Portability and Packaging

Chapel Team, Cray Inc.

Chapel version 1.15

April 6, 2017

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

Portability

- Chapel on Amazon Web Services
- Chapel on 64-bit ARM
- Chapel on Windows 10 WSL

Packaging

- Debian Package

Configuration and Build

- Build Directory
- Configuration Paths

Other

- Other Portability and Packaging Improvements

Chapel on Amazon Web Services

AWS: Background

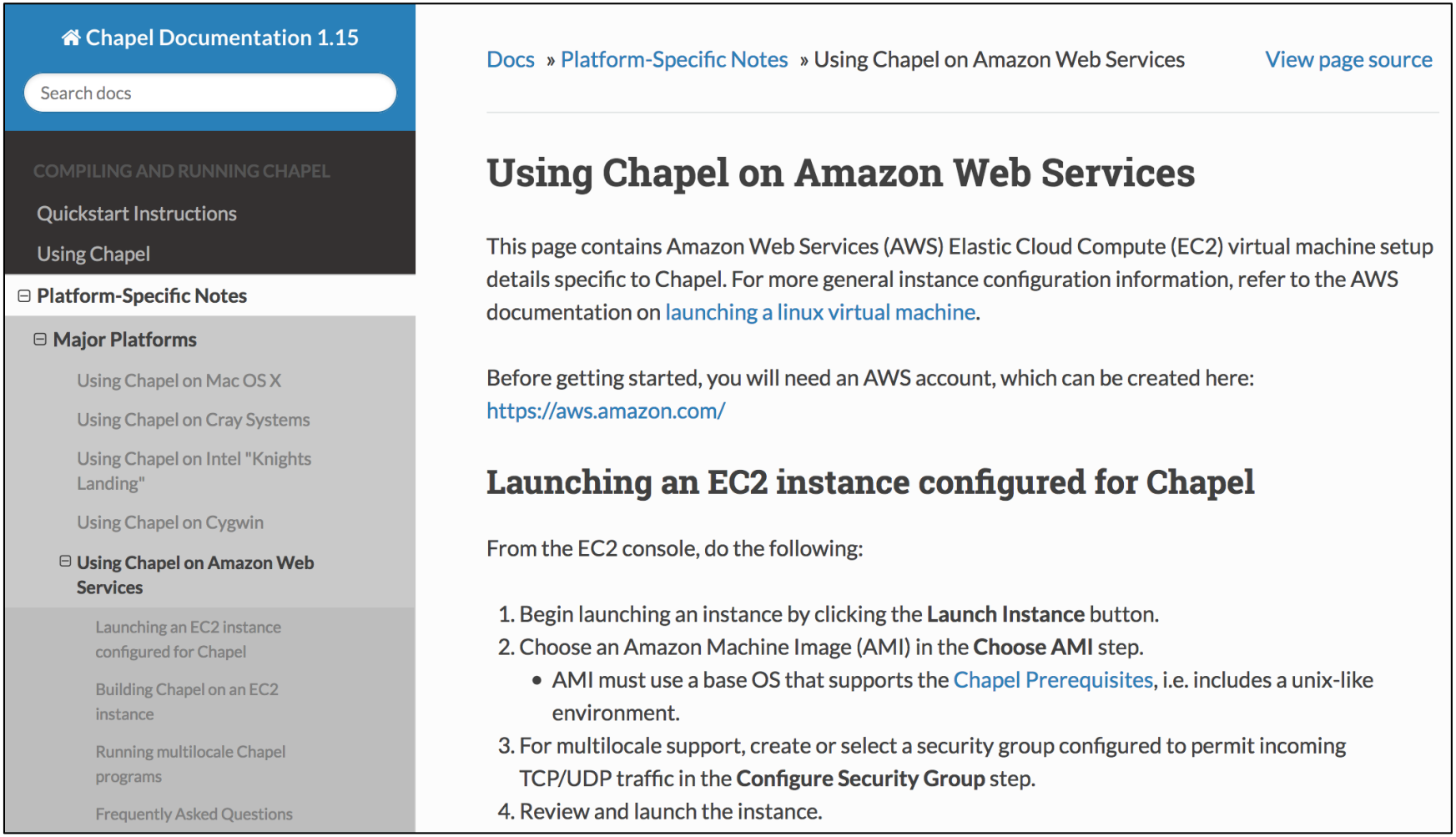
- **Some users have been using Chapel on AWS EC2**
  - (Amazon Web Services Elastic Compute Cloud)
  - Particularly useful for running test suite
- **No official guide existed for setting up Chapel on AWS**

AWS: This Effort

- **Document how to set up Chapel on AWS EC2**
  - Assuming no prior experience with AWS

- **Documentation Covers:**
  - Setting up and launching an EC2 instance
  - Building Chapel on an EC2 instance
  - Setting up an EC2 instance for multilocale Chapel programs
  - Executing multilocale Chapel programs over EC2 instances
  - Other general caveats

AWS: Impact



Chapel Documentation 1.15

Search docs

COMPILING AND RUNNING CHAPEL

Quickstart Instructions

Using Chapel

Platform-Specific Notes

Major Platforms

Using Chapel on Mac OS X

Using Chapel on Cray Systems

Using Chapel on Intel "Knights Landing"

Using Chapel on Cygwin

Using Chapel on Amazon Web Services

Launching an EC2 instance configured for Chapel

Building Chapel on an EC2 instance

Running multilocale Chapel programs

Frequently Asked Questions

Docs » Platform-Specific Notes » Using Chapel on Amazon Web Services View page source

Using Chapel on Amazon Web Services

This page contains Amazon Web Services (AWS) Elastic Cloud Compute (EC2) virtual machine setup details specific to Chapel. For more general instance configuration information, refer to the AWS documentation on launching a linux virtual machine.

Before getting started, you will need an AWS account, which can be created here: https://aws.amazon.com/

Launching an EC2 instance configured for Chapel

From the EC2 console, do the following:

1. Begin launching an instance by clicking the **Launch Instance** button.
2. Choose an Amazon Machine Image (AMI) in the **Choose AMI** step.
   - AMI must use a base OS that supports the Chapel Prerequisites, i.e. includes a unix-like environment.
3. For multilocale support, create or select a security group configured to permit incoming TCP/UDP traffic in the **Configure Security Group** step.
4. Review and launch the instance.

⋮

AWS: Next Steps

- **Provide prebuilt Amazon Machine Image for Chapel**
  - to make it easier to run Chapel on AWS
- **Extend support to other providers**
  - Google Cloud Platform
  - Microsoft Azure
  - …
- **Explore integrating test infrastructure with cloud services**

Chapel on 64-bit ARM

Chapel on ARM: Background

- **64-bit ARM was added as a target platform in Chapel 1.14**
  - supported cross-compilation and cross-launching
    - i.e., host node need not be an ARM
  - only single-locale ARM was supported

- **GASNet over UDP did not support cross launching**
  - the amudprun launcher was compiled for target, not the launch host

Chapel on ARM: This Effort

- **We updated Chapel's ARM configuration**
  - enabled use of GASNet over UDP for ARM
  - corrected GASNet ARM configuration

- **Arranged to compile the amudprun launcher separately**
  - when compiled as part of GASNet, it is compiled for the target
  - we now additionally compile it for the host

Chapel on ARM: Impact, Status, and Next Steps

Impact:

- Chapel programs can now be compiled for multilocale 64-bit ARM
  - may be cross-launched (launching node is not required to be an ARM)

Status:

- Detailed instructions available in the Chapel documentation
  - 64-bit ARM platform-specific notes
  - multilocale Chapel execution notes

Next Steps:

- Study and optimize for ARM
- Add power-aware features

Chapel on Windows 10 WSL

Chapel on Windows 10 WSL

Background: Chapel on Windows supports Cygwin, but

- requires using fifo rather than qthreads tasking layer
- generated executables require Cygwin DLL to run
- Cygwin GPL / LGPL license may be problem for some

This Effort: Demonstrate Chapel in WSL

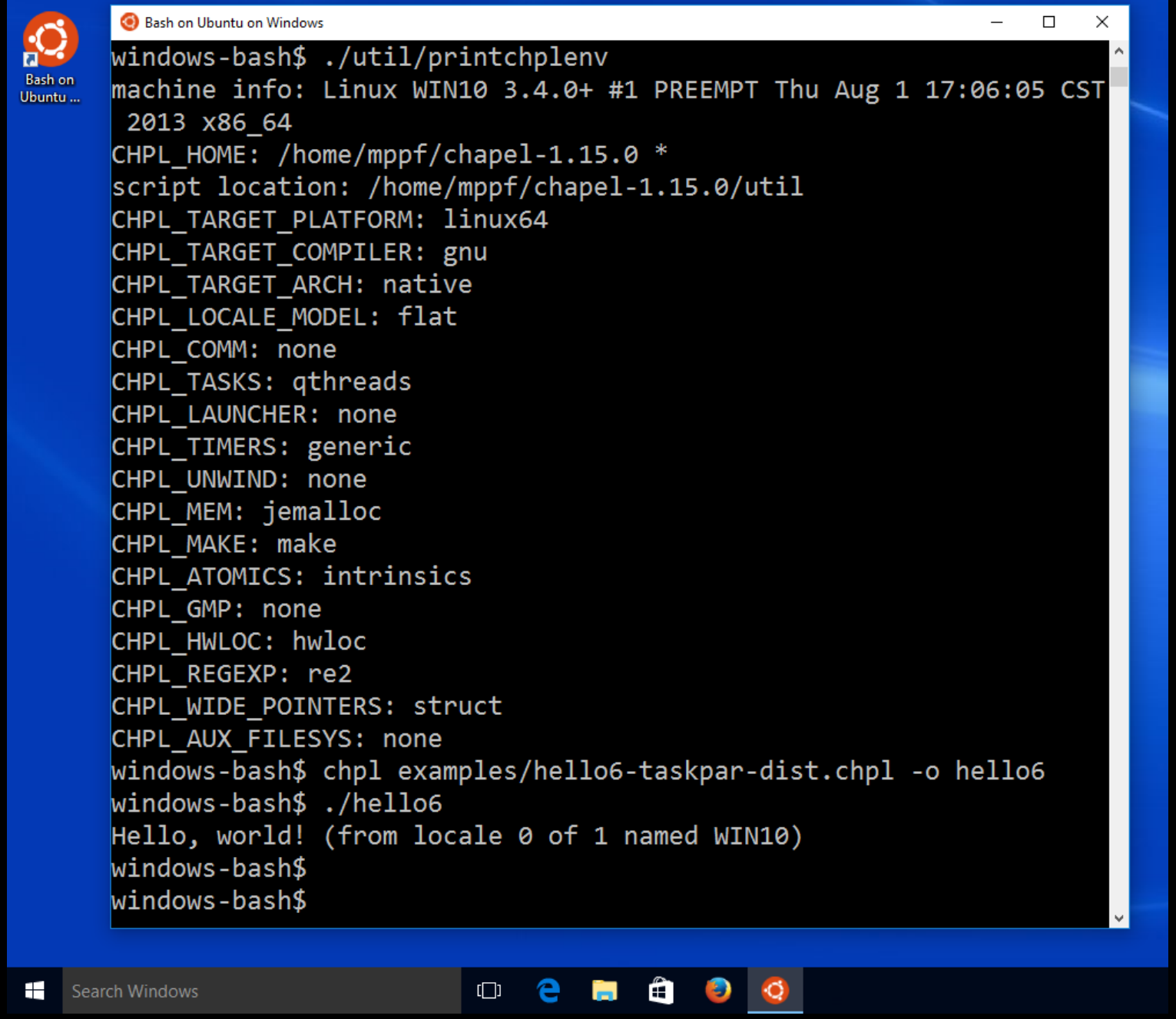
- WSL is Windows Subsystem for Linux, a.k.a. Windows Bash Shell
  - Beta is available for Windows 10
  - Runs an Ubuntu image
  - Enables portability by running programs in an Ubuntu environment

Impact: More seamless Chapel user experience on Windows

- qthreads works in WSL, unlike Cygwin

Next Steps: Update docs, do regular testing, fix GASNet issue

- GASNet/UDP worked in 1.14 under WSL, doesn't work under 1.15


Bash on Ubuntu ...
Bash on Ubuntu on Windows
windows-bash$ ./util/printchplenv
machine info: Linux WIN10 3.4.0+ #1 PREEMPT Thu Aug 1 17:06:05 CST 2013 x86_64
CHPL_HOME: /home/mppf/chapel-1.15.0 *
script location: /home/mppf/chapel-1.15.0/util
CHPL_TARGET_PLATFORM: linux64
CHPL_TARGET_COMPILER: gnu
CHPL_TARGET_ARCH: native
CHPL_LOCALE_MODEL: flat
CHPL_COMM: none
CHPL_TASKS: qthreads
CHPL_LAUNCHER: none
CHPL_TIMERS: generic
CHPL_UNWIND: none
CHPL_MEM: jemalloc
CHPL_MAKE: make
CHPL_ATOMICS: intrinsics
CHPL_GMP: none
CHPL_HWLOC: hwloc
CHPL_REGEXP: re2
CHPL_WIDE_POINTERS: struct
CHPL_AUX_FILESYS: none
windows-bash$ chpl examples/hello6-taskpar-dist.chpl -o hello6
windows-bash$ ./hello6
Hello, world! (from locale 0 of 1 named WIN10)
windows-bash$
windows-bash$
Search Windows

Debian Package

Debian Package: Background and Effort

Background: Debian package under review

- Submitted *Request for package* (RFP) and *intent to package* (ITP)
  - Package submitted is 'chapel-minimal', with third-parties removed
- Review has been challenging
  - Debian has strict packaging policies
  - Chapel has a non-traditional build system
- Missed 'stretch' deadline in January 2017
  - However, Chapel can still be back-ported after acceptance

This Effort: Address reviewer feedback

- Enabled hardening for dpkg-buildflags
  - Requires patch to Makefiles to recognize CPPFLAGS
- Reached agreement that utf8-decoder is acceptable in package
- Made 'make clean' idempotent for stripped down package
- Removed stray .c files in include-directories
- Addressed several minor formatting errors

Debian Package: Impact and Next Steps

Impact: Closer to a Debian package release

Next Steps: Release a Debian package

- Next challenge: conforming to filesystem hierarchy standard
  - Roughly equivalent to supporting a 'make install'
- Expedite propagation of package downstream
  - *sid* → *stretch* (back-ported)
  - *Debian* → *Ubuntu*
- Pursue full-featured Chapel package
  - Including third-parties as package-dependencies
- Extend to other Linux distributions
  - Fedora, SUSE, Arch, …
- Encourage community members to build packages

Build Directory

Build Directory

Background: Object files were stored in ‘gen/’ subdirs

- For example

```
runtime/src/gen/
compiler/main/gen/
compiler/passes/gen/
```

- Complicated ‘make clobber’ and ‘make cleanall’
  - Led to surprises where 'make clobber' failed to clean up some object files

This Effort: Build object files into a top-level ‘build/’ dir

```
runtime/src/gen/ → build/runtime/…/src/
compiler/*/gen/ → build/compiler/…/*/
```

Impact: Object files stored in more predictable locations

- Simplified and improved ‘make clobber’

Configuration Paths

Configuration Paths

Background: Build target paths encode their configuration

- For example:

```
lib/darwin.clang.arch-native.loc-flat.comm-none...
```

- Configuration-encoded paths used '.' delimiter
- These paths exceeded max filename length for some filesystems
  - Some users ran into this error

This Effort: Configuration-encoded paths use '/' delimiter

- For example:

```
lib/darwin/clang/arch-native/loc-flat/comm-none/...
```

- Now generates chplconfig file in each path leaf
- Used in the new build/ directory paths as well

Impact:

- Chapel works on more filesystems
- Easy to use generated chplconfig files to select a built configuration

Other Packaging and Portability Improvements

Other Packaging and Portability Improvements

Packaging:

- Updated Docker images with
  - improved README
  - chapel-gasnet images

Portability:

- Fixed an occasional cygwin failure with pthread_attr_init()
- Removed extra -O flags from cray-prgenv-cray compiles
- Improved portability of libunwind support

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