Dynamic Partitioned Global Address Spaces for Power Efficient DRAM Virtualization

Jeffrey Young, Sudhakar Yalamanchili
School of Electrical and Computer Engineering, Georgia Institute of Technology
Talk Outline

- Why Worry About DRAM Power?

- Increasing Memory Efficiency with Virtual DIMMs using Dynamic Partitioned Global Address Spaces (DPGAS)
  - Architectural Support
  - Memory Management
  - Performance Evaluation

- Lessons Learned and Conclusions
Inefficient DRAM Usage Leads to Power Inefficiency

- 1.5% of all U.S. energy costs go to datacenters and costs could double\(^1\)
- DRAM power can consume from 20-30% of total HW budget\(^2\)
  - Increased use of virtualization increases need for more DRAM
  - Projects like RamCloud\(^3\) and in-memory databases lead to increased usage of memory
- DRAM background power hard to reduce due to need to refresh state
- DRAM is overprovisioned due to time-varying workloads.
  - Actual memory requirements can vary with time\(^4\)
  - Data centers typically have low utilization\(^5\)

---

1) EPA Report to Congress on Server and Data Center Efficiency, 2007

---

Photo from http://eetd.lbl.gov
Techniques for Power Efficient DRAM Usage

- What about sharing underutilized DRAM between nodes?
  - Existing techniques have high overhead
    - RDMA is fast but has high set up cost
    - MPI and other high-level sharing mechanisms use OS/network stack
  - Or require custom interconnects
    - Supercomputing clusters typically use custom interconnects with NUMA

- How can we enable DRAM sharing that is high performance and uses commodity infrastructure?
Proposed Approach – Dynamic Partitioned Global Address Spaces (DPGAS)

- Create a “virtual DIMM” abstraction that allows for transparent, low-latency DRAM sharing over commodity interconnects
  - Remote access is handled at hardware layer with OS control path interaction for setup
  - OS handles “control path” setup while “reference path” bypasses traditional networking stack
Proposed Approach – Dynamic Partitioned Global Address Spaces (DPGAS)

- Create a “virtual DIMM” abstraction that allows for transparent, low-latency DRAM sharing over commodity interconnects.
  - Remote access is handled at hardware layer with OS control path interaction for setup.
  - OS handles “control path” setup while “reference path” bypasses traditional networking stack.
Proposed Approach – Dynamic Partitioned Global Address Spaces (DPGAS)

- Create a “virtual DIMM” abstraction that allows for transparent, low-latency DRAM sharing over commodity interconnects
  - Remote access is handled at hardware layer with OS control path interaction for setup
- OS handles “control path” setup while “reference path” bypasses traditional networking stack
Dynamic Partitioned Global Address Spaces (DPGAS)

- **Dynamically managed system-wide global address space**
  - 64-bit physical address spaces dynamically mapped across memory controllers as needed
  - Builds on existing Partitioned Global Address Space (PGAS) model that uses “private” and “shared” memory
    - (UPC, Co-array Fortran, X10, etc.)

- **Integrated network interface and memory mapping unit**
  - Memory mapping integrated into a HyperTransport interface
  - Bridge to commodity or specialized interconnection networks
    - Ethernet used for this work

- **Remote memory accesses built on spill/receive model**
  - One node “spills” requests to remote node with unused DRAM, which “receives” remote requests.
  - OS daemon handles memory allocation and updates to lower-level HW
DPGAS System View

- Portion of the virtual address space mapped to remote physical memory
- Protection issues handled by virtual memory system
  - Bridge mapping handled and coordinated by OS
- “Dynamic” updates allow for flexibility in sharing
Architectural Support – Reference Path

- Address translated into a node address and remote local memory address
- Low latency memory bridge: encapsulation takes 24 – 72 ns in current FPGA implementation
  - Referred to as HyperTransport Ethernet Adapter (HTEA)
- Bridge ➔ 1300-1500 FPGA slices (Virtex 4 FX140)

1) J. Young, et al., A HyperTransport-enabled global memory model for improved memory efficiency, WHTRA '09
## Memory Allocation with DPGAS – Control Path

<table>
<thead>
<tr>
<th>Node A</th>
<th>Node B</th>
</tr>
</thead>
<tbody>
<tr>
<td>App1</td>
<td>App1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Northbridge / SRI</td>
<td>Northbridge / SRI</td>
</tr>
<tr>
<td>Local HT</td>
<td>Local HT</td>
</tr>
<tr>
<td>HTEA</td>
<td>HTEA</td>
</tr>
</tbody>
</table>

- Node A hits memory threshold (pg faults or % of physical memory)
Memory Allocation with DPGAS – Control Path

- Node A hits memory threshold (pg faults or % of physical memory)
- Node A requests to “spill” to Node B via OS daemon
Memory Allocation with DPGAS – Control Path

- Node A hits memory threshold (pg faults or % of physical memory)
- Node A requests to “spill” to Node B via OS daemon
- Node B approves and agrees to “receive” remote accesses from Node A
  - OS or hypervisor updates available memory (possibly with libnuma hints)
  - System Request Interface is updated to direct requests to HTEA
  - HToE mapping table is updated on Node A
  - If memory is to be unshared, Node B OS updates its available physical memory
Memory Allocation with DPGAS – Control Path

- Node A hits memory threshold (pg faults or % of physical memory)
- Node A requests to “spill” to Node B via OS daemon
- Node B approves and agrees to “receive” remote accesses from Node A
  - OS or hypervisor updates available memory (possibly with libnuma hints)
  - System Request Interface is updated to direct requests to HTEA
  - HToE mapping table is updated on Node A
  - If memory is to be unshared, Node B OS updates its available physical memory
- Node A can make remote accesses to Node B’s memory via the HTEA
DPGAS Test Infrastructure

- **Demonstrate how DPGAS can reduce DRAM power inefficiency**
  - Also investigate DPGAS effects on network and DRAM
- **Simulation infrastructure used with NS-3, DRAMSIM, custom C++ code**
  - NS-3 handles event scheduling, network
  - DRAMSIM handles memory access latency, DRAM power
- **2 to 16 node simulations with different levels of DPGAS sharing**
- **Synthetic traces used for this evaluation**
Test Setup

<table>
<thead>
<tr>
<th>Num Nodes</th>
<th>Apps / node</th>
<th>Spill Nodes</th>
<th>Receive Nodes</th>
<th>Mem Size (GB)</th>
<th>Apps / Receive Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4,16</td>
<td>0</td>
<td>0</td>
<td>4,16</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4/8</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6,16</td>
<td>1</td>
<td>1</td>
<td>4,16</td>
<td>2,8</td>
</tr>
<tr>
<td>4</td>
<td>6,16</td>
<td>2</td>
<td>2</td>
<td>4,16</td>
<td>2,8</td>
</tr>
<tr>
<td>8</td>
<td>6,16</td>
<td>4</td>
<td>4</td>
<td>4,16</td>
<td>2,8</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>16/20</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>6,16</td>
<td>8</td>
<td>8</td>
<td>4,16</td>
<td>2,8</td>
</tr>
<tr>
<td>8</td>
<td>6,16</td>
<td>6</td>
<td>2</td>
<td>4 (8 on Recv Nodes), 16</td>
<td>2,8</td>
</tr>
<tr>
<td>16</td>
<td>6,16</td>
<td>14</td>
<td>2</td>
<td>4 (8 on Recv Nodes), 16</td>
<td>2,8</td>
</tr>
</tbody>
</table>

- Synthetic traces represented large memory footprint applications
  - DRAM accesses every 4000 - 4500 cycles for 3000 MHz CPU
  - Random, clustered random, strided access each with 50,000 accesses
  - 1 – 16 applications with ½ of the nodes with 2-8 applications spilling
  - Memory blade scenario has 6 or 14 nodes spilling to 2 nodes
- DRAM timing/power stats match Micron’s MT47H512M8 TwinDie 4 GB DDR2
- 10 Gbps Ethernet network simulated with 200 ns latency
  - Additional component latency drawn from other studies, datasheets

Test Setup (continued)

- **Metrics studied**
  - **Background power for DRAM** – How much power could DPGAS spill/receive save by reducing overprovisioning of DIMMs?
  - **Link and buffering latency** – How much latency is incurred by DPGAS-enabled sharing?
  - **Network utilization** – How does sharing with DPGAS affect demand on a shared 10 Gbps Ethernet link?
  - **Memory Controller Access Latency** – Does DPGAS dramatically increase the local access latency of “receive” nodes?
    - Experiments described in paper – access latency within 2 ns between DPGAS/non-DPGAS tests

- **How much power could DPGAS spill/receive save by reducing overprovisioning of DIMMs?**
- **How much latency is incurred by DPGAS-enabled sharing?**
- **How does sharing with DPGAS affect demand on a shared 10 Gbps Ethernet link?**
- **Does DPGAS dramatically increase the local access latency of “receive” nodes?**
  - Experiments described in paper – access latency within 2 ns between DPGAS/non-DPGAS tests
Impact on DRAM Power Savings

- ½ of all nodes have reduced number of DIMMs
  - Reduces background power – refresh, standby
- Power Savings from Removing one 4 GB DIMM
  - 2 to 19 Watts (4 GB)
  - 1.5 to 16 Watts (16 GB)
  - Savings for a 10,000 core data center would be 3,540 Watts

**DPGAS Impact on Network Latency**

<table>
<thead>
<tr>
<th>Component</th>
<th>Latency (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Northbridge</td>
<td>40</td>
</tr>
<tr>
<td>On-chip memory access</td>
<td>60</td>
</tr>
<tr>
<td>Heidelberg HT Cave Device</td>
<td>45</td>
</tr>
<tr>
<td>HTEA</td>
<td>48</td>
</tr>
<tr>
<td>10 Gbps Ethernet MAC</td>
<td>500</td>
</tr>
<tr>
<td>10 Gbps Ethernet Switch</td>
<td>200</td>
</tr>
<tr>
<td>Average Component Delay</td>
<td>893</td>
</tr>
<tr>
<td>Measured Transmission and Buffering Delay (NS3)</td>
<td>185 - 939</td>
</tr>
</tbody>
</table>

- Network latency calculated based on NS-3 simulations and estimates from other work
- One-way latency varies from 1042 to 1238 ns (4 GB), 1057 to 1593 ns (16 GB), and 1478 to 1832 ns (memory blades)
- Two-way latency is on the order of 2.242 µs for cache line read
DPGAS Impact on Network Utilization

- Link utilization measured for “peak” times when many applications were “spilling” via DPGAS
  - Represents a worst-case scenario for data center machines that are typically underutilized
  
- Utilization ranges from 31.3 MB/s to 555 MB/s (4GB), 31.3 MB/s to 250.65 MB/s (16 GB), and 324 MB/s to 756 MB/s (memory blade).
  - Utilization lower for 16GB case due to more spread out accesses

Related Work

- **Memory Efficiency**
  - Lim, et al. - Memory Blades for disaggregated memory
  - Tolentino, Cameron – Memory Miser OS level support
  - Lefurgy, et al. – DRAM server power and DRAM consolidation

- **PGAS**
  - Software approaches - UPC, X10, Titanium, Gasnet
  - RDMA - Liang ’05 low-level implementation for page swapping
  - RNA Networks – RDMA for high BW, low latency sharing

- **Power and Cost Analysis**
  - Google
  - Lim, et al. – Warehouse Computing
Conclusions

- Introduced Dynamic Partitioned Global Address Spaces as abstraction for *efficient* sharing of memory
  - HyperTransport over Ethernet offers commodity, low-latency substrate that can access “virtual” DIMMs
  - Simulation framework allows for investigation of network and memory

- Low-latency virtual DIMMs enable power savings for time-varying workloads
  - 18% to 49% background power savings result from removing underutilized DIMMs

- Network utilization may require additional network infrastructure for “memory blades”
  - Large-scale memory blade used over 6 Gbps of BW in experiments
Future Work

- Model the effects of DPGAS latency on system performance
  - Longer run-times may lead to increased power draw by system, network
  - DPGAS still has potential for power savings
    - Modern processors geared to overlap computation with DRAM access
    - Remote DRAM access generally much faster than swapping to disk
    - Network hardware to support HToE already exists in data centers

- Add page migration support and evaluation
  - Remote accesses are reduced for most frequently used pages

- Fat nodes versus thin nodes
  - Where can we position dedicated “receive” nodes?
Thank you!

- Questions?
- More information at http://www.ece.gatech.edu/research/labs/casl/hec.html
Modular approach and encapsulation allows software to be portable as processor physical address space grows.

- Extension from the 40-bit to 64-bit physical address
- Creation of a HyperTransport packet which includes a 64-bit extended address
- Map the most significant 24 bits of destination address to a 48-bit MAC address and encapsulation into an Ethernet frame.
Impact of Memory Latency

- **DPGAS causes slight increase in latency for “receive” node**
  - Average DRAM access latency across 2 nodes rose/decreased by 2 ns.
  - DRAM accesses are more evenly split between heavily loaded “spill” and lightly loaded “receive” nodes.

- **High-performance DRAM mapping policy and random traces reduced potential row buffer hit rates**
  - DPGAS might increase latency more for “receive” nodes with high row buffer hit rate

<table>
<thead>
<tr>
<th>Simulation, DRAM Size</th>
<th>Ave. Mem Latency (ns)</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 node, 4/8 GB</td>
<td>54.28</td>
<td>6.21</td>
</tr>
<tr>
<td>2 node, 4 GB</td>
<td>53.06</td>
<td>2.5</td>
</tr>
<tr>
<td>4 node, 4 GB</td>
<td>69.42</td>
<td>5.58</td>
</tr>
<tr>
<td>8 node, 4 GB</td>
<td>66.29</td>
<td>9.5</td>
</tr>
<tr>
<td>16 node, 4 GB</td>
<td>64.35</td>
<td>11.5</td>
</tr>
<tr>
<td>8 node, 4/8 GB</td>
<td>67.74</td>
<td>10.89</td>
</tr>
<tr>
<td>16 node, 4/8 GB</td>
<td>69.98</td>
<td>10.65</td>
</tr>
<tr>
<td>2 node, 16 GB</td>
<td>68.11</td>
<td>12.51</td>
</tr>
<tr>
<td>4 node, 16 GB</td>
<td>68.27</td>
<td>13.28</td>
</tr>
<tr>
<td>8 node, 16 GB</td>
<td>68.72</td>
<td>14.21</td>
</tr>
<tr>
<td>16 node, 16/20 GB</td>
<td>68.17</td>
<td>9.11</td>
</tr>
<tr>
<td>16 node, 16 GB</td>
<td>68.84</td>
<td>7.2</td>
</tr>
</tbody>
</table>

No DPGAS – ½ of nodes had more applications and DRAM
Memory Blade – 2 nodes had more DRAM. Other nodes spill.