Development and Extension of Atomic Memory Operations in OpenSHMEM

Pavel Shamis, ORNL
Manjunath Gorentla Venkata, ORNL
Stephen W. Poole, ORNL
Swaroop Pophale, Mellanox
Michael Dubman, Mellanox
Richad Graham, Mellanox
Dror Goldenberg, Mellanox
Gilad Shainer, Mellanox
Goal of The Paper

• Highlight the challenges of implementing AMOs in OpenSHMEM

• Start a discussion with the community about potential solutions and extensions for AMOs in OpenSHMEM
Challenges

1. Underlying hardware does not support full set of atomic operations
   - Common issue for new technologies that aim to support OpenSHMEM (Mellanox Connect-X2, Cray Gemini ASIC)

2. Multiple network devices
   - Two or more NICs on a node

3. Heterogeneous systems with multiple types of devices
   - Very common (!): Shared Memory + NIC
Challenges – Common Denominator

• The typical solution – switch to a “common” AMO mechanism
  – Typically results in software based implementation of AMOs
  – Some implementations use two-sided semantic (!)
Potential Solution - 1

• Hints by programmer
  – Programmer declares ahead which AMOs are used by application (e.g. command line option, etc)

• Motivation
  – If programmer uses only a limited set of AMOs support by HW, OpenSHMEM can disable the software AMOs

• Pros/Cons
  + Requires only minimal changes to the application or runtime parameters
  – Programmer has to to review the code and identify list of used AMOs
  – Still no guaranty that OpenSHMEM implementation will be able to provide optimized AMOs
  – Does not address all the challenges (only #1)
Potential Solution - 2

• Hints by the OpenSHMEM library implementation
  – OpenSHMEM provides an indication about supported / optimized AMOs

• Motivation
  – If an application detects that some operations are unsupported it may replaces these with an alternative flow

• Pros/Cons
  + Simplifies the OpenSHMEM library development and potentially enables efficient utilization of underlying hardware capabilities
  – The burden of AMO management is upon the application developers
  – May not be portable
  – Does not address all the challenges (only #1)
Potential Solution - 3

• Manage a different memory region for each variation of AMOs
  – Assignment of a different symmetric heap/storage for each atomic provider (hardware, software, etc)

• Motivation
  – Ensure that AMO performance is not degraded when there are multiple AMO providers

• Pros/Cons
  + Address all the challenges
  – The burden of AMO and memory storage management is upon the application developers; developer is responsible to enquire about different groups of AMOs, allocate and manage AMOs over a particular region of memory
Potential Solution - 4

• Leveraging already existing AMOs in order to implement operations that are not directly supported by hardware
  – For example implementation of 32bit atomics through 64bit atomic GET and CSWAP

• Motivation
  – Avoid fallback to “slow” software based atomics for all AMOs

• Pros/Cons
  + The approach is transparent to user
  – Does not address all the challenges
System Configuration

- HP ProLiant DL380p system (ORNL’s ESSC)
- Two compute nodes, each with two Intel Xeon E5-2650 CPUs, for a total of 16 CPU cores and 32 threads.
- Compute nodes are interconnected with Mellanox ConnectX-3 VPI HCA connected back-to-back (no switch)
- CentOS release 6.5 with MLNX-OFED-2.2-1.5.5 and OpenSHMEM-UCCS v0.3
- OSU OpenSHMEM Atomic Test v4.4.
  - Used to evaluate the blocking atomics: FADD, FINC, SWAP, CSWAP
  - Checks how these atomic operations are affected by the quality of the AMO implementation by measuring the full round trip communication latency.
AMO Algorithm for 32bit AMOs Using 64bit AMOs

• Initiator atomically issues atomic get (Fetch-and-add 0)
• Once value is fetched, it updates (based on the requested operation) the relevant bits
• Use CSWAP with previously fetched value as a compare argument and the updated value as the swap argument
• If the fetched value is identical to previously fetched value – Success, otherwise - Retry
Summary

• Introduction of new AMOs (without hardware support) may have side-effects and negatively affect the performance of already existing AMOs in OpenSHMEM

• OpenSHMEM community must closely collaborate with hardware vendors to ensure that new atomic operations in OpenSHMEM are supported at the hardware level and the gap between software requirements and hardware AMOs support does not increase

• OpenSHMEM community has to provide a mechanism to enable the coexistence of software and hardware-based AMOs
Acknowledgements

This work was supported by the United States Department of Defense & used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory.

Empowering the Mission
Questions ?