Efficient Memory Disaggregation with Infiniswap

Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, Kang G. Shin
Agenda

• Motivation and related work

• Design and system overview

• Implementation and evaluation

• Future work and conclusion
Memory-intensive applications

- VOLTDB
- memCached
- powergraph
- GraphX
Memory-intensive applications

Your computer is low on memory

To restore enough memory for programs to work correctly, save your files and then close or restart all open programs.

OK
Performance degradation

- VoltDB (TPC-C)
- Memcached (Facebook/FB SYS)
- PowerGraph (TunkRank)
- GraphX (PageRank)

Normalized Performance

100% working sets in memory
Performance degradation

100% working sets in memory   75% working sets in memory
Performance degradation

\[\text{Normalized Performance} = \begin{array}{c|c|c|c|c}
\text{VoltDB (TPC-C)} & \text{Memcached (Facebook/FB SYS)} & \text{PowerGraph (TunkRank)} & \text{GraphX (PageRank)} \\
\hline
0.18 & 0.47 & 1 & 1
\end{array}\]

Legend:
- Blue: 100% working sets in memory
- Orange: 75% working sets in memory
Performance degradation

![Graph showing performance degradation with normalized performance values and working sets in memory for different systems.]
Performance degradation

<table>
<thead>
<tr>
<th>System</th>
<th>Normalized Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoltDB (TPC-C)</td>
<td>0.18</td>
</tr>
<tr>
<td>Memcached (Facebook/FB SYS)</td>
<td>0.47</td>
</tr>
<tr>
<td>PowerGraph (TunkRank)</td>
<td>0.06</td>
</tr>
<tr>
<td>GraphX (PageRank)</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- **100% working sets in memory**
- **75% working sets in memory**
- **50% working sets in memory**
Performance degradation

- VoltDB (TPC-C)
- Memcached (Facebook/FB SYS)
- PowerGraph (TunkRank)
- GraphX (PageRank)

Normalized Performance

- 100% working sets in memory
- 75% working sets in memory
- 50% working sets in memory

3/30/17
## Performance degradation

<table>
<thead>
<tr>
<th></th>
<th>Normalized Performance</th>
<th>100% working sets in memory</th>
<th>75% working sets in memory</th>
<th>50% working sets in memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoltDB (TPC-C)</td>
<td>0.18</td>
<td>0.04</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>Memcached (Facebook/FB SYS)</td>
<td>0.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PowerGraph (TunkRank)</td>
<td>0.94</td>
<td></td>
<td>0.04</td>
<td>0.06</td>
</tr>
<tr>
<td>GraphX (PageRank)</td>
<td>0.97</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance degradation

- **VoltDB (TPC-C)**: 0.18
- **Memcached (Facebook/FB SYS)**: 0.47
- **PowerGraph (TunkRank)**: 0.94
- **GraphX (PageRank)**: 0.97

Legend:
- Blue: 100% working sets in memory
- Orange: 75% working sets in memory
- Red: 50% working sets in memory
Performance degradation

Normalized Performance

- VoltDB (TPC-C): 0.18, 0.04
- Memcached (Facebook/FB SYS): 0.47, 0.06
- PowerGraph (TunkRank): 0.94, 0.12
- GraphX (PageRank): 0.97, 0.04

- 100% working sets in memory
- 75% working sets in memory
- 50% working sets in memory
Performance degradation

Memory overestimation

100% working sets in memory  75% working sets in memory  50% working sets in memory
Memory underutilization

- Google Cluster Analysis\textsuperscript{[1]}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Memory allocation and usage over time.}
\end{figure}

\textsuperscript{[1]} Reiss, Charles, et al. “Heterogeneity and dynamicity of clouds at scale: Google trace analysis.” SoCC’12.
Memory underutilization

- Google Cluster Analysis\(^{[1]}\)

---

\(^{[1]}\) Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." SoCC'12.
Memory underutilization

• Google Cluster Analysis[1]

Memory underutilization

- Google Cluster Analysis\(^1\)

![Graph showing memory utilization](image)

---

\(^{1}\) Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." SoCC'12.
Memory underutilization

- Google Cluster Analysis\(^1\)

Can we utilize this memory?

---

Machine 1

Machine 2

Machine 3

Machine 4

Machine N

Used memory

Free memory

Remote memory
Disaggregate free memory

Memory Disaggregation Layer

Machine 1

Machine 2
Used memory
Free memory
Remote memory

Machine 3
 Machine 4
 Machine N

...
Disaggregate free memory

Machine 1

Memory Disaggregation Layer

Machine 2
Machine 3
Machine 4
Machine N

Used memory
Free memory
Remote memory
What are the challenges?

• Minimize deployment overhead
  • No hardware design
  • No application modification

• Tolerate failures
  • e.g. network disconnection, machine crash

• Manage remote memory at scale
## Recent work on memory disaggregation

<table>
<thead>
<tr>
<th></th>
<th>No HW design</th>
<th>No app modification</th>
<th>Fault-tolerance</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Blade</strong> [ISCA’09]</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>HPBD [CLUSTER’05] / NBDX [1]</strong></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>RDMA key-value service</strong> (e.g. HERD [SIGCOMM’14], FaRM [NSDI’14])</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Intel Rack Scale Architecture (RSA) [2]</strong></td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Infiniswap</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

---

[1] https://github.com/accelio/NBDX
Agenda

• Motivation and related work

• Design and system overview

• Implementation and evaluation

• Future work and conclusion
System Overview

User Space

Application1
Application2

Kernel Space

Virtual Memory Manager (VMM)

Infiniswap Block Device

Local Disk

Async

Sync

Machine 1

RNIC

Machine 2

Infiniswap Daemon

Application

User Space

RNIC
System Overview

Infiniswap Block Device
- Swap space
- Request router

Machine 1
- Application
- Virtual Memory Manager (VMM)
- Infiniswap Block Device
- Local Disk
- RNIC

Machine 2
- Infiniswap Daemon
- Application
- RNIC

User Space
- Kernel Space
- Infiniswap

Async
Sync
System Overview

Local disk
- [ASYNC] backup swapped-out data
- Tolerate remote memory failure
System Overview

Infiniswap Deamon
- Local memory region
- Remote memory service

Machine 1
- Application1
- Application2
- Virtual Memory Manager (VMM)
- Infiniswap Block Device
- Local Disk
- RNIC
- Sync
- Async

Machine 2
- Application
- User Space
- Infiniswap Daemon
- RNIC
System Overview

RDMA
- One-sided operations
- Bypass remote CPU
### How to meet the design objectives?

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>No hardware design</td>
<td>Remote paging</td>
</tr>
<tr>
<td>No application modification</td>
<td></td>
</tr>
<tr>
<td>Fault-tolerance</td>
<td>Local backup disk</td>
</tr>
</tbody>
</table>
One-to-many

Machine 1

- Application 1
- Application 2
- Virtual Memory Manager (VMM)
- Infiniswap Block Device
- Local Disk
- Async
- Sync
- RNIC

Machine 2

- Infiniswap Daemon
- RNIC

Machine 3

- Application
- User Space
- Infiniswap Daemon
- RNIC
Many-to-many
How to scale remote memory?

- How to find remote memory in the cluster?
- Which remote mapping should be **evicted**?
How to meet the design objectives?

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>No hardware design</td>
<td>Remote paging</td>
</tr>
<tr>
<td>No application modification</td>
<td></td>
</tr>
<tr>
<td>Fault-tolerance</td>
<td>Local backup disk</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td><strong>Decentralized</strong> remote memory management</td>
</tr>
</tbody>
</table>
Management unit: memory page?

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon
Management unit: memory page?

Infiniswap Block Device

<table>
<thead>
<tr>
<th>Local Page</th>
<th>Remote Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>p100</td>
<td>&lt;s1, p1&gt;</td>
</tr>
</tbody>
</table>

1GB = 256K entries
1GB = 256K RTTs
Management unit: memory slab!

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon
Management unit: memory slab!

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon
Which remote machine should be selected?
Which remote machine should be selected?

Infiniswap Block Device

Infiniswap Daemon

Goal: balance memory utilization
Which remote machine should be selected?

- Central controller

- Infiniswap Block Device

- Infiniswap Daemon

- Infiniswap Daemon

- Infiniswap Daemon

- Infiniswap Daemon
Which remote machine should be selected?

- Central controller
- Decentralized approach
Infiniswap Block Device

Power of two choices

Power of two choices

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon

Slab eviction

Infiniswap Daemon

Remote Memory

Used Memory

Mapped Slab

Unmapped Slab
Slab eviction

Remote Memory

Used Memory

Mapped Slab

Unmapped Slab

Infiniswap Daemon
Slab eviction
Which slab should be evicted?

Daemon: Does not know the swap activities
Which slab should be evicted?

Infiniswap Daemon

Daemon: Too expensive to query all the slabs
Power of multiple choices \cite{Park2011}

Select E least-active slabs from E+E’ random slabs

\begin{figure}
\centering
\includegraphics[width=\textwidth]{infiniswap_daemon}
\caption{Infiniswap Daemon}
\end{figure}

\textsuperscript{[1]} Park, Gahyun. "A generalization of multiple choice balls-into-bins." PODC'11
Power of multiple choices\textsuperscript{[1]}

Select E least-active slabs from $E+E'$ random slabs

\textsuperscript{[1]} Park, Gahyun. "A generalization of multiple choice balls-into-bins." PODC’11
Power of multiple choices[1]

Select E least-active slabs from E+E’ random slabs

Agenda

• Motivation and related work

• Design and system overview

• Implementation and evaluation

• Future work and conclusion
Implementation

- **Connection Management**
  - **One** RDMA connection per active block device - daemon pair
- **Control Plane**
  - **SEND**, **RECV**
- **Data Plane**
  - **One-sided** RDMA READ, WRITE
What are we expecting from Infiniswap?

- Application performance
- Cluster memory utilization
- Network usage
- Eviction overhead
- Fault-tolerance overhead
- Performance as a block device
Evaluation

32-node cluster

InfiniBand Network

2 x 8 cores (32 vcores)
64GB DRAM
56Gbps InfiniBand NIC

VOLTDB memCached powergraph GraphX
Application performance

- 50% working sets in memory

- Application performance is improved by 2-16x
Application performance

- 50% working sets in memory

![Bar chart showing normalized performance comparison]

- Application performance is improved by 2-16x
Application performance

- 50% working sets in memory

Application performance is improved by 2-16x
Cluster memory utilization

- 90 containers (applications), mixing all applications and memory constraints.

- Cluster memory utilization is improved from **40.8%** to **60%** \((1.47x)\)
Agenda

• Motivation and related work

• Design and system overview

• Implementation and evaluation

• **Future work and conclusion**
Limitations and future work

• **Trade-off in fault-tolerance**
  • Local disk is the bottleneck
  • Multiple remote replicas
    • Fault-tolerance vs. space-efficiency

• **Performance isolation among applications**
  • W/o limitation on each application’s usage
  • W/o mapping between remote memory and applications
Conclusion

• Infiniswap: remote paging over RDMA
  • Application performance
  • Cluster memory utilization

• Efficient, practical memory disaggregation
  • No hardware design
  • No application modification
  • Fault-tolerance
  • Scalability

Source code is coming soon!
https://github.com/Infiniswap/infiniswap.git
Thank You!