Designing Efficient Systems Services and Primitives for Next-Generation Data-Centers[*]

K. Vaidyanathan S. Narravula P. Balaji D. K. Panda
Department of Computer Science and Engineering
The Ohio State University
{vaidyana, narravul, balaji, panda}@cse.ohio-state.edu

Abstract

Current data-centers lack in efficient support for intelligent services, such as requirements for caching documents and cooperation of caching servers, efficiently monitoring and managing the limited physical resources, load-balancing, controlling overload scenarios, that are becoming a common requirement today. On the other hand, the System Area Network (SAN) technology is making rapid advances during the recent years. Besides high performance, these modern interconnects are providing a range of novel features and their support in hardware (e.g., RDMA, atomic operations). In this paper, we extend our previously proposed framework comprising of three layers (communication protocol support, data-center service primitives and advanced data-center services) that work together to tackle the issues associated with existing data-centers. We present the performance results using data-center services such as cooperative caching and active resource monitoring and data-center primitives such as distributed data sharing substrate and distributed lock manager, which demonstrate significant performance benefits achievable by our framework as compared to existing data-centers in several cases.

1 Introduction

There has been an incredible growth of highly data-intensive applications such as medical informatics, genomics and satellite weather image analysis in the recent years which generate multi-terabytes of data. With technology trends, the ability to store and share these datasets is also increasing, allowing scientists to create such large dataset repositories and making them available for use by others, typically through a web-based interface forming web-based data-centers. Such data-centers are not only becoming extremely common today, but are also increasing exponentially in size, currently ranging to several thousands of nodes. With increasing interest in web-based data-centers [16], more and more datasets are being hosted online. Several clients request for either the raw or some kind of processed data simultaneously. However, current data-centers are becoming increasingly incapable of meeting such sky-rocketing processing demands with high-performance and in a flexible and scalable manner.

Current data-centers rely on TCP/IP for communication even within the cluster. The host-based TCP/IP protocols are known to have high latency, low bandwidth, and high CPU utilization limiting the maximum capacity of data-centers. Together with raw performance, data-centers also lack in efficient support for intelligent services, such as requirements for caching documents and cooperation of caching servers, monitoring and managing the limited physical resources, load-balancing, that are becoming a common requirement today. Not only are current data-centers expected to handle these with high-performance, but also in a scalable manner on clusters ranging to thousands of nodes. However, currently there is no mechanism to achieve this.

On the other hand, the System Area Network (SAN) technology is making rapid advances during the recent years. SAN interconnects such as InfiniBand (IBA) [2] and 10-Gigabit Ethernet (10GigE) [11] have been introduced and are currently gaining momentum for designing high-end data-centers. Besides high performance, these modern interconnects are providing a range of novel features and their support in hardware, e.g., Remote Direct Memory Access (RDMA), Remote Atomic Operations, Offloaded Protocol support and several others.

In our previous work [6], we have shown the capabilities of these current generation SAN technologies in dealing with the limitations of existing data-centers. Specifically, we presented a novel framework comprising of three layers (communication protocol support, data-center service primitives and advanced data-center services) that work together to tackle the issues associated with existing data-centers. In this paper, we extend our work to provide efficient data-center services and primitives to further improve the data-center performance. We also present performance results using data-center system services such as cooperative caching and active resource monitoring and primitives such as *distributed data sharing substrate* and *distributed lock manager*, which demonstrate significant performance benefits achievable by our framework as compared to existing data-centers in several cases.

The remaining part of the paper is organized as follows: Our proposed framework is discussed in detail in Section 2. In Section 3, we briefly describe the initial work on advanced communication protocols and data-center services which was completed last year. This year, we extended our framework in designing efficient data-center primitives and data-center services as discussed in Sections 4 and 5, respectively. We present the discussion and future work in Section 6 and conclude the paper in Section 7.

2 Proposed Framework

To satisfy the needs of the next generation data-center applications, we propose a three-stage research framework for designing data-centers as shown in Figure 2. This framework is aimed to take advantage of the novel features provided by advances in

[*]This research is supported by NSF grant #CNS-0509452 and NSF RI equipment grant #CNS-0403342.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

networking technologies.

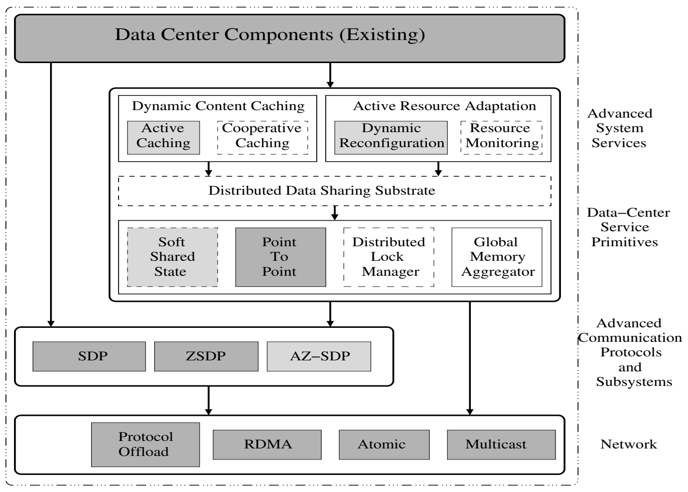


Figure 1. Proposed Framework

The framework is broken down into three layers, namely, communication protocol support, data-center service primitives and advanced data-center services as illustrated in the figure. Broadly, in the figure, all the dark colored boxes are the components which exist today. The light gray colored boxes are the components which we developed last year. The white boxes are the ones which need to be designed to efficiently support next-generation data-center applications. Amongst these, for this paper, we concentrate on the boxes with the *dashed lines* by providing either complete or partial solutions. The boxes with the *solid lines* are aspects which are deferred for future work.

The advanced communication protocols layer aims at *transparently* improving the communication performance of data-center applications by taking advantage of the features provided by modern interconnects. More details about this layer are presented in [3]. The data-center primitives layer takes advantage of the features of modern networks to provide higher-level utilities for data-center applications and services through a distributed data sharing substrate. For the most efficient design of the higher-level data-center services, several primitives such as soft shared state, enhanced point-to-point communication, distributed lock manager, and global memory aggregator are necessary. In this paper, however, we limit our study to only the distributed data sharing substrate, soft shared state primitive and the distributed lock manager as described in Section 4. The advanced data-center services such as active caching, cooperative caching, dynamic reconfiguration and resource monitoring are intelligent services that are critical for the efficient functioning of data-centers. In Section 5, we discuss two of these services: (i) cooperative caching and (ii) active resource monitoring.

3 Overview of Initial Work

In this section, we present our initial work on advanced communication protocols and services such as active caching and dynamic reconfiguration which was completed last year.

As discussed earlier, several data-center applications have traditionally relied on TCP/IP sockets for communication. Thus, a communication protocol stack that creates a pseudo sockets-like interface can allow such applications to transparently utilize the capabilities of the network. In [5], we had demonstrated the potential of one such stack, namely the Sockets Direct Protocol (SDP), for data-center applications. In [3], we proposed a new design for SDP, namely Asynchronous Zero-copy SDP (AZ-SDP), to extend the capabilities of the stack by allowing for better network utilization using asynchronous zero-copy communication. To transparently provide asynchronous communication capabilities for the widely sockets, two goals need to be met: (i) the interface should not change; the application can still use the same interface as earlier, i.e., the synchronous interface and (ii) the application can assume the synchronous sockets semantics, i.e., once the control returns from the communication call, it can read or write from/to the communication buffer. In AZ-SDP, the key idea in meeting these design goals is to memory-protect the user buffer (thus disallow the application from accessing it) and to carry out communication asynchronously from this buffer, while *tricking* the application into believing that we are carrying out data communication in a synchronous manner.

As mentioned earlier, the size, scale and complexity of data-centers is increased tremendously in recent years and hence has necessitated complex caching schemes to scale the performance correspondingly. In our previous work [12], we presented a complete architecture to support strong cache coherency for dynamic content caches. Our architecture, which is based primarily on the shared state primitive using one sided RDMA operations, is designed to handle caching of responses composed of multiple dynamic dependencies. We propose a complete architecture to handle two issues: (i) caching documents with multiple dependencies and (ii) being resilient to load on servers. In our work, we have explored mechanisms for maintaining necessary information on the application servers to achieve the above objectives.

Active resource adaptation services help in improving the utilization of the nodes in the data-center by dynamically reallocating the resources based on system load and traffic pattern. In [7], we had shown the strong potential of using the advanced features of high-speed networks in designing such services. In [4], we extended the service to support Quality of Service and prioritization for different websites hosted in the data-center. In our approach, we used the RDMA and atomic operations for providing efficient active resource adaptation service. Some of the other major design challenges and issues we solved in [7] are: (i) concurrency control to avoid live-locks and starvation, (ii) avoiding server thrashing through history aware reconfiguration and (iii) tuning the reconfigurability module sensitivity.

4 Data-Center Service Primitives

While all the system-level primitives are important for the efficient functionality of the advanced data-center services, in this paper, we limit our scope to the discussion of soft shared state and distributed lock manager.

4.1 Distributed Data Sharing Substrate

Several higher level services such as active resource adaptation, caching and resource monitoring use some sort of a shared state for exchanging the system state. Further, some of these services require the data only at periodic intervals and can also sustain staleness in the data; However, other services have strong requirements with coherency, consistency, scalability, etc. In this section, we present a novel distributed data sharing substrate (DDSS) that provides a soft shared state and addresses all the issues mentioned above. In our design, we use RDMA and atomic operations to support a variety of operations such as *get*, *put*, *allocate*, *free*, etc., for reading and writing to the shared state.

Figure 2 shows the various components of DDSS. The IPC management module helps in virtualizing the shared state services to multiple processes in a single node using IPCs. The memory management module takes care of the allocation and release operations. Data placement module allows the shared data to be placed either on local or remote node. Locking services provide basic locks for shared data to avoid multiple processes accessing the data simultaneously. Coherency and Consistency maintenance module support various coherency models such as *strict coherence*, *write coherence*, *read coherence*, *null coherence*, *delta coherence*, *temporal coherence*. To meet the consistency needs of applications, DDSS also supports versioning of cached data and ensures that requests from multiple sites view the data in a consistent manner. More details are discussed in [20].

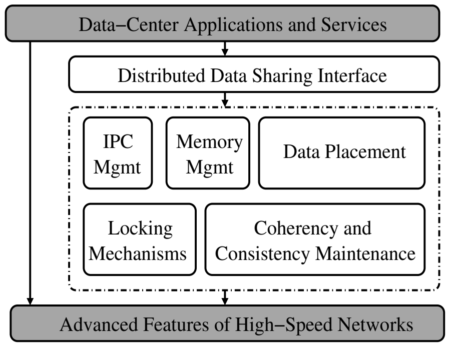

Data−Center Applications and Services
Distributed Data Sharing Interface
IPC Mgmt
Memory Mgmt
Data Placement
Locking Mechanisms
Coherency and Consistency Maintenance
Advanced Features of High−Speed Networks

Figure 2. Proposed Distributed Data Sharing Substrate

Figure 3a shows the *put()* operation latency of different coherence models of DDSS using InfiniBand. We observe that, for all coherence models, the maximum 1-byte latency achieved is only around 55μs. We also observe close to 19% improvement with distributed STORM in comparison with traditional implementations, as shown in Figure 3b. Detailed performance results are presented in [20].

4.2 Distributed Lock Manager

In this section, we present the design details of a novel distributed lock manager based on advanced network features. Our algorithm enables the use of one sided RDMA operations for both shared and exclusive locking modes.

Recent advances in parallel computing has lead to a deluge of applications that use these computing environments. Efficient cooperation among these parallel processes and applications has necessitated highly efficient distributed locking services. On the other hand, InfiniBand provides remote memory atomic (RMA) operations (compare_and_swap and fetch_and_add) that can be used to atomically operate on a remote memory location. Our distributed lock manager design utilizes these RMA operations to provide one-sided atomic-based locking operations.

Figure 4 shows the two basic scenarios for the locking operations. In our design, for each lock we designate a 64-bit memory window that is used to provide the primary location for lock's current state. We divide this 64-bit window into two 32-bit parts that are used to store the following: (i) the first 32-bits store the location of the tail of the current distributed queue of the nodes that requested an exclusive lock and (ii) the second 32-bit store the number of shared lock requests received after the enqueuing of the last exclusive request. The RMA operation used for enqueuing exclusive requests is *compare_and_swap* and the RMA operation used for shared locks is *fetch_and_add*. Figure 4 shows the scenarios for shared only and exclusive only locking scenarios. Further details are described in [14].

Our results shown in Figure 5 present the basic performance improvement that our scheme (*N-CoShED*) shows over existing schemes: (i) basic Distributed Queue based Non-shared Locking (*DQNL*) [10] and (ii) traditional Send/Receive-based Server Locking (*SRSL*). *N-CoShED* scheme shows 39% improvement over the *SRSL* scheme. We also observe a significant (up to 317% for 16 nodes) improvement over the *DQNL* scheme. Further performance results are presented in [14].

5 Data-Center Systems Services

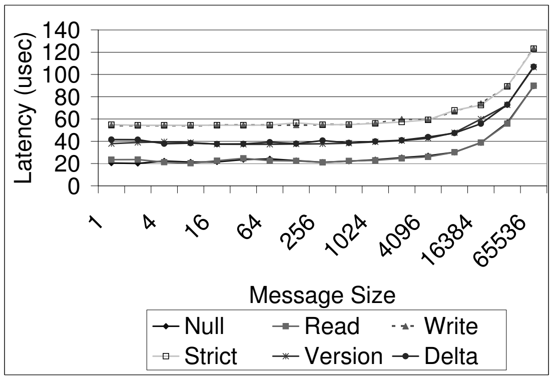
As mentioned earlier, multi-tier data-centers need efficient support for many higher level services for efficient functioning of the data-center. In this section, we present two such services: (i) cooperative caching and (ii) resource monitoring.

5.1 Co-operative Caching

Caching has been a very important technique in improving the performance and scalability of web-serving data-centers. The research community has long proposed cooperation of caching servers to achieve higher performance benefits. Many of the existing cooperative caching mechanisms often partially duplicate the cached data redundantly on multiple servers for higher performance (by optimizing the data-fetch costs for multiple similar requests) [15, 8]. With the advent of RDMA enabled interconnects, these basic data-fetch cost estimates have changed significantly. Further, the effective utilization of the vast resources available across multiple tiers in today's data-centers is of obvious interest.

In the following, we briefly describe our designs [13] that provide highly efficient cooperative caching schemes using network based RDMA operations while controlling the amount of duplicated content in the data-center to maximize the resource utilization. We design the following three schemes: (i) Basic RDMA based Cooperative Cache (*BCC*) - a basic scheme for aggregating cache across the caching nodes using RDMA for data transfers, (ii) Cooperative Cache Without Redundancy (*CCWR*) - a caching scheme in which duplicate copies are eliminated across all the caching nodes and (iii) Multi-Tier Aggregate Cooperative Cache (*MTACC*) - a caching scheme in which we have memory aggregated (for caching) from multiple tiers of a data-center. Based on our study, we further propose a Hybrid Cooperative Caching Scheme (*HYBCC*) that uses multiple of the above schemes based on a method that can achieve the best possible performance. Additional details are available in [13].

Figure 6 shows the performance results of our schemes (Scheme *AC* denotes basic Apache Cache). The throughput improves up to 35% for certain cases for our advanced schemes over the RDMA based simple cooperative caching schemes and



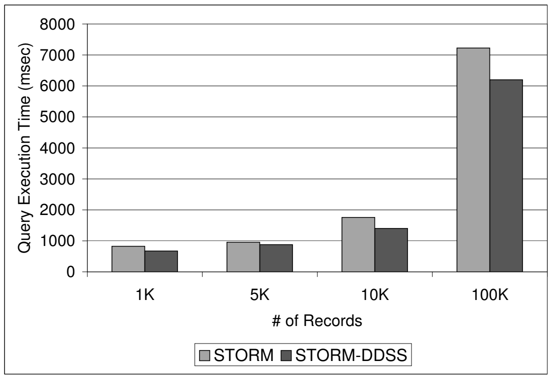
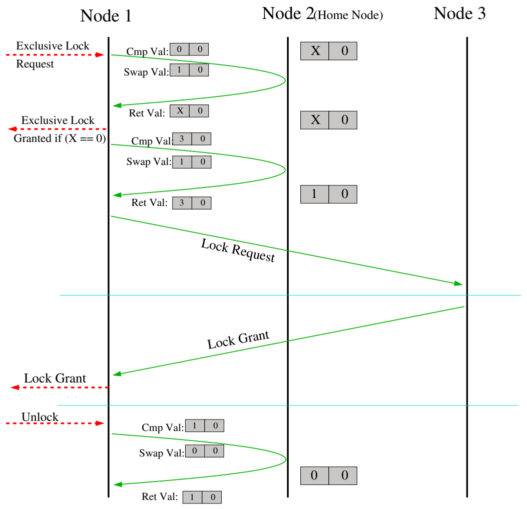


Figure 3. Distributed Data Sharing Substrate Performance: (a) ***put*** **Latency and (b) Distributed STORM**



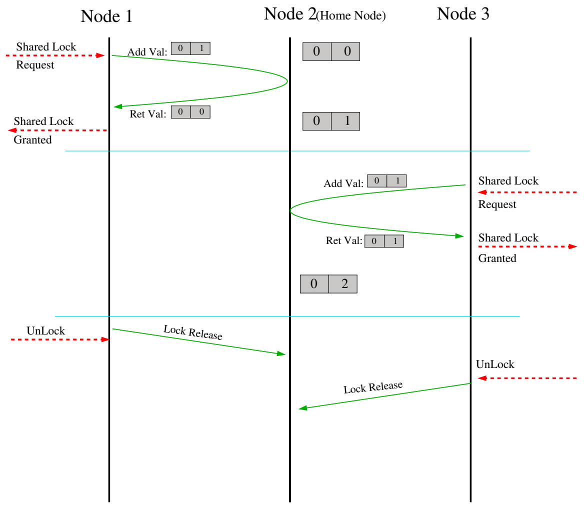
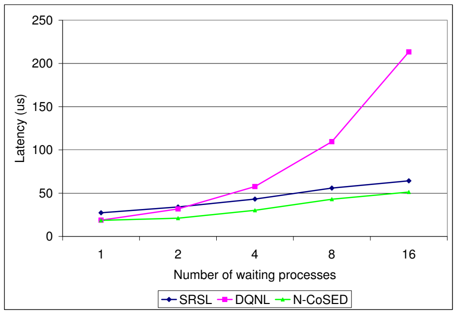


Figure 4. Locking protocols: (a) Exclusive only and (b) Shared Only



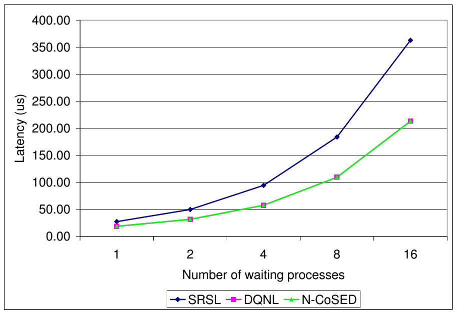
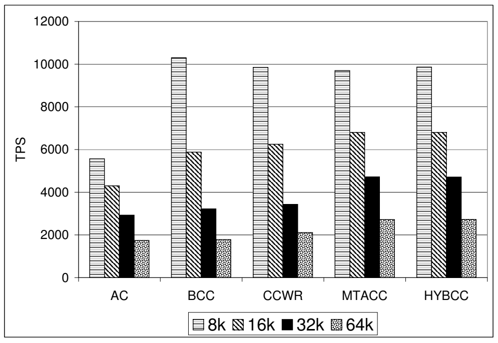


Figure 5. Lock Cascading Latency: (a) Shared Lock Queue and (b) Exclusive Lock Queue



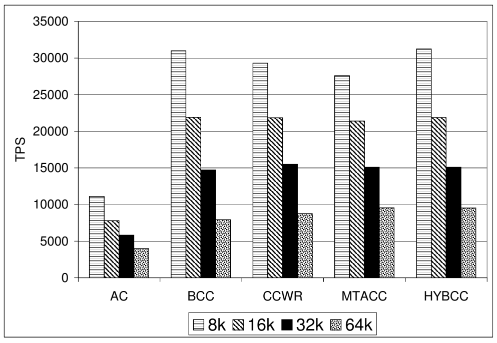


Figure 6. Data-Center Throughput: (a) Two Proxy Nodes and (b) Eight Proxy Nodes

improves up to 180% over simple caching methods. The figure further shows that our schemes scale well for systems with large working-sets and large files. Further results are available in [13].

5.2 Active Resource Monitoring

Efficiently identifying the amount of resources used in data-center environments has been a critical research issue in the past several years. With the amount of resources used by each application becoming more and more divergent and unpredictable [21, 17], the solution to this problem is becoming increasingly important. Traditionally, several techniques periodically monitor the resources used in the cluster and use this information to make various decisions such as load-balancing, reconfiguration, etc. However, these techniques relied on coarse-grained monitoring in order to avoid the overheads associated with fine-grained resource monitoring. On the other hand, the resource usage of requests is becoming increasingly divergent [9], thus increasing the need for fine-grained monitoring resources.

In this section, we describe our approach [19] for fine-grained resource monitoring in data-center environments. We achieve three broad goals: (i) to get an accurate picture of the current resource usage in data-centers at very high granularity (in the order of milliseconds), (ii) to avoid the overhead of an extra monitoring processes on the node that is being monitored and (iii) to be resilient to loaded conditions in a data-center environment. In our approach we use RDMA operations in kernel space to actively monitor the resource usage of the nodes. As shown in Figure 7, we register the necessary kernel data structures that directly monitor the resource usage and we perform RDMA read operation from the front-end node to capture the current load information. Such a design has two major advantages: (i) it removes the need for an extra process in the back-end server and (ii) it can exploit the detailed resource usage information in kernel space to report accurate load information.

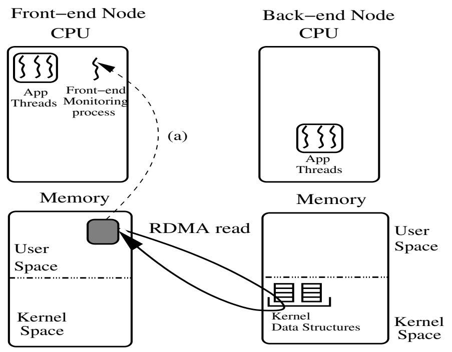


Figure 7. Synchronous RDMA-based Resource Monitoring Mechanism

In our experimental evaluation, we compared our scheme with the traditional sockets-based communication schemes. Figure 8a shows the deviation of the number of threads running on the server for all four schemes in comparison to the actual number of threads running on the remote node. We observe that, RDMA-based schemes report very less or no deviations with the actual number of threads running on the back-end node, thus giving an accurate picture of load on the back-end node. We also measure the performance of a data-center environment hosting two web services: (i) a Zipf trace with varying α value (i.e., higher the α value, higher is the temporal locality of the document accessed) and (ii) RUBiS auction benchmark [1] simulating an e-commerce website developed by Rice University. We observe close to 35% improvement with RDMA-based schemes in comparison with traditional sockets-based implementation.

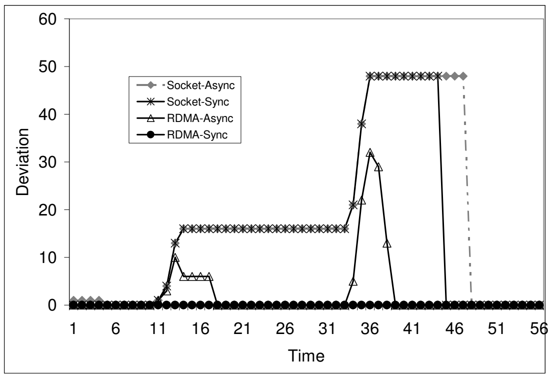
6 Discussion and Work-in-Progress

Our proposed framework mentioned in Section 2 builds multiple layers of efficient designs. Apart from the services mentioned in this paper, these different layers can also be utilized to design other data-center applications and services as needed. More importantly, however, our designs have already been integrated into current data-center applications such Apache, PHP and MySQL. Also, though this work has been done in the context of InfiniBand and 10GigE, our designs rely on quite common features provided by most RDMA-enabled networks and can be easily extended to work with several other networks such as Myrinet, Quadrics etc. In the current context, we intend to focus on several aspects as described in this section.

Like many other communication middleware, SDP utilizes a copy-based approach for transferring small messages. In this approach, the sender copies the data into a temporary buffer and transmits it to a temporary buffer on the receiver side. The receiver copies this data to the final destination buffer. To improve pipelining of data communication, researchers have proposed flow-control mechanisms such as the credit-based flow-control, where the receiver preposts multiple buffers and the sender is given as many credits. Whenever a message is sent, the sender loses a credit; after the data is copied to the final destination buffer, the receiver returns credit to the sender. While this approach is simple, it suffers from significant under-utilization of the temporary buffer and hence loss of performance. For example, let us assume that each temporary buffer is about 8KB large. Now, if the sender sends two 1-byte messages, each of them use up a separate temporary buffer on the receiver side, thus wasting the remaining 99.98% of the buffer space. In packetized flow control, we use RDMA communication to allow the sender to manage both the sender side buffers as well as the receiver side buffers. This allows the sender to pack the transmitted data more precisely and avoid buffer wastage. Preliminary results in this approach demonstrate close to an order of magnitude bandwidth improvement for some message sizes.

Further, resource monitoring services can be combined with the active resource adaptation in providing fine-grained resource dynamism in data-centers. As part of the current study, we have developed a fine-grained active resource adaptation module that uses the resource usage information through resource monitoring schemes. Further, our fine-grained reconfiguration module not only reconfigures the web servers but also other applications such as application servers and database servers. Preliminary evaluations show that the reconfiguration module can achieve an order of magnitude performance benefit compared to existing schemes. Also, we plan to extend the knowledge gained in our previous study [18] in utilizing the remote memory on a file system cache miss to avoid cache corruption.

Several of the challenges and solutions described in the previous few sections are not completely independent. For example,



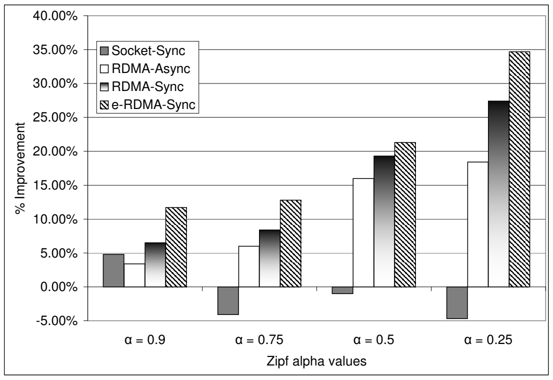


Figure 8. Active Resource Monitoring Performance: (a) Accuracy of Connection Load and (b) Throughput with Zipf Trace

the active resource adaptation schemes focus on reallocating the data-center resources to the varying load. However, blindly reallocating resources might have negative impacts on the proposed caching schemes due to cache corruption that can potentially occur. Thus, each of these designs cannot be evaluated in a standalone fashion, but needs to be seen in an integrated environment. We plan to carry out such integrated evaluation.

7 Concluding Remarks

In this paper, we have extended our previously proposed framework comprising of communication protocol support, data-center primitives and data-center services for addressing two primary drawbacks of current data-centers: (i) low performance due to high communication overheads and (ii) lack of efficient support for advanced features such as caching dynamic data, monitoring and managing limited physical resources, load-balancing and prioritization and QoS mechanisms. Specifically, we focused on data-center systems services such as cooperative caching and active resource monitoring and data-center service primitives such as *distributed data sharing substrate* and *distributed lock manager*. Our experimental results demonstrate that this framework is quite promising in tackling the issues with current and next-generation data-centers and can provide significant performance benefits as compared to existing solutions.

References

[1] RUBiS: Rice University Bidding System. http://rubis.objectweb.org.

[2] Infiniband Trade Association. http://www.infinibandta.org.

[3] P. Balaji, S. Bhagvat, H. W. Jin, and D. K. Panda. Asynchronous Zero-Copy Communication for Synchronous Sockets Direct Protocol (SDP) over InfiniBand. In *CAC 2006; In Conjunction with IPDPS 2006*, 2006.

[4] P. Balaji, S. Narravula, K. Vaidyanathan, H. W. Jin, and D. K. Panda. On the Provision of Prioritization and Soft QoS in Dynamically Reconfigurable Shared Data-Centers over InfiniBand. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2005)*, 2005.

[5] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. Sockets Direct Protocol over InfiniBand in Clusters: Is it Beneficial? In *ISPASS '04*.

[6] P. Balaji, K. Vaidyanathan, S. Narravula, and H. W. Jinand D. K. Panda. Designing Next-Generation Data-Centers with Advanced Communication Protocols and Systems Services. In *Workshop on NSF Next Generation Software(NGS) Program; held in conjuction with IPDPS*, 2006.

[7] P. Balaji, K. Vaidyanathan, S. Narravula, K. Savitha, H. W. Jin, and D. K. Panda. Exploiting Remote Memory Operations to Design Efficient Reconfiguration for Shared Data-Centers. In *RAIT '04*.

[8] Enrique V. Carrera, Srinath Rao, Liviu Iftode, and Ricardo Bianchini. User-level communication in cluster-based servers. In *HPCA*, 2002.

[9] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar andAmin Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, 2001.

[10] A. Devulapalli and P. Wyckoff. Distributed queue based locking using advanced network features. In *ICPP*, 2005.

[11] J. Hurwitz and W. Feng. End-to-End Performance of 10-Gigabit Ethernet on Commodity Systems. *IEEE Micro '04*.

[12] S. Narravula, P. Balaji, K. Vaidyanathan, H. W. Jin, and D. K. Panda. Architecture for Caching Responses with Multiple Dynamic Dependencies in Multi-Tier Data-Centers over InfiniBand. In *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, 2005.

[13] S. Narravula, H.-W. Jin, K. Vaidyanathan, and D. K. Panda. Designing Efficient Cooperative Caching Schemes for Multi-Tier Data-Centers over RDMA–enabled Networks. In *Proceedings of Int'l Symposium on Cluster Computing and the Grid (CCGrid)*, May 2006.

[14] S. Narravula, A. Mamidala, A. Vishnu, K. Vaidyanathan, and D. K. Panda. High Performance Distributed Lock Management Services using Network-based Remote Atomic Operations. In *Proceedings of Int'l Symposium on Cluster Computing and the Grid (CCGrid)*, May 2007.

[15] Vivek S. Pai, Mohit Aron, Gaurav Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich M. Nahum. Locality-aware request distribution in cluster-based network servers. In *Architectural Support for Programming Languages and Operating Systems*, pages 205–216, 1998.

[16] H. V. Shah, D. B. Minturn, A. Foong, G. L. McAlpine, R. S. Madukkarumukumana, and G. J. Regnier. CSP: A Novel System Architecture for Scalable Internet and Communication Services. In *the Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, pages pages 61–72, San Francisco, CA, March 2001.

[17] Weisong Shi, Eli Collins, and Vijay Karamcheti. Modeling Object Characteristics of Dynamic Web Content. *Special Issue on scalable Internet services and architecture of Journal of Parallel and Distributed Computing (JPDC)*, Sept. 2003.

[18] K. Vaidyanathan, P. Balaji, H. W. Jin, and D. K. Panda. Workload-driven Analysis of File Systems in Multi-Tier Data-Centers over InfiniBand. In *Computer Architecture Evaluation with Commercial Workloads (CAECW-8), in conjunction with the International Symposium on High Performance Computer Architecture (HPCA)*, 2005.

[19] K. Vaidyanathan, H. W. Jin, and D. K. Panda. Exploiting RDMA operations for Providing Efficient Fine-Grained Resource Monitoring in Cluster-based Servers. In *Workshop on Remote Direct Memory Access (RDMA): Applications, Implementations, and Technologies (RAIT 2006), in conjunction with Cluster Computing September*, 2006.

[20] K. Vaidyanathan, S. Narravula, and D. K. Panda. DDSS: A Low-Overhead Distributed Data Sharing Substrate for Cluster-Based Data-Centers over Modern Interconnects. In *International Conference on High Performance Computing (HiPC)*, 2006.

[21] J. Yin, L. Alvisi, M. Dahlin, and A. Iyengar. Engineering Web Cache Consistency. *ACM Transactions on Internet Technology, 2:3,*, August. 2002.