Supporting Strong Cache Coherency for Active Caches in Multi-Tier Data-Centers over InfiniBand

S. Narravula, P. Balaji, K. Vaidyanathan, S. Krishnamoorthy, J. Wu and D. K. Panda

The Ohio State University
Presentation Outline

- Introduction/Motivation
- Design and Implementation
- Experimental Results
- Conclusions
Introduction

- Fast Internet Growth
  - Number of Users
  - Amount of data
  - Types of services

- Several uses
  - E-Commerce, Online Banking, Online Auctions, etc

- Types of Content
  - Images, documents, audio clips, video clips, etc - Static Content
  - Stock Quotes, Online Stores (Amazon), Online Banking, etc. - Dynamic Content (Active
Presentation Outline

- Introduction/Motivation
  - Multi-Tier Data-Centers
  - Active Caches
  - InfiniBand
- Design and Implementation
- Experimental Results
- Conclusions
Multi-Tier Data-Centers

- Single Powerful Computers
- Clusters
  - Low ‘Cost to Performance’ Ratio
  - Increasingly Popular
- Multi-Tier Data-Centers
  - Scalability – an important issue
A Typical Multi-Tier Data-Center

Tier 0
- Clients
- Proxy Nodes

Tier 1
- Application Servers
- Web Servers

Tier 2
- Database Servers

Apache

PHP

WAN
Tiers of a Typical Multi-Tier Data-Center

- **Proxy Nodes**
  - Handle Caching, load balancing, security, etc

- **Web Servers**
  - Handle the HTML content

- **Application Servers**
  - Handle Dynamic Content, Provide Services

- **Database Servers**
  - Handle persistent storage
Data-Center Characteristics

- The amount of computation required for processing each request increases as we go to the inner tiers of the Data-Center
- Caching at the front tiers is an important factor for scalability
Presentation Outline

- Introduction/Motivation
  - Introduction
  - Multi-Tier Data-Centers
  - Active Caches
  - InfiniBand
- Design and Implementation
- Experimental Results
- Conclusions
Caching

- Can avoid re-fetching of content
- Beneficial if requests repeat
- Static content caching
  - Well studied in the past
  - Widely used
Active Caching

- Dynamic Data
  - Stock Quotes, Scores, Personalized Content, etc
- Simple caching methods not suited
- Issues
  - Consistency
  - Coherency
Cache Consistency

- Non-decreasing views of system state
- Updates seen by all or none
Cache Coherency

- Refers to the average staleness of the document served from cache
- Two models of coherence
  - Bounded staleness (Weak Coherency)
  - Strong or immediate (Strong Coherency)
**Strong Cache Coherency**

- An absolute necessity for certain kinds of data
  - Online shopping, Travel ticket availability, Stock Quotes, Online auctions
  - Example: Online banking
    - Cannot afford to show different values to different concurrent requests
<table>
<thead>
<tr>
<th>Caching policies</th>
<th>Consistency</th>
<th>Coherency</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Caching</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Client Polling</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Invalidation *</td>
<td>√</td>
<td>✗</td>
</tr>
<tr>
<td>TTL/Adaptive TTL</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Presentation Outline

- Introduction/Motivation
  - Introduction
  - Multi-Tier Data-Centers
  - Active Caches
  - InfiniBand

- Design and Implementation

- Experimental Results

- Conclusions
InfiniBand

- High Performance
  - Low latency
  - High Bandwidth
- Open Industry Standard
- Provides rich features
  - RDMA, Remote Atomic operations, etc
- Targeted for Data-Centers
- Transport Layers
  - VAPI
  - IPoIB
  - SDP
Performance

- Low latencies of less than 5us achieved
- Bandwidth over 840 MB/s

* SDP and IPoIB from Voltaire’s Software Stack
Performance

- Receiver side CPU utilization is very low
- Leveraging the benefits of One sided communication
## Caching policies

<table>
<thead>
<tr>
<th></th>
<th>Consistency</th>
<th>Coherency</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Caching</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Client Polling</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Invalidation</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>TTL/Adaptive TTL</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Objective

- To design an architecture that very efficiently supports strong cache coherency on InfiniBand
Presentation Outline

- Introduction/Motivation
- Design and Implementation
- Experimental Results
- Conclusions
Basic Architecture

- External modules are used
  - Module communication can use any transport

- Versioning:
  - Application servers version dynamic data
  - Version value of data passed to front end with every request to back-end
  - Version maintained by front end along with cached value of response
Mechanism

- **Cache Hit:**
  - Back-end Version Check
  - If version current, use cache
  - Invalidate data for failed version check

- **Cache Miss**
  - Get data to cache
  - Initialize local versions
Architecture

Front-End

Request

Cache Hit

Response

Cache Miss

Back-End
Design

- Every server has an associated module that uses IPoIB, SDP or VAPI to communicate

- VAPI:
  - When a request arrives at proxy, VAPI module is contacted.
  - Module reads latest version of the data from the back-end using one-sided RDMA Read operation
  - If versions do not match, cached value is invalidated
VAPI Architecture

Front-End

Request

Cache Hit

Response

Back-End

RDMA Read

Cache Miss
Implementation

- Socket-based Implementation:
  - IPoIB and SDP are used
  - Back-end version check is done using two-sided communication from the module
- Requests to read and update are mutually excluded at the back-end module to avoid simultaneous readers and writers accessing the same data.
- Minimal changes to existing software
Presentation Outline

- Introduction/Motivation
- Design and Implementation
- Experimental Results
  - Data-Center Throughput
  - Data-Center Response Time
  - Data-Center Break-up
  - Zipf and WC Trace Throughput
- Conclusions
Experimental Test-bed

- Eight Dual 2.4GHz Xeon processor nodes
- 64-bit 133MHz PCI-X interfaces
- 512KB L2-Cache and 400MHz Front Side Bus
- Mellanox InfiniHost MT23108 Dual Port 4x HCAs
- MT43132 eight 4x port Switch
- SDK version 0.2.0
- Firmware version 1.17
Data-Center: Performance

- The VAPI module can sustain performance even with heavy load on the back-end servers.
The VAPI module responds faster even with heavy load on the back-end servers.
Response Time Breakup

- Worst case Module Overhead less than 10% of the response time
- Minimal overhead for VAPI based version check even for 200 compute threads
The drop in the throughput of VAPI in World cup trace is due to the higher penalty for cache misses under increased load. VAPI implementation does better for real trace too.
Conclusions

- An architecture for supporting Strong Cache Coherence
- External module based design
  - Freedom in choice of transport
  - Minimal changes to existing software
- Sockets API inherent limitation
  - Two-sided communication
  - High performance Sockets not the solution (SDP)
- Main benefit
  - One sided nature of RDMA calls
Web Pointers

http://nowlab.cis.ohio-state.edu/

E-mail: {narravul, balaji, vaidyana, savitha, wuj, panda} @cis.ohio-state.edu