TSHMEM: Shared-Memory Parallel Computing on Tilera Many-Core Processors

Bryant C. Lam (speaker)
Alan D. George
Herman Lam

NSF Center for High-Performance Reconfigurable Computing (CHREC), University of Florida
Motivation

- Emergent many-core devices vary considerably in architecture and performance characteristics
  - e.g., scalable mesh networks, ring-bus topologies
- Platforms provide separate APIs to individually leverage underlying hardware resources
  - Difficult to develop efficient cross-platform apps

Objective

- Bridge the gap between many-core hardware utilization and developer productivity

Approach

- **Quantify performance** of many-core devices via application kernels and microbenchmarks
- **Develop TSHMEM** to increase device utilization of various many-core platforms with arch-specific insights
Outline

- Tilera Many-Core
  - Chip Architectures
  - Microbenchmarks
    - Shared Memory Copy
    - UDN Latency
    - TMC Spin/Sync Barriers
- TSHMEM Computing
  - Design Infrastructure
  - Performance Analysis
    - Put/Get Transfers
    - Barrier Synchronization
    - Collectives
  - Application Case Studies
- Conclusions, Q&A
Tilera TILE-Gx Architecture

- 64-bit VLIW processors
- 32k L1i cache, 32k L1d cache
- 256k L2 cache per tile
- Up to 750 BOPS
- Up to 200 Tbps of on-chip mesh interconnect
- Over 500 Gbps memory bandwidth
- 1 to 1.5 GHz operating frequency
- Power consumption: 10 to 55W for typical applications
- 2-4 DDR3 memory controllers
- mPIPE delivers wire-speed packet classification, processing, distribution
- MiCA for cryptographic acceleration
TILE-Gx36 vs. TILE Pro64

- **TILE-Gx8036**
  - 64-bit VLIW processors
  - 32k L1i cache, 32k L1d cache
  - 256k L2 cache per tile
  - Up to 750 billion op/sec
  - Up to 60 Tbps of on-chip mesh interconnect
  - Over 500 Gbps memory bandwidth
  - 1 to 1.5 GHz operating frequency
  - Power consumption: 10 to 55W for typical applications
  - Two DDR3 memory controllers

- **TILE Pro64**
  - 32-bit VLIW processors
  - 16k L1i cache, 8k L1d cache
  - 64k L2 cache per tile
  - Up to 443 billion op/sec
  - Up to 37 Tbps of on-chip interconnect
  - Up to 50 Gbps of I/O bandwidth
  - 700 MHz and 866 MHz operating frequency
  - Power consumption: 19 – 23W @700MHz under full load
  - Four DDR2 memory controllers
**iMesh On-Chip Network Fabric**

**TILEPro64**: MDN (Memory), CDN (Coherence), IDN (I/O), TDN (Tile), UDN (User), STN (Static Network)

**TILE-Gx36**: Five networks instead of six (removed static network)

- MDN replaced by QDN (reQuest) and RDN (Response) dynamic network
  - Memory controller has two QDN (0/1) and RDN (0/1) network connections
  - Reduced congestion in memory accesses from TILEPro
- SDN (Share)
  - Provides accesses and coherency for cache system
- IDN (Internal)
  - Used primarily for communication with external I/O
- UDN (User)
  - Allows user applications to send data directly between tiles
  - UDN and IDN can be accessed by users
TILE-Gx36 Benchmarking

- Why microbenchmark?
  - Determines *empirically realizable* TILE-Gx performance
  - Defines *realistic upper bound* for TSHMEM performance

- Benchmarking Overview
  - Bandwidth of shared memory copy
  - UDN latency varying test pairs of sender/receiver tiles
  - TMC spin-sync barrier primitives
Shared-Memory-Copy Bandwidth

- Bandwidth on iMesh networks to caches and memory controllers
  - Shared memory performance critical for TSHMEM
  - Bandwidth of memory operations influenced by 3 of 5 iMesh networks
    - QDN: memory request network
    - RDN: memory response network
    - SDN: cache sharing network
  - Significant bandwidth performance transitions occur at cache-size limits
    - L1 data cache: 3100 MB/s
    - L2 cache: 2700 MB/s
    - Memory-to-memory performance thereafter
UDN Performance on Gx36

- Tilera User Dynamic Network (UDN)
  - Hardware support for routing of data packets between CPU cores (tiles)

- UDN microbenchmark
  - Measures average latency between two tiles by ping-ponging numerous packets

UDN benchmarks transfer between two tiles:
- **Red** for neighbors
- **Green** for side to side
- **Blue** for corner to corner
UDN Latency

- Average latency of one-way UDN transfers
  - TILEPro64’s slightly faster latency is attributed to 32-bit switching fabric
    - TILE-Gx uses 64-bit fabric; yields roughly twice as much data throughput over TILEPro
  - Low latency of UDN attractive for fast inter-tile communication

<table>
<thead>
<tr>
<th>Type (6x6 area)</th>
<th>Direction</th>
<th>Sender</th>
<th>Receiver</th>
<th>Time (ns) TILE-Gx36</th>
<th>Time (ns) TILEPro64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbors</td>
<td>left</td>
<td>14</td>
<td>13</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>14</td>
<td>15</td>
<td>22</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>up</td>
<td>14</td>
<td>8</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>down</td>
<td>14</td>
<td>20</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>28</td>
<td>27</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>28</td>
<td>29</td>
<td>22</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>up</td>
<td>28</td>
<td>22</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>down</td>
<td>28</td>
<td>34</td>
<td>22</td>
<td>18</td>
</tr>
<tr>
<td>Side-to-Side</td>
<td>right</td>
<td>6</td>
<td>11</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>11</td>
<td>6</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>down</td>
<td>1</td>
<td>31</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>up</td>
<td>31</td>
<td>1</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>23</td>
<td>18</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>18</td>
<td>23</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>down</td>
<td>33</td>
<td>3</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>up</td>
<td>3</td>
<td>33</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td>Corners</td>
<td>down-right</td>
<td>0</td>
<td>35</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>up-left</td>
<td>35</td>
<td>0</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>down-left</td>
<td>5</td>
<td>30</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>up-right</td>
<td>30</td>
<td>5</td>
<td>32</td>
<td>33</td>
</tr>
</tbody>
</table>
TMC Spin/Sync Barriers

- Barrier primitives provided by Tilera’s TMC library
  - Spin uses spinlock, discouraging process context switching
    - Sync barrier rectifies this with performance penalty
  - Barrier performance on TILEPro is order of magnitude slower than TILE-Gx
    - TILEPro barriers potentially too slow for use in SHMEM

![TMC Barrier Performance](chart.png)
TSHMEM Overview

HPC acceleration with SHMEM on many-core processors

Investigate SHMEM reference design directly over *Tilera TILE-Gx* architecture and libraries

- Stay true to SHMEM principles
  - High performance with low overhead
  - Portability with easy programmability
- Maximize architectural benefits
  - Tile interconnect, mPIPE, MiCA
- Extend design to multi-device systems
  - Evaluate interconnect capabilities
  - Explore optimizations to point-to-point transfers and collectives

OpenSHMEM and TSHMEM

Achieved

- Dynamic symmetric heap management
- Point-to-point data transfer
- Point-to-point synchronization
- Barrier synchronization
- Broadcast, Collection, Reduction
- Atomics for dynamic variables
- Extension to multiple many-core devices

Ongoing

- Optimizations for multiple many-core
- Exploration of new SHMEM extensions

TILERA®

TSHMEM with OpenSHMEM API

<table>
<thead>
<tr>
<th>Setup</th>
<th>Data Transfers</th>
<th>Sync</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tilera Libraries&lt;br&gt;(TMC, gxio, etc.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Device Functionality</td>
<td></td>
</tr>
</tbody>
</table>

TSHMEM reference design on TILE-Gx36
TSHMEM Initialization

Running TSHMEM applications

- Compile C or C++ program with TSHMEM library
  - *tile-gcc* provided as cross compiler for TILE-Gx
- Transfer and run program using *tile-monitor* and *shmemrun* executable launcher

TSHMEM setup and initialization – *shmemrun* and *start_pes()*

- Initializes TMC Common Memory (CMEM) for shared memory allocation
- Initializes UDN for barrier sync and basic inter-tile communications
- Creates processes and binds them to unique tiles
- Allocates shared memory and broadcasts pointer
- Synchronizes PEs before exiting routine
Put/Get Transfers – Design

- Template functions generated with macro definitions
  - Dynamic symmetric variable transfers
    - Performed via `memcpy` and `shmeme_ptr` into TMC common memory
  - Point-to-point transfers for static symmetric variables
    - Handled via UDN interrupts and dynamic shared memory buffer allocation
TSHMEM dynamic performance closely matches shared-to-shared performance profile for both TILE-Gx36 and TILEPro64 devices.
TSHMEM static transfers attempt to offload request if possible, otherwise a temporary buffer is necessary and drastically degrades performance.
Barrier Sync. – Design

- Barrier uses UDN for messages
  - SHMEM allows for barriers with overlapping active sets
    - Difficult to enable overlapping active sets with TMC barriers
  - Barrier primitives for TILEPro too slow

- Barrier design leverages split-phase barrier internally but externally appears as blocking
  - Semantics for shmem_barrier is blocking
    - Cores (tiles) linearly forward notify message until all receive it
      - Active-set information transmitted to avoid overlap errors from multiple barriers
    - Lead tile broadcasts release message to leave barrier routine
  - Significant performance improvements with increasing number of tiles on TILE-Gx
    - 1.5 µs latency at 36 tiles
TSHMEM barriers leverage UDN for better scaling than most Tilera TMC barriers for TILE-Gx36 and TILEPro64
Collectives – Types

- **Broadcast**
  - One-to-all transfer
    - Active-set tiles except root tile receive data from root tile’s memory

- **Collection**
  - All-to-all transfer
    - Active-set tiles linearly concatenate their data

- **Reduction**
  - All-to-all transfer
    - Active-set tiles perform reduction arithmetic operation on their data
Pull-based broadcast up to 46 GB/s aggregate bandwidth at 29 tiles; 37 GB/s aggregate at 36 tiles
Performance – FCollect, Reduction

- Results are for naïve fast-collect and reduction; additional algorithms planned for testing.

**Fast-Collect on TILE-Gx36**

**Reduction on TILE-Gx36**
App Case – 2D-FFT

- TILE-Gx executes order of magnitude faster than TILEPro due to *improved floating-point support*

- Speedup at 32
  - TILE-Gx: 5
  - TILEPro: 12
App Case – CBIR

- **Content-based image retrieval** with searches based on feature extraction of input image

- **Speedup at 32**
  - TILE-Gx: 25
  - TILEPro: 27
Conclusions

- **TSHMEM is a new OpenSHMEM library focused on fully leveraging Tilera many-core devices**

- Microbenchmarks define the expected and maximum realizable device performance
  - TSHMEM reaches toward these microbenchmark results with *very little overhead* for variety of functions
  - UDN leveraged for SHMEM barriers due to lacking performance of TMC barrier primitives on TILEPro

- Performance, portability, scalability demonstrated via two app case studies
Thanks for listening!

Questions?
References


