CHAPEL 1.24.1 RELEASE NOTES: INFINIBAND OPTIMIZATIONS

Chapel Team
April 15, 2021
Background

• Historically, we have focused primarily on improving performance for Cray networks
  • Intent was to ensure we had the right language features/semantics first, then prioritize other networks

• Recently, we have been focusing on improving performance for InfiniBand (IB) networks
  • We use the GASNet communication library with the ibv conduit to target InfiniBand (gasnet-ibv)
INFINIBAND OPTIMIZATIONS

Background

- Memory must be registered with the network in order to do one-sided GETs/PUTs (RDMA)
  - gasnet-ibv supports two registration modes:
    - Static: All memory is registered at startup—fast communication, but hurts NUMA affinity and leads to long startup times
    - Dynamic: Memory is registered at communication time—can add overhead, but good NUMA affinity and fast startup
- Chapel defaults to dynamic registration to get good NUMA affinity and fast startup times
  - We believe this is the right choice for most users getting started
    - Have recommended static registration to some users with certain communication-heavy idioms
  - Ideally, we just want to have one mode with no, or few, downsides
- Late in the 1.24 release cycle, we identified some pre-existing performance issues on InfiniBand at scale
  - Diagnosed the root causes, leading to improvements for both static and dynamic registration
    - These improvements motivated April's 1.24.1 release
INFINIBAND STATIC REGISTRATION IMPROVEMENTS
BACKGROUND AND THIS EFFORT

**Background:** Registration will fault memory in if it hasn’t been already
- gasnet-ibv static registration is a serial operation, which meant fault-in was serial
  - This was slow and resulted in poor NUMA affinity
  - In practice when allocating memory, the allocation would live on a single NUMA domain, limiting memory bandwidth

**This Effort:** Parallelize and interleave memory fault-in prior to registration
- Interleaving improves memory bandwidth by spreading memory accesses across NUMA domains
- Parallelizing speeds up fault-in time

INFINIBAND STATIC REGISTRATION
Impact

- Improved performance for NUMA-sensitive codes under static registration (gn-ibv-fast)
INFINIBAND STATIC REGISTRATION

Impact

- Faster startup time

- Startup time is faster than before, but still linear with respect to the amount of memory
  - On nodes with 1 TB of memory, startup takes ~60 seconds
    - Most of this time is spent registering memory, not faulting it in
    - May be able to improve this, though static registration will always have more startup overhead compared to dynamic
INFINIBAND DYNAMIC REGISTRATION IMPROVEMENTS
INFINIBAND DYNAMIC REGISTRATION

Background and This Effort

**Background:** gasnet-ibv dynamic registration only registers memory at communication time

- Results in fast startup, but registration cost can limit communication performance
- NUMA affinity is based on user first-touch rather than as a side-effect of registration
- Requires tracking which memory regions are currently registered

**This Effort:** Identified bottleneck in registration tracking code that limited performance and scalability

- Core issue was that we were running out of dynamic registration entries
  - Number of entries was hard-coded to what older generation networks could support
- Collaborated with the GASNet team to resolve this issue
  - Number of entries is now based on execution-time query of hardware capabilities
INFINIBAND DYNAMIC REGISTRATION

Impact

- Significant performance improvements for codes with large point-to-point communication patterns
INFINIBAND DYNAMIC REGISTRATION

Impact

- Significant performance improvements for codes with all-to-all communication patterns
INFINIBAND DYNAMIC REGISTRATION

Impact

• Significant performance improvements for Arkouda
INFINIBAND OPTIMIZATIONS

Next Steps

• Explore ways to speed up static registration, possibly by parallelizing it
  • Consider supporting a mode that runs a process per NUMA domain

• Continue to improve dynamic registration performance
  • ISx and some other communication-intensive applications still lag behind static registration

• Look at using On-Demand-Paging (ODP) as an alternative registration mechanism
  • Hardware/firmware takes care of registration on-demand rather than tracking in software
  • Current prototype hangs
    – Needs more investigation and collaboration with the GASNet team

• Investigate experimental ucx conduit
  • Unified Communication X (UCX) is likely the future for targeting InfiniBand with GASNet
  • We have done preliminary testing and want to track ucx as upstream support becomes more stable
THANK YOU

https://chapel-lang.org
@ChapelLanguage