Hewlett Packard
Enterprise

CHAPEL 1.24 RELEASE NOTES: IMPLEMENTATION IMPROVEMENTS

Chapel Team

March 18, 2021

OUTLINE

- LLVM Backend Improvements
- 'ofi' Communication Layer
- Chapel on HPE Cray EX Systems
- Python 3 Compatibility
- Other Implementation Improvements

LLVM BACKEND IMPROVEMENTS

LLVM IMPROVEMENTS

Background and This Effort

Background: Working to make LLVM the default compiler backend

- rationale:
  - reduce effort spent supporting and testing multiple C compilers/versions
  - convey semantic information more directly to the back-end
  - leverage open-source efforts, community familiarity, GPU back-ends, etc.

This Effort: Increase LLVM testing and fix any problems discovered

- Discovered several issues with LLVM code generation for multilocale tests
  - Fixed most of the issues: incorrect flags to codegen, mishandling of signedness, debugging errors, etc.
  - Multilocale 'C' interop not currently working with LLVM, but close
- PRs exist that will make LLVM the default post-release
  - If 'CHPL_LLVM' is unset, default to…

```
CHPL_LLVM=bundled  # if $CHPL_HOME/third-party/llvm is already built
CHPL_LLVM=system   # if a working external system llvm is detected
```

  - If 'CHPL_LLVM' is unset after the above defaults, issue an error requesting that it be explicitly set

LLVM IMPROVEMENTS

Status and Next Steps

Status: Nearly ready to switch to LLVM by default

- Cleaned up several bugs for multilocale LLVM testing
- Have open PRs ready to flip the switch

Next Steps: Flip the switch

- Finish getting multilocale interop feature working with LLVM
- Merge the open PRs
- Address any issues uncovered in nightly testing
- Update test configurations to continue testing the C back-end

'OFI' COMMUNICATION LAYER

'OFI' COMMUNICATION LAYER

Background and This Effort

Background:

- This communication layer is based on libfabric, defined by the Open Fabrics Interfaces Working Group (thus 'ofi')
- Libfabric is the native network interface on HPE Cray EX systems, and is portable to others such as AWS/EFA
  - Defines an interface to an abstract network
  - Application selects a *provider* which instantiates that abstraction in terms of underlying interfaces
- The 'ofi' communication layer had some known functional and performance flaws
  - Conformance to the Chapel Memory Consistency Model (MCM) was somewhat unprincipled, had excess overhead
  - Selected providers correctly, but was less than ideal about enabling/disabling related capabilities and modes

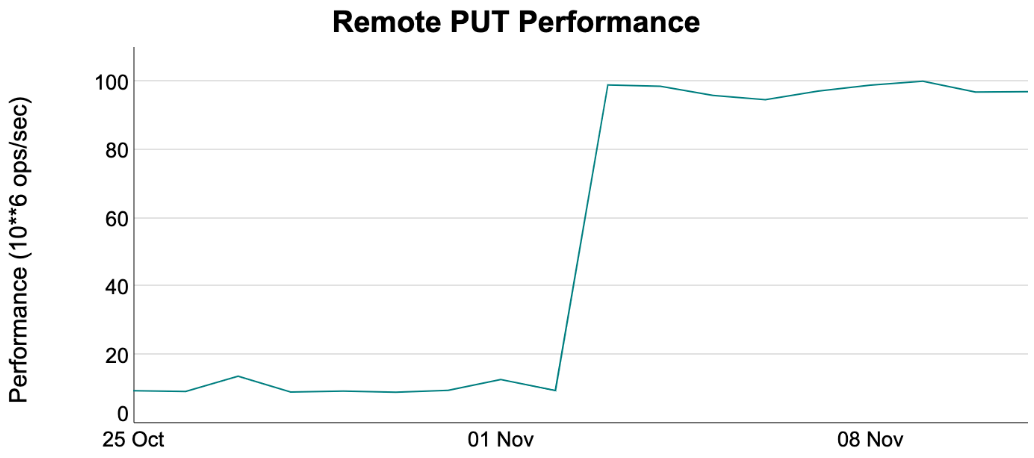
This Effort:

- Reduced overheads in MCM conformance
- Improved integration with providers' capability sets
- Tuned based on exposure to a wider variety and scale of target platforms, especially HPE Cray EX

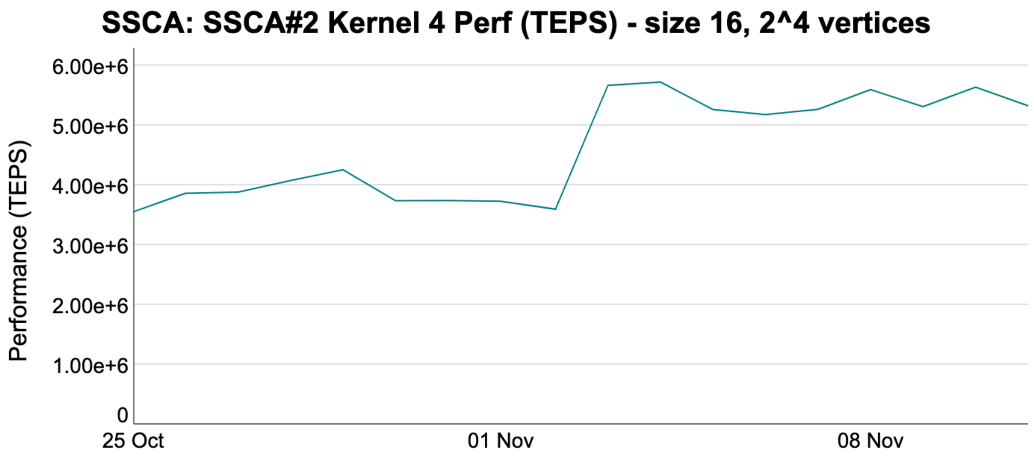
'OFI' COMMUNICATION LAYER

Impact

- MCM conformance speedup improved PUT performance significantly



~10x better on a microbenchmark, 16-node Cray CS



~1.5x better on a 'macro' benchmark, 16-node Cray CS

'OFI' COMMUNICATION LAYER

Status and Next Steps

Status:

- Ready for production use
- There are still some areas where performance is worse than desired
  - Memory registration needed with some providers produces poor NUMA locality
  - Active Message (on-statement) rates are much lower than with 'gasnet' communication

Next Steps:

- Address known performance issues:
  - NUMA
  - Bottleneck due to use of a single AM handler (progress thread)
- Ongoing provider- and capability-related improvements
- Add regular testing on more systems and networks

CHAPEL ON HPE CRAY EX SYSTEMS

CHAPEL ON HPE CRAY EX SYSTEMS

Background:

- Need to ensure Chapel continues to work on EX systems throughout the early-release process

This Effort:

- Adjusted Chapel module to integrate with HPE/Cray PE group's new Lmod module system
- As of Shasta v1.4, unbundled the Chapel package from the OS and included it with Analytics & AI instead
  - However, release timing required building that package from 1.23.1 rather than 1.24.0

Status:

- Chapel continues to be available as the EX product line progresses

Next Steps: (not necessarily in order)

- Unbundle Chapel module from Analytics & AI, so it’s a standalone package
- Release Chapel 1.24.x for EX
- Add comm=gasnet configurations
- Continue tracking EX product changes and releases

PYTHON 3 COMPATIBILITY

PYTHON 3 COMPATIBILITY

Background and This Effort

Background:

- Python 2 was officially deprecated in January 2020, but we were still relying on it more than ideal
  - Needed to update so that systems that don't include Python 2 would still be able to use Chapel effectively
  - But wanted to minimize impact on systems that use older operating systems

- Python is used by 'chpldoc' and the scripts that support 'printchplenv'
  - This meant we were relying on older versions of dependencies to maintain Python 2 compatibility
  - Eventually the older versions of these dependencies would become unavailable, too

This Effort:

- Updated 'printchplenv' support scripts to use Python 3
  - and fall back to Python 2 if Python 3 is unavailable

- Updated 'chpldoc' dependencies to latest versions as of November 2020

PYTHON 3 COMPATIBILITY

Impact and Next Steps

Impact:

- ‘chpldoc’ now relies solely on Python 3
  - Users have already started encountering issues with ‘chpldoc’ from previous releases
  - This emphasizes how important this update was

- ‘printchplenv’ is now usable on any system, including systems with ‘python3’ but not ‘python’ in the path

Next Steps:

- Continue to track compatibility with various Python 3 versions

OTHER IMPLEMENTATION IMPROVEMENTS

OTHER IMPLEMENTATION IMPROVEMENTS

For a more complete list of implementation changes and improvements in the 1.24 release, refer to the following sections in the CHANGES.md file:

- ‘Packaging / Configuration Changes’
- ‘Compilation-Time / Generated Code Improvements’
- ‘Portability’
- ‘Runtime Library Changes’
- ‘Launchers’
- ‘Bug Fixes’
- any of the ‘Developer-oriented changes’ sections

THANK YOU

https://chapel-lang.org
@ChapelLanguage