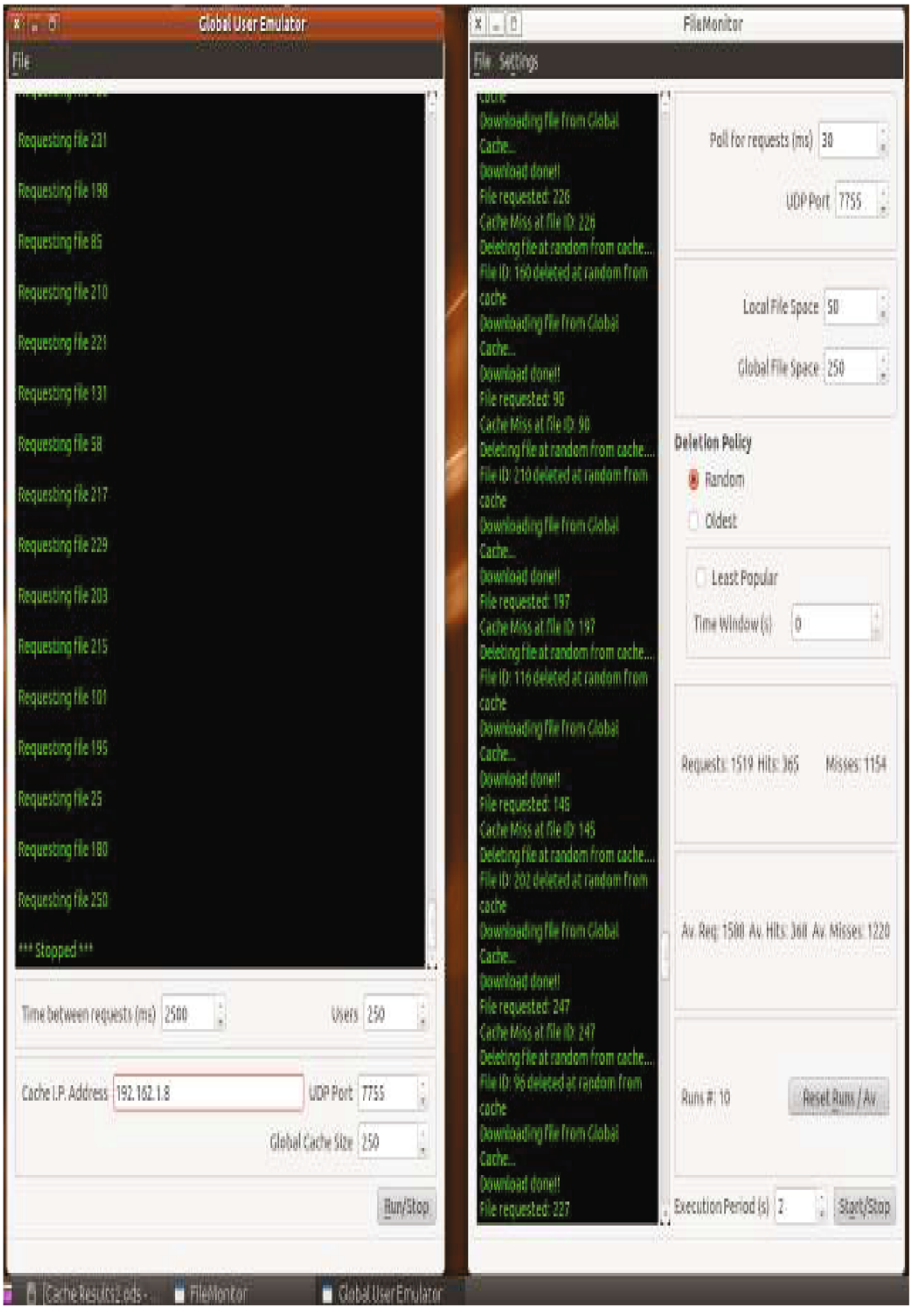


**Figure 2** Screenshot of the input and output parameters of the experiments for cache performance analysis.

### Definitions of scenarios for the experimental study

A simplification is made to Equations 1 - 4, developed in 2.1 and 2.2 so that the experimental analyses of the impact of user runtime behavior on the local cache hits and miss ratios can be carried out by expressing the local cache and remote storage capacities in terms of the maximum users that fill up the cache and master storage respectively. Thus, the formula, $L = \frac{C_L}{S_F N_F}$, represents the maximum number of users that can use cache storage before capacity misses occur, and the equation, $S = \frac{C_S}{S_F N_F}$, relates to the maximum users whose data are kept on the remote storage space. Figure 3 shows the experimental setup of the scenarios based on the use of simplified input parameters for user load levels, local cache and remote storage capacities.

Therefore, in situations where user requests for the experiments are defined according to each user having a unique file set for exclusive access, $L$ becomes relevant

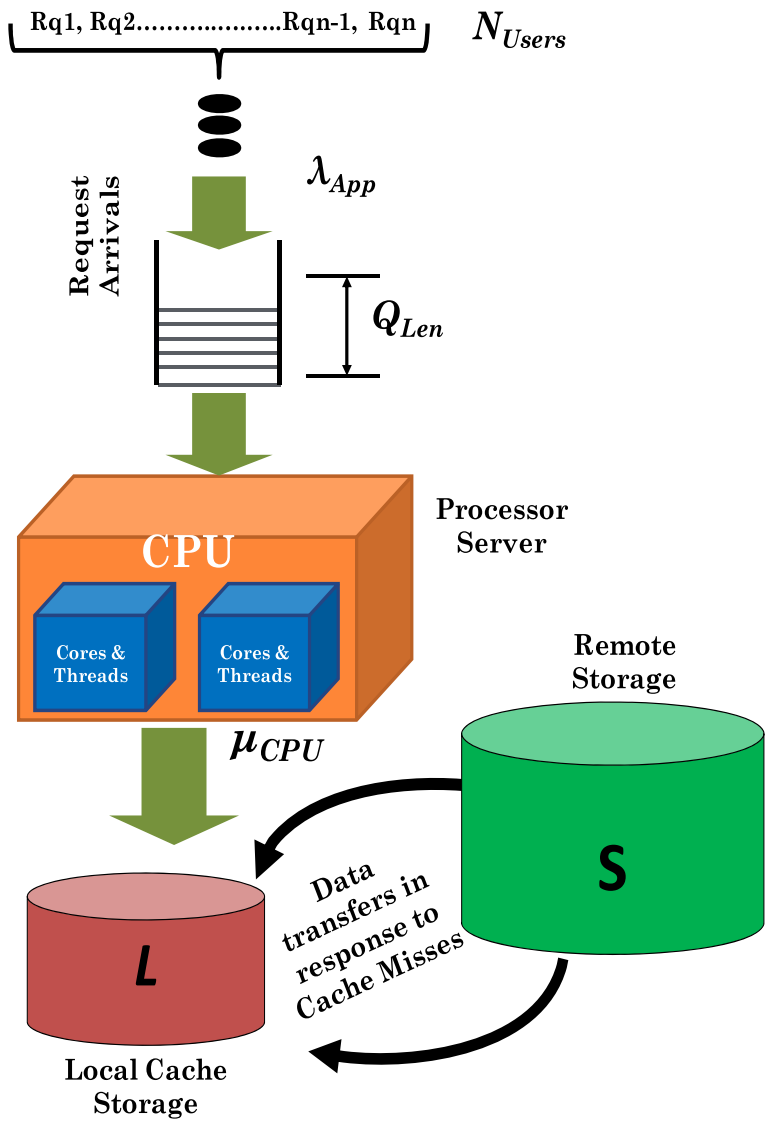


**Figure 3** Experimental setup the for cache performance studies.

to the analysis of cache performance according to the expression,

$$P_L = \begin{cases} 1 & \text{if } N_{UUsers} \leq L \\ \dfrac{L}{N_{UUsers}} & \text{if } N_{UUsers} > L. \end{cases} \tag{5}$$

The cache performance analysis associated with requests for private data is thus considered in terms of the number of active users, $N_{UUsers}$, generating data requests and the number of users, $L$, that fill up local cache capacity. Similarly, when public data objects are requested randomly from the list of shared data objects that are kept in the remote storage, the respective sizes of the local cache and remote storage affect cache performance according to the equation,

$$P_L = \begin{cases} \left(\dfrac{L}{S}\right) & \text{if } L < S \\ 1 & \text{if otherwise.} \end{cases} \tag{6}$$

It is important to reiterate that the specific definitions for the scenarios considered in the experimental studies take into account the fact that the levels of content availability at storage nodes in the cloud are governed by the interplay of the principal factors considered in the derivations of cache hit ratios, which are the data request patterns in terms of whether they specify private or shared data, the amounts of storage space in both the cache and

remote nodes, the applied user load according to volumes of data requests and file sizes associated with each request, and the efficacy of the storage management and cache replacement policies in keeping content that closely matches the needs of anticipated user requests. Hence, we define scenarios for experimental investigations to bring out the impact the respective factors to cache performance by setting the key experimental parameters as follows:

- Scenario 1 considers Cache Performance with User Requests accessing Private Data
- Scenario 2 considers Cache Performance with User Requests accessing Public Data
- Within each of the two primary scenarios, three separate studies of cache performance are conducted based on cache optimisation policies for Random, Popularity and Age-based File Evictions.
- The applied user load, based on the number of users and the average file sizes per request, is uniformly increased to levels that are beyond the assigned cache storage, $C_L$, and the resulting scalability patterns for cache misses and hits are recorded for comparisons with the theoretical ones.

## Validations of cache performance trends for user requests of private data

The arrival and service processes for the user requests received at the local cache nodes server are assumed to be Poisson, and in order to ensure a stable queue on the storage node, the relationship, $\frac{1}{\lambda_{App}} > \frac{1}{\mu_{CPU}}$, must hold for the respective magnitudes of the mean arrival and service times. Reference can be made to the list of input parameters shown in Table 1 for the actual values of arrival and service intervals. To investigate the sensitivity of the local cache ratios to user load, the local cache and remote storage capacities are assigned fixed values of $L = 50$, and $S = 400$ respectively. The applied user load parameter, $N_{UUsers}$, is increased uniformly from 10 to 250 with each active user accessing his own set of data whenever requests are sent to the local cache.

### Random eviction criterion

The pseudocode representing key functional features of the cache optimisation program is presented in Algorithm 1. It is important to point out that for the purpose of providing a complete summary of the experimental operations, Algorithm 1 also includes the primary functionalities of the Load Generator and Storage Manager programs. Once all start up misses have been dealt with and the cache is space is filled up, the Random Cache Eviction algorithm responds to any further misses by choosing the victim files in the local cache that are marked for deletion. The victim files are then replaced with the requested content, which is brought from the remote storage in order to satisfy the cache miss event. The durations of inter-arrival and service times of the user requests follow the exponential distribution with mean values of $\frac{1}{\lambda_{App}} = 2.5$ and $\frac{1}{\mu_{CPU}} = 1.5$ seconds respectively.

The results in Figure 4 showing both the measurements and theoretical trends confirm a decrease in the cache miss ratio, $P_L$, as the applied user load, $N_{UUsers}$, is increased. Even more significant from the graphs is the observation that the practical results track the modelled trends very closely, with the cache performance levels of the measurement results being higher than the theoretical ones.

An important factor leading to better performance for the measurement results is that the analytical models are based on the worst-case situations, in which the consideration of excess requests within each average cycle of user requests by $N_{UUsers}$ does not take into account the possibility that some of the requests generated in the practical scenarios would be accessing data already in the cache. Hence because of the inability of the performance analysis to quantify exactly the extent of improvement in cache hits caused by the repeat requests that can occur within the $\{N_{UUsers} - L\}$ excess requests inside each average load cycle, there are lower cache hit ratios for the theoretical trends.

The use of the Random Eviction criterion in selection of the files for deletion in the cache is another factor contributing to the better cache performance achieved in the practical measurements. Since the identities of requested files is determined by the Load Generator according to the probability, $(\frac{1}{N_{UUsers}})$, and that of the files to be deleted according to the probability, $(\frac{1}{L})$, identical strategies are thus used for the respective actions of file requests and cache optimisation. Having such alignment in the Load Generator and Caching Algorithm functional patterns therefore helps improve the cache hit ratios obtained from practical measurements. Despite the consistently lower performance levels for the modelled cache hit ratios, the comparisons in Figure 4 nonetheless show that the theoretical trends can be a reliable indicator of achievable cache performance in practice.

### Least frequently used criterion

The LFU Algorithm is structurally similar to the Random Eviction criterion, the important difference as shown in the program module below, being that update and sort functions on the popularity list are performed according to the frequencies of the requests for the cached files. The lowest ranked file on the list is marked for deletion and its place in the cache is taken by the newly requested content brought into the local cache from the remote storage.

**Algorithm 1** : ***Cache Eviction Simulator*** ($N_{UUsers}$, ***L, S***)

1: $Measurement.start \leftarrow true$
Beginning program execution
2: $((|LocalCache| \leftarrow L) \wedge (|RemoteStore| \leftarrow S)) \mid (LocalCache \subsetneq RemoteStore)$
Setting the cardinalities for, Local Cache and Remote Storage
3: **for** $j \leftarrow 1$ to $N_{UUsers}$ **do**
4: $StorageFile[j].Popularity \leftarrow default$
Initialising file popularity parameters to default value.
5: **end for**
6: $(t_i \leftarrow t_{(i-1)} + t_w) \mid (Pr(t_w) \leftarrow exp(\lambda_{App}))$
Setting the waiting time for next user request, $t_i$
7: $i \leftarrow \{0 + \%rand(N_{UUsers})\}$
Fixing User ID associated with next request, $i$
8: $Listen.Request[i] \leftarrow StorageFile[i].[Content]$
Setting the affinity between User ID and Target File ID
Execution of File Manager upon resolving startup misses
9: **while** $((\neg Measurement.stop) \wedge (CacheMiss.EventType \neg Compulsory))$ **do**
10: **if** $(Request[i].Content \in LocalCache)$ **then**
11: $LocalCacheRequest.Total.Update$
12: $LocalCacheHit.Total.Update$
13: **else**
14: $LocalCacheRequest.Total.Update$
15: $LocalCacheMiss.Total.Update$
16: $m \leftarrow \{0 + \%rand(L)\}$
Local Cache position ID, $m$, marked for Random Deletion.
17: $LocalCache.Position[m] \leftarrow Request[i].Content$
18: **end if**
19: $(CPUTime.Request[i] \leftarrow t_{CPU}) \mid (Pr(t_{CPU}) \leftarrow exp(\mu_{CPU}))$
Setting the service time for current request
20: $Request[i].execute$
21: return $Request[i].Result$
22: **end while**

In order to perform the comparisons for cache performance trends, two graphs are used in the validations: one based on measurement results and the other on the derived theoretical trends. The trends for the analytical characterisations are based on capacity misses occurring on the local cache space assuming that fully-associative mapping policies are enforced i.e. cached objects brought in from external nodes can reside anywhere within the entire cache storage area, $L$, that is set aside for local caching service. Apart from compulsory misses, the impact of conflict misses on the cache performance is also disregarded in the analysis, the assumption being that the incidence of predictive errors of cache policies (i.e. whenever the algorithms evict content that should have been retained) will have a negligibly low impact on the overall cache ratio, $P_L$.

1: **if** $(Request[i].[Content] \in LocalCache)$ **then**
2: $LocalCacheRequest.Total.Update$
3: $LocalCacheHit.Total.Update$
4: $Popularity.List.Update \leftarrow Request[i].Content.$
5: **else**

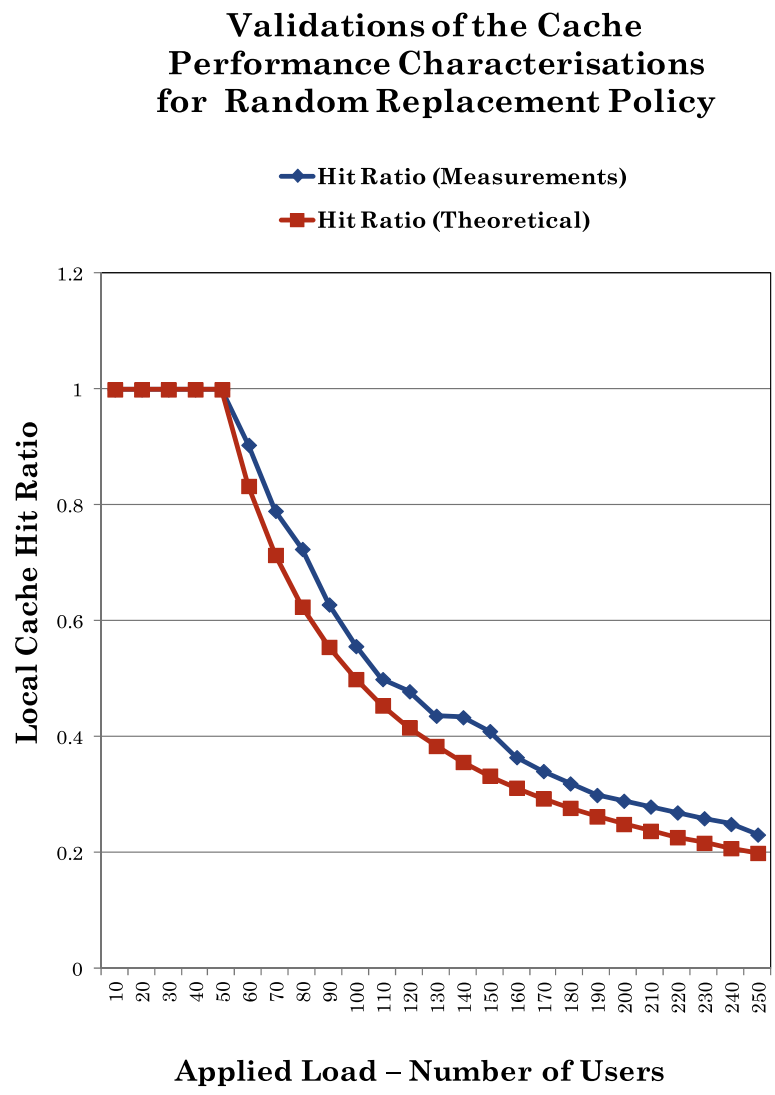


**Figure 4** Cache performance vs. applied load with random file evictions for private data.

6: *LocalCacheRequest.Total.Update*
7: *LocalCacheMiss.Total.Update*
8: *Popularity.List.Update* ← *Request*[ *i*] *.Content*.
9: *Popularity.Rank.Consult*
10: *m* ← *Lowest.Rank.CachePosition.Get*
Local Cache Position, *m*, marked for deletion according to LFU criterion.
11: *LocalCache.Position*[ *m*] ← *Request*[ *i*] *.Content*
12: **end if**

As in Figure 4, the overall cache hit ratio trend in Figure 5 shows a decrease according to the relationship, $P_L = (\frac{L}{N_{Users}})$ whenever the data requirements of user requests exceed the capacity of the local cache space.

Measurements for scenarios based on the popularity of cached data produce lower cache hits than in the case of optimisation techniques that use the Random Eviction policy. As can be seen in Figure 5, the practical values of cache hit ratios are much nearer to the theoretical ones than those shown in Figure 4. The reduced level of cache performance using the popularity based criteria suggests that the LFU algorithm is less efficient than the random deletions of cached files in response to cache misses. The performance knock resulting from the LFU algorithm is likely accounted for by the fact that mechanisms, which rank cached files by virtue of the frequencies of previous requests are not employing the relevant strategy given that the identities of requested files are in fact specified by the Load Generator at random, based on uniform probability of occurrence of magnitude, $(\frac{1}{N_{Users}})$. Hence, the LFU approach for ranking cached content only produces nonexistent patterns of file popularity, which in turn, results in reductions of cache hit events.

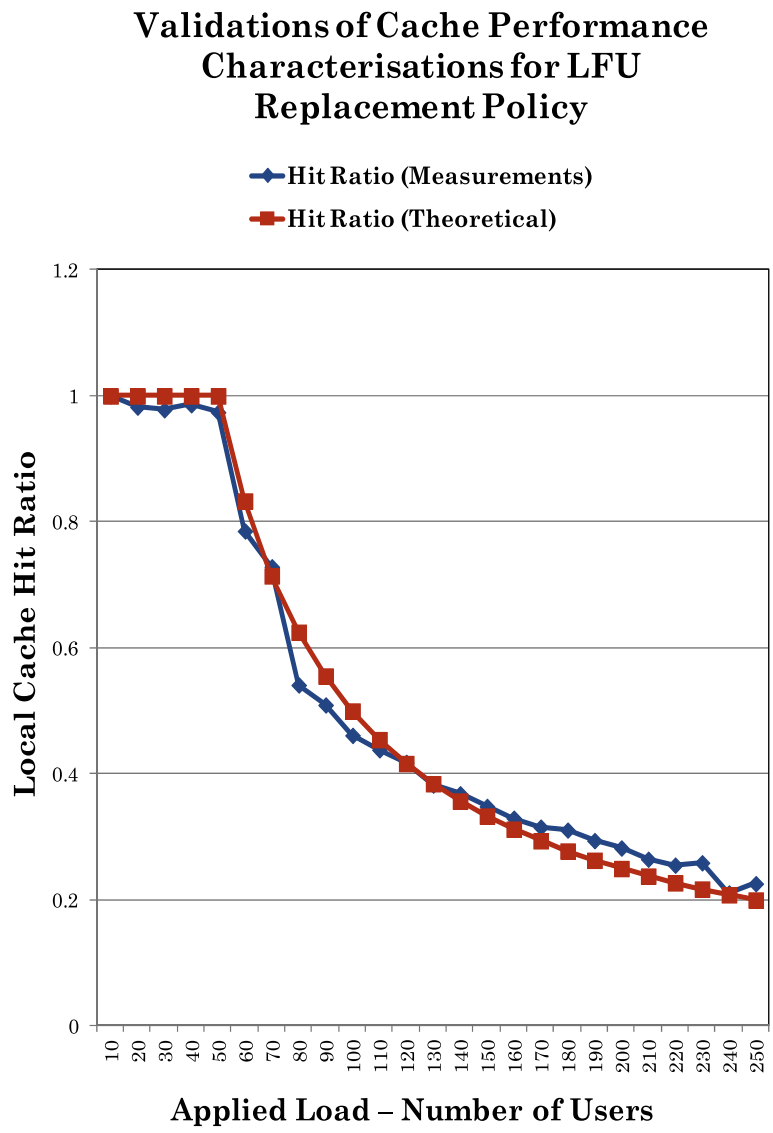


**Figure 5** Cache performance vs. applied load with LFU cache optimisation for private data.

**Least recently used criterion**

The LRU algorithm is based on rating cached files according to age so that file objects that have been kept in the local cache the longest are assigned the lowest indices of usefulness with respect to data needs of future requests. The oldest files are thus selected for deletion whenever cache miss events occur and the victim files are replaced by the newly requested content, which is transferred from the master storage.

The trends shown in Figure 6 for the comparisons of both the theoretical and measurement results confirm that there is a drop in local cache ratio performance as the load is increased beyond the local cache capacity. The results for the LRU algorithm are almost identical to those associated with the LFU criterion presented in Figure 5. As load levels exceed the local capacity cache corresponding to $L$ files, the cache hit ratios track the modelled trends very closely according to the analytical formula, $P_L = (\frac{L}{N_{Users}})$.

As in the case of the LRU, the lower cache hit ratio performance is probably an outcome of the mismatch between the cache optimisation strategy and the patterns associated with data requests coming onto the cache. The age-based approach of quantifying the likelihood of experiencing repeat requests in the future for cached files is not a useful optimisation technique given the random manner in which requested files are specified by users. Thus, any apparent difference in the ages of stored files that may be computed by the LRU provides no predictive value on the likely patterns of future file requests, which

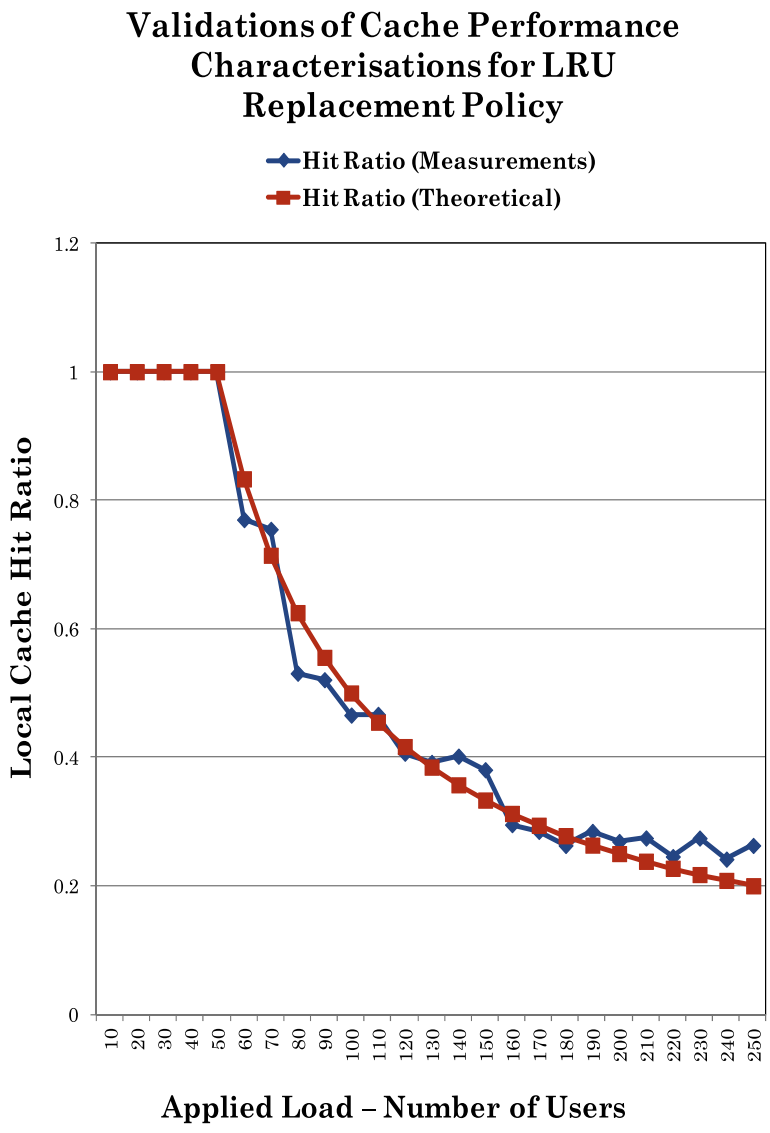


**Figure 6** Cache performance vs. applied load with LRU cache optimisation for private data.

according to Algorithm 1, are stochastic and follow a discrete uniform distribution with probability, $(\frac{1}{N_{Users}})$, for each request action.

### Summary of validation experiments for private data requests

The practical measurements that were conducted using the three cache optimisation algorithms produced results that were closely matched to the modelled theoretical trends. In terms of the actual cache hit ratios, the use of the Random Cache Eviction criterion resulted in better cache performance over the age and popularity-based LRU and LFU policies respectively. The inefficiencies in the LRU and LFU Cache Eviction criteria are due to the non-existence of age and popularity-based behavioural patterns in the requests for cached content. Since each of the public data objects, like private content, are requested with identical probability, the next set of validations of cache performance models for the requests for public data are therefore carried out on the basis that our initial results show superiority of Random Eviction policy over LRU and LFU by considering the scenarios involving use of the Random Cache Eviction policy only.

## Validations of cache performance trends for user requests of public data

In conducting the practical measurements, we simplify the cache performance models for public data requests developed in Subsection "Cache performance analysis of user requests for public data" by expressing the storage

capacities of the local cache and external storage in terms of $L$ and $S$, which are the respective numbers of active user requests that can fill up the storage entities. Content access patterns to the $S$ publicly available files on the remote storage are thus governed by the discrete uniform distribution with the probability, $\frac{1}{S}$, applying to each file request. Consequently, when the requested data objects can be specified randomly from the list of shared data objects that are kept in the remote storage, the respective sizes of the local cache and remote storage capacities impact on cache performance according to Equation 3.

To ensure that user requests cycle through all the $S$ files in the remote storage device, the duration of the experiment runs should be sufficient to cover at least $S$ unique file requests from active users. If the assumption is made that no repeat requests are generated in each request cycle, the minimum time period for the experiment should equal $\frac{S}{\mu_{CPU}-\lambda_{App}}$. Hence, the duration of the experiments for publicly hosted content can be expressed by the equation, $T_{Exp} = \frac{MS}{\mu_{CPU}-\lambda_{App}}$, where $M \geq 1$. Figure 7 shows the setup for the cache performance studies of accesses generated in $M$ request cycles.

Based on the setup shown in Figure 7, the derivation and practical evaluations of cache performance emanating from scattered requests take into account that of the $S$ possible files that can be specified by each user request with equal probability, $\frac{1}{S}$, there are $L$ files already in the cache if compulsory misses are ignored. Hence, assuming that the $M$ request cycles are sufficient to produce a record of data accesses to all the $S$ files, the probability that a requested data object is found inside the local cache is given by equation,

$$P(L)|(T_{Exp} >> \{\frac{S}{\mu_{CPU} - \lambda_{App}}\}) \approx \left(\frac{L}{S}\right). \quad (7)$$

As an input parameter the practical validation of cache performance trends, the applied load based on $N_{UUsers}$ is increased from 10 to 250 in uniform incremental steps as shown in Figure 8. Since the LRU and LFU algorithms proved ineffective in tracking file request patterns that are associated with the File Generator, our use of the cache optimisation techniques in the second practical study is confined to the technique of Random Eviction of least useful data in the cache. Figure 8 also presents three cases of modelled cache performance based on the ratio, $\frac{L}{S}$, which were chosen for comparison. The theoretical performance trends featured in the validations are based on the predefined hit ratios, $\frac{L}{S}$ =0.25, $\frac{L}{S}$ =0.5, and $\frac{L}{S}$ =0.75.

The results from the three scenarios confirm that in the event of data requests predominantly going to publicly hosted content, which all active users are free to access the overall cache performance is independent of load in accord with the theoretical approximation. Another important observation from the graphs in Figure 8 is that for the modelled cache performances of 25 and 50%, the theoretical and measurement results are very similar, with the practical results marginally better than the theoretical estimates in some places for cache hit ratio of 25%. As the size of the local cache is increased to 75% of the remote storage, the practical performance also goes even though it stays within the 70% range. We attribute the lower values of cache hit ratios in the measurements at high cache capacity to a further need for calibrating the number of user request cycles, $M$, that governs the duration for the measurements to capture the events so that the impact data of the requests to all $S$ files is accurately reflected by practical observations. Despite the discrepancies in the modelled and practical results shown in the graphs, particularly for higher values of hit ratios, the theoretical approximations of cache performance (in terms the respective sizes of local cache and remote storage capacities) is a reliable guide of cache performance for file requests that are spread across publicly available content.

Section "Performance characterisations for concurrent accesses to private and public content" follows the theoretical derivations of cache performance patterns with scenarios where mixed requests are generated simultaneously by users to access both public and private content.

## Performance characterisations for concurrent accesses to private and public content

The analysis of simultaneous requests patterns to public and private data considers two cases of cache space allocation, one which features separate cache partitions for private and public data and the other involving the use of a common cache partition that is shared by both types of content. In both cases of cache space assignment, the sizes of input parameters used in the performance analysis remain the same, i.e. $N_{UUsers}$ is uniformly varied from 10 to 250, while $L$ is equal to 50, and the remote storage space for public data, $S$, is 200. The list of input and output parameters used in the derivations of cache performance models that are associated with the simultaneous requests of private and public is provided in Table 2.

### Data access patterns for content on separate caches

We define separate cache partitions, $L_1$ and $L_2$, for private and public data respectively on the local cache storage. The cache partitions are both equal to 50 and the respective probabilities of each active user requesting private and public data are $P_1$ and $P_2$ i.e. a generated data request either specifies personal user or publicly available content, which means $P_1 + P_2 = 1$. Figure 9 shows the setup involving data requests going to separately allocated cache spaces.

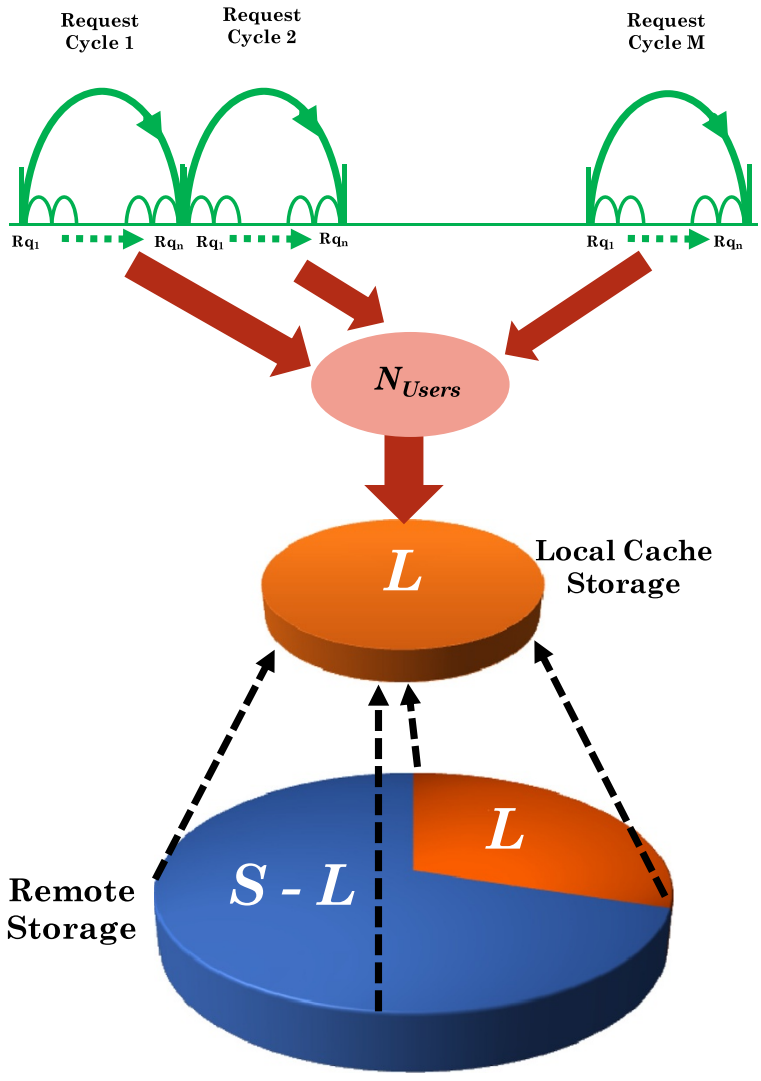


**Figure 7** Setup for validations of cache performance trends for dispersed file accesses.

We recall that the local cache hit performance, $P_{HitL1}$, on the cache partition assigned for private user content, $L_1$, is given by the expression:

$$P_{HitL1} = \begin{cases} 1 & \text{if } N_{Users} \leq L_1 \\ \dfrac{L_1}{N_{Users}} & \text{if } N_{Users} > L_1. \end{cases} \tag{8}$$

Similarly, the local cache performance, $P_{HitL2}$, on the cache space that is set aside for hosting public content is given by the expression,

$$P_{HitL2} = \begin{cases} \left(\dfrac{L_2}{S_2}\right) & \text{if } L_2 < S_2 \\ 1 & \text{if otherwise.} \end{cases} \tag{9}$$

Hence, taking into account the preference weights associated with user request patterns to both sets of cached data, the cache hit ratio for private data accesses becomes:

$$P_{HitPrivate} = \begin{cases} P_1 & \text{if } N_{Users} \leq L_1 \\ \left(\dfrac{P_1 L_1}{N_{Users}}\right) & \text{if } N_{Users} > L_1. \end{cases} \tag{10}$$

The cache hit ratio associated with requests to public data is given by equation,

$$P_{HitPublic} = \begin{cases} \left(\dfrac{P_2 L_2}{S_2}\right) & \text{if } L_2 < S_2 \\ 1 & \text{if otherwise.} \end{cases} \tag{11}$$

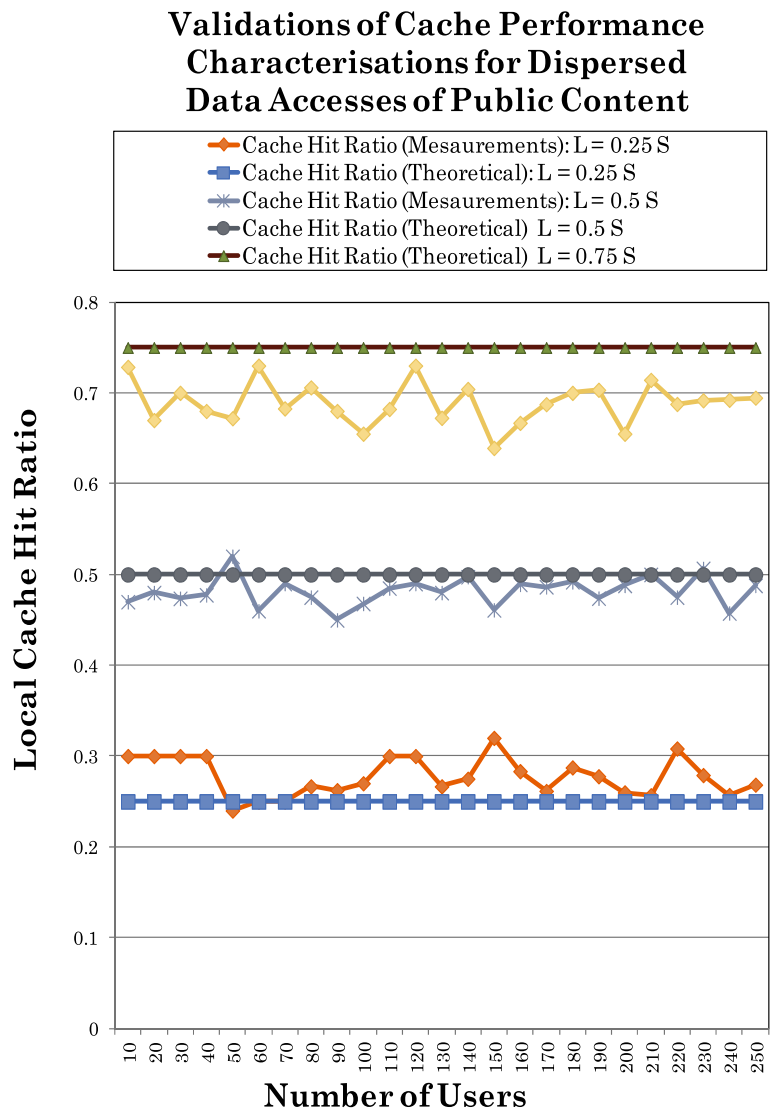


**Figure 8** Comparisons of cache performance for dispersed access of content.

The overall cache hit ratio for data accesses over the both cache partitions is then given by the expression,

$$P_L = \begin{cases} P_1 + \left(\frac{P_2 L_2}{S_2}\right) & \text{if } N_{UUsers} \leq L_1 \\ (\frac{P_1 L_1}{N_{UUsers}}) + \left(\frac{P_2 L_2}{S_2}\right) & \text{if } N_{UUsers} > L_1. \end{cases} \quad (12)$$

The overall trends for the cache hit and miss ratios associated with the requests to both sets of hosted data are shown in Figures 10 and 11 respectively.

As shown by both Figures 10 and 11, the Private Access Ratios i.e. the preference weights associated with requests to private data, $P_1$, are varied in uniform steps of 0.1 from 0 to 1. Conversely, the access weights for public data, $P_2$, vary in reverse order from 1 to 0 for the featured scenarios in the graphs, given that the request events to public

**Table 2 Parameters for concurrent requests of private and public data**

| Parameter | Description |
|---|---|
| $P_1$ | Probability of Requesting Private Data |
| $P_2$ | Probability of Requesting Public Data |
| $L_1$ | Capacity of Cache Partition for Private Data (users) |
| $L_2$ | Capacity of Cache Partition for Public Data (users) |
| $S_1$ | Capacity Remote Storage Partition for Private Data (users) |
| $S_2$ | Capacity Remote Storage Partition for Public Data (users) |
| $P_{HitPrivate}$ | Cache Ratio from the requests of Private Data |
| $P_{HitPublic}$ | Cache Ratio from the requests of Private Data |
| $P_L$ | Overall Hit Ratio on the Local Cache |

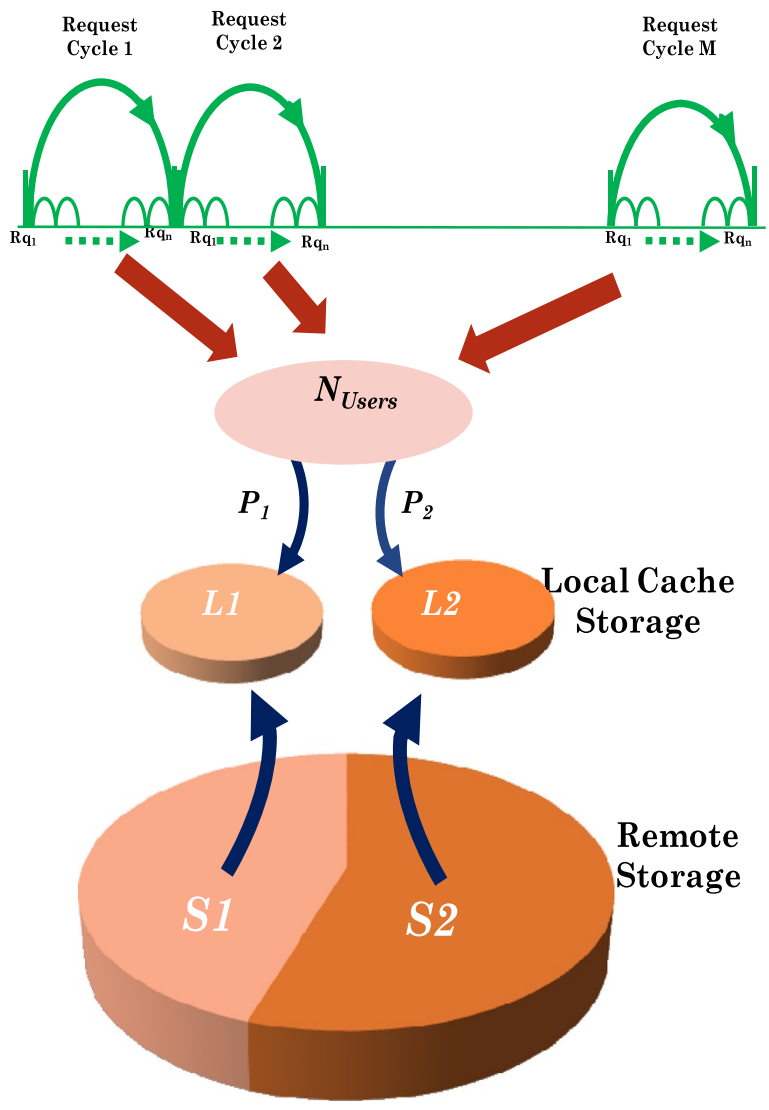


**Figure 9** Access to public and private content on separate cache partitions.

and private content are mutually exclusive. Hence, when Private Access Ratio is 0, the cache hit and miss trends solely emanate from requests that are directed at public content. It also follows that when Private Access Ratio is equal to 1, the cache performance complete derives from the requests for private user content. Between the two extremes as $P_1$ is raised gradually, the cache performance patterns associated with private data requests become more dominant.

It can be deduced from the equation of the overall cache hit ratio, $P_L$, that the assignment of separate cache partitions provides the ability to isolate and individually control the respective cache performance trends associated with private and public data requests. Thus, within the boundary fixed by $P_1$, there is the ability in the spilt cache configuration to tune $L_1$ and fix the cut-off point, at which cache performance begins to fall exponentially with increase in $N_{UUsers}$ for requests of private data. Within the bounds of $P_2$, the cache capacity, $L_2$, can similarly be adjusted with respect to $S_2$ to determine the average cache performance associated with the requests of public data.

The trends for overall cache misses, $M_L$, shown in Figure 11 can have serious QoS implications, should there be considerable delays associated with data retrievals from external source storage whenever requested content is not found in the local cache. If the respective data access times that are experienced in the event of cache hits are also taken in account, the tuning of cache performance can be carried out to deliver output performance that keeps average storage access within SLA thresholds. With the allocation of split caches providing the flexibility of enabling individual adjustments of cache sizes,

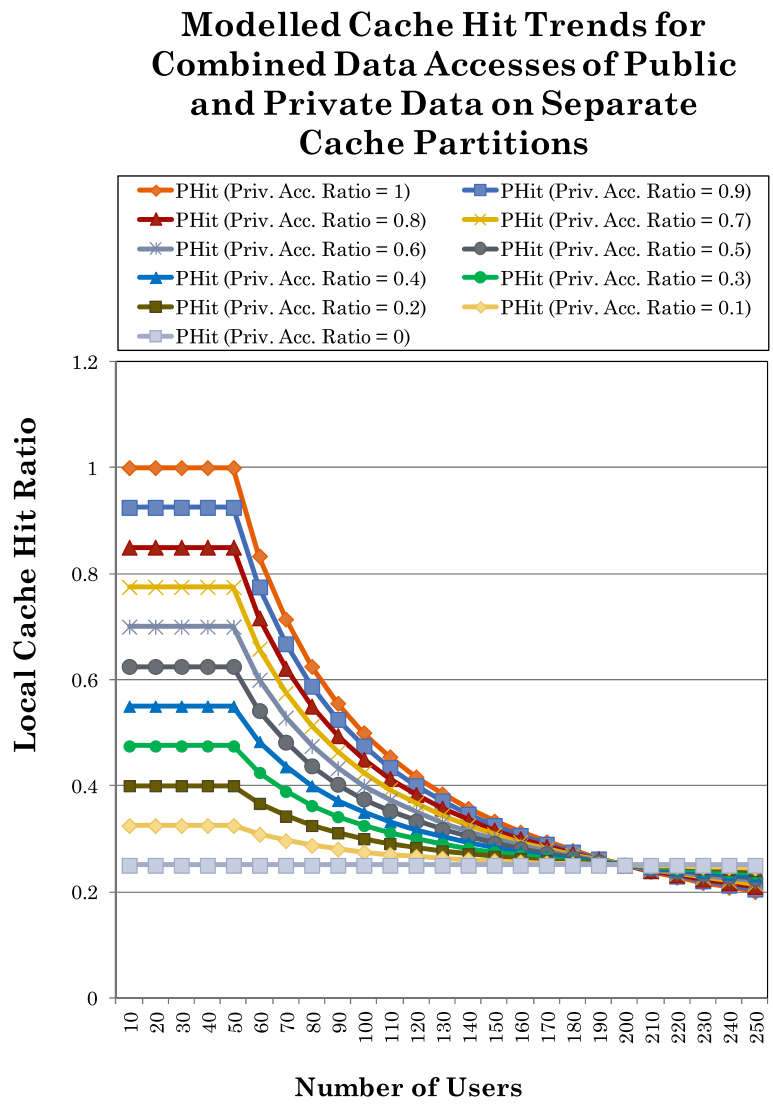


**Figure 10** Cache hit trends: public and private content on separate cache partitions.

performance can thus be managed in a way that discriminates between different sets of content according to their desired QoS ratings.

### Data access patterns for content on a shared cache partition

We begin the analysis of cache performance trends for mixed data access patterns on a shared cache partition by assigning a value of 50 to the common cache space, $L$, which is made up of $L_1$ and $L_2$ as the component caches for holding the private and public data respectively. The preference weights associated with the data request patterns to $L_1$ and $L_2$ are $P_1$ and $P_2$. The remote storage capacity for $S_2$ is set to 200 and the parameter, $N_{UUsers}$, for number of active users that generate data requests increases uniformly from 0 to 250. Figure 12 shows the basic setup for user requests accessing data on a shared cache partition.

Given that the total cache space, $L$, is divided up between public and data requests, we can express the amount of space allocated to $L_1$ as follows:

$$L_1 = \begin{cases} P_1 N_{UUsers} & \text{if } N_{UUsers} \leq L \\ P_1 L & \text{if } N_{UUsers} > L. \end{cases} \tag{13}$$

Thus, the cache space for private data is a subset of the data requested by active users according to the proportional factor which is equivalent to probability, $P_1$, if the number of users does not exceed $L$. Whenever $N_{UUsers}$ becomes greater than $L$, the average space occupied by public data is $P_1 L$. Similarly, the amount of cache

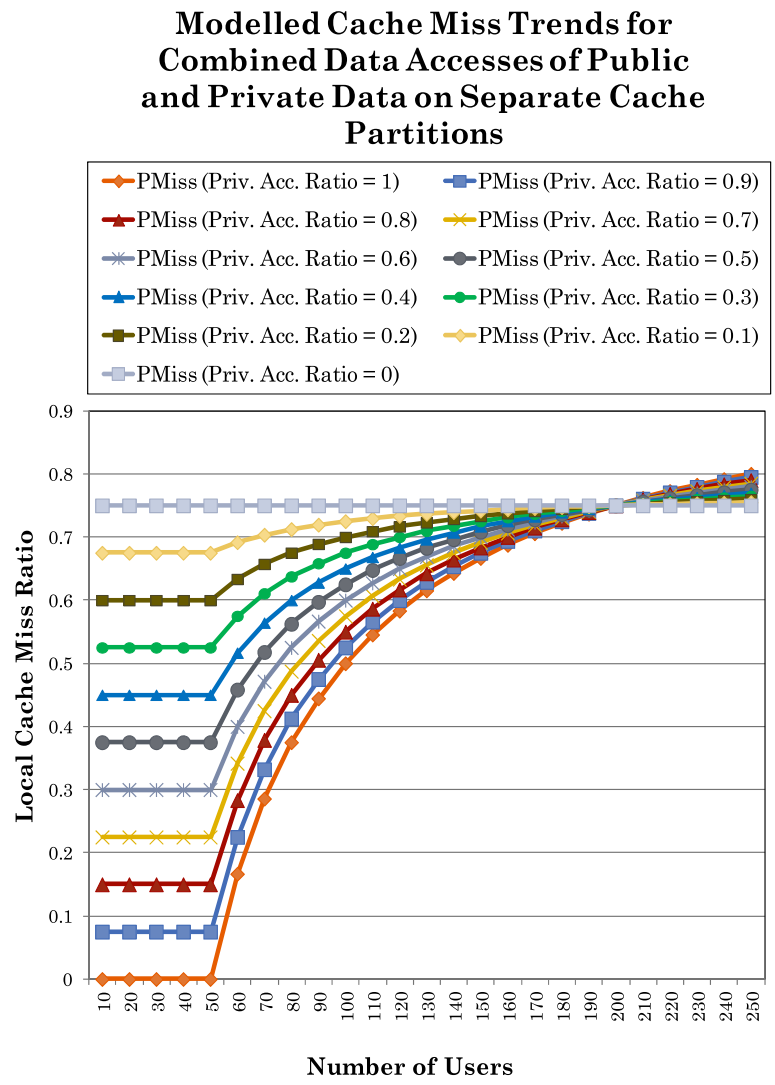


**Figure 11** Cache miss trends: public and private content on separate cache partitions.

space, $L_2$, that is occupied by public data is given by the expression:

$$L_2 = \begin{cases} (L - P_1 N_{UUsers}) & \text{if } N_{UUsers} \leq L \\ P_2 L & \text{if } N_{UUsers} > L. \end{cases} \tag{14}$$

Since $S_2$ is greater than $L$, the cumulative requests for public content will inevitably fill all space (equivalent to $L - P_1 N_{UUsers}$) that is left by public data requests if the number of active users remains lower than $L$. Once $N_{UUsers}$ goes beyond the cache capacity, $L$, the storage space is shared proportionally according to the ratios $P_1$ and $P_2$.

The expression for the cache performance associated with private data requests becomes,

$$P_{HitPrivate} = \begin{cases} P_1 & \text{if } N_{UUsers} \leq L \\ \dfrac{P_1^2 L}{N_{UUsers}} & \text{if } N_{UUsers} > L. \end{cases} \tag{15}$$

For the data accesses to public content the cache trends are given by the expression,

$$P_{HitPublic} = \begin{cases} \dfrac{(L - P_1 N_{UUsers}) P_2}{S_2} & \text{if } N_{UUsers} \leq L \\ \dfrac{P_2^2 L}{S_2} & \text{if } N_{UUsers} > L. \end{cases} \tag{16}$$

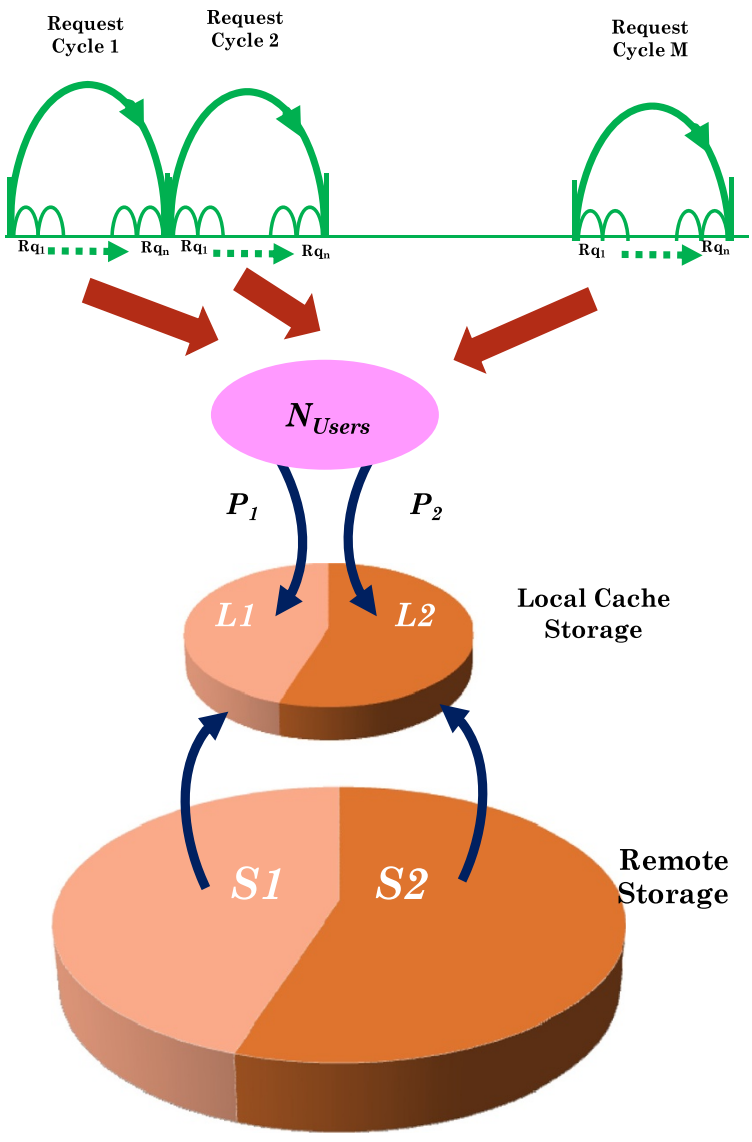


**Figure 12** Access to public and private content on a shared cache partition.

From the constituent cache performance trends of the public and private data requests, the overall cache hit ratio is therefore given by the expression,

$$P_L = \begin{cases} P_1 + \dfrac{(L - P_1 N_{UUsers})P_2}{S_2} & \text{if } N_{UUsers} \leq L \\ \dfrac{P_1^2 L}{N_{UUsers}} + \dfrac{P_2^2 L}{S_2} & \text{if } N_{UUsers} > L. \end{cases} \tag{17}$$

Figure 13 shows the overall cache hit trends on a common cache partition as the access weight, $P_1$, that is assigned for private data is uniformly increased from 0 to 1.

As in the case of separate cache partitions, the cache hit ratios trends for the scenario, $P_1 = 0$, correspond to data requests that are going to completely public content, while that for $P_1 = 1$, applies solely to accesses to private content. Between these two extremes, the cache hit ratios fall more steeply compared to the corresponding scenarios considered for split caches as shown in Figure 12. The rate of performance drops as $P_1$ is raised from 0.1 to 0.9 is due to limited space on $L_1$, which is divided up between the two sets of cached content.

The impact of having a shared cache can be further emphasised by Figure 14, which shows the corresponding cache miss ratios as $P_1$ varies between 0 and 1. Comparisons with Figure 10, which has the family of cache miss

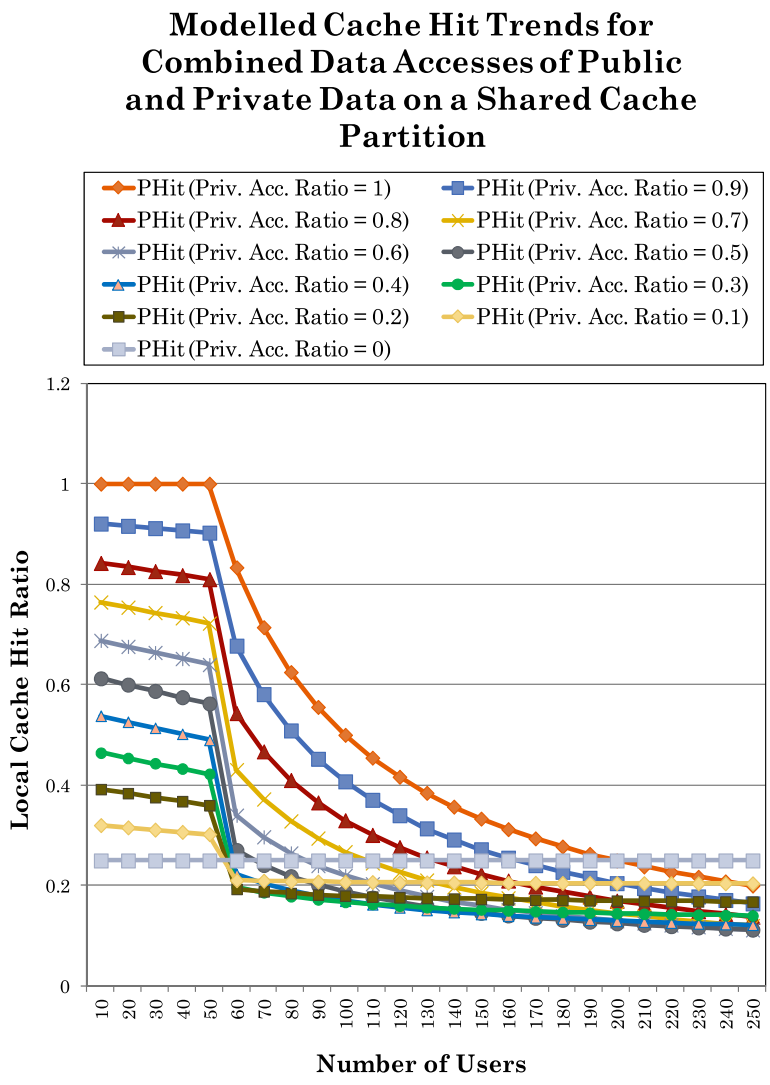


**Figure 13** Cache hit trends: public and private content on a shared cache partition.

ratio trends for corresponding scenarios of $P_1$, reveal that with $N_Users$ reaching the value of 250, the cache miss ratios associated with the shared cache partition range between 75 and 90% , while those for split cache configuration are between 75 and 80%. The impact of the higher cache misses on overall performance is amplified if the data access operations that are associated with content transfers from external storage are subject to huge delays.

The equations for cache performance trends of private and public data requests are subject to the size of the available space, $L$, in the common cache. As such it is not possible to individually change the storage allocations for given sets of content without affecting other cached datasets. Hence, even though the shared cache configuration is simpler to implement and is less computationally expensive because of having all the cached datasets on a single global list for cache optimisation, the design does not permit flexible allocation of cache space that would grade various sets of content according their assigned QoS categories.

## Discussion

We began the discussion by highlighting the need for having capabilities for scalable solutions in storage cloud domains so that infrastructure-based responses can be achieved for maintaining performance within SLA thresholds in the event of such challenges as increases in user demand or, interruptions to the operating states of the service entities making up the cloud resource fabric. We went on to argue that for SLA-compliant services to be provided consistently over a wide range of load levels, an in-depth understanding of the performance trends associated with storage cache resource

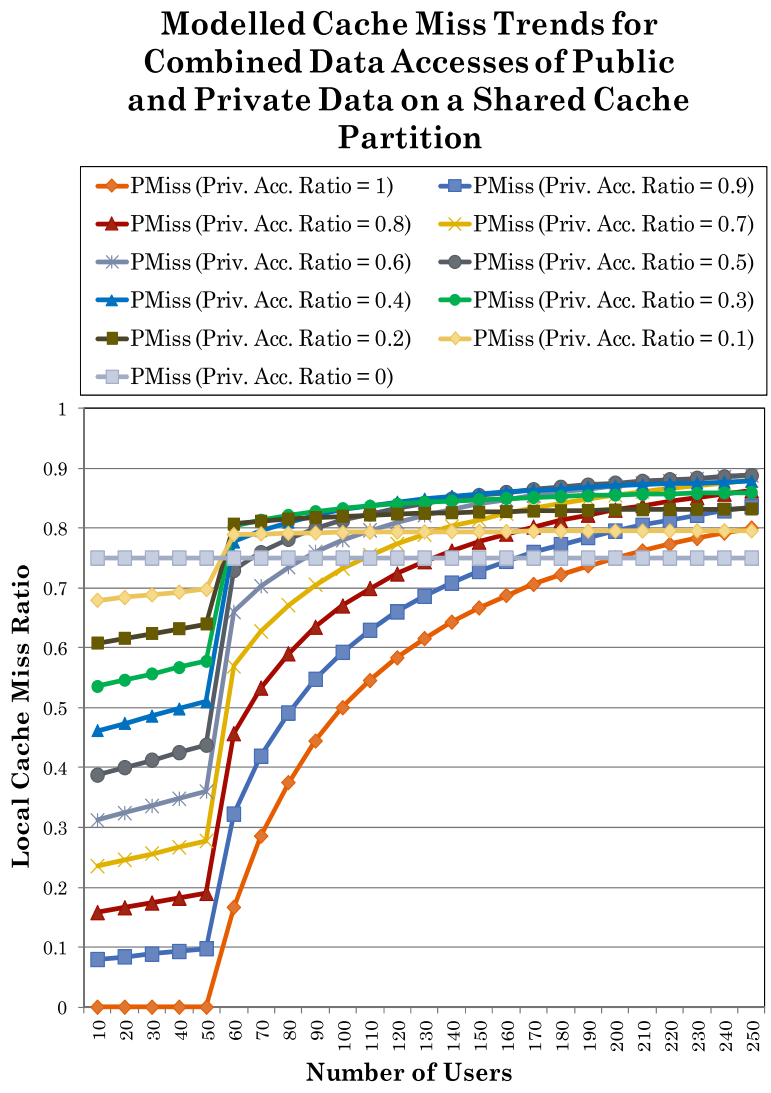


**Figure 14** Cache miss trends: public and private content on a shared cache partition.

entities in the cloud infrastructure, is an important foundation on which to base QoS-ready solutions. It was further pointed out that from the performance characterisations of storage cache entities, storage resource management decisions on infrastructure sizing can be made, which are relevant to important stages of resource deployment such as initial roll-outs, short-term expansions to deal with overflow requests, and permanent upgrades.

Theoretical models were proposed for estimating performance trends occurring at individual cache entities as the levels of user demand for content increase. In order to validate the derived theoretical trends, three suites of experiments (based on Random, LFU and LRU eviction policies) were defined for studying the sensitivity of cache performance to applied loads. A noteworthy observation from the results was that whenever the data request patterns are characterised by rigid affinity to content (i.e. each user accessing only its own data) and with the requested data objects being of comparable popularity, the decay trends for the measured cache hits exhibit high fidelity to the theoretical characterisations. Additionally, the Random Cache replacement algorithm provides better results than the LFU and LRU algorithms, which although still conforming to theoretical estimates, have lower levels of cache hit ratio performance. Thus, the LFU algorithm is more effective if there are distinct categories of data popularity from the users generating the requests. Similarly, the LRU algorithm performs better for cases where the usefulness of cached content is indexed by age, and thus the algorithm is not equipped deal with the even scatter of requests over a wide range of file objects.

Overall, the results for private data requests demonstrate the potential utility of the models for estimating the cache storage needs associated with computing scenarios involving enterprise application routines such as Sales and Distribution (SD), Assembly-to-Order (ATO) or Employee Self-Service Portal (EP-ESS), whereby user requests work with their own sets of customer account data [11].

Another important observation from the results was the confirmation of the validity of trends for the second set of cache performance models that is associated with data requests for wide selections of files [12], where there is loose coupling between users and public content. The cache performance levels in such cases characterised by dispersed requests are independent of the levels of input user load i.e. the local cache hit ratio can be expressed as a function of the respective ratios of the local cache and remote storage capacities, $P_L = \frac{L}{S}$. The accuracy of performance measurements for scattered file requests is based on setting the durations of the observation time window for the experiments long enough to cover data requests to the all the $S$ files kept in the remote storage. Additionally, the measurement results of the three cases of predefined cache performance (corresponding to $P_L$ = 0.25, 0.50 and 0.75) show that the actual values of cache hit ratios obtained from practical investigations are close to the theoretical estimates. However performance from results is slightly lower than the theoretical one when the size of the local cache is increased to $0.75S$. It therefore has to be emphasised that the potential usefulness of second set of characterisations in predicting cache performance trends in scenarios where users interact with public content (to which there is unrestricted access) is subject to durations of the observation time window.

The cache model extensions for the characterisations of concurrent requests to private and public data were developed in Sections "Validations of cache performance trends for user requests of public data" and "Performance characterisations for concurrent accesses to private and public content", the modelled trends derived for mixed requests apply to scenarios of shared and separate storage cache partitions holding the cached content.

## Related research

A number of initiatives are being pursued towards maturing performance management capabilities in cloud computing infrastructures so that scalable, secure and reliable IT services can be delivered to computing environments, most of which run business-critical applications. Below, we briefly highlight some of the work that is underway to develop SLA-based strategies for supporting firm guarantees of performance delivery.

In [13], estimates of output performance based on the levels of applied user loads and the mean service rates at the resource entities are used and, from applying the Laplace Stieltjes Transform, the overall response times at each service node in the cloud infrastructure are calculated, with SLA mappings being derived from probability distributions of the calculated response service times. The study in [14] also features use of input parameters such as the number of service requests from the consumers and the service capacities offered by deployed resource entities in developing SLA indices and, a trust model is then obtained for performing predictive estimates of the levels of resource and service availability in the cloud infrastructure. The strategy for SLA enforcement that is presented in [15] categorises workload instances that are despatched to server entities into four basic classes of resource consumption of the processor, memory and disk entities, and from taking into account the service constraints in the cloud infrastructure, the optimisation function determines the number of VM instances of each resource consumption class, which can be hosted by the provider. The work that is presented in [16] features fault tolerant and redundancy techniques for identifying and filtering out compromised resource elements in the infrastructure in order to ensure service availability and continuity in the cloud. Apart from applying redundancy strategies on the matrix-mapped resource collections, the SLA enforcement in [16] also employs predefined performance constraints on the constituent resource entities in the cloud together with integer linear programming methods that eliminate faulty and malicious elements with the greatest likelihood of compromising service quality.

By making high-level considerations regarding the overall resource capabilities in developing strategies for SLA guarantees, the approaches described above thus treat the runtime operations of CPU execution, memory and storage data access as a single composite service functionality, which differs from our work, whose focus is exclusively on establishing internal cache performance trends pertaining to storage access.

The studies in [17-19] have a similar focus to our approach of isolating subsystems of server hardware in order to characterise their resource consumption patterns for SLA and QoS support, the difference being that the strategies presented in all the three contributions consider CPU and memory utilisations associated with processor-bound workloads. Specifically, the strategy in [17] aims to guarantee CPU QoS delivery by overcoming the common problem of runtime interference effects that usually arises when running multiple instances of applications that are derived from virtualisation technologies. The interferences between the active VM application instances are minimised through the control of the working set sizes of allocated memory pages, thereby ensuring predictability of memory fetch times, CPU utilisations and ultimately, processor QoS support. In [18], the standardised metric,

EC2 Compute Unit (ECU), developed by Amazon is used for rating available computing power on various CPU hardware architectures. Based on the ECU metric, thresholds margins can be defined for identifying the resource utilisation levels, at which SLA violations are approached and the reallocation of the CPU resources can be initiated to protect SLA contracts. The framework presented in [19] features a dynamic SLA template that is designed to deal with changing user requirements by mapping consumer requirements to existing capabilities in the cloud infrastructure, with the focus also being on the allocation of processor cores as the primary resource entities of user interest.

In a related contribution on data caching mechanisms featured in [20], models have been developed for cache hit performance, with emphasis however being on the performance of multilevel cache configurations based on hierarchical and cooperative models for data sharing across distributed environments. Another endeavour on developing caching solutions for improve data availability is in the form of the Tuxedo caching framework presented in [21], which is based on the use of protocols to enhance traditional CDN and local caching strategies and thereby ensure that user requests for both personalised and public content are fulfilled incurring minimum latency. While the objectives behind Tuxedo are very similar to the motivations for our work particularly as considered in Section "Performance characterisations for concurrent accesses to private and public content", the approach taken in the former approach is different from ours in that the emphasis of Tuxedo is on an architecture-based solution as opposed to the quantitative analyses for cache performance that we consider in this paper.

## Future directions

It has been highlighted that the relative inadequacies of LRU and LFU cache algorithms in the scenarios that were featured in our studies, stem from the inherent bias of the cache optimisation logic to index the usefulness of cached content according to age and popularity respectively. It is therefore necessary to build within our cache algorithms the ability to capture and respond to the complexity in the behaviours of user requests. Hence, one strand of further work will proceed in the direction of establishing and characterising the relevant dynamics affecting the likelihood of repeat requests of cached content based on both popularity and passage of time. A significant part of proposed investigations on this aspect will consider developing strategies for breaking down the cached content into principal categories of popularity (such as High, Moderate and Low popularity) and building time profiles for the request events so that the decay of content popularity is defined as a function of time. The proposed extension will be a further step in the study of the heterogeneity of data access patterns, which in our current models involves two broad classes of data requests; private and public content requests. From the results of this study, we intend to calibrate the cache replacement algorithms on the basis of hybrid criteria that employ adjustable time windows for rating content value.

The allocation of storage cache space for accessed data in our initial studies was simplified through the choice of uniform file sizes. In the next phase of the study, we will therefore also investigate approaches, by which cached content is classified according file sizes. In working towards the overall estimates of the required cache capacity, it will be important to investigate how to characterise patterns of the variability of the range of all file sizes grouped together within each category.

Another dimension worth exploring in the future work is employing the utilisation of the strategy proposed in [22] to harness the cloud infrastructure as a data gathering and dissemination engine to achieve ready availability of context information in supporting informed caching decisions. The information collection and dissemination technique considered in [22] is predicated on the idea that context data exhibits predominantly temporal trends. Hence, cache optimisation mechanisms (most likely in the form of enhanced versions of the LRU policy) can be developed for characterising the time-related properties of cached items in such a way that their values are indexed and, the eviction and retention of content can then follow formal criteria. An additional aspect of scoring the cached files would determine how to categorise the rates of expiration of cached objects based on the frequency of modifications to original files. Typically, the public content which becomes stale more quickly would be based on volatile updates such as live sports news and business feeds.

As has been highlighted in Section "Experimental facility for validations of cache performance Trends", our experimental scenarios employ the Tier 2 configuration i.e. application executions and data fetch operations are conducted on the same physical server. In the next phase, part of the focus will involve deployments based on the Tier 3 setting, whose configuration is such that application routines and data transfer operations are handled in server entities. Based on the outcome of the experiments conducted so far, we consider network delays that are associated with the transfers of requested data to be the most likely factor that can impact the accuracy of the future experiments. Hence, an important aspect of the work on analysing cache performance in Tier 3 server settings will involve characterisations of the network delays so that the time windows for the measurement epochs are properly calibrated according to prevailing conditions on the data transfer paths

such as available bandwidth, propagation and congestion delays. And since our current theoretical models basically apply to standalone cache configurations, the Tier 3 scenarios can also be considered in the context of more complex caching environments based on redundant and hybrid physical deployments. Thus, the follow up work will study of the joint use of network management and replica location services on our infrastructure-monitoring framework in order to characterise service performance profiles associated with wide-area data accesses in cloud environments.

**Competing interests**
The author declare that they have no competing interests.

**Author's contributions**
ES and GP designed the basic caching strategies for the experimental studies and the combined data scenarios that were featured in the paper. SM and BS developed the analytical models for characterising the caching scenarios that were considered. AM developed the test cloud infrastructure and the code that was deployed in the User Request, File Manager and Master Storage Virtual Machines. DB and AM helped with the structure and the Introduction of the paper. All authors read and approved the final manuscript.

**Acknowledgements**
The authors acknowledge support for this work from the Engineering and Physical Sciences Research Council (Grant References EP/G051674/1 and EP/J016748/1). Any views or opinions presented herein are those of the authors and do not necessarily represent those of IU-ATC, their associates or their sponsors.

**Author details**
[1]Networking and Computing Technologies Laboratory, University of Ulster at Coleraine, Coleraine - BT52 1SA Northern Ireland, United Kingdom.. [2]School of Computing and Information Engineering, University of Ulster at Coleraine, Coleraine - BT52 1SA Northern Ireland, United Kingdom.

Received: 11 March 2012 Accepted: 28 November 2012
Published: 10 January 2013

**References**

1. Oracle Inc. Oracle platform for SaaS. http://www.oracle.com/us/technologies/saas/index.html. Accessed 12 December 2012
2. SAP Inc. On-demand solutions from SAP that fit your needs now. http://www.sap.com/solutions/technology/cloud/index.epx
3. Varia J (2010) Amazon Web Services., Architecting for the cloud: best practices, AWS white paper, (January 2010), pp 1–21. https://aws.amazon.com/whitepapers/
4. VMWare Inc. (2010) Architecting a vCloud, technical white paper, version 1.0, pp 1–30. http://www.vmware.com/solutions/cloud-computing/index.html
5. Piech M (2009) Oracle Corporation., Platform-as-a-service private cloud with oracle fusion middleware, oracle whitepaper (October 2009), pp 1–20. http://www.oracle.com/us/technologies/cloud/index.htm
6. VMware Inc. (2010) VMware vSphere - The best platform for building cloud infrastructures. http://www.vmware.com
7. VMware Inc. (2009) vSphere basic system adminstartion VCenter server 4.0, (2009), pp 1–370. http://www.vmware.com
8. Rackspace Hosting, Hosting solutions for business. http://www.rackspace.co.uk/managed-hosting/solutions-for-business/. Accessed 12 December 2012
9. Mozy, Mozy products for business. http://mozy.co.uk/products. Accessed 12 December 2012
10. Eucalyptus Systems, Eucalyptus open source cloud computing infrastructure - an overview, Euclayptus Whitepaper. http://www.eucalyptus.com/. Accessed 12 December 2012
11. Alexa - Site Inforamtion. http://www.alexa.com/siteinfo. Accessed 14 December 2012
12. Finkelstein S, Brendle R, Jacobs D, Hirsch M, Marquard U (2008) The SAP transaction model: know your applications. In: ACM. SIGMOD. Conference
13. Xiong K, Perros H (2009) Service performance and analysis in cloud computing. In: SERVICES '09 proceedings of the 2009 congress on Services - I, Los Angeles, CA, USA, July 6-10, 2009
14. Kim H, Lee H, Kim W, Kim Y (2010) A trust evaluation model for QoS guarantee in cloud systems. Int J Grid Distributed Comput 3(1): 1–10
15. Tordsson J, Montero RS, Moreno-Vozmediano R, Llorente (2012) IM cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Generation Comput Syst 28(2): 358–367
16. Deng J, Huang SCH, Han YS, Deng JH (2010) Fault-tolerant and reliable computation in cloud computing. In: Globecom Workshops (GC Wkshps 2010), Miami Florida (December 2010)
17. Nathuji R, Kansal A, Ghaffarkhah A (2010) Q-Clouds: managing performance interference effects for QoS-Aware clouds. In: Proceeding EuroSys '10 proceedings of the 5th European conference on computer systems, Paris, France, 13-16 April 2010
18. Goiri I, Julia F, Fito JO, Macias M, Guitart J (2010) Resource-level QoS metric for CPU-based guarantees in cloud providers. In: GECON'10 proceedings of the 7th international conference on economics of grids, clouds, systems, and services, Ischia, Italy, August 31, 2010
19. Maurer M, Emeakaroha VC, Brandic I, Altmann J (2012) Cost-benefit analysis of an SLA mapping approach for defining standardized cloud computing goods. In: GECON'10 proceedings of the 7th international conference on economics of grids, clouds, systems, and services. Vol.28, No.1, Jan. 2012
20. Dykes SG, Robbins KA (2001) A viability analysis of cooperative proxy caching. In: IEEE INFOCOM, Anchorage, Alaska, USA, April 2001
21. Shi W, Shah K, Mao Y, Chaudhary V (2003) Tuxedo: a peer-to-peer caching system. In: Intl Conf on Parallel and Distributed Processing Techniques and Applications (PDPTA) 2003
22. Kiani SL, Anjum A, Antonopoulos N, Munir K, McClatchey R (2011) Towards Caching in the Clouds. In: International Workshop on Intelligent Techniques and Architectures for Autonomic Clouds (ITAAC 2011) co-located with 4th IEEE/ACM international Conference on Utility and Cloud Computing (UCC 2011), Melbourne, Australia, 8th December, 2011

doi:10.1186/2192-113X-2-1
**Cite this article as:** Sithole *et al.*: **Cache performance models for quality of service compliance in storage clouds.** *Journal of Cloud Computing: Advances, Systems and Applications* 2013 **2**:1.