vIC: Interrupt Coalescing for Virtual Machine Storage Device IO

Irfan Ahmad
Ajay Gulati, Ali Mashtizadeh

USENIX Annual Technical Conference
June 15, 2011
Outline

Interrupts

Interrupts Coalesced

Virtual Interrupts

Virtual Interrupts Coalesced

Inter-Processor Interrupts Coalesced
Interrupts

User → Kernel/Driver → Intr Handler → I/O Device

Interrupt Fired → IO Requested → Interrupt Fired

vlC: Interrupt Coalescing for Virtual Machine Storage Device IO
Interrupts

“It was a great invention, but also a Box of Pandora.”

-- E.W. Dijkstra

Source: EWD 1303
http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html
Electrologica X-1

Source: People Behind Informatics, An exhibition in memory of Dahl, Dijkstra, Nygaard
http://cs-exhibitions.uni-klu.ac.at/
Picture: http://cs-exhibitions.uni-klu.ac.at/fileadmin/template/pictures/Dijkstra_electrologica.jpg

vlC: Interrupt Coalescing for Virtual Machine Storage Device IO
E. W. Djikstra

“Halfway the functional design of the X1, I guess early 1957, Bram [J. Loopstra] and Carel [S. Scholten] confronted me with the idea of the interrupt, and I remember that I panicked, being used to machines with reproducible behaviour. How was I going to identify a bug if I had introduced one?”

Source: EWD 1303 http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
E. W. Djikstra

“After I had delayed the decision to include the interrupt for 3 months, Bram and Carel flattered me out of my resistance, it was decided that an interrupt would be included and I began to study the problem.”

Source: EWD 1303 http://www.cs.utexas.edu/users/EWD/transcriptions/EWD13xx/EWD1303.html

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Typical techniques modify one or both of:

- Maximum Interrupt Delay Latency (MIDL)
- Maximum Coalesce Count (MCC)

Source: Mark Smotherman
http://www.cs.clemson.edu/~mark/interrupts.html

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Virtual Interrupts are Different?

• Real HW I/O controllers are embedded systems
• Device emulation executes on general purpose, multi-user, time-shared architectures
• Can’t install timers for 100 microseconds intervals
  – Host would be overwhelmed by interrupt storm
  – Other VMs would be impacted
  – Shouldn’t solve interrupt coalescing for VMs by increasing interrupt rate on host!

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
First Intuition Behind vIC

• Let’s pretend HW IO completions are “timers”
  – But, just can’t program them to our desired rate
  – So, let’s piggyback the ShouldDeliverInterrupt() logic on real HW completion handlers
• HW controllers: deliver when internal timers fire
• vIC: let’s only deliver in line with HW completion
• Motivates using a delivery ratio instead of timer
  – Deliver a virtual interrupt for every $n^{th}$ completion
Delivery Ratio

• Naïve implementation: deliver an interrupt for 1 of every \( n \) HW completions
• Equivalent of the typical \texttt{max coalesce count} (MCC) parameter in HW controllers
• Problem in MCC: limits delivery ratio to be \( 1/n \)
  – E.g. 1/1, 1/2, 1/3, 1/4, etc.
  – Can’t express, say, 80% delivery ratio
• Experiments suggest 1.0->0.5 jump too drastic
Delivery Ratio

• Use two counting parameters (MCC has one)
  1. countUp
  2. skipUp

• Express arbitrary fractional delivery rate

80% delivery: Deliver up to 4, Skip up to 5
Second Intuition Behind vIC

- Suppose a scheme coalesces 2 completions
- With CIF of 32, pipeline remains mostly full
- With CIF of 4, pipeline is half empty!
  ➩ make delivery ratio a function of CIF

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Delivery Ratio: CIF Dependence

Latency vs. CIF

Throughput vs. CIF
Delivery Ratio: CIF Dependence

- Measure dynamic Commands in Flight (CIF)
- Vary delivery ratio \( R \) inversely with CIF

<table>
<thead>
<tr>
<th>CIF</th>
<th>Intr Delivery Ratio ( R ) as %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>100%</td>
</tr>
<tr>
<td>4-7</td>
<td>80%</td>
</tr>
<tr>
<td>8-11</td>
<td>75%</td>
</tr>
<tr>
<td>12-15</td>
<td>66%</td>
</tr>
<tr>
<td>CIF ( \geq ) 16</td>
<td>8 / CIF</td>
</tr>
<tr>
<td>e.g., CIF = 64</td>
<td>12%</td>
</tr>
</tbody>
</table>

Interrupt delivery ratio \( (R) \) as a function of CIF.

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Loose ends

• What if next HW completion never comes?
  – There is always a future I/O when CIF > 0 😊
  – Still, short-circuit to deliver f/ low CIF situations

• What if the hardware completions are too far apart: could cause high latency?
  – Measure and automatically enable/disable vIC

• Trickle I/O
  – Measure and automatically enable/disable vIC
vIC Implementation

- Portable to other hypervisors on any CPU architecture. Also to firmware/hardware
- No floating point
- No int div or RDTSC in critical path
- Increase in the 64-bit VMM:
  - .text: +400 bytes
  - .data: +104 bytes.
- LSI Logic emulation in VMM: <120 new LoC
- IPI coalescing logic in the Vmkernel: 50 new LoC
Results

• Application benchmarks
  – GSBlaster and SQLIOSim
  – Throughput (IOPS) increase by up to 19%
  – Improve CPU efficiency up to 17%

• Let’s look at TPC-C next
  – transaction rate increased by up to 5%
Internal TPC-C Testbed

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>T Diff</th>
<th>Users</th>
<th>IOPS</th>
<th>Intr/Sec</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>No vIC</td>
<td>43.3</td>
<td></td>
<td>80</td>
<td>10.2K</td>
<td>9.9K</td>
<td>7.7ms</td>
</tr>
<tr>
<td>( cifT = 4 )</td>
<td>44.6</td>
<td>+3.0%</td>
<td>90</td>
<td>10.4K</td>
<td>6.4K</td>
<td>8.5ms</td>
</tr>
<tr>
<td>( cifT = 2 )</td>
<td>45.5</td>
<td>+5.1%</td>
<td>90</td>
<td>10.5K</td>
<td>5.8K</td>
<td>9.2ms</td>
</tr>
</tbody>
</table>

Throughput Increased
IOPS Increased
Proportional Latency increase
More Users
Interrupts Decreased

\(^1\)Non-comparable implementation; results not TPC-C\(^{TM}\) compliant; deviations include: batch benchmark, undersized database.

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Dynamic Adaptation (TPC-C)

Virtual interrupt coalescing rate, $R$.
Online adaptation by vIC to burstiness in outstanding IOs of the workload

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO
Dynamic Adaptation (TPC-C)

Virtual interrupt coalescing ratios, $R$, during our TPC-C run.

x-axis log-scale.

vlC: Interrupt Coalescing for Virtual Machine Storage Device IO
vIC Deployment Experience

• vIC default in VMware’s LSI Logic virtual adapter on ESX (since v. 4.0 released 2Q ’09)
• Till now, no performance bug reports
Key Takeaways

• 60-yr old problem revisited
• Encouraging results
  – TPC-C by 5%, other by 18%+
  – Take another look at your interrupt subsystem
  – IPI coalescing very beneficial
• More optimization opportunities exist in vIC
• Change the rules when they weigh you down
  – What about networking?
  – An equivalent of CIF there (TCP window size?)

vIC: Interrupt Coalescing for Virtual Machine Storage Device IO