International Journal of Computer Applications Technology and Research
Volume 5– Issue 6, 364 - 367, 2016, ISSN:- 2319–8656

A Review of Storage Specific Solutions for Providing Quality of Service in Storage Area Networks

Joseph Kithinji
Department Computer Science and Information Technology,
School of Information Technology and Engineering,
Meru University of Science and Technology,
Meru, Kenya

Abstract: Predictable storage performance is a vital requirement for promising performance of the applications utilizing it and it is the systems administrators' job to ensure that storage performance meets the requirements of the applications. Most storage solutions are able to virtualize the amount of storage presented to the host in a flexible way, but the same storage devices have no QOS features .Storage level agreements provided by storage devices do not provide predictability in service delivery due to the absence of prioritization (QOS) mechanisms in storage devices. This paper reviews some of the storage specific solutions developed to implement quality of service in storage area networks.

Keywords: Starvation, latency, burst handling, quanta, performance isolation.

1. INTRODUCTION

Storage area networks play a key role in business continuity, enterprise wide storage consolidation and disaster recovery strategies in which storage resources are most often distributed over many distant data centers[10].Future storage systems are required to scale to large sizes due to the amount of information that is being generated. In a SAN large numbers of magnetic disks are attached to a network through custom storage controllers or general purpose pcs and provide storage to application servers [6].

In a SAN, a single host request may flood the resources of a storage pool causing poor performance of all hosts utilizing that particular pool [5]. Hence, the performance of a given host utilizing a shared pool resource is unpredictable by the nature of resource sharing. To address this problem a mechanism of providing QOS based on some policy is required. Storage service level agreements provide for predictability in service delivery which is not effective due to the absence of QOS mechanisms in storage devices [5].

QOS is essential in the mixed environment where various users with different levels of priorities and preferences are accessing the storage systems simultaneously. For example in an enterprise network, web hosting, data analysis and data editing may be running at the same time[10].Providing QOS to SANs has been a challenge which has led to the design of many approaches.

2. STORAGE SPECIFIC QUALITY OF SERVICE SOLUTIONS

2.1 Stonehenge

QOS is essential in mixed environment where various users with different levels of priorities and preferences are accessing the storage systems simultaneously.it is important to ensure that critical tasks get satisfying performance given limited resources.[8] Developed Stonehenge to solve the issues of storage scalability, manageability and quality of service. Stonehenge is built on IP networks IDE hard drives, IDE controllers and off-the shelf low end personal computers. To implement QOS Stonehenge dedicates a set of storage servers to manage disk arrays and single personal computers to perform the controlling functions such as storage reservation and run-time management [7].

[1]Developed pClock based on arrival curves that capture the bandwidth and burst requirements of applications. When implemented pClock showed efficient performance isolation and burst handling.it also showed an ability to allocate spare capacity to either speed up some applications. When a request arrives the pClock algorithm performs three functions; updating the number of tokens, checking and adjusting tags and computing the tags. The update number of tokens function updates the arrival upper bound function for the present arrival time while the check adjust tags is used to-resynchronize flows to avoid starvation and the compute tags assigns start and finish tags. The pClock algorithm allows multiple workloads to share storage, with each workload receiving the level of service it requires. PClock allows each workload to specify its throughput, burst size and desired latency [1] [8].

The pClock algorithm is as follows:-

Request arrival:

Let t be arrival time of request r from fi;

Update Numtokens();

CheckandAdjustTags();

ComputeTags();

Request scheduling:

Choose the request w with minimum finish tag *fjw* and dispatch to the server

Let the chosen request be from flow fk with start tag swk;

Minsk=sk; [5].

2.2.1 updateTokens

In order to assign tags the arrival upper bound function Uia() to the current time t.it maintains a variable numtokens for each flow fi.

This means that the difference Uia(t)-Ri(0,t) is the difference between AUB at time t and the cumulative number of arrivals up to that time. The value obtained indicates the number of requests that can be made by fi at t without violating the arrival constraints.

Hence when [9];

Uia(t)-Ri(0,t)<1, means that a well behaved flow cannot make any request at t.

2.2.2 computeTags

This function assigns start and finish tags (Sir + Fir) to the request r from fi arriving at time t.The value assigned to the start tag Sir depends on whether the request is within the AUB or exceeds it.When numtokensi>=1, Sir is set to the current time t.If the total number of requests made by fi through time t exceeds AUB(numtokensi<1),the start tag will be assigned a future time greater than t.In particular the start tag is set to the time it would have taken a well behaved flow to send a number of requests[2].

Pclock guarantees that the well behaved flows are not missed and the requests of the background jobs are done in batches, which can lead to better disk utilization since many background jobs tend to be sequential [8]. The algorithm has the ability to allocate spare capacity to the workloads or to the background jobs. The algorithm is also lightweight to implement and efficient to execute. However it does not offer control of how QOS mechanisms interact with storage devices [7].

2.3 Argon

The argon storage server explicitly manages its resources to bind the inefficiency arising from interservice disk and cache interference in traditional systems. The goal is to provide each service with at least a configured fraction of the throughput it achieves when the storage server to itself. Argon uses automatically configured prefetch/write back sizes to insulate streaming efficiency from disk seeks introduced by competing workloads. Argon uses prefetching and write back aggregation as a tool for performance insulation [4] [6].

Argon adapts, extends and applies some existing mechanisms to provide performance insulation for shared storage servers. Many operating systems such as eclipse operating system use time slicing of disk head time to achieve performance insulation. Argon goes beyond this approach by automatically determining the lengths of time slices required and by adding appropriate and automatically configured cache partitioning and prefetch/write back [8].

Argon uses QOS aware disk scheduler in place of strict time slicing, for workloads whose access patterns would not interfere when combined.to implement fairness or weighted fair sharing between workloads argon uses amortization cache partitioning and quanta based scheduling. Assumes that network bandwidth and CPU time has no effect on efficiency [9]. To achieve complete isolation argon does not allow requests from different workloads to be mixed, instead it uses a strict quanta based scheduling .This ensures that each client gets exclusive access to the disk during a scheduling quantum which avoids starvation because active clients quanta are scheduled in a round robin manner[5].

Traditional disk and cache management allow interference among services access patterns to significantly reduce efficiency [7]. Argon combines and automatically configures prefetch/write back cache partitioning and quanta based disk time scheduling to provide each service with a configurable fraction of efficiency it would receive without competition. This increases both efficiency and predictability when services share storage server [4]

However as with all other storage specific solutions Argon runs on the storage device itself which requires multiple instances of it to be implemented in all the devices. This increases overhead and CPU time. Again since there is no centralized management of QOS when the storage data is in transit from the source to destination QOS is not taken care of. The argon design also assumes that bandwidth is not a factor in QOS however with IP SANs bandwidth management is very important since the storage data will be moving from source to destination via IP network [6].

2.4 Facade

[3]Developed Façade as a dynamic storage controller for controlling multiple input/output streams going to a shared storage device and to ensure that each of the input/output streams receives a performance specified by its service level objective. Façade provides performance guarantees in highly volatile scenario. To achieve QOS Façade is implemented as a virtual store controller that sits between hosts and storage devices in the network, and throttles individual input/output requests from multiple clients so that devices do not saturate [2].

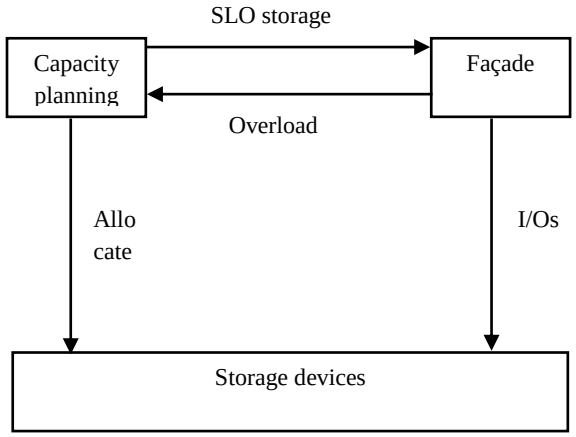

Figure: Facade structure [3]

The capacity planner allocates storage for each workload on the storage device and ensures that the device has adequate

capacity and bandwidth to meet the aggregate demands of the workloads assigned to it.The allocation is adjusted depending on the workload. Requests arriving at façade are queued in per workload input queues.to determine which requests are admitted to the storage devices façade relies on three components that is the I/O scheduler, statistics monitor and controller [8].

The I/O scheduler maintains a target queue depth value and per workload latency target which it tries to meet using earliest deadline first (EDF) scheduling. The deadline for a request from a workload WK is arrivalTime (WK) + latenctyTarget (Wk), where arrivalTime (WK) is its arrivalTime and latencyTarget (WK) is a target supplied for WK by the controller. Requests are admitted into the devices in two cases; if the device queue depth is now less than the current queue length target or if the deadline for any workload is already past. The intent of controlling queue depth is to allow workloads with low latency requirements to satisfy their SLOs [3].

2.4.1 Statistics monitor

This receives I/O arrivals and completions.it reports the completions to the I/O scheduler and also computes the average latency and read and write request arrival rates for active workloads every P seconds and reports them to the controller [10].

2.4.2 Controller

The controller adjusts the target workload latencies and the target device queue length. Target workload latencies must be adjusted because the workload request rates vary and therefore it is necessary to give those requests a different latency based on the workload SLO.The device queue depth must also be adjusted to meet the varying workload requirements[8] .The controller tries to keep the queue as full as possible to enhance device utilization. However this increases the latency. This means when any.

Workload demands a low latency, the controller reduces the target queue depth. The controller uses the I/O statistics it receives from the monitor every P seconds to compute a new latency target based on the SLO for each workload as follows;

Let the SLO for WK be ((r,tr1,tw1),(r2,tr2,tw2),…,(rn,trn,twn)) with a window w and the fraction of reads reported is fr.

Let r0=0, rn+1=∞, trn+1=twn+1=∞ then latencyTarget (WK) =trifn+twi (1-fr)

If ri-1<=readRate (WK) + writeRate (WK) <ri [7].

Facade is able to efficiently utilize resources and balance the load among multiple backend devices while satisfying the performance requirement of many different client applications. Facade is also able to adopt to workloads whose performance requirements change overtime. However façade cannot handle large workloads. This is because multiple instance of façade that are in every storage device cannot be able to cooperate in order to handle large workloads [3].

2.5 PARDA

PARDA enforces proportional share fairness among distributed hosts accessing a storage array without assuming any support from the array itself.PARDA uses latency measurements to detect overload and adjust issue queue lengths to provide fairness [7]. Numerous algorithms for network QOS have been proposed, including many variants of fair queueing.However these approaches are suitable only in centralized setting where a single controller manages all requests for resources [2].

3. Discussion and Conclusion

PARDA enforces proportional fairness among distributed hosts accessing a storage array, without assuming any support from the array itself. PARDA uses latency measurements to detect overload and adjust issue queue lengths to provide fairness. However these technique require each storage device to run an instance of the algorithm which results in overhead caused by running the algorithm. Facade provides performance guarantees by throttling individual input/output requests from multiple clients so that devices do not saturate. Facade provides performance isolation in that the performance experienced by the workload from a given customer must not suffer because of variations on the workloads from other customers. Façade is able to use resources much more efficiently and to balance the load among multiple back end devices while satisfying the performance requirements of many different client applications. However it cannot handle well large workloads. It also requires multiple instances of the same algorithm to run in all storage devices.

Stonehenge was developed as a technique for providing QOS guarantees in storage area networks. All the above techniques require that multiple instances of the same algorithm runs on every storage device. These increases overhead which is caused by the processing of the algorithms. These techniques are implemented on the storage device and therefore so do not provide service guarantees when storage traffic is traversing the network which is important since with IP SAN storage traffic will be interacting with other traffic in the network.

4. ACKNOWLEDGMENTS

My thanks goes to all my friends who have contributed towards development of the paper.

5. REFERENCES

[1] Gulati, A., Merchant, A., & Varman, P. J. (2007). p Clock : An Arrival Curve Based Approach For QoS Guarantees In Shared Storage Systems. ACM.

[2] Gulati, A., & Waldspurger, C. A. (2007). PARDA : Proportional Allocation of Resources for Distributed Storage Access. In 7th USENIX Conference on File and Storage Technologie (pp. 85–98).

[3] Lumb, C. R., Merchant, A., & Alvarez, G. A. (2003). Fac , ade : virtual storage devices with performance guarantees. In File and Storage Technologies (pp. 131–144).

[4] Wachs, M., Abd-el-malek, M., Thereska, E., & Ganger, G. R. (2007). Argon : performance insulation for shared storage servers. In File and Storage Technologies (pp. 61–76).

[5] A. Gulati and I. Ahmad(2008). Towards distributed storage resource management using flow control. ACM SIGOPS Operating Systems Review, 42(6):pp 10–16.

[6] Bjørgeengen, J. (2010). Using TCP / IP traffic shaping to achieve iSCSI service predictability. In Proceedings of LISA '10: 24th Large Installation System Administration Conference (pp. 91–107).

[7] Traver, L, Tarin, C, Cardona, N.(2009)Bandwidth Resource Management for Neural Signal Telemetry, Information Technology in Biomedicine, IEEE , Vol.13, no.6(December), pp.1083-1084.

[8] M.Wachs, M. Abd-El-Malek, E. Thereska, and G.R. Ganger (2007). Argon: performance insulation for shared storage servers. In Proceedings of the 5th USENIX conference on File and Storage Technologies,pages 5–5. USENIX Association.

[9] Van der Stok,P, D. Jarnikov,D,Kozlov, S,van Hartskamp,M & Lukkien,J.J(2009)Hierarchical Resource Allocation for Robust In-Home Video Streaming, Journal of Systems and Software.Vol. 80, no. 7(February), pp. 951–961.

[10] Peter, M.O, and Babatunde, P.J. (2012) Software Prototyping: A Strategy to Use When User Lacks Data Processing Experience, ARPN Journal of Systems and Software.Vol.2,No.6(June),pp 219-223

Table1.Comparison of storage specific quality of service solutions

	QOS measures				
QOS solutions	**Performance isolation**	**Burst handling**	**Bandwidth sharing**	**Centralized control of QOS**	**Control latency**
Argon	✓	×	✓	×	✓
Façade	✓	×	✓	×	✓
Stonehenge	✓	×	✓	×	✓
PARDA	✓	✓	✓	×	✓
pClock	✓	✓	✓	×	✓