Nswap: a network swapping module for Linux clusters

Tia Newhall, Sean Finney, Kuzman Ganchev, Michael Spiegel

Computer Science Department
Swarthmore College
Swarthmore, PA USA

newhall@cs.swarthmore.edu
Network Swapping

- Let cluster nodes transparently share each other’s RAM as swap space
  - when one node’s memory is overcommitted it swaps to the idle memory of other nodes in the cluster

- Part of SSI support for cluster systems
  - Cluster as single, large parallel machine
  - Idle cluster RAM as a single, large, shared swap partition

- Also applicable to any NW of PCs/WS
Why Network Swapping?

- Network speeds are getting faster, disk speeds are not keeping up

There is almost always some idle memory in the cluster even when some nodes are overloaded
  - Usually 2/3 idle, about 1/3 idle under heavy loads
Nswap

- Network swapping module for Linux clusters
  - Transparency
    - Processes don't need to do anything special to use Nswap
  - Efficiency and Portability
    - Make swapping in and out fast to node doing the swapping
    - Kernel level implementation as Linux lkm
  - Scalability
    - Point-to-Point model
    - Don't require complete, nor accurate, global state info.
      => Each node independently w/o complete info. chooses the remote server to which to swap
  - Adaptability
    - Grow/Shrink each node's remote swap cache size based on its local memory needs
    - Remote page migration from server to server
      • avoid writing to disk when a server is full
Nswap Architecture

- Each node runs multi-threaded client & server
- **Nswap client** device driver for network swap “device”
  - Kernel makes swap-in & swap-out requests to it
- **Nswap server** manages part of RAM for caching remotely swapped pages (Nswap Cache)
Nswap Communication Protocol

- **PUTPAGE (swap-out)**

  Node A
  - IP Table
    | HOST | AMT | SOCK |
    |------|-----|------|
    | B    | 20  |      |
    | C    | 10  |      |
    | F    | 35  |      |

  Nswap client
  - Shadow slot map

  Swap out page

- **GETPAGE (swap-in)**

  Node A
  - Swap in page

  Node B
  - Nswap Server

PUTPAGE

GETPAGE
Nswap Communication Protocol

Page Migration (PUNTPAGE)

When Server B is full, it migrates A’s page to server C

Server C tells A that it now has A’s page

Client A tells Server B that it can drop its copy of A’s page

Nswap, EuroPar’03
Some Complications

Kernel doesn’t inform swap device driver when a slot is no longer being used

- For disk swap devices this is fine
- For NW swap devices this results in “dead” pages remaining cached on remote nodes

Nswap removes dead pages in 2 ways:

1) A re-use of a slot map results in an INVALIDATE message being set to the server caching the old, dead page

2) A garbage collector thread runs on each node, detecting dead slots and sending INVALIDATES
More Complications

Simultaneous Conflicting Operations:

EX: PUNTPAGE for slot i and a new PUTPAGE for slot i:

- A better not overwrite slot i with "C" losing its new location of "D"
- old page i at B and C should be dropped

To detect & handle cases like these:
- extra state kept in shadow slot map
- extra state sent with protocol msgs
Our project so far...

- Implemented as lkm for Linux 2.4.18
- Running on cluster of 8 nodes connected with switched 100 BaseT
  - All nodes have faster disk than Network
    - PII's disk is up to 176 Mb/sec
    - PIII's disk is up to 494 Mb/sec
    -> We expect to be slower than swapping to disk
- On 100BaseT Ethernet, Nswap is comparable in speed to swapping to faster disk
  - For several workloads Nswap on slower network is faster than swapping to fast disk
Experiments

- Workload 1: sequential R & W to large chunk of memory
  - Best case for swapping to disk
- Workload 2: random R & W to mem
  - Disk arm seeks w/in swap partition
- Workload 3: 1 large file I/O, 1 W1
  - Disk arm seeks between swap and file partitions
- Workload 4: 1 large file I/O, 1 W2
## Results

<table>
<thead>
<tr>
<th>Workload</th>
<th>PIII Disk (494 Mb/s)</th>
<th>Nswap (TCP 100BaseT)</th>
<th>Nswap (UDP 100BaseT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) 1 proc</td>
<td>13.1</td>
<td>154.3</td>
<td>61.3</td>
</tr>
<tr>
<td>(1) 4 proc</td>
<td>577.0</td>
<td>1507.9</td>
<td>614.4</td>
</tr>
<tr>
<td>(2) 1 proc</td>
<td>266.8</td>
<td>1071.8</td>
<td>155.5</td>
</tr>
<tr>
<td>(2) 4 proc</td>
<td>68.6</td>
<td>189.3</td>
<td>50.3</td>
</tr>
<tr>
<td>(3) 1 proc</td>
<td>770.2</td>
<td>1111.0</td>
<td>811.0</td>
</tr>
<tr>
<td>(3) 4 proc</td>
<td>727.1</td>
<td>1430.5</td>
<td>619.5</td>
</tr>
<tr>
<td>(4) 1 proc</td>
<td>923.9</td>
<td>1529.3</td>
<td>821.7</td>
</tr>
<tr>
<td>(4) 4 proc</td>
<td>502.5</td>
<td>498.7</td>
<td>429.2</td>
</tr>
</tbody>
</table>

- Nswap faster than swapping to much faster disk for several workloads
- TCP latency hurting Nswap performance
# Nswap on Faster Networks

<table>
<thead>
<tr>
<th>Workload</th>
<th>Disk</th>
<th>10BaseT</th>
<th>100BaseT</th>
<th>1Gb</th>
<th>10Gb</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) PIII TCP</td>
<td>580.10</td>
<td>5719.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>158.3</td>
<td>1075.0</td>
<td>1034.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>speed up 3.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.3</td>
<td>5.5</td>
</tr>
<tr>
<td>(1) PIII UDP</td>
<td>12.27</td>
<td>306.69</td>
<td></td>
<td>28.9</td>
<td>26.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>56.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(5.4)</td>
<td>10.6</td>
<td>11.6</td>
</tr>
<tr>
<td>(2) PIII UDP</td>
<td>266.79</td>
<td>847.74</td>
<td>153.5</td>
<td>77.3</td>
<td>70.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(5.5)</td>
<td>10.9</td>
<td>12.1</td>
</tr>
<tr>
<td>(4) PIII UDP</td>
<td>6265.39</td>
<td>9605.91</td>
<td>1733.9</td>
<td>866.2</td>
<td>786.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(5.54)</td>
<td>11.1</td>
<td>12.2</td>
</tr>
</tbody>
</table>

Measured on Disk, 10 BaseT and 100 BaseT

Calculated speed-up values for 1 Gbit and 10 Gbit

\[
\text{Speedup} = \frac{1}{(1 - \text{FracBandwidth} + \text{FracBandwidth}/\text{SpeedupBandwidth})}
\]
Conclusions

- Space efficient and time efficient implementation of Network Swapping
  - Designed to scale to large clusters
  - Adapts to local memory use on cluster nodes
- Nswap better than swapping to faster disk in several cases
- Nswap on faster NW will out perform disk in most cases
  - Based on NW vs. Disk speed trends, Nswap will be even better in the future
Future Work

- Develop better growing/shrinking policy
- Add reliability scheme to Nswap
- Test on larger, faster, heterogeneous clusters
- Implement faster reliable NW protocol
- Develop a swapping scheme that changes based on workload
  - For some workloads NW swapping may not be best choice