Replica: A Wireless Manycore for Communication-Intensive and Approximate Data

Vimuth Fernando$^1$, Antonio Franques$^1$, Sergi Abadal$^2$, Sasa Misailovic$^1$, Josep Torrellas$^1$

$^1$University of Illinois at Urbana-Champaign

$^2$Universitat Politècnica de Catalunya

CCF-1629431
CCF-1703637
Motivation

Computations with broadcast and fine-grained data sharing do not scale well in shared-memory multiprocessor architectures.

Master Thread

counter++;  
barrier_wait(b)

Worker Threads

barrier_wait(b)  
x = counter;
Manycore with a Network on Chip

Master Thread

counter++;

barrier_wait(b)

Worker Threads

barrier_wait(b)

x = counter;
WiSync: On-chip Wireless Communication for Synchronization

WiSync: On-chip Wireless Communication for Synchronization

WiSync: On-chip Wireless Communication for Synchronization

Master Thread

barrier_wait(b)
WiSync: On-chip Wireless Communication for Synchronization

Master Thread
barrier_wait(b)
WiSync: On-chip Wireless Communication for Synchronization

Master Thread
barrier_wait(b)
In WiSync, ordinary data uses the wired network

Master Thread
counter++;

Worker Threads
x = counter;
Can we leverage wireless communication to speed-up transfers of ordinary shared data?
Contributions: Replica

• A manycore architecture and software interface for wireless communication (sync and ordinary data)

• Hardware innovations
  • Adaptive wireless protocol
  • Selective packet dropping

• Software innovations
  • Transformations and tools to adapt applications to wireless
  • Optimizations for approximate computing

• For 64 core execution: speedup applications by 1.89x over a conventional multicore
Replica Architecture

Core

Controller

L1 Cache

Directory

L2 Cache

Wired network

Transceiver

Antenna

BMem

32-512 KB
Example

\[
\text{int}^* \ A = (\text{int}^*) \ \text{wireless_malloc}(\text{size})
\]
Write
Write

Atomic update of local and all remote BMems
Broadcast Memory for ordinary data

Master Thread

counter++;

Core0

... counter:0 ...

Core3

... counter:0 ...

Core15

... counter:0 ...

(counter:0)
Broadcast Memory for ordinary data

Master Thread
  counter++;

Core0
  ...
  counter:1
  ...

Core3
  ...
  counter:1
  ...

Core15
  ...
  counter:1
  ...
Replica: Wireless channel

- One channel shared by all the cores
- Everyone receives what one core transmits
- Only one core can transmit at a given time
  - ensures the same order of updates across all BMems
Reads

Read: Local access
Challenges

• Limited wireless bandwidth: Only one core can transmit at a time

• Bounded size of the BMem: Arbitrary data structures will not fit
Solutions

• Limited wireless bandwidth: Only one core can transmit at a time
  • Adaptive wireless protocol
  • Selective message dropping
  • Approximate transformations to use less bandwidth

• Bounded size of the BMem: Arbitrary data structures will not fit
Solutions

• Limited wireless bandwidth: Only one core can transmit at a time
  • Adaptive wireless protocol
  • Selective message dropping
  • Approximate transformations to use less bandwidth

• Bounded size of the BMem: Arbitrary data structures will not fit
  • Software transformations to fit most important structures in BMem
  • Approximate transformations to use BMem effectively
  • Tools to identify/autotune highly-shared data structures
Wireless Protocol

• Wireless protocol organizes the accesses to the wireless network

• Two wireless protocols can be used based on application behavior
  • Broadcast Reliability Sensing protocol (BRS)
  • Token Ring protocol
## Wireless Message

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>C</th>
</tr>
</thead>
</table>

4 cycles at 20Gb/s*

Broadcast Reliability Sensing Protocol (BRS)

- Start sending message if the medium is free
- Two cores starting at the same time results in a collision
Broadcast Reliability Sensing Protocol (BRS)

0 1 2 3 4 5

Time

Check if medium is free
Check if collision occurred
Broadcast Reliability Sensing Protocol (BRS)
Broadcast Reliability Sensing Protocol (BRS)

- No wasted cycles if low contention
- Lot of collisions if high contention
Token Ring Protocol

• Pass conceptual token among cores
• Can send wireless message only if the core *owns* the token
Token Ring Protocol

Core 0

Core 1

Core 2

Core 3

No wasted cycles if high contention

Unnecessary delays if low contention
Adaptive Wireless Protocol

• In Replica, the utilization of the wireless network vary across applications and within an application
  • Sparse traffic – BRS
  • Bursty traffic – Token Ring

• Replica uses an adaptive dynamic protocol that switches between the two by observing communication behavior
  • Number of collisions
  • Number of skipped token slots
Approximate transformations to use less bandwidth

• Every write to data in the BMem results in a message being broadcasted

• We can reduce the pressure on the network by skipping some of the writes
  • Reducing communication at the cost of accuracy

• Many programs have shared data structures that are amenable to approximations
Opportunity in Replica: Dropping Messages

• All cores see the contention in the wireless network
• Can drop messages while maintaining the same state across all cores
Approximate stores

• Developers indicate approximable data structures
  \texttt{approx\_wireless\_malloc(size)}

• Stores to approximable variables are dropped if they cannot access the wireless network before a given threshold
Approximate stores

• Developers indicate approximable data structures
  `approx_wireless_malloc(size)`

• Stores to approximable variables are dropped if they cannot access the wireless network before a given threshold
Approximate transformations to use less bandwidth

• We used the approximate stores to implement primitives such as **Approximate Locks**
  • Spin lock that gives up trying to acquire a lock after some time

• Existing approximate techniques that reduce communication more useful in this resource constrained setting
  • Example: Skipping negligible updates to shared data
Addressing Bounded size of the BMem

- Software transformations to fit most important structures in BMem
- Approximate transformations to use BMem effectively
  - Example: Numerical precision reduction, Cyclic collection update
- Tools to identify highly-shared data and tune the application

See the paper for more details
Evaluation

• Cycle-level architectural simulations using Multi2sim
  • 64 core chip
  • 32-512 KB BMem
  • 2D Mesh wired network

• Applications
  • 10 benchmarks from PARSEC and CRONO
  • Multiple domain: Scientific simulations, computer vision, and graph applications
## Benchmarks: Communication Patterns

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Sharing Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water</td>
<td>Broadcast</td>
</tr>
<tr>
<td>BFS</td>
<td>Irregular: many-to-many</td>
</tr>
<tr>
<td>Bodytrack</td>
<td>One-to-many</td>
</tr>
<tr>
<td>SSSP</td>
<td>Irregular: many-to-many</td>
</tr>
<tr>
<td>Canneal</td>
<td>Irregular</td>
</tr>
<tr>
<td>CC</td>
<td>Irregular: many-to-many</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>One-to-many, reduction</td>
</tr>
<tr>
<td>Pagerank</td>
<td>Irregular: many-to-many</td>
</tr>
<tr>
<td>Community</td>
<td>Irregular: many-to-many</td>
</tr>
<tr>
<td>Volrend</td>
<td>One-to-many</td>
</tr>
</tbody>
</table>
BMem for sync variables (WiSync)

1.4x speed up over conventional wired multicore (Geometric Mean)
BMem for shared data

1.76x speed up (Geometric Mean)
## Benchmarks: Approximation

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Sharing Pattern</th>
<th>Approximations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water</td>
<td>Broadcast</td>
<td>Precision reduction and Approximate Locks</td>
</tr>
<tr>
<td>BFS</td>
<td>Irregular: many-to-many</td>
<td>Approximate Stores</td>
</tr>
<tr>
<td>Bodytrack</td>
<td>One-to-many</td>
<td>Approximate Stores</td>
</tr>
<tr>
<td>SSSP</td>
<td>Irregular: many-to-many</td>
<td>Approximate Stores</td>
</tr>
<tr>
<td>Canneal</td>
<td>Irregular</td>
<td>Approximate Locks</td>
</tr>
<tr>
<td>CC</td>
<td>Irregular: many-to-many</td>
<td>Approximate Stores</td>
</tr>
<tr>
<td>Streamcluster</td>
<td>One-to-many, reduction</td>
<td>Cyclic collection updates</td>
</tr>
<tr>
<td>Pagerank</td>
<td>Irregular: many-to-many</td>
<td>Skipping negligible updates</td>
</tr>
<tr>
<td>Community</td>
<td>Irregular: many-to-many</td>
<td>Approximate Stores</td>
</tr>
<tr>
<td>Volrend</td>
<td>One-to-many</td>
<td>Approximate Stores</td>
</tr>
</tbody>
</table>
BMem for shared data + approximations

On average 1.89x speed up
Energy and area

- Since faster execution: 33% energy reduction
- Replica components: 9% of total energy consumed
Energy and area

- Since faster execution: 33% energy reduction
- Replica components: 9% of total energy consumed

- 15% increase in the area
  - 11% from the BMem + 4% from the transceiver/antenna
  - Using the same area to increase the L2 cache has little impact on performance (1.04x speedup)
Also in the paper

• Scalability analysis
• Power evaluation
• Area consumption
• Architecture sensitivity analysis
• Effectiveness of profiler and autotuner
• Statistics on developer effort to adapt programs
Conclusions

• Replica: a manycore that uses a wireless NoC to communicate ordinary data

• Hardware and Software innovations
  • Adaptive wireless protocol
  • Selective packet dropping
  • Software techniques to identify and allocate shared data in BMem
  • Software transformations for approximate computing

• Effectively supports communication-intensive computations

• Average speedup of 1.89x over conventional machines