xBGAS: A Global Address Space Extension on RISC-V for High Performance Computing

Xi Wang\textsuperscript{1}, John D. Leidel\textsuperscript{2,3}, Brody Williams\textsuperscript{3}, Alan Ehret\textsuperscript{4}, Miguel Mark\textsuperscript{4}, Michel A. Kinsy\textsuperscript{4}, Yong Chen\textsuperscript{3}

RISC-V International Open Source Laboratory\textsuperscript{1}, Tactical Computing Laboratories\textsuperscript{2}, Texas Tech University\textsuperscript{3}, Texas A&M University\textsuperscript{4}
Outline

• Introduction

• Background

• xBGAS Design and Philosophy

• Evaluation

• Summary
What is xBGAS?

• **Extended Base Global Address Space (xBGAS)**

• **Goals:**
  • Provide extended addressing capabilities without ruining the base ABI
    • EG, RV64 apps will still execute without an issue
  • Extended addressing must be flexible enough to support multiple target application spaces/system architectures
    • Traditional data centers, clouds, HPC, etc..
  • Extended addressing must not specifically rely upon any one virtual memory mechanism
    • EG, provide for object-based memory resolution
  • RISC-V based large-scale HPC systems and datacenters

• **What is xBGAS NOT?**
  • ...a direct replacement for RV128
Why xBGAS

• Traditional message passing paradigm limitations
  • User library overhead, driver overhead
  • Optimized for large data transfers of regular workloads
  • Limited scalability for Exascale-class systems

• Little hardware/uArch support in existing PGAS paradigms
  • LBNL/UCB: UPC
  • PNNL: Global Arrays/ARMCI
  • Cray: Chapel
  • OpenSHMEM
Why xBGAS?

OpenSHMEM Get Profiling (OSHMEM 3.0.4 + UCX 1.6.0)
Why xBGAS?

Existing address space extensions have limited Generalizability

- RISC-V RV128 & Cray T3D/T3E projects

- Customized OS
  - e.g. UNICOS for T3D/T3E

- Customized ABI
  - Non-portable to existing binaries

- Complicated Microarchitecture Ext.
  - Extended width of registers, data path, HW address translation support, etc.
Why xBGAS?

Applicability

• **HPA-FLAT**
  • High performance analytics flat addressing
  • For extremely large datasets that are too difficult/time consuming to shard

• **MMAP-IO**
  • Map storage tiers into address space
  • Potential for object-based addressing

• **Cloud-BSP**
  • Potential for global object visibility for in-memory cloud infrastructures (Spark)
  • Reduce the time/cost to port Java to a full 128-bit addressing model

• **Security**
  • Fine grained, tagged security extensions to base addressing model
  • Tags are stored/maintained as ACL’s for secure memory regions

• **HPC-PGAS**
  • High Performance Computing: Partitioned Global Address Space
  • The focus of this paper
xBGAS ISA Extension

xBGAS Instructions are split into
- Ext. address management
- Base integer load/store
- Raw integer load/store
- Remote atomic instructions
xBGAS Addressing Model

- **Extended Addr**: Serves as *namespace* (object ID) to specify target remote node
- **Base Addr**: Standard 64-bit address to locate data in remote memory
- *No complicated address translations or 128-bit register/data-path needed*
xBGAS Architecture

A Processor in Node 1

- xBGAS Core
- L1 D-$
- L1 I-$
- L2 Cache (LLC)
- Mem Interface
- NLB
- Local Memory
- Net Interface
- Coherent Memory System

A Processor in Node 2

- xBGAS Core
- L1 I-$
- L1 D-$
- L2 Cache (LLC)
- Mem Interface
- NLB
- Local Memory
- Net Interface
- Coherent Memory System

Remote Access

From Core ①
② Remote Access
From Core ②
xBGAS Architecture

- Detailed Request path and data flow
xBGAS Architecture

- xBGAS uses the upper 64/32-bits of an extended address as the Namespace of shared data objects
  - Serves as data object indices

- A Namespace Lookaside Buffer (NLB) is introduced to translate extended addresses
  - Provides a mapping between namespace and remote node ID/physical location

<table>
<thead>
<tr>
<th>NLB of Node 1</th>
<th>NLB of Node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>Node ID</td>
</tr>
<tr>
<td>0x90df</td>
<td>3</td>
</tr>
<tr>
<td>0xbbbf</td>
<td>4</td>
</tr>
<tr>
<td>0x1111</td>
<td>8</td>
</tr>
<tr>
<td>0x0088</td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Ext. Addr (Tag)

- Bit [127:64] in RV64
- Bit [63:32] in RV32

Memory System Design & Optimization

Namespace

Node ID

0x90df
0xbbbf
0x1111
0x0088

0x90df
0xbbbf
0x1111
0x0088

...
xBGAS Runtime and Memory Management

xBGAS Broadcast Example

```c
#include "xbtime.h"
int main(){
    /* Init */
    xbttime_init();
    /* Shared memory allocation */
    int* p = (int*)xbtime_malloc(2*sizeof(int));
p[0] = xbttime_mtype();
p[1] = p[0];
    xbttime_barrier();
    /* xBGAS broadcast. Arguments: dest, src, nelems, stride, root */
    xbttime_int_bcast(p, p, 1, 1, 0);
    /* Sync */
    xbttime_barrier();
    xbttime_close();
    return 0;
}
```

P[0] = PE ID
P[1]: unused

Symmetric (8-byte per PE)

PE0 0 Shared Memory Slot 0
PE1 1 Shared Memory Slot 0
PE2 2 Shared Memory Slot 0
PE7 7 Shared Memory Slot 0

.......

TACTICAL COMPUTING LABS
xBGAS Software Stack

- **APP**
  - xBGAS Applications and benchmarks

- **Runtime**
  - xBGAS Runtime library (APIs to manage the remote memory accesses)

- **Compiler Toolchain**
  - xBGAS GNU and LLVM toolchain support to generate the xBGAS binaries.

- **Simulator**
  - RISCV Spike + Sandia SST simulation infrastructures
xBGAS Tools & Implementations

• Software
  • xBGAS compiler toolchains
    • Extended from GNU 8.3.0 & LLVM 8.0.0
  • xBGAS runtime library
    • Provides PGAS-style programming model written in ANSI C
  • xBGAS simulation infrastructure
    • Built upon the RISC-V Spike simulator and Sandia Structural Simulation Toolkit (SST)

• Hardware
  • Open-source BRISC-V Design Space Exploration Platform
  • Microarchitecture extension based on 7-stage in-order pipeline RISC-V cores
  • Synthesis is performed using Quartus Prime version 18.0.0 and Cyclone V (5CSEMA5F31C6) FPGA
## xBGAS Evaluation

### xBGAS HW Overhead Against the BRISC-V

<table>
<thead>
<tr>
<th>Resources</th>
<th>Baseline</th>
<th>xBGAS</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic utilization</td>
<td>1,088</td>
<td>1,783</td>
<td>63.87 %</td>
</tr>
<tr>
<td>(ALMs) Total registers</td>
<td>586</td>
<td>1798</td>
<td>206.82 %</td>
</tr>
<tr>
<td>Non-regFile registers</td>
<td>586</td>
<td>774</td>
<td>32 %</td>
</tr>
<tr>
<td>Total block memory bits</td>
<td>2,048</td>
<td>3,072</td>
<td>50 %</td>
</tr>
<tr>
<td>Fmax</td>
<td>78.7MHz</td>
<td>77.7MHz</td>
<td>1.2 %</td>
</tr>
<tr>
<td>Average IPC</td>
<td>0.63</td>
<td>0.63</td>
<td>0 %</td>
</tr>
</tbody>
</table>
xBGAS Evaluation

Simulation Environment Configuration

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base ISA</td>
<td>RV64I</td>
</tr>
<tr>
<td>Node &amp; Core</td>
<td>64 Nodes, 1 Core/Node, 2 GHz</td>
</tr>
<tr>
<td>CPU $</td>
<td>8-Way, 16-KB L1, 8-MB L2</td>
</tr>
<tr>
<td>NLB</td>
<td>Fully associative, 16 KB, 512 Entries</td>
</tr>
<tr>
<td>Memory</td>
<td>DDR4, 2 GB per</td>
</tr>
<tr>
<td>Network</td>
<td>Node 2D-meshed NoC, 32-bit FLIT</td>
</tr>
</tbody>
</table>
Performance

- On average, xBGAS increases performance of the tested workloads by 21.96% (up to 37.29%)

- Performance gain attributed to the reduction of software-based overheads
Software Overhead Reduction

xBGAS significantly reduces the software cost of accessing a remote register-width data element to 9.7% of that demonstrated by OpenSHMEM.

Implies a software overhead reduction of 78.43%
Payload Transfer Analyses

- xBGAS provides lower RMA latency
  - Optimizes both fine/coarse-grain request granularities
  - Applicable to both regular/irregular workloads
  - Relying solely on individual ISA-level RMA instructions can amplify software overheads
### Portability Analyses

#### API Mappings

<table>
<thead>
<tr>
<th>OpenSHMEM</th>
<th>xBGAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>shmem_init()</td>
<td>xbrtime_init()</td>
</tr>
<tr>
<td>shmem_malloc()</td>
<td>xbrtime_malloc()</td>
</tr>
<tr>
<td>shmem_ulonglong_get()</td>
<td>xbrtime_ulonglong_get()</td>
</tr>
<tr>
<td>shmem_int_put()</td>
<td>xbrtime_int_put()</td>
</tr>
<tr>
<td>shmem_bcast_64()</td>
<td>xbrtime_int64_broadcast()</td>
</tr>
<tr>
<td>shmem_barrier_all()</td>
<td>xbrtime_barrier()</td>
</tr>
<tr>
<td>shmem_free()</td>
<td>xbrtime_free()</td>
</tr>
<tr>
<td>shmem_finalize()</td>
<td>xbrtime_finalize()</td>
</tr>
</tbody>
</table>

---

On average, only 4.99% of the lines of a program need to be modified to port benchmarks from OpenSHMEM to xBGAS.
Current and Future work

• Beyond our investigation of xBGAS as a means for improving the HPC-PGAS application domain, we have also been investigating:

  • The potential of xBGAS-accelerated distributed file systems
    • Utilizing xBGAS to improve metadata and small, random I/O performance

  • xBGAS-based microarchitecture security
    • Integration of existing data protection mechanisms with the extended addressing capabilities of xBGAS to build scalable enclave systems

  • Memory centric computing systems
    • Built upon xBGAS to provide high-performance access to centralized, heterogenous memory resources
Summary

• xBGAS demonstrates the impacts on performance of ISA and micro-architecture support for applications with distributed shared memory (DSM).
  • PGAS models, MPI RMA, etc.

• The proposed xBGAS extended addressing methodology
  • Has no need for customized ABIs or operating systems
  • Is scalable to large-scale HPC systems and datacenters
  • Is applicable to diverse application domains such as HPC-PGAS, security, cloud BSP, memory-mapped I/O, etc.
Acknowledgements

• The xBGAS project is a collaboration between
  • Texas Tech University
  • Tactical Computing Labs
  • Texas A&M University
  • Lawrence Berkeley National Laboratory
  • Massachusetts Institute of Technology
  • University of Cambridge
xBGAS Specification & Codebases

- xBGAS Spec: [https://github.com/tactcomplabs/xbgas-archspecc](https://github.com/tactcomplabs/xbgas-archspecc)
- xBGAS Toolchain: [https://github.com/tactcomplabs/xbgas-tools](https://github.com/tactcomplabs/xbgas-tools)
- xBGAS ISA Tests: [https://github.com/tactcomplabs/xbgas-asm-test](https://github.com/tactcomplabs/xbgas-asm-test)
- xBGAS Runtime: [https://github.com/tactcomplabs/xbgas-runtime](https://github.com/tactcomplabs/xbgas-runtime)
- xBGAS Benchmarks: [https://github.com/tactcomplabs/xbgas-bench](https://github.com/tactcomplabs/xbgas-bench)

We welcome comments/collaborators!
xBGAS: A Global Address Space Extension on RISC-V for High Performance Computing

Xi Wang\(^1\), John D. Leidel\(^{2,3}\), Brody Williams\(^3\), Alan Ehret\(^4\), Miguel Mark\(^4\), Michel A. Kinsy\(^4\), Yong Chen\(^3\)

RISC-V International Open Source Laboratory\(^1\), Tactical Computing Laboratories\(^2\), Texas Tech University\(^3\), Texas A&M University\(^4\)
x3GAS