Designing Next-Generation Data-Centers with Advanced Communication Protocols and Systems Services

Presented by: Jitong Chen
Outline

- Architecture of Web-based Data Center
- Three-Stage framework to benefit from InfiniBand
- Optimize Communication Protocol
- Data-Center Service Primitives
- Dynamic Content Caching
- Active Resource Adaptation
Architecture of Web-based Data Center

Figure 1: Web-based data-centers
Problems of Traditional Web-based Data Center

- TCP/IP Protocols have high latency, low bandwidth
- Two-sided communication incur CPU overhead at two sides.
- Low scalability of Strong Cache Coherence for Dynamic Content Caching
- Poor Service-Level Load-balancing Support to fully utilize limited physical Resource
Three-Stage Framework to Benefit from InfiniBand

Figure 2: Proposed Framework
Optimize Communication Protocol

- **AZ_SDP** (Asynchronous Zero-Copy SDP)

---

**Figure 3:** AZ-SDP Performance: (a) Latency and (b) Unidirectional Throughput
Data-Center Service Primitives

- Soft shared state primitive efficiently share information across cluster by creating a logical shared memory region using IBA’s RDMA operation.

![Diagram showing shared state and communication between Proxy Server and App Server](image)

Figure 4: A Soft Shared State Scenario
Data-Center Service Primitives

- Soft shared state primitive efficiently share information across cluster by creating a logical shared memory region using IBA’s RDMA operation
Dynamic Content Caching

- Client Polling Protocol Using RDMA read
- Coherent Invalidation
- The New Caching Design achieve 20% improvement for overall data center throughput
Active Resource Adaptation

Figure 8: RDMA based Protocol for Dynamic Reconfigurability
Active Resource Adaptation

![Graph showing Basic Reconfigurability Performance with 8 nodes and 14 nodes comparison. The x-axis represents Burst Length (requests) ranging from 1K to 16K, and the y-axis represents Transactions per Second. The graph compares Rigid (Small), Reconf, and Rigid (Large) configurations.]
Summary

Proposed a three-layer framework

- AZSDP reduce communication overhead
- Soft State Primitives eases the sharing of information across cluster
- RDMA-based Dynamic Content Caching increase throughput
- RDMA-based Active Resource Adaptation Protocol
DDDS: A Low-Overhead Distributed Data Sharing Substrate for Cluster-Based Data-Centers over Modern Interconnects

Presented by: Jitong Chen
Outline

- The Design Goals of DDSS
- DDSS Framework
- Implementation
- Evaluation
The Design Goals of DDSS

- Allow efficient sharing of information across the cluster by creating a logical shared memory region
- Support local and remote allocation in the shared state
- Support the access, update and deletion of data for all threads in a transparent manner
- Be resilient to load imbalances and should have minimal overheads to access to data
The Design Goals of DDSS

Support a range of coherency models:

- **Strict Coherence** (obtain the most current version and excludes concurrent writes and reads)
- **Write Coherence** (obtain the most current version and excludes concurrent writes)
- **Read Coherence** (obtain the most current version and excludes concurrent reads)
- **No Coherence**
- **Delta Coherence** (data is no more than $x$ versions stale)
- **Temporal Coherence** (data is no more than $t$ time units stale)
Non-Coherent/Coherent Distributed Data Sharing

Fig. 2. DDSS using the proposed Framework (a) Non Coherent Distributed Data Sharing Mechanism (b) Coherent Distributed Data Sharing Mechanism
DDSS Framework

Fig. 3. Proposed DDSS Framework
Implementation

- IPC: create a run-time daemon support user process or thread to access DDSS
- Data Placement: try to distribute allocations among different nodes to avoid NIC contention
- Data Access: use one-sided operations to access remote memory without interrupting the remote node
Implementation

- **Locking Mechanism:**
  use atomic operation Compare-and-Swap to acquire and check the status of locks

- **Coherence Maintenance:**
  use atomic operation Fetch-and-Add to update the version of every put() operation,
Implementation

- **DDSS Interface:**

<table>
<thead>
<tr>
<th>DDSS Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int allocate_ss(nbytes, type, ...)</td>
<td>allocate a block of size nbytes in the shared state</td>
</tr>
<tr>
<td>int release_ss(key)</td>
<td>free the shared data segment</td>
</tr>
<tr>
<td>int get(key, data, nbytes, ...)</td>
<td>read nbytes from the shared state and place it in data</td>
</tr>
<tr>
<td>int put(key, data, nbytes, ...)</td>
<td>write nbytes of memory to the shared state from data</td>
</tr>
<tr>
<td>int acquire_lock_ss(key)</td>
<td>lock the shared data segment</td>
</tr>
<tr>
<td>int release_lock_ss(key)</td>
<td>unlock the shared data segment</td>
</tr>
</tbody>
</table>
Evaluation

- Micro benchmark

Increasing clients accessing different portions from a single node using get()
Evaluation

- Dynamic reconfiguration
Evaluation

- Application-level evaluation
Supporting Strong Coherency for Active Caches in Multi-Tier Data-Centers over InfiniBand

Presented by: Jitong Chen
Outline

- Architecture of Multi-Tier Data Center
- Web Cache Coherence
- Strong Cache Coherency Model
- Strong Cache Coherency Model over InfiniBand
- Experiment Results
Architecture of Multi-Tier Data Center
Web Cache Coherence

- Average staleness of the documents present in the cache, i.e., the time elapses between the current time and the time of the last update of the document in the back-end.

- Strong Coherence means average staleness is zero. i.e., a client get the same response whether a request is answered from cache or from the back-end.
Strong Cache Coherency Model

Figure 2. Strong Cache Coherence Protocol
Strong Cache Coherency Model

Figure 3. Interaction between Data-Center Servers and Modules
Strong Cache Coherency Model over InfiniBand

Figure 4. Strong Cache Coherency Protocol: InfiniBand based Optimizations
Experiment Results

TCP/IP over InfiniBand (IPoIB)
Sockets Direct Protocol over InfiniBand (SDP)
native InfiniBand Verbs layer (VAPI)

Datacenter: Response Time

Datacenter: Throughput

Figure 6. Data-Centers Performance Analysis
Experiment Results

Figure 8. Data-Center Response Time Breakup: (a) 0 Compute Threads, (b) 200 Compute Threads
Summary

- RDMA operations provide low latency and high bandwidth communication between tiers in data center.

- One sides communication provided by native InfiniBand leave more CPU free for the data center nodes to perform other operations.

- When Application Server is busy, one sided communication doesn’t require much CPU to request coherence status from back-end tier, therefore cache verification is not slowed down too much even when the application server is heavily loaded.
Thank You !