Parallel Router Design for Content Centric Networks

Sankaralingam Panneerselvam and Vinod Ramachandran
Computer Sciences Department
University of Wisconsin, Madison, WI
{sankarp,vinorama}@cs.wisc.edu

Abstract

Today's routing infrastructure is built using special purpose hardware due to the demand for very high throughput. With the growth of hardware techniques, satisfying this huge demand is made possible. However the downside of this hardware based architecture is that upgradation is very rare due to high costs and extensibility is limited which heavily limits the deployment of additional features or new routing policies. Recently, Software routers have gained more traction due to it extensible nature. Though their performance is still behind hardware routers, solutions like RouteBricks are trying to close this gap. In order to scale to higher speeds, a parallelized architecture has been proposed where a cluster of commodity machines are used to achieve higher throughput. In this paper, we analyze the generic issues in porting routing policies on to this type of parallel architecture. We illustrate this by parallelizing Content Centric Routing using different designs and explain the key issues with every design and also help in choosing the optimal design based onthe underlying architecture.

1 Introduction

In modern times a plethora of applications are Web based applications and depend heavily on the internet. Since the late 90's we have been seeing many innovative Web based applications. More recently social networking websites like facebook , twitter and streaming websites like youtube have taken the importance of Web based applications to the next level. While using the services of these applications the modern internet user could get impatient on experiencing any delays while using such applications. Therefore it is very important to minimize any latency experienced by the user. Hence it is crucial that routers operate with the greatest efficiency and employ every optimization necessary inorder to minimize latency and to provide for high throughput for all the modern internet applications. They need to be able to process packets at a high speed and minimize any delay experienced by the user.

To scale to increasing demand for bandwidth, early router research emphasized the use of hardware routers. Hardware routers make use of ASIC components to implement important functions like packet forwarding. Functions that were implemented once in software were now ported to hardware. More resources were spent on enhancing the router hardware which led to improved overall system performance.

Hardware routers today operate at a very high rates and provide good throughput. Standard hardware routers start operating from about 1 Tbps and can scale upto 92 Tbps[1]. They make use of a fast crossbar switched backplane and make use of techniques like virtual output queueing to obtain such high operational rates.

Modern Networks are built using special purpose hardware and software. This philosophy of using special purpose hardware has one main limitation. It limits the extent to which one can extend, program or experiment with the routing infrastructure. Any change required to a core router would require a costly upgrade which forces the network providers to use a conservative strategy of making minimal changes. This limitations has forced the researchers to turn back the focus on software based routers.

Software routers can provide for all the functionalities of a hardware router like per-packet protocol processing, route look up and forwarding. The use of software routers has its advantages of low cost, familiar programming environment and large-volume manufacturing. Innovations in commodity hardware drive the need to shift hardware oriented routers to cheaper software routers.

The main challenge with software routers is

achieving high scalability. Modern single server software routers do not scale beyond 1-3 Gbps which is a stark contrast to carrier grade routers that start at 40 Gbps and scale as high as 92 Tbps. This need for increased scalability lead to solutions like RouteBricks[2], a router architecture that parallelizes router functionality[1] across multiple servers and across multiple cores within a single server.

Keeping this requirement of scalability in software routers we decided to study the challenges in achieving thread level parallelism in a software based router.

The initial internet was designed for host to host communication like FTP and Telnet. But cost of the internet applications today works on the basis of data access and service access. This has motivated researchers to consider the need for an architecture focusing primarily on Data. Content Centric Network [3] is one such architecture which motivated us to implement a parallel software router for Content Centric Networking.

We begin by motivating the problem in Section 2 and overview of CCN routing in Section 3. We follow with the different design models in Section 4. We discuss the evaluation results in Section 5, and finish with related work, future work and conclusions.

2 Motivation

The main challenge in the internet routing infrastructure is implementing any new changes to the router'ss core networking behavior. Modern routers provide high throughput in the order of 92Tbps but it is hard to deploy any additional features in them. This drawback has led to a keen interest in software routers. Software routers can run on commodity machines and one of the benefits of using commodity servers is that they are inexpensive and can be upgraded regularly with the development of technology. Developers can pick a language of their choice and use a familiar programming environment to implement their router. Such routers can also be very useful to the research community as they could deploy such routers in commodity PCs in a University environment in order to conduct their experiments. Researchers could use such a router to implement new protocols and study their behavior with much less effort then they have to do now.

However there are issues with software routers. They do not scale as much as modern hardware routers. This performance limitation has motivated many researchers to work on the scalability of software routers. More thought has been put into parallelizing the architecture of software routers. Projects such as RouteBricks focus on converting a software router from being single server based to a cluster based software router. They explain that such an architecture can help improve scalability. Their cluster architecture revolves around using a cluster of N servers in which each server represents an input port. If the individual line rate is R bps then they aim in scaling this cluster architecture to scale to the rate of an individual router having a throughput of O (NR). This interesting idea has received much attention from the research community and has pushed the need for more parallelism in software router design.

Another important project which has garnered significant attention in recent times is PacketShader which is a software router that utilizes the benefits of GPUs to enhance scalability. They route core packet processing to GPUs and make I/O optimizations to obtain maximum parallelism. They state that GPUs reduced memory latencies and provide for greater memory bandwidth which provides for greater throughput. Their router scaled to a throughput of 40Gbps which is the best achieved by a software router so far.

These ideas provide an important motivation for our work. We feel that utilization of thread level parallelism in a software router could be an important design benefit. Many challenges are faced while moving from single server implementation to a parallelized design. One challenge is to maintain a consistent view of routing table entries across multiple threads. We also need to handle locking issues and prevent the occurrence of deadlocks between the threads. One can try many strategies in balancing load across threads and also partition routing tables to increase the concurrency of data accessed by threads for routing purposes. These challenges motivated us in designing a parallel CCN router with the objective of getting a high throughput.

Content Store	
Name	**Data**
/BBC/news	...
/BBC/music	...
Pending Interest Table (PIT)	
Prefix	**Requesting Ports**
/BBC/news	0
/BBC/music	1
Forwarding Information Base	
Prefix	**Port List**
/BBC/news	2, 3, 4
/BBC/music	0, 2

Table 1: **Tables used by CCN Routing**

3 Content Centric Networking

3.1 CCN Overview

Content Centric Networking is motivated from the fact that the modern user of the Internet only cares about the content he wishes to access and not how it is accessed. Yet communication in the Internet is based on "where" and not "what". This aberration in the Internet model has created a few problems. In order to provide availability one has to rely on application specific mechanisms like CDNs and P2P networks, which could lead to greater bandwidth costs. Security is also an issue as it becomes harder to trust content causing more reliance on untrustworthy location and connection information. Network configuration and implementation of network services becomes harder due to location dependence which requires mapping of content to host locations.

Keeping these issues in mind one has to consider replacing "where" with "what". This thought process led to the proposal of Content Centric Networking (CCN)[3]. In CCN one has no notion of a host, packet address names relate to content and not location. One does not have to rely on application specific mechanisms to provide for availability. The security mechanism in CCN is based on content based security. The protection and trust are provided along with the content itself instead of being a property of the connections over which it travels. The location independence provided by CCN also helps in better network configuration and in the easier deployment of services.

This approach of redesigning the data access protocols on the Internet is very interesting and has become a rapidly evolving line of thought. The DONA[7] project at UC Berkley and CCNx[4] project at PARC labs have come out with initial CCN prototypes. CCNx has made its implementation open source and has invited researchers to use it and make extensions to it.

3.2 CCN Routing

In this section we give a brief overview of CCN routing. In CCN there are two main packet types, namely the CCNINTEREST packet and the CCNDATA packet. The CCNINTEREST packet is sent when the user requests for content he is interested in. The CCNDATA packet is a response packet sent by the content provided with the requested content. Both these packets are sent as broadcast messages and neither the users ipaddress nor the content providers ip address is used for the routing of these packets.

Table 1 shows all the routing tables used in a content centric router. When the router receives an Interest packet from a certain input port it does longest prefix match for the content path with the entries of the Forwarding Information Block Table. It then forwards the Interest along these ports. After forwarding the Interest it adds an entry in the Pending Interest Table for the given interest at the given input port. Once the router receives a data packet for the given interest, it forwards the data packet to the input ports present in the Pending Interest Table entry for the given Interest and removes the Pending Interest Entry. It then caches the data packet in the Content Store. In the future if the router receives a request for the given Interest it serves the request from its content store and does not forward the Interest.

4 CCN Parallelized

We came up with three designs of parallel CCN router. The first and second design focuses on exploiting the parallelism in a single multi-core machine whereas the third design targets NUMA architecture and can also be extended to a cluster of machines. In the following subsections, we explain each of the designs in detail focusing on choices made for each model like locality, data partitioning, locking mechanisms etc.

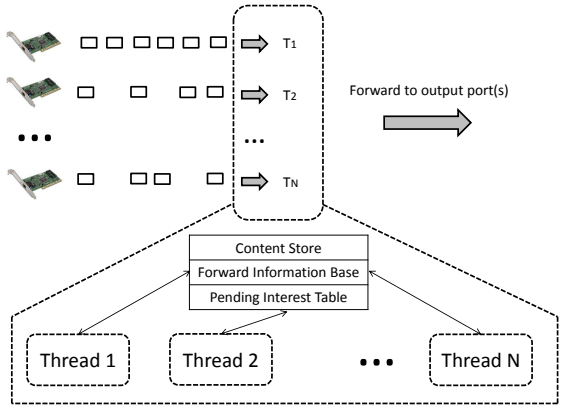

Figure 1: **Parallelized CCN router architecture optmized for single multi-core machine.**

4.1 Lock based Parallel CCN router

As discussed in the previous section on CCN, the router maintains global structures such as Pending Interest table, Forwarding table and content store which act as the cache. The parallelization is brought in by introducing number of threads which are mapped on to the different cores in the system. This is commonly referred as Thread Level Parallelism. Since the router machine accommodates multiple NICs, worker threads are mapped onto separate network cards. If the input arrival rate is higher than the processing capability of each worker thread then multiple threads can be mapped to the same NIC.

Figure 1 represents design of this model. Each worker thread now acts as a separate CCN router by itself. They receive the packets from the network ports and perform routing based on CCN policy. It is to be noted that the structures are present in the shared memory and can be accessed by all worker threads. However these concurrent accesses to the global tables are not thread safe since both the read and write operations are frequent.

The consistency of the global tables is to be ensured through the use of locks. The challenge here lies in choosing the granularity of locks. A single lock to preserve the entire structures would make things easy for the developer but the scalability of the system would be drastically reduced. To enable concurrent access to the tables and also to ensure consistency, we logically divide the tables into multiple partitions. The packets are mapped onto different partition through a simple hash function. The intuition behind the choice of partitioning is that the incoming packets will map onto different partitions with fairly equal probability.

Other techniques like dynamic partitioning [13] can also be employed if load imbalance occurs. It is also to be noted that since the global tables are accessed by all the threads in the system and no locality in the data accessed from the tables, cache conflicts can reduce the scalability of the system. Ensuring locality will limit threads generality in that they can access only certain partitions.

4.2 Transaction based Parallel CCN router

In the world of parallel programming, Transactional Memory has gained much importance due to the combined properties of simple programming style and concurrency benefits of that of fine grained locks. Transactional memory are similar to transactions in Databases where in Atomic, Isolation and Consistency properties are ensured to all the transactions running in the system. All transactions can run Concurrently unless they conflict which is the simultaneous access to same memory location where at least one of the access is a write.

There have been several works on Transactional memory like Intel STM[9], Tiny STM[10] etc. which are software based solutions (STM) and LogTM[11], TCC[12] etc. which are hardware based solutions (HTM). Software solutions are currently available in the market whereas the hardware solutions are not. Though HTM offers significant performance than their counterpart STM, they are not available in the market since they require additional hardware logic or change in coherence protocols. There still has been interest in TM based systems [14, 15] and recent works also shows efforts in getting Transactional memory on commodity systems [14]. This makes us believe that hardware based transactional memory solutions will become prevalent in the near future.

Our second design is based on the transactional memory. It follows the same thread level parallelism based approach as the previous design. The global tables need not be partitioned since transactional memory can allow concurrency at the same level as that of locks on each entry of the tables.

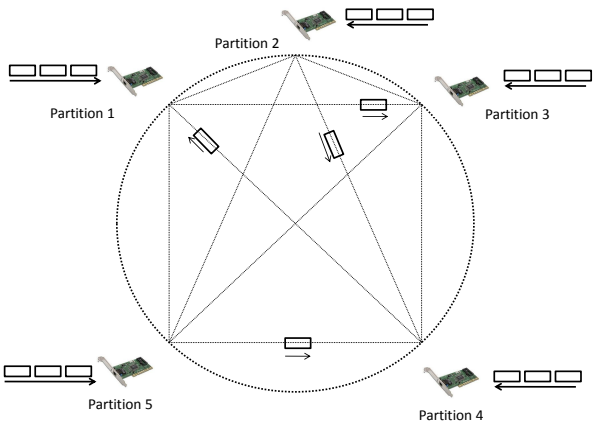

Figure 2: **Partition based CCN router architecture suitable for NUMA or cluster of machines**

However the property of the TM solution like granularity of concurrency (word level granularity or cache line based granularity) has to be considered when designing the global data structures.

4.3 Partition based Parallel CCN router

The above two models are designed for a router that runs on a single multi-core machine with uniform memory access. It is hard to extend them to NUMA based machines or a router formed with cluster of machines. The Forwarding table, Pending Interest table and the content store which can be globally accessed in the previous models cannot suit these types of architectures. It is hard or not possible to maintain the consistency of the structures without any high latency solutions.

The consistent structures were possible in previous designs mainly due to the cache coherence in the underlying hardware architecture. Though cache coherence is present in NUMA type of architecture, the overhead of accessing cache blocks or table entries from memory present in other nodes is really high. For cluster of machines, virtualizing the tables to provide a global view is really hard or close to not possible since the consistency of the structures has to be ensured manually through software solutions.

We employ a partition based solution to overcome the consistency issue similar to consistency hashing [reference]. Each node in the cluster or in the NUMA is made the owner of a one or more partition and they will be responsible to forward any packets belonging to that partition. Each node uses the same hash function to map the packets onto the partition. Simple version of this model is shown in figure 2. The ownership table is maintained by each node carrying the same mapping between the partitions and its respective owner. Since this mapping is not going to be changed frequently, maintaining their consistency does not incur high overhead.

The packets can flow from one node to another since they are to be forwarded to their owners. Though this can be a overhead, the overhead is lesser than trying to provide global view of structures where any node in the system can forward any packets. We can employ optimization like batching in RouteBricks . instead of forwarding the packets immediately to its owners, wait till a batch of packets gets accumulated to be forwarded to the destination - to reduce contention in the intra cluster network and also to improve effective usage bandwidth. Load imbalance can be overcome by dynamic partitioning techniques thereby shifting some load to other nodes.

5 Evaluation

All the experiments were run on a machine with Intel Xeon X5550 processors, memory capacity of 24 GB and Linux OS with kernel version 2.6.18.

We developed our own application framework to simulate the functionality of CCN router. The framework consists of a packet generator which is reposnsible for generating CCN packet and forward them to the core CCN router. The packet generation is completely randomized that it generates both types of CCN packets - Forward and Interest and also takes care of generating appropriate input ports. Another property of the generator is that the packets are equally distributed in count among the different partitions in the system.

The main router functionality is implemented as a separate pluggable component. Separate routers were implemented for each design model. The router tables - Forwarding Information Base, Pending Interest table, Content Store - were implemented as static arrays indexed through packet ID. The router tables are part of the core routers since the structure of these tables can vary for different design models. In order to avoid the tables getting stored in the L3 cache, each table used was configured with

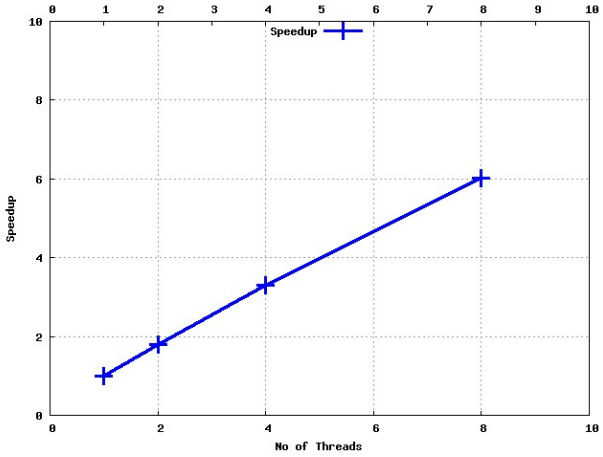

Figure 3: **Speedup of the router with the number of threads in the lock based parallel router**

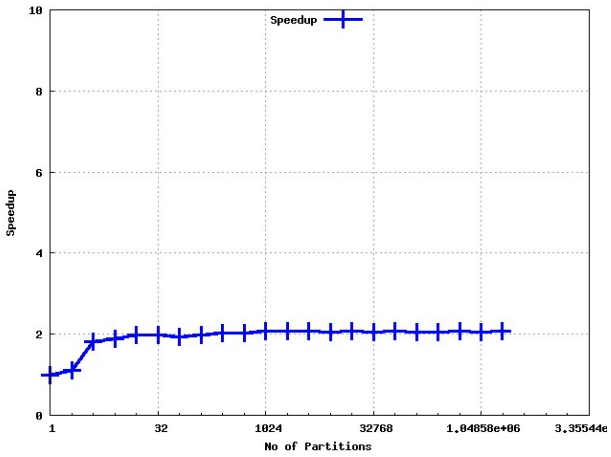

Figure 4: **Concurrency in router with number of logical partititions in the locak based system**

minimum size of 32 MB. The core routers were configured with different number of threads upto eight threads to measure the speedup.

5.1 Lock based Parallel CCN router

The lock based design made use of spin locks available with the pthread library. We wanted to evaluate two key factors - speedup achieved by the router and the amount of concurrency exploited with the number of partitions. The partitions referred in this design determines the granularity of locking.

Figure 3 shows the speedup that we obtain with the number of threads. The number of partitions were fixed to 1024 for this experiment. The graph shows that our model was able to achieve good speedup but it is not close to linear. Load imbalance is not a concern here because we observed that the randomly generated packets were mapped across different partitions and not skewed to any particular partition. Since all threads can access the global structures, cache conflicts among different cores are possible. This might result in reduced scalability and thus eventually affect the speedup.

The concurrency obtained with number of partitions is shown in Figure 4. The graph shown here was obtained for four threads. But we observed similar results for other thread count. The idea of this experiment is to illustrate the effect of the granularity of locks on the amount of speedup exhibited by the router. It can be clearly seen that coarse grained locking affects the speedup since it reduces the number of logical partition and thereby increasing the

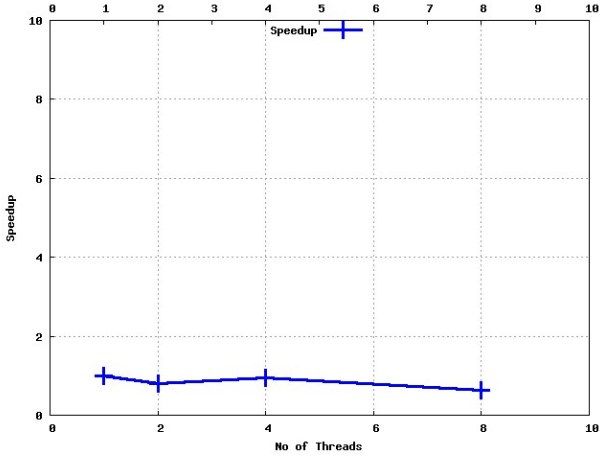

Figure 5: **Speedup of the router with the number of threads in the transaction based parallel router**

contention and the execution time. But the effect of partitioning does not extend till finest locking granularity. This is because the effect of contention is not really high to see any improvement with more partitions and so the concurrency obtained reaches maximum with certain number of partitions after which the graph flattens.

5.2 Transaction based Parallel CCN router

Transactional memory can ideally obtain performance of the finest grained locking mechanism. We used Intel STM which is a software based transactional memory for this experiment. Intel STM operates on cache line granularity - independent threads can access different cache lines without generating any conflicts. Access to the same cache line will

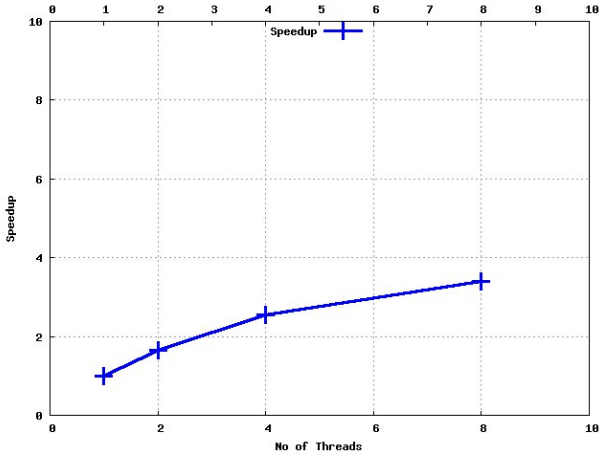

Figure 6: **Speedup of the router with the number of threads in the partition based parallel router**

trigger conflict and one of the transaction has to be restarted based on the policy implemented in the system. There are systems like Tiny STM which can achieve word level granularity.

Figure 5 shows that the speedup obtained is worse than the single thread performance. Since the implementation was very basic, running the experiments with multiple threads ended up with multiple conflicts. The global structures were not optimized to avoid or reduce the conflicts among multiple transactions. The number of conflicts increased with the number of threads which was reflected in the statistics logged by the STM framework. We ran simple microbenchmark to analyze the effect of this optimization. The sample application basically spawned multiple threads and they were made to access different logical partitions of a global array. It was noticed that the number of conflicts were reduced if the size of the logical partition was increased.

5.3 Partition based Parallel CCN router

The partitions referred in this model should not be confused with the logical partitions referred in the previous models. The global tables are physically partitioned and any node in the system will be able to see only the partition that is assigned to it. We implemented a very simple consistency hashing like technique to partition the global structure and distribute the ownership among multiple nodes. We treated different threads as different nodes for this simulation. Two sets of threads are considered (i) Distributor threads which sends the packet to the appropriate owner based on the packet ID (ii) Classifier threads which performs the actual routing and forwarding based on the partitioned structure local to itself.

Since this was simulated on a shared memory machine, classifier thread in each node maintains separate queues for every distributor queue to avoid any contention while forwarding. The speedup of this model is shown in Figure 6. It achieves a good speedup but not comparable to the lock based design. It is to be noted that the implementation of this model is very primitive and not much optimizations were done to improve the speedup.

6 Related Work

Software routing has been around for quiet a while now. Vyatta[8] is a company which provides open source software routers that can run on x86 hardware as well as virtualization and cloud platforms. However there is a key difference between their product and the work we do in this project. Their router design does not explore the benefits of parallelism that be used in a software router architecture. Likewise the thought of focusing on Content based data transfer has been tried in content based publish and subcribe projects like Sienna[5] and Hermes[6]. Sienna differs from CCN as its aim is to have fixed number of subscribers and many notifications from content providers. Similarly Hermes focusses on creating matchmaking rendezvous nodes between a publisher and subscriber, which is again different than CCN. CCN in contrast aims at focusing on Content based networks which can scale to many subscribers. This makes CCN stand out in solving the problem of content based routing.

7 Future Work

There is are a lot of interesting challenges involved in implementing a multi-threaded software router for Content Centric Networking. The experiments on a simple Lock based multi-threaded CCN router implementation reveals that we can get linear speedup. However we have conducted our experiments upto 8 threads, we would like to try the same with larger thread pool sizes going upto 1000 threads. Our partitioning strategies suggest that there is potential for speedup on applying parti-

tion techinques on routing tables, we would like to explore this further and analyze more trends based on varied partition sizes.

We considered using transactional memory with our implementation but found that with the current implementation we get a negative speedup, we would like to explore the necessary optimizations required for a transactional memory implementation as transactional memory performs poorly in the cases of routing entry conflicts for the same cache line entry.

Finally the CCN router implementation needs to be integrated with RouteBricks, this would require porting to Click infrastructure.

8 Conclusion

Hardware routers are very efficient and operate at very high rates. These routers make use of special purpose hardware and software inorder to provide high throughput. The main limitation of using such routers is that they cannot be modified that easily. This led to the development of the need for software routers. Modern Software routers cannot provide for high data rates provided by hardware routers. But efforts are being made to parallelize software routing inorder to get higher speedup. We have thought along these lines and implemented a primitive parallel Content Centric Router and have analyzed its performance for a few workloads. We believe combining parallelism with data partitioning in software could lead to greater througput in software routing and think that the study provided in this implementation could be a 'food for thought' for more experimentation with software routers on these lines.

References

[1] Mihai Dobrescu et al., "Route Bricks: Exploiting Parallelism to Scale Software Routers" *22nd ACM Symposium on Operating Systems Principles (SOSP)*,October 2009.

[2] Katerina Argyraki et al, "Can Software Routers Scale? ," *ACM Sigcomm Workshop - PRESTO*, August 2008.

[3] V. Jacobson et al, "Networking Named Content ," *CoNEXT 2009*, Rome, December, 2009.

[4] "What is Project CCNx?",retrieved from http://www.ccnx.org/content/welcome

[5] Carzaniga,A.,"Siena-Software", http://www.inf.unisi.ch/carzaniga/siena/software/index.html

[6] Peter R. Pietzuch and Jean M. Bacon." Hermes: A Distributed Event-Based Middleware Architecture". *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops*, p.611-618, July 02-05, 2002

[7] T. Koponen (ICSI/HIIT) et al, "A Data-Oriented (and Beyond) Network Architecture".*Proceedings of Sigcomm* ,2007

[8] http://www.vyatta.com/

[9] http://software.intel.com/en-us/articles/intel-c-stm-compiler-prototype-edition/

[10] http://www.tmware.org/tinystm

[11] Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill, David A. Wood: "LogTM: Log-Based Transactional Memory", *Proc. Symposium on High-Performance Computer Architecture*, February 2006

[12] Transactional Memory Coherence and Consistency Lance Hammond, Vicky Wong, Mike Chen, Ben Hertzberg, Brian D. Carlstrom, John D. Davis, Manohar K. Prabhu, Honggo Wijaya, Christos Kozyrakis, and Kunle Olukotun *Proceedings of the 31st Annual International Symposium on Computer Architecture*, M, Germany, June 19-23, 2004.

[13] Karger, D.; Sherman, A.; Berkheimer, A.; Bogstad, B.; Dhanidina, R.; Iwamoto, K.; Kim, B.; Matkins, L.; Yerushalmi, Y. (1999). "Web caching with consistent hashing". Computer Networks 31 (11): 1203.1213

[14] Hardware Acceleration of Transactional Memory on Commodity Systems Jared Casper, Tayo Oguntebi, Sungpack Hong, Nathan Bronson, Christos Kozyrakis, Kunle Olukotun, *ASPLOS 2011*

[15] Hybrid NOrec: A Case Study in the Effectiveness of Best Effort Hardware Transactional Memory Luke Dalessandro, Fraincois Carouge, Sean White, Yossi Lev, Mark Moir, Michael Scott, Michael Spear, *ASPLOS 2011*