Compiler / Tools

Chapel Team, Cray Inc.

Chapel version 1.18

September 20, 2018

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

Outline

- Mason Improvements
- LLVM Back-end Improvements
- Tab Completion Improvements
- Error Message Improvements

Mason Improvements

Mason Improvements: Background

- **Mason originally added in Chapel 1.16.0**
  - Command line tool for package management and building
  - Considered under development
    - No breaking changes made to date, but reserving that right until version 1.0
- **Package metadata centralized in registries**
  - Official registry located on Github: chapel-lang/mason-registry
  - Users can create their own internal or public registries
- **Package source code decentralized across git repositories**
  - Supports any valid git address
    - Github, Gitlab, local git repositories, etc.
- **Has supported Chapel packages only**
  - This prevented many user packages from being mason packages

Mason: This Effort

- **Improved general usability of mason**
  - Added “--no-update” flag for better offline support
  - Added “mason {add, rm} <package>” for managing dependencies
  - Added support for package tests and examples
  - Improved documentation
- **Added build-on-last-modified behavior**
  - Eliminates unnecessary rebuilds
- **Added support for non-Chapel packages in mason**
  - Supports Spack packages (mason external)
  - Supports system packages (mason system)

Mason: This Effort – Improved Usability

- **Added 'mason {add, rm}' for dependency management**
  - Allows managing dependencies without editing manifest file

```
mason add MatrixMarket@0.1.0
mason add --external hdf5@1.10.1 # Spack package
mason rm MatrixMarket
```

- **Improved offline support**
  - --no-update flag added to skip registry update, which requires internet
  - Many mason commands invoke a registry update by default
    - this would cause connection warnings for offline users
  - For example:

```
mason build --no-update
```

- **Improved documentation**
  - Created a "Basic Usage" section
    - walks through how to use mason
    - includes copy/paste-friendly examples

Mason: This Effort – Improved Usability

- **Added 'mason test'**
  - Tests can be added to '<package>/test/'
  - 'mason test' will compile and run all tests, reporting pass/fail results
    - any top-level Chapel program in 'test/' assumed to be a test program
    - pass/fail is determined by an exit code (non-zero means fail)
    - test output can be piped to stdout with flag: 'mason test --show'
  - For example:

```
$ mason test
--- Results ---
Test: myPackageTest Passed

--- Summary: 1 tests run ---
-----> 1 Passed
-----> 0 Failed
```

Mason: This Effort – Improved Usability

- **Added support for package examples**
  - Examples can be added to '<package>/example/'
  - 'mason build --example <filename>' will compile an example
    - Omitting the filename will build all examples
  - 'mason run –example <filename>' will run an example
    - Omitting the filename will list all available examples to run
  - Examples can be specified in the manifest file
    - compiler and execution options can also be specified
    - if omitted from manifest, examples will be found automatically in 'example/'

```
[examples]
examples = ["myPackageExample.chpl"]

[examples.myPackageExample]
execopts = ["--count=20"]
compopts = ["--savec tmp"]
```

Mason: This Effort – Build on Last Modified

- **Mason skips compilation when project "not modified"**
  - Similar to 'make' behavior when build dependencies are unchanged
- **Project considered "not modified" when:**
  - Target binary already exists
  - Lock file already exists
  - Source files have not been modified since binary last built
    - this includes dependency code as well
  - Manifest file has not been modified since binary last built
    - this accounts for modified dependencies, versions, compilation flags, etc.
  - Force flag is not thrown, '--force'
    - this flag was added to override this feature
- **User notified when skipping build:**

```
$ mason build
Updating mason-registry
Skipping Build... No changes to project
```

Mason: This Effort – Spack Integration

- **Spack is a system package manager**
  - Developed with HPC users in mind
  - Developed by LLNL
  - Spack has ~3000 packages in its registry
  - Supports multiple configurations, platforms, and compilers
- **Mason uses Spack to manage non-Chapel dependencies**
- **Relying on a single package manager has tradeoffs:**
  - Using only Spack means Mason's success is tied to Spack's success
  - Using only Spack enables version resolution for external packages
    - Mason offloads version resolution of external packages onto Spack

Mason: This Effort – Spack Integration

- **Accessed through 'mason external' command**
  - Users are required to install Spack backend to use this feature:

```
$ mason external install openssl@1.0.2k
To use `mason external` call `mason external --setup`
$ mason external --setup
```

  - Spack is installed within $MASON_HOME
- **Provides subcommands, which call down to Spack:**
  - 'mason external search <search-string>'
    - searches packages on Spack registry
  - 'mason external info <package>'
    - shows information about external package
  - 'mason external compiler'
    - lists available compilers on system

Mason: This Effort – Spack Integration

- **External packages require explicit installation**
  - This behavior differs from how mason packages are installed
    - external packages tend to have a long installation time
    - we decided that this should be explicitly opted into to avoid surprise
- **Subcommands available for installing/uninstalling:**
  - 'mason external install <spack spec expression>'
  - 'mason external uninstall <package>'
- **"Spack spec expressions" allow specifying constraints**
  - For example: '<package>@<version>%<compiler>'
  - Compiler defaults to 'CHPL_TARGET_COMPILER' if unspecified
  - Spec expression documentation shown with 'mason external --spec'

Mason: This Effort – Spack Integration

- **Manifest files distinguish external packages in [external]**

```
[brick] name = "myPackage"
version = "0.1.0"
chplVersion = "1.18.0"

[external]
openSSL = "1.0.2k”
```

- **External packages can be managed with ‘mason add/rm’**

```
$ mason add --external openSSL@1.0.2k
Adding external dependency with spec openssl@1.0.2k
$ mason rm openSSL
```

Mason: This Effort – System Packages

- **Mason uses pkg-config to access packages on system**
  - Feature intended for prototyping purposes
    - only available for top-level packages
    - allows quick and easy access to the libraries available on the system
    - cannot publish packages with system dependencies
  - Pkg-config provides compiler flags for linking to the provided library
- **Accessed through "mason system" command**
- **Provides subcommands, which call down to pkg-config:**
  - 'mason system search <search-string>'
    - searches packages installed on system
  - 'mason system pc <package>'
    - prints pkg-config file of package

Mason: This Effort – System Packages

- **Manifest files distinguish system packages in [system]**
  - For example:

```
[brick] name = "myPackage"
version = "0.1.0"
chplVersion = "1.18.0"

[system]
openSSL = "0.9.8zh”
```

  - System packages can be added/removed with mason add/rm:

```
$ mason add --system openSSL@0.9.8zh
Adding system dependency openSSL version 0.9.8zh
$ mason rm openSSL
```

Mason: Impact & Status

- **Mason is becoming more mature and feature-rich**
- **Many Chapel repositories on Github can now be packaged**
  - Supporting non-Chapel dependencies was a prerequisite for many
- **Mason is still considered under development**
  - Still gathering feedback on features before locking down 1.0

Mason: Next Steps

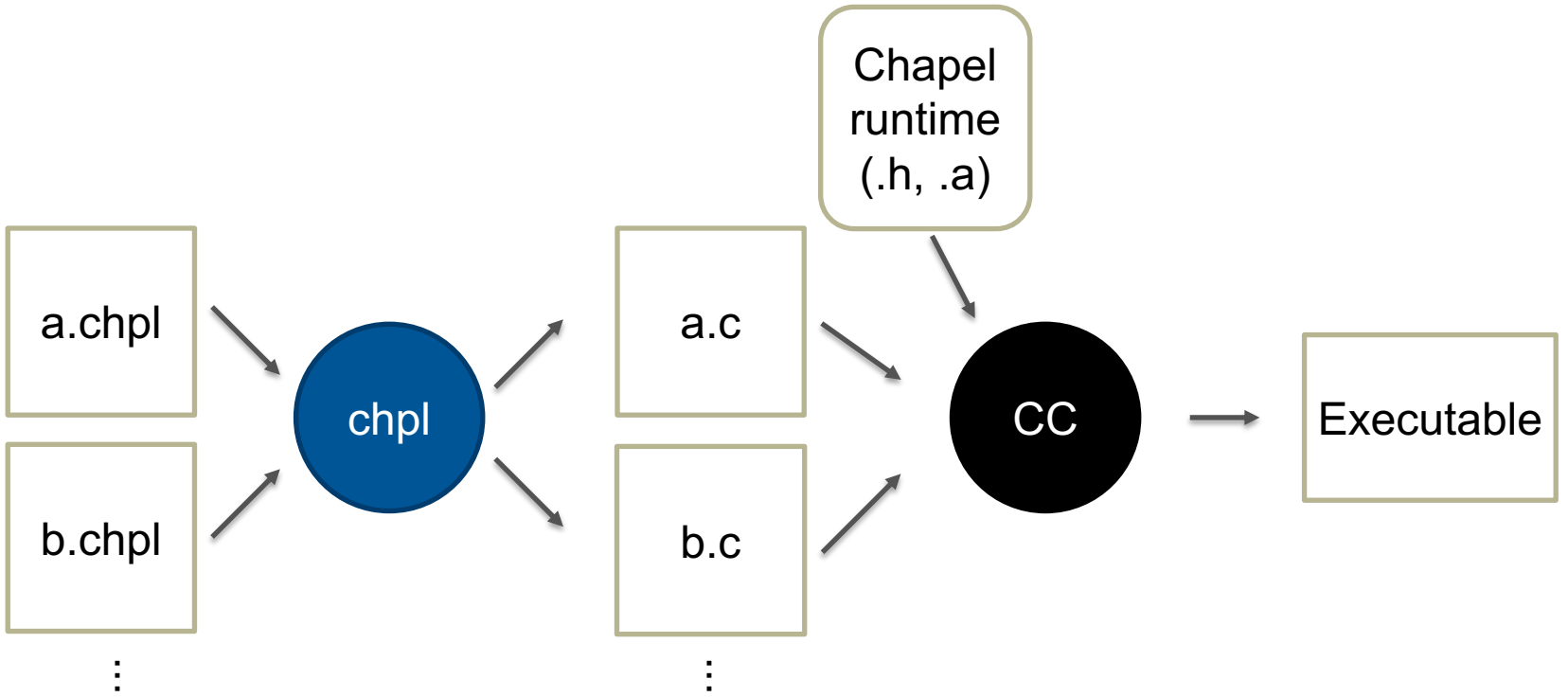
- **Implement essential package manager features**
  - Support licensing
  - Support signed packages (GPG)
  - Support a more secure way to pin package versions
  - Support packages as applications
  - Improve process for publishing packages
- **Continue to improve offline support**
  - Support environment variable for “offline mode”
- **Handle package collisions better**
  - Current namespace rules will cause many package incompatibilities
- **Cache packages on a server**
  - Ensures availability of packages in the registry
- **More next steps tracked in #7106**

LLVM Back-end Improvements

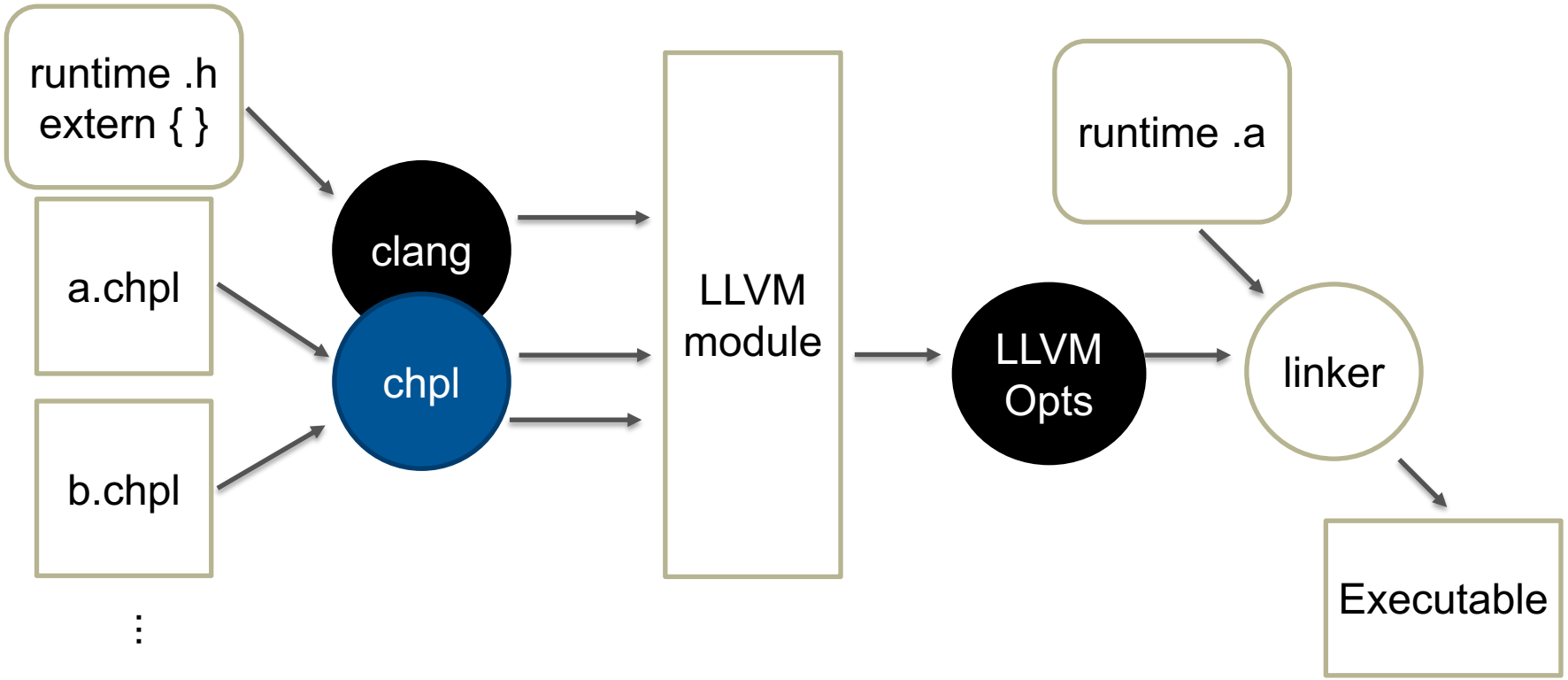
LLVM: Background

- **LLVM is a compiler optimization framework**
  - actively developed and constantly improving
- **The Chapel compiler generates C code by default**
  - runs a C compiler to compile the generated code
  - but can generate LLVM Intermediate Representation instead
- **We want the Chapel compiler to use LLVM by default**
  - to reduce maintenance vs. depending on many C compilers
  - to improve optimization and enable communication optimization

Chapel compilation flow



Chapel --llvm compilation flow



LLVM: This Effort

- **Broadened LLVM support for assorted use cases**
  - improved ARM support
  - made LLVM work with dynamic linking on Cray XC systems
  - updated llvm.invariant.start emission for records using initializers
  - enabled ‘chpl’ to build with LLVM 7 pre-release

LLVM: Next Steps

- **Improve ABI compatibility for non-x86 architectures**
  - particularly ARM
- **Integrate Chapel alias analysis metadata with that for C**
  - improves optimization opportunities with imported/exported functions
- **Allow --llvm to link user programs statically**
- **Improve performance for code generated with --llvm**
- **Make --llvm the default**

Tab Completion Improvements for Chapel Options

Tab Completion Improvements

Background: Tab completion of 'chpl' options was added in 1.17

- Tab completion searched against all compiler options
  …including developer options
- Tab completion completed paths looking for .chpl files
  …but mishandled paths that included the home directory marker '~'

```
% chpl ~/test.ch<tab>   ➡   % chpl \~/test.chpl
```

This Effort: Fix the issues listed above

- Only complete non-developer options unless developer mode is on

```
% chpl --g<tab>
--gasnet-segment --gmp
```

```
% chpl --devel --g<tab>
--gasnet-segment --gdb --gen-ids --gmp
```

- Paths including '~' complete successfully

```
% chpl ~/test.ch<tab>   ➡   % chpl ~/test.chpl
```

Error Message Improvements

Error Messages: Background + This Effort

Background:

- The Chapel compiler's error messages have often been lacking
  - confusing, not written with end-users in mind, internal errors, …
  - we've been focused more on supporting correct code than incorrect
  - however, as we work to attract new users, this becomes a bigger problem

This Effort:

- Striving to improve error message problems for reported cases

Error Message Improvements

- Applying '.type' to a type is consistently an error

```
writeln(uint.type: string);
```

```
uintType.chpl:1: error: can't apply '.type' to a type (uint(64))
```

- Bad accesses to type-tuples result in compile-time errors

```
type t = (int, real);

writeln(t(3): string);
writeln(t(2, 1): string);
```

```
t.chpl:3: error: type index expression '3' out of bounds
t.chpl:4: error: too many arguments to type index expression
```

Error Message Improvements (continued)

- **Recursive records result in compile-time errors**

```
record R {
  var x : R;
}
var r = new R();
```

```
r.chpl:2: error: record 'R' cannot contain a recursive field 'x' of type 'R'
```

- **And many others…**

For More Information

For a more complete list of compiler and tool changes in the 1.18 release, refer to the ‘New Tools / Tool Changes’, ‘Compiler Flags’, ‘Error Messages’, and ‘Bug Fixes’ sections in the **CHANGES.md** file.

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

CRAY

THE SUPERCOMPUTER COMPANY