Devirtualizing Memory in Heterogeneous Systems

Swapnil Haria, Mark D. Hill, Michael M. Swift
University of Wisconsin-Madison

• Accelerators going mainstream
• Benefit from Shared Memory
### 3Ps for Memory Management

<table>
<thead>
<tr>
<th></th>
<th>Physical Addressing</th>
<th>Virtual Addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protection</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Performance</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Programmability</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

How do we get the best for both worlds?
Devirtualized Memory (DVM)

1. (Usually) Allocate data such that its VA == PA!
How is Heap Memory Allocated Today?

Allocation Request ➔ OS allocates VAs ➔ PAs allocated lazily

Virtual AS

0x1000:0x1FFF

Physical AS

0x1800 0x2000
Allocating VA == PA

Allocation Request

OS allocates PAs

OS finds matching VAs

Virtual AS

Physical AS

0x1000:0x1FFF

0x1000:0x1FFF
Devirtualized Memory (DVM)

1. (Usually) Allocate data such that its VA == PA!

2. Exploit PA == VA to mostly skip translation

VM overheads down to ~2% w/o TLB!
## Devirtualizing Memory in Heterogeneous Systems

<table>
<thead>
<tr>
<th></th>
<th>Physical Addressing</th>
<th>Virtual Addressing</th>
<th>Our Proposal (DVM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protection</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Performance</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Programmability</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Today in session **Memory 2 (9 AM)!**