Network optimizations for PV guests

J. Renato Santos
G. (John) Janakiraman
Yoshio Turner

HP Labs

Xen Summit
September 7-8, 2006
Motivation

- Network I/O has very high CPU cost
  - 300% higher than Linux (~1 year old data)
- Goal:
  - Reduce CPU cost of network I/O
**Domain 0 profile for network I/O**

Profile results for dom0 while doing network I/O (receive) for domU

<table>
<thead>
<tr>
<th>Kernel (dom0)</th>
<th>Xen (in dom0 context)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>samples</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>15020</td>
<td>11.8946</td>
</tr>
<tr>
<td>12670</td>
<td>10.0337</td>
</tr>
<tr>
<td>9118</td>
<td>7.2207</td>
</tr>
<tr>
<td>6958</td>
<td>5.5101</td>
</tr>
<tr>
<td>6003</td>
<td>4.7539</td>
</tr>
<tr>
<td>4646</td>
<td>3.6794</td>
</tr>
<tr>
<td>4209</td>
<td>3.3331</td>
</tr>
<tr>
<td>3959</td>
<td>3.1352</td>
</tr>
<tr>
<td>3544</td>
<td>2.8071</td>
</tr>
<tr>
<td>2840</td>
<td>2.2490</td>
</tr>
<tr>
<td>3367</td>
<td>2.6675</td>
</tr>
<tr>
<td>72334</td>
<td>57.2843</td>
</tr>
</tbody>
</table>

- Focus on high cost components
  - Bridge
  - Memory management in Xen for I/O (grant tables)
Part 1

Bridge optimization
Network alternative to bridge

- Netback keeps a mapping of virtual interfaces to physical interfaces
- Netback hooks at bottom of network stack (dev.c) intercept packets:
  - received from remote host on a physical interface
  - transmitted by dom0 to a physical interface
- Netback route packets to right destination (based on MAC address):
  - Either to guest, physical interface or dom0 network stack (or to ALL if broadcast or multicast)
Performance comparison with bridge

- Alternative approach has better perf. than original bridge
- However, bridge overhead can be reduced
  - Disabling netfilter in bridge (CONFIG_BRIDGE_NETFILTER)
- Both approaches have equivalent performance
  - Direct routing in netback slightly better
Advantages of removing bridge

• Easier to configure network setup
  – No bridge to configure
  – No loopback interfaces for dom0
  – No need to rename interfaces
  – No need to stop and restart physical interfaces

• Less complexity implies lower probability of errors
  – Direct mapping of vifs to devices. Less opportunity for configuration errors.
  – No network setup script. Thus, no script bugs.
  – No errors due to insufficient loopback devices (8 in default bridge config)
  – No error due to uninstalled bridge-utils. (reduced SW requirements in dom0)

• Can be used with NFS root in dom0
  – Network connectivity not interrupted during network setup
Part 2

Optimization opportunities related to page grants
Cost of page flip

- Cost of data copy almost equivalent to page flip
  - At least when dom0 runs on a separate CPU

Estimated break even packet size: 1280 bytes

Estimated break even packet size: 480 bytes
Cost of mapping I/O page

*dom0 and domU in different CPUs*

*dom0 and domU on same CPU (SEDF)*

-pre-mapped: skb data pool mapped once by dom0 at initialization and reused for all packets*

- If cost of mapping/unmapping guest page is eliminated
  – copy becomes more efficient than page flipping in all cases.
Cost of mapping I/O page for TX

- Results are for Xen version before TSO optimizations
  - Numbers should be different now
  - But removing page mapping/unmapping for each packet should always help
    - No copy cost on transmit path
domU to domU communication

• Current implementation:
  – First copy page from transmitting guest to dom0
  – Then flip dom0 page with receiving guest page
  – Page flip just adds overhead
    • We should instead just do the copy
Some suggestions for discussion

• Replace page flip with data copy as default mechanism
  – Costs are similar
  – Better performance on domU to domU communication
  – Possibility for additional improvements (e.g. avoid mapping/unmapping of I/O page on each packet)
  – Future smart devices may place data directly in guest memory.
     (eliminating copy cost)

• Possible mechanism for reducing page mapping cost
  – Do not unmap guest page after each use in netback.
  – Keep a table of current guest pages mappings in netback
  – Current skb data allocation mechanism which uses a page cache will likely re-use guest pages already mapped by netback in the past.

• Alternative mechanism (proposed by Ian on Xen roadmap)
  – Static buffer between netfront and netback with 2 data copies on RX
    • Probably, simpler to implement but with higher cost (2 data copies)
Next steps

• Get new performance numbers on latest Xen unstable version with TSO optimizations

• Evaluate cost of new “copy” grant mechanism
  – Only one hypercall does everything (map page, copy, un-map page)
    • less expensive than “access” grant