An Analysis of QoS specific Coherence Issues in Distributed Networks

Aaqif Afzaal Abbasi[1] and Andreea Florentina[2]

[1]National University of Sciences and Technology (NUST), Islamabad, Pakistan
[2]Academia de Studii Economice (ASE), Bucureşti, Romania
aaqif@nust.edu.pk, besliuandreea06@stud.ase.ro

Abstract

Distributed Systems remained one of the most recent development fields in computing research. The parallel applications development faced enormous hindrance in QoS delivery due to resource hungry queuing models and complex memory patterns. Being a collection of multiple autonomous systems, a distributed system's efficiency lies in best coordination of its middle ware and coordinating modular layers. The concurrency is maintained by implementing Integrity constraint rules. The paper assesses various coherence addressing schemes for QoS delivery in Distributed Network environment.

Keywords: *Distributed Systems, Quality of Service, Computer Networks, Coherence, Bandwidth, Cache*

1. Introduction

Distributed Systems mainly focus on solving complex problems by making the optimum use of parallel processing, introduced in the multi-core processors. They share data across boundaries/ nodes transparently where each memory location can be address directly [15].

A significant expansion has been noticed in Distributed System's models, frameworks, architectures and supportive hardware structures. There is a general trend of using internodes cooperation for improving performance of parallel and distributed file systems. Cooperative caching is a good example of this concept [16].

The performance up gradation has always focused on determining a reliable solution to the coherence and memory/ buffer management techniques. The cache coherence and fault tolerance has seen an enormous change in the recent decades. The paper mainly lays focus on various cache and coherence management techniques in a Distributed System. It helps in evaluating a balanced overview of structures, used techniques and schemas.

2. Literature Review

A distributed system is a combination of several autonomous systems, sharing memory resources for accomplishment of complex query operations. The section briefly discusses the coherence issues in distributed networked environments and the work performed on them in detail;

In [1], authors discussed cache coherence in centralized shared memory and distributed shared memory architectures. In centralized shared-memory architectures all processors communicate with the same central memory. This kind of architecture is very useful for multiprocessor workstations such as Sun Ultra-SPARC workstations. For large arrays of multiprocessors a distributed shared memory approach is more suitable as shown in Figure 1.

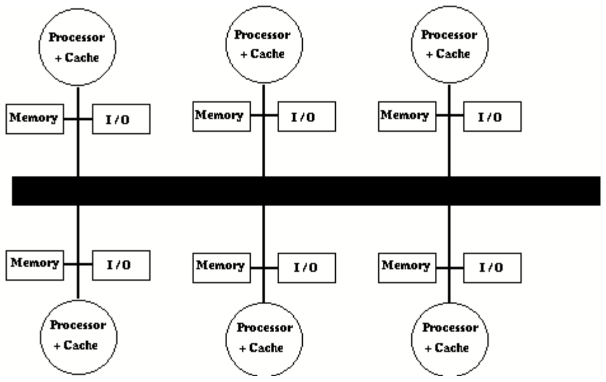

Figure 1. Distributed Shared Memory Architecture from [1]

Significant reduction with the problem of memory bandwidth can be resolved by inclusion of large caches with processors. Caching of shared data introduces a cache coherence problem. In computing, cache coherence (also cache coherency) refers to the consistency of data stored in local caches of a shared resource. The protocols to maintain coherence for multiple processors are called cache – coherence protocols. Snooping Protocol is used for centralized shared memory architectures. The caches are usually on a shared – memory bus and all cache controllers monitor or snoop on the bus to determine whether or not they have a copy of a block that is requested on the bus. This protocol is useful for small systems as shown in Figure 2.

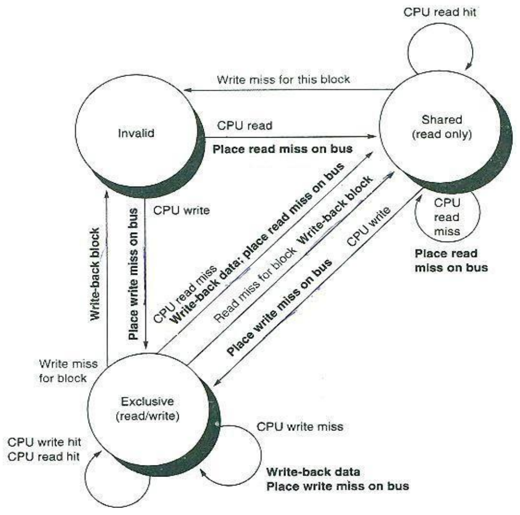

Figure 2. Cache – Coherence State Transition Diagram with the State Transitions Induced by the Local Processor and the Bus Activities from [1]

The proposed distributed memory framework scales well where the shared memory is easy to program and the shared memory is distributed logically. It is Page based and shares pages through shared pages, by demanding paging between nodes. The distributed memory is difficult to program and shared memory is difficult to build, whereas the snooping protocols are not scalable and Directories on the other hand tend to have longer latencies.

In [2], a common exponent in most of the parallel I/O and parallel/distributed file systems is the idea of cooperation between nodes in order to achieve a better system performance –

cooperative caches, where all nodes work together in order to make a global cache -. This way the cache size and the hit ratio are increased and the file system performance is improved. Cooperation between nodes raises a very important problem: maintaining the shared information coherent. A simple and efficient solution to this problem is PAFS –a parallel/distributed file system designed to work in a parallel machine or network of workstation. Each node in the network runs a micro-kernel operating system and all services are handled by user-level servers. More than that, each node may have none, one or even several discs connected to it.

In [3], cooperative caching cuts as a inter-node cooperation example aimed at parallel and distributed file system performance improvement. Here cooperation between nodes raises design problems that need to be addressed. There are different approaches to solve the afore-mentioned problem: i.e. through a cache that avoids replication, or to distribute the cache buffers among the servers in such a way they are able to adapt load variations. To increase write operation's performance, delayed-write policy is used. The proposed cache has the advantage of cooperation and avoids the problems derived from the coherence mechanism. One of the main ideas presented in this paper is that exploiting the local-hit ratio is not the only way to achieve a high-performance cooperative cache. In order to see whether the dynamic redistribution is really needed, a Fixed-partition policy where the buffers were assigned at booting timestamps was evaluated. The paper also simulated two different workloads through two basic architectures. The influence of bandwidth rate was measured. On the X axis the ratio between the local-memory bandwidth (L-BW) and the interconnection-network bandwidth (R-BW) were evaluated as shown in Figure 3 from [3].

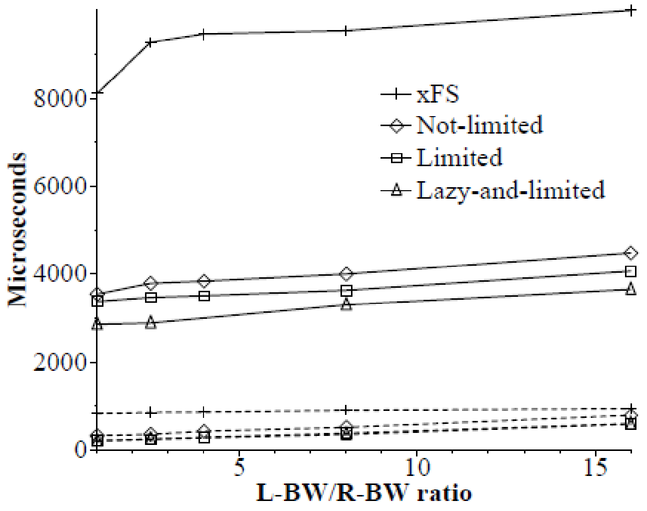

Figure 3. Network-bandwidth Influence from [3]

The coherence avoidance problem not only simplifies the file-system design but also increases system performance. The described redistribution policies should neither be too aggressive nor too conservative and the parity-based fault-tolerance mechanism might obtain a high level of fault-tolerance as it allows a high-performance file system.

In [4], Coherence decoupling breaks a traditional cache coherence protocol into two protocols: a Speculative Cache Lookup (SCL) protocol and a safe, backing coherence protocol. The SCL protocol produces a speculative load value, typically from an invalid cache line, permitting the processor to compute with incoherent data. The result demonstrated that coherence misses are a significant fraction of total L2 misses, ranging from 10% to80%, and averaging around 40% for large caches. Coherence decoupling has the potential to hide the miss latency for about 40% to 90% of all coherence misses, miss speculating roughly 20% of the time. The proposed approach, called coherence decoupling, breaks up the cache coherence protocol, which is used to implement coherent inter-processor communication, into a Speculative Cache Lookup (SCL) protocol that returns a speculative value, and a coherence correctness protocol that confirms the correctness of the speculation. A better SCL protocol can be designed for enhanced speculation accuracy with focus on hierarchical multiprocessors.

In [5], a distributed architecture for cooperative spoken dialogue agents with high domain extensibility, different spoken dialogue agents handling different domains was proposed for independent development. While a user interface agent can access the correct spoken dialogue agent through a domain switching protocol, and carry over the dialogue state and history so as to keep the knowledge processed persistently and consistently across different domains as shown in Figure 4 from [5] where Spoken Dialogue Agent (SDA) interacts with the State Dependent Data(SDD).

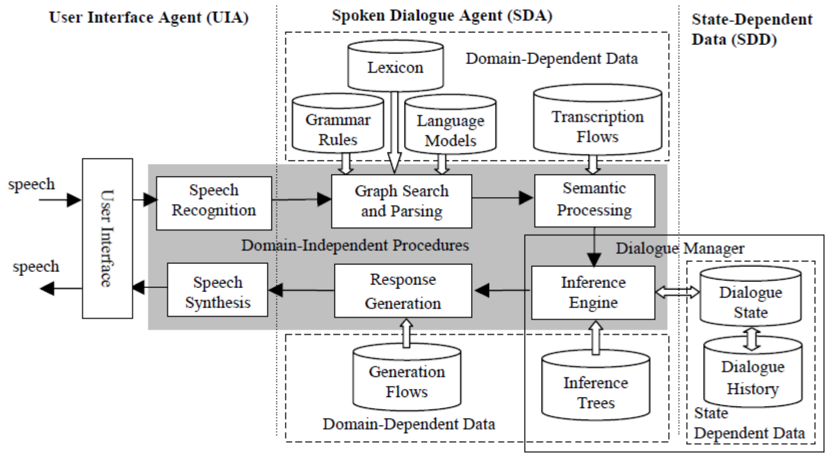

Figure 4. Partition of a Spoken Dialogue System into a User Interface Agent, a Spoken Dialogue Agent, and State Dependent Data from [5]

A dialogue system was developed for train, weather, and bus information services. All the lexicons, grammar rules, language models, and inference trees for the different SDA's were developed independently, and a common module for domain switching control is augmented to each SDA. The user is not aware of the change of the agents during dialogue, because the User Interface Agent (UIA) hides the details of the negotiation and the knowledge integrity maintained by delivering the dialogue state and history across different domain agents make the user feel that he or she dialogues with a single agent. The proposed distributed agent architecture for multi-domain spoken dialogue was used to build a multi-domain spoken dialogue system for travel information services, and very encouraging results were obtained.

In [6], Distributed shared memory offers the power of parallel computing using multiple processors as well as a single system memory view. Consistency in a distributed shared memory system is an important issue because there might be some potential consistency problems when different processors access, cache and update the shared single memory space. To get acceptable performance from a Distributed Shared Memory System, data have to be placed near the processors who are using it. This is done by replicating and replacing data for read and write operations at a number of processors. The write-shared protocol buffers the write accesses thus allows multiple writers update concurrently. The time of creating diff in Lazy Diff Creation Protocol (LDC) is postponed until the modifications are requested, which differs from that of write-shared protocol. Midway is a software-based distributed shared memory system which optimizes the propagation of updates by using Entry Consistency. The communication cost inherited in the underlying network is very expensive, thus limits the scalability of distributed shared memory systems. For write detection, TreadMarks system has to scan the entire shared data region, although only a small portion of it may have been updated. Entry Consistency outperforms Lazy Release Consistency if its coherence unit is larger than a page. In addition to the algorithms, protocols and memory models, new network technologies may play an important role in improving Distributed Shared Memory Systems efficiency.

In [7], the FIFO and LRU policies and others such as LFU were developed for processor or disk level caching. The underlying assumption in these policies is to incur the same amount of delay to be fetched, which is valid for processor or disk level caching. But in the case of web caching this basic assumption does not hold true. The proposed adaptive coherence-replacement mechanism for Web caches is based on system like Squid, which caches Internet data. It does this by accepting request for objects that people want to download and by processing their request at their sites. The model is defined according to the following ideas:

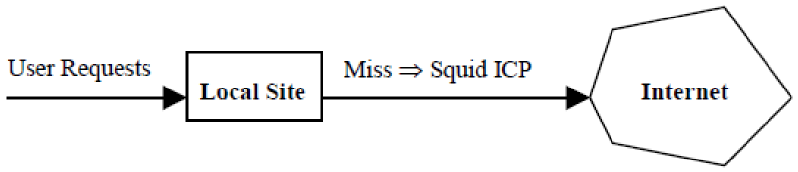

Figure 5. The Replacement-Coherence System from [7]

The coherence-replacement protocol is able to assess the performance of proxy caches. Our approach includes additional information/factors such as frequency of block use, state of the block etc. in replacement decisions. It takes into consideration that coherency and replacement decisions affect each other. This adaptive policy system has been validated by experimental work.

In 8, Shared memory multiprocessor architectures employ cache coherence protocols such as MSI, MESI, and Dragon protocol, to guarantee data integrity and correctness when data are shared and cached within each processor. The paper presents two implementations using commercially available embedded processors: PowerPC755, Write-back Enhanced Intel486 and ARM920T. PowerPC755 uses the MEI protocol, Intel486 supports a modified MESI protocol, and no cache coherence is supported in ARM920T as shown in Figure 6 from [8].

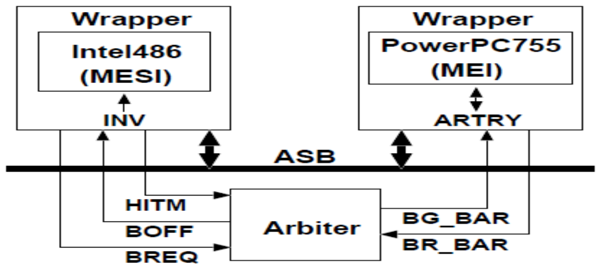

Figure 6. PowerPC755, Intel486 Coherence from [8]

Simulations were performed using a worst-case scenario (WCS), a typical-case scenario (TCS), and a best-case scenario (BCS) microbench programs. In the microbench programs, one task runs on each processor. Each task will try to access a critical section (i.e. shared memory), which is protected by a lock mechanism.The proposed solution shows 136% performance improvement compared to the baseline and better performance than the software solution by at least 2.56% for all WCS simulations. The solution speedup the number of accessed cache lines. Simulation with 32 cache lines shows 58.2% improvement over the software solution. The integrated coherence protocol will at most consist of all the common states from various protocols in a system. In future, authors plan to apply their approach to emerging technologies that tightly integrate between a main processor and specialized I/O processors such as network processors.

In [9], authors implement a logical time system in which all messages appear to travel at the same speed—one unit of logical distance per unit of logical time. Exploiting this capability changes the programming model— instead of using locks or barriers to enforce atomicity, a processor issues batch of requests called isochrons and the memory system executes the requests in each isochron so that they appear to be executed at the same time. Delta protocols enforce SC without limiting concurrency and enforce atomicity and SC using isotach ordering guarantees without the locks and restrictions on pipelining required in conventional systems. They don't require acknowledgments and processors can execute multiple requests atomically without locks. Delta protocols offer a significantly higher level of concurrency than existing coherence protocols, while a prototype isotach network implementation demonstrates that the cost of providing this additional concurrency is low.

In [10], authors support Software Transactional Memory (STM) systems as a powerful paradigm to develop concurrent applications. The proposed architecture for a distributed replicated STM for the FenixEDU system is elaborated below:

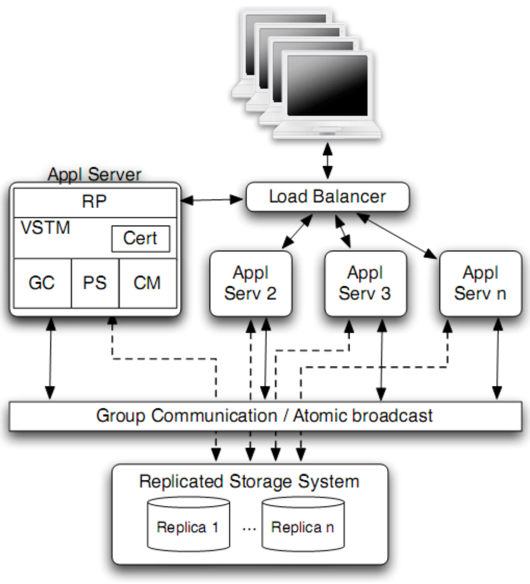

Figure 7. Proposed Architecture of a Distributed Replicated STM for the FenixEDU System from [10]

A transaction is started whenever the application server receives a request from a client, via a load balancer and it may require the retrieval of some objects from the storage system.

Transactions can be validated globally without the need for any other messages to be exchanged. The false positive rate is constant and known and its value may be adjusted for performance gains of reducing the size of the messages broadcast. The most relevant drawback associated with the use of bloom filters is the existence of false positives. The number of hash functions needs to be configured in order not to greatly affect the performance. The work summarizes that the most promising techniques to maintain the consistency of replicated STMs are the certification based protocols, based on totally ordered broadcast primitives.

In [11], authors address the problem of implementing efficient distributed and replicated software transactional memory systems. All the experiments were performed in a cluster of 8 nodes, each one equipped with an Intel QuadCore Q6600 at 2.40GHz with 8 GB of RAM running Linux 2.6.27.7. The nodes are interconnected via a private Gigabit Ethernet. The work introduced a novel distributed certification based protocol which relies on the properties of an Atomic Broadcast primitive to serialize conflicting transactions and uses Bloom filters as a technique to reduce the size of the messages exchanged during the validation phase by compressing the read sets.

In [12], authors introduce a new distributed CMP coherence directory that uses an alternative home to alleviate the hot-home conflict. Virtutech Simics 3.0, a whole-system execution-driven simulator, is used to evaluate an in-order x86 64-core CMP with Simics Micro-architecture Interface (MAI). The set selection within each home for the first three organizations is direct-mapped. The block distribution, the hit/miss improvement, and the invalidation traffic will be presented and compared. The histograms of cached blocks distributed among 64 cores for the five simulated coherence directory organizations are plotted in Figure 8 from [12].

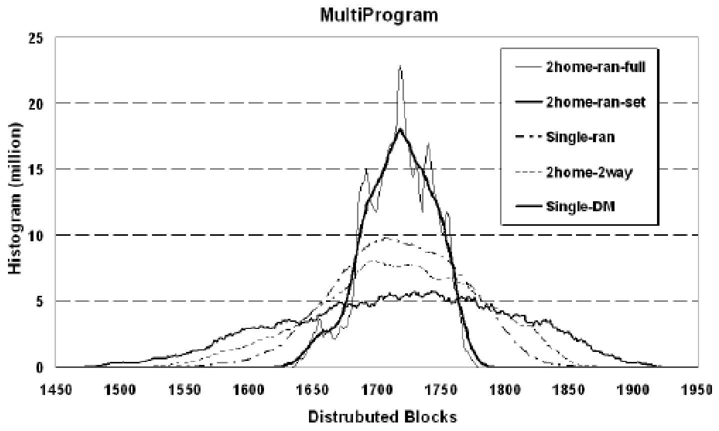

Figure 8. Histogram of Block Distributions of Five Distributed Coherence Directories from [12]

In [13] authors identify that the performance of distributed transactional memory systems is determined by two factors: the contention manager and the cache-coherence protocol used. Authors present a proof of the worst-case competitive ratio of the Greedy contention manager with an arbitrary cache coherence protocol and present location-aware cache-coherence protocols called LAC protocols. The Greedy manager still guarantees the progress of transactions: the priorities of transactions are assigned when they start. At any time, the transaction with the highest priority (the earliest timestamp for the Greedy manager) never waits and is never aborted by a synchronization conflict.

In [14], authors explained the Total Order Broadcast, also known as Atomic Broadcast (an essential building block). In pervasive systems the tolerance to frequent disconnections becomes one of the key performance aspects. Through it, a global coherent view of a data set can be provided, as messages that modify this data set are received by all correct processes and executed in the same order. Although fault-tolerant algorithms are designed to tolerate a certain number of process failures, sooner or later the accumulated number of failed or brutally disconnected processes. The View Synchronous Communication protocol manages the creation and the maintenance of a set of processes during the execution. When evaluating the performance of a Total Order Broadcast, two important metrics are the throughput and the delivery latency. The delivery latency and k-resiliency is shown in Figure 13 from [14].

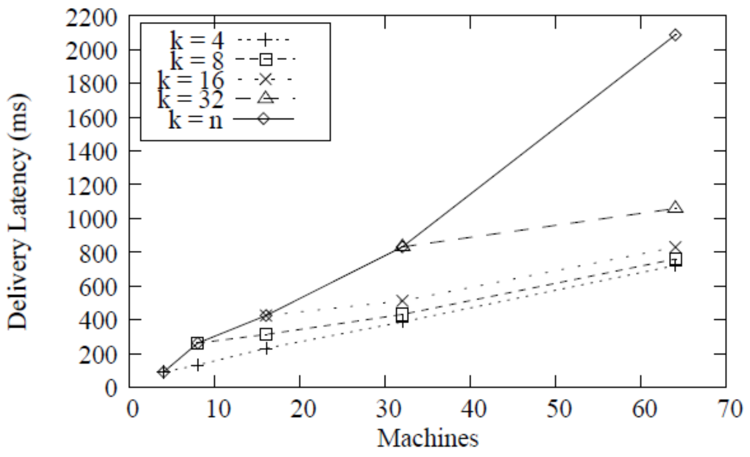

Figure 9. Delivery Latency and k-resiliency from [14]

By limiting the k-resiliency we can bound the delivery latency and by reducing the overhead on the i-view changes, we reduce the impact of wrong suspicions due to aggressive failure detectors. The delivery latencies may represent a problem on a large network. The work addressed in [14] describes the problem of Total Order Broadcast in the context of pervasive distributed systems. Traditional algorithms are not fit for these environments therefore a self-stabilizing distributed solution that can operate in environments subjected to frequent disconnections while providing good performance rates was proposed.

9. Conclusion and Future Work

For the reviews conducted above, we can determine that a distributed shared memory system provides a platform for parallel processing. Though expensive, the coherence, scalability and fault tolerance issues can be resolved to a greater extent. The enhancement in development of parsers and thread management in an operating system varies.

A new definition based on system designs, memory architectures and compiler for multi-architectures can ensure a consistency in all data objects. A new hardware design can be introduced for enhance shared memory efficiency and reduced communication costs.

References

[1] S. Deshpande, P. Ravale and S. Apte, "Cache Coherence in Centralized Shared Memory and Distributed Shared Memory Architectures", International Journal on Computer Science and Engineering (IJCSE), **(2010)**, Special Issue, ISSN: 0975-3397, pp. 39-44.

[2] T. Cortes, S. Girona and J. Labarta, "Avoiding the Cache-Coherence Problem in a Parallel/Distributed File System", Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking, **(1997)** April 28-30, pp. 860-869.

[3] T. Cortes, S. Girona and J. Labarta, "Design issues of a cooperative cache with no coherence problems", Proceedings of the fifth workshop on I/O in parallel and distributed systems, **(1997)** November 17, San Jose, California, USA, pp. 37-46.

[4] J. Huh, J. Chang, D. Burger and G. S. Sohi, "Coherence Decoupling: Making Use of Incoherence", Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'04), **(2004)** October, Boston, MA, USA

[5] B. -s. Lin, H. -m. Wang and L.-S. Lee, "A Distributed Architecture For Cooperative Spoken Dialogue Agents With Coherent Dialogue State And History", Proc. ASRU-99, **(1999)**, http://www.iis.sinica.edu.tw /~whm/publish/papers/asru99.pdf.

[6] C. Chai, "Consistency Issues in Distributed Shared Memory Systems", http://crystal.uta.edu/~kumar/ cse6306/papers/Chingwen.pdf.

[7] J. Aguilar and E. L. Leiss, "A coherence-replacement protocol for web proxy cache systems", Computacion y Sistemas, vol. 8, no. 1, **(2004)**, ISSN 1405-5546, Mexico, pp. 001-014.

[8] T. Suh, D. M. Blough and H. -h. S. Lee, "Supporting Cache Coherence in Heterogeneous Multiprocessor Systems", Proceedings of the Design Automation and Test in Europe, **(2003)**.

[9] C. Williams, P. F. Reynolds, B. R. de Supinski, "Delta Coherence Protocols", IEEE Concurrency, vol. 8 no. 3, pp. 23-29.

[10] M. I. C. Couceiro, "Cache Coherence in Distributed and Replicated Transactional Memory Systems", http://www.gsd.inesc-id.pt/~ler/reports/mariacouceiro-midterm.pdf.

[11] M. I. C. Couceiro, "Cache Coherence in Distributed and Replicated Transactional Memory Systems", http://www.gsd.inesc-id.pt/~ler/reports/mariacouceiromsc.pdf.

[12] Z. Huang, X. Shi, Y. Xia and J.-K. Peir, "Alternative Home: Balancing Distributed CMP Coherence Directory", In CMP-MSI: 2nd Workshop on Chip Multiprocessor Memory Systems and Interconnects, Beijing, China, **(2008)** June 22.

[13] B. Zhang, B. Ravindran, "Location-Aware Cache-Coherence Protocols for Distributed Transactional Contention Management in Metric-Space Networks", Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems, **(2009)** September 27-30, pp. 268-277.

[14] L. A. Steffenel, M. K. Pinheiro and Y. Bebers, "Total order broadcast on pervasive sytems", Proceedings of the 2008 ACM symposium on Applied computing (SAC'08), **(2008)**, ISBN: 978-1-59593-753-7, New York, NY, USA, pp. 2202-2206.

[15] K. Li and P. Hudak, "Memory Coherence in Shared Virtual Memory Systems", ACM Transactions on Computer Systems, vol. 7, no. 4.

[16] M. D. Dahlin, R. Y. Wang, T. E. Anderson and D. A. Patterson, "Cooperative caching: Using remote client memory to improve file system performance", 1st International Symposium on Operating System Design and Implementation, vol. 194, pp. 267-280.

Authors

Aaqif Afzaal Abbasi holds Bachelor (Honors) and Master Degrees in Information Technology. He has experience in IT Management, Operations and Research. He is member of Research Faculty at National University of Sciences and Technology (NUST), Islamabad Pakistan. His research interests include Computer Networks, QoS in networks and Information security.

(Email: aaqif@nust.edu.pk)

Andreea Florentina graduated at Facultatea de Cibernetică, with Informatică Economică (IE) as major from Statistică şi Informatică Economică (CSIE), Academia de Studii Economice (ASE), Bucureşti, Romania. She has worked in Colegiul Naţional "Mihai Viteazul", Bucureşti and Business Soft, Romania.

(Email: besliuandreea06@stud.ase.ro)