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.
Performance degradation

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

100% 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

3/30/17
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
Performance degradation

- VoltDB (TPC-C): 0.18
- Memcached (Facebook/FB SYS): 0.47
- PowerGraph (TunkRank): 1.0
- GraphX (PageRank): 1.0

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

Normalized Performance

- VoltDB (TPC-C)
  - 100% working sets in memory: 0.18
  - 75% working sets in memory: 0.04
  - 50% working sets in memory: 0.06

- Memcached (Facebook/FB SYS)
  - 100% working sets in memory: 0.47

- PowerGraph (TunkRank)
  - 100% working sets in memory: 1.0

- GraphX (PageRank)
  - 100% working sets in memory: 1.0
Performance degradation

- VoltDB (TPC-C): 0.18 (100% working sets in memory), 0.04 (75% working sets in memory), 0.04 (50% working sets in memory)
- Memcached (Facebook/FB SYS): 0.47 (100% working sets in memory), 0.06 (50% working sets in memory)
- PowerGraph (TunkRank): 0.94 (100% working sets in memory)
- GraphX (PageRank): 0.94 (100% working sets in memory)
Performance degradation

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

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

- VoltDB (TPC-C)
  - 100% working sets in memory: 0.18
  - 75% working sets in memory: 0.04
  - 50% working sets in memory

- Memcached (Facebook/FB SYS)
  - 100% working sets in memory
  - 75% working sets in memory: 0.47
  - 50% working sets in memory: 0.06

- PowerGraph (TunkRank)
  - 100% working sets in memory: 0.94
  - 75% working sets in memory: 0.12
  - 50% working sets in memory: 0.04

- GraphX (PageRank)
  - 100% working sets in memory: 0.97
  - 75% working sets in memory
  - 50% working sets in memory

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

- **VoltDB (TPC-C)**: Normalized performance drops from 0.18 to 0.04, 0.47 to 0.06, and 0.94 to 0.12 as working sets in memory decrease from 100% to 75% and then to 50%.
- **Memcached (Facebook/FB SYS)**: Performance drops from 0.47 to 0.06 as working sets in memory decrease from 100% to 75%.
- **PowerGraph (TunkRank)**: Performance drops from 0.94 to 0.12 as working sets in memory decrease from 100% to 75%.
- **GraphX (PageRank)**: Performance drops from 0.97 to 0.04 as working sets in memory decrease from 100% to 75%.

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

Memory overestimation

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

Legend:
- 100% working sets in memory
- 75% working sets in memory
- 50% working sets in memory
Memory underutilization

- Google Cluster Analysis\(^1\)

![Graph showing Allocated and Used memory over time](image)

---

Memory underutilization

- Google Cluster Analysis[1]

Memory underutilization

- Google Cluster Analysis\textsuperscript{[1]}

![Graph showing memory allocation and usage over time](image)

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

- Google Cluster Analysis\[1\]

![Graph showing memory allocation and usage over time]

- Allocated memory portion:
  - Approximately 30% underutilization

- Used memory:
  - Consistent usage with slight fluctuations

---

Memory underutilization

• Google Cluster Analysis\[1\]

Allocated

Used

Portion of Memory

=30%

Can we utilize this memory?

---

Disaggregate free memory

Machine 1

Memory Disaggregation Layer

---

Machine 2
Machine 3
Machine 4
... Machine N

Used memory
Free memory
Remote memory
Disaggregate free memory

Memory Disaggregation Layer

Machine 1

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</strong> [1]</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)</strong> [2]</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](https://github.com/accelio/NBDX)
Agenda

• Motivation and related work

• Design and system overview

• Implementation and evaluation

• Future work and conclusion
System Overview

- Application 1
- Application 2
- Virtual Memory Manager (VMM)
- Infiniswap Block Device
- Local Disk
- RNIC
- Machine 1
- Infiniswap Daemon
- RNIC
- Machine 2
- User Space
- Kernel Space
- Async
- Sync
- S
- Sync
- Async
System Overview

Infiniswap Block Device
- Swap space
- Request router
System Overview

Machine 1

User Space Application1 Application2 ...
Kernel Space Virtual Memory Manager (VMM)

Async

Local Disk

Sync

RNIC

[ASYNC] backup swapped-out data
Tolerate remote memory failure

Machine 2

Infiniswap Daemon

Application

RNIC

User Space
System Overview

Infiniswap Deamon
- Local memory region
- Remote memory service

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

Sync
Async

Machine 2
- Application
- Infiniswap Daemon
- RNIC

Local memory region
Remote memory service
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>Local backup disk</td>
</tr>
<tr>
<td>Fault-tolerance</td>
<td></td>
</tr>
</tbody>
</table>
One-to-many

Virtual Memory Manager (VMM) → Infiniswap Block Device

Machine 1:
- Application 1
- Application 2
- Infiniswap Daemon
- RNIC
- Local Disk

Machine 2:
- Infiniswap Daemon
- RNIC

Machine 3:
- Infiniswap Daemon
- RNIC
Many-to-many
Many-to-many

How to scale remote memory?

- How to find remote memory in the cluster?
- Which remote mapping should be evicted?
<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>Local backup disk</td>
</tr>
<tr>
<td>Fault-tolerance</td>
<td><strong>Decentralized</strong> remote memory management</td>
</tr>
</tbody>
</table>
Management unit: memory page?

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

...
Management unit: memory page?

<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>

$1 \text{GB} = 256 \text{K entries}$

$1 \text{GB} = 256 \text{K RTTs}$
Management unit: memory slab!

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon
Management unit: memory slab!

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

...
Which remote machine should be selected?
Which remote machine should be selected?

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

- Central controller
Which remote machine should be selected?

- Central controller
- Decentralized approach
Power of two choices

Infiniswap Block Device

Infiniswap Daemon

Infiniswap Daemon

Infiniswap Daemon

Power of two choices [1]

Slab eviction

Infiniswap Daemon

1 2 3 4

Remote Memory

Mapped Slab

Unmapped Slab

Used Memory
Slab eviction

Remote Memory

Infiniswap Daemon

Used Memory

Mapped Slab

Unmapped Slab
Slab eviction

Remote Memory

Used Memory

Mapped Slab

Unmapped Slab
Which slab should be evicted?

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

Daemon: Too expensive to query all the slabs
Select $E$ least-active slabs from $E + E'$ random slabs.

Power of multiple choices \[1\]

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

Power of multiple choices\cite{1}

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

\cite{1} Park, Gahyun. "A generalization of multiple choice balls-into-bins." PODC’11
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

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

- 50% working sets in memory

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

- 50% working sets in memory

- 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!