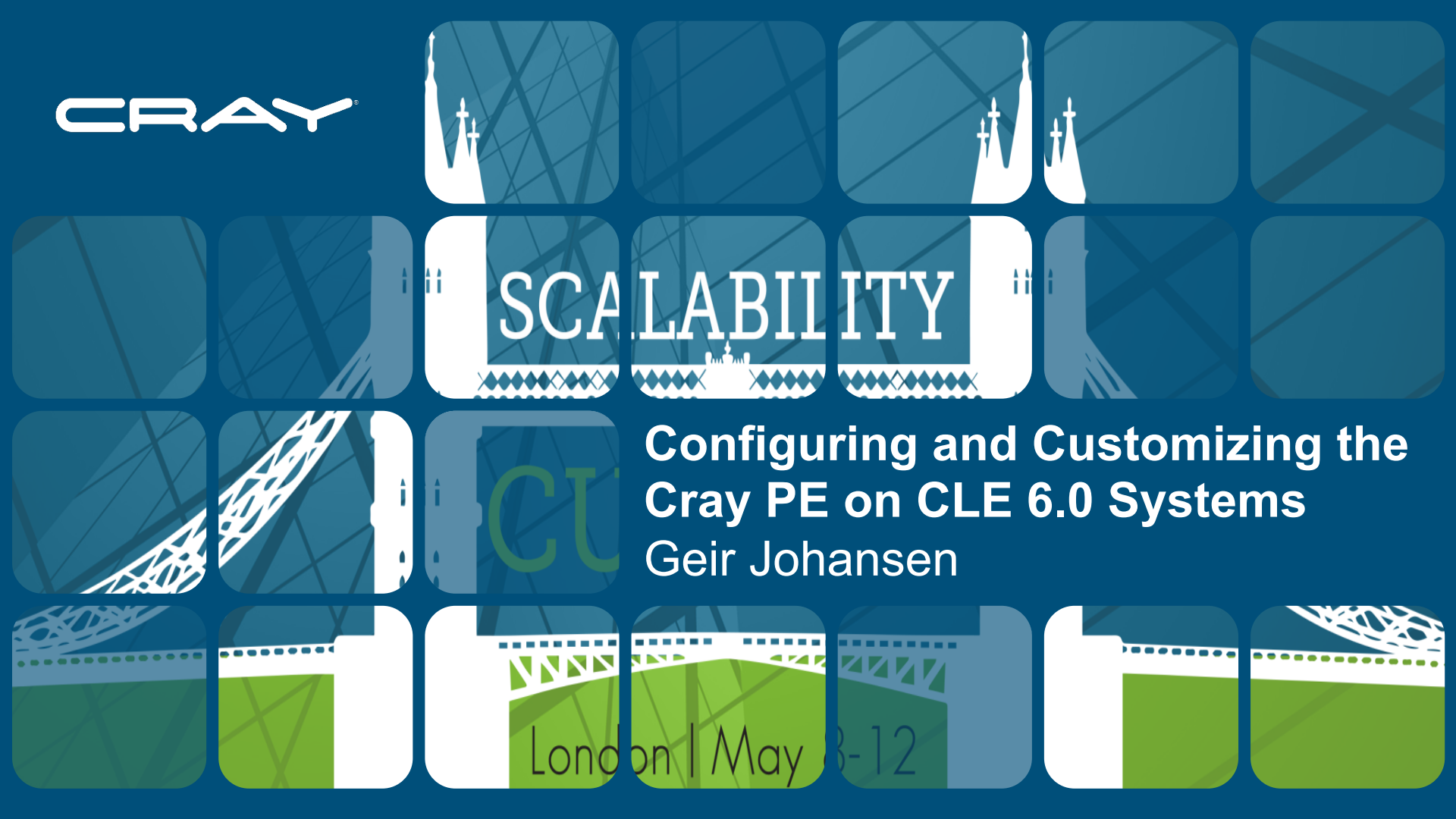
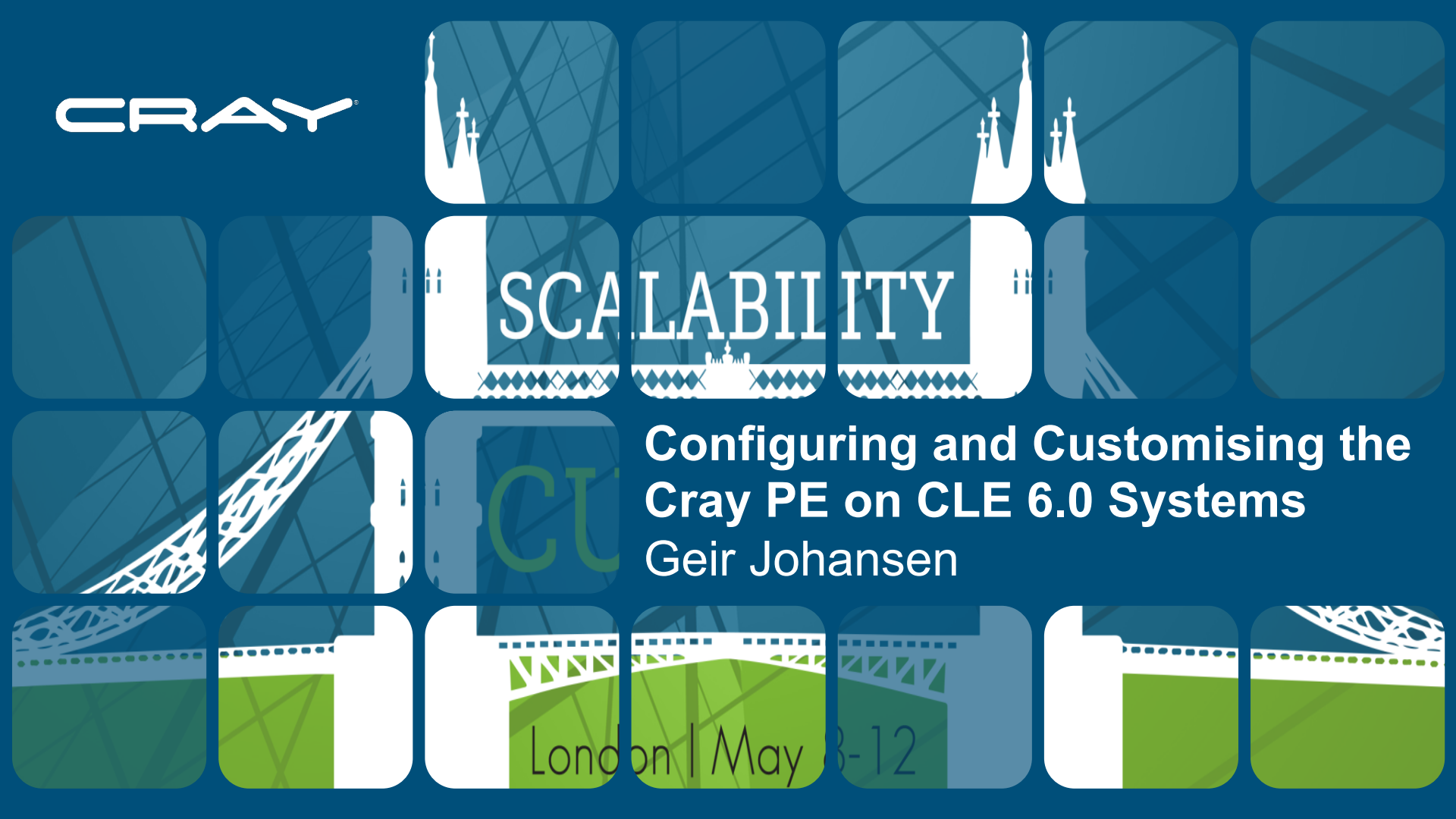


Configuring and Customizing the Cray PE on CLE 6.0 Systems

Geir Johansen



Configuring and Customising the Cray PE on CLE 6.0 Systems

Geir Johansen

Agenda

- **Purpose**
  - Outline changes to the Cray Programming Environment in CLE 6.0
  - Provide current information on customization of the programming environment
- **Benefit/Value**
  - Talk is targeted for site administrators, and system consultants responsible for assisting their user community in building and analyzing programs.
- **CLE 6.0 PE installation and configuration**
- **New CLE 6.0 PrgEnv modulefile features**
- **Topics in porting and integrating third party software with Cray PE**
- **Summary**
- **Q&A**

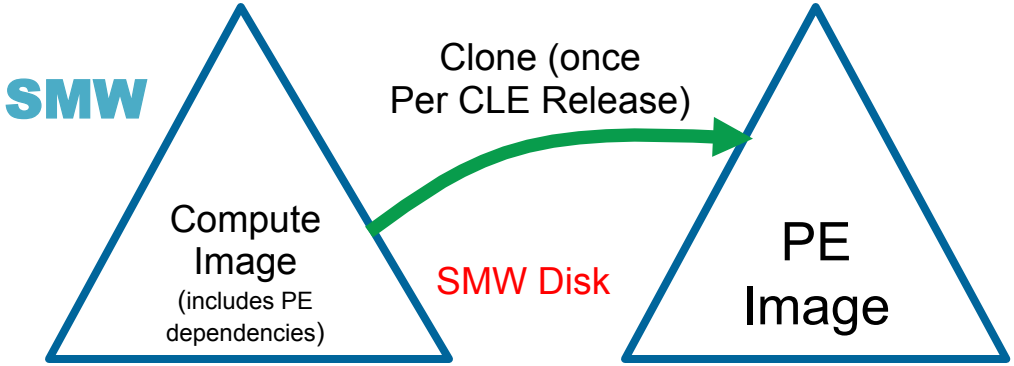
CLE 6.0 Installation & Configuration Info @ CUG

- *Cray Management System for XC Systems with SMW 8.0/CLE 6.0* -- Cray Tutorial
- *eLogin Made Easy - An Introduction and Tutorial on the new Cray External Login Node* – Cray tutorial
- *Crossing the Rhine - Moving to CLE 6.0 System Management* -- NERSC presentation
- *How to Automate and not Manage under Rhine/ Redwood* -- LANL presentation

CLE6.0/SMW 8.0 Cray Management System

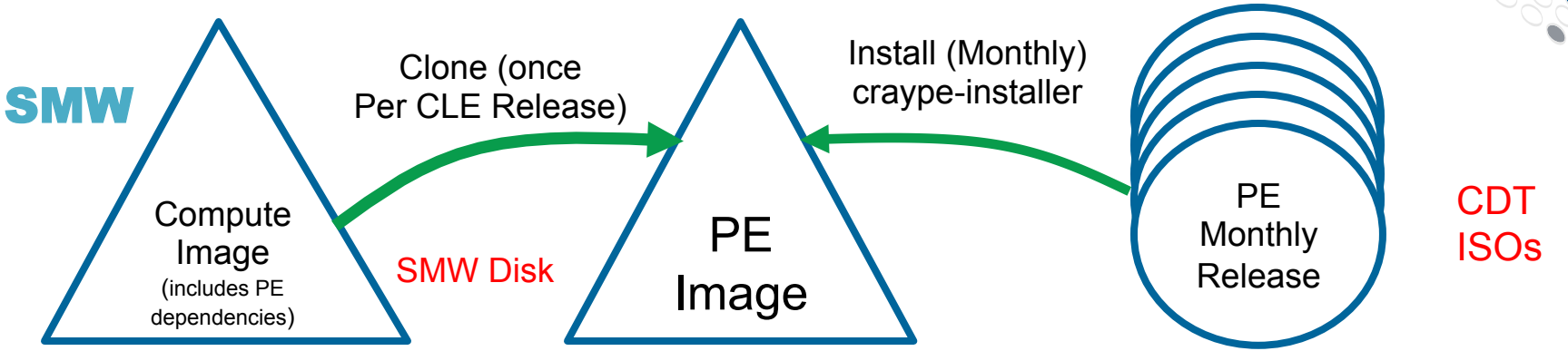
- CLE 6.0 system installation and configuration methodology is significantly different than previous versions of CLE.
- Move away from proprietary tools to Linux tools (rpm, zypper, yum, …)
- Processes can be shared across platforms (XC, SMW, eLogin, …)

CLE 6.0 PE Management



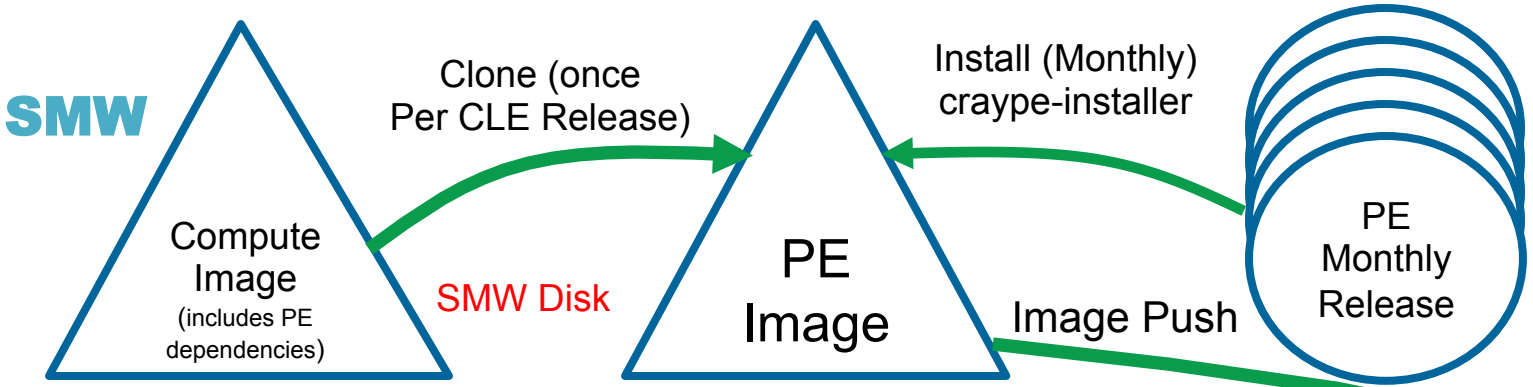
- PE installation takes place on SMW,
- No longer installed on a shared root on boot node
- PE software installed into a *PE root image*
- System agnostic process – same PE image root can be used on different systems
- Cloning feature – easier to test new PE releases, revert back to previous PE releases

CLE 6.0 PE Management



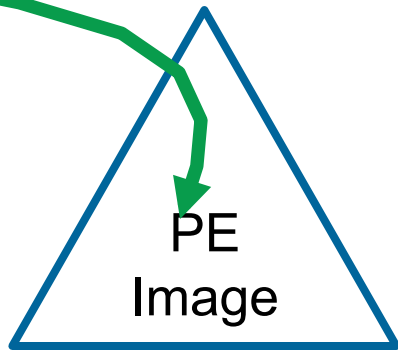
- Same Cray Developer Toolkit (CDT) ISOs are used for CLE 5.2 and CLE 6.0
- `craype-installer` will continued to be used to install the Cray PE
- PE software installed into a *PE root image*
- For CLE 6.0: the `install-cdt.yaml` configuration file has a new variable IMAGE_DIRECTORIES that designates the PE root image
- Cray PE software now installed into `/opt/cray/pe` directory

CLE 6.0 PE Management



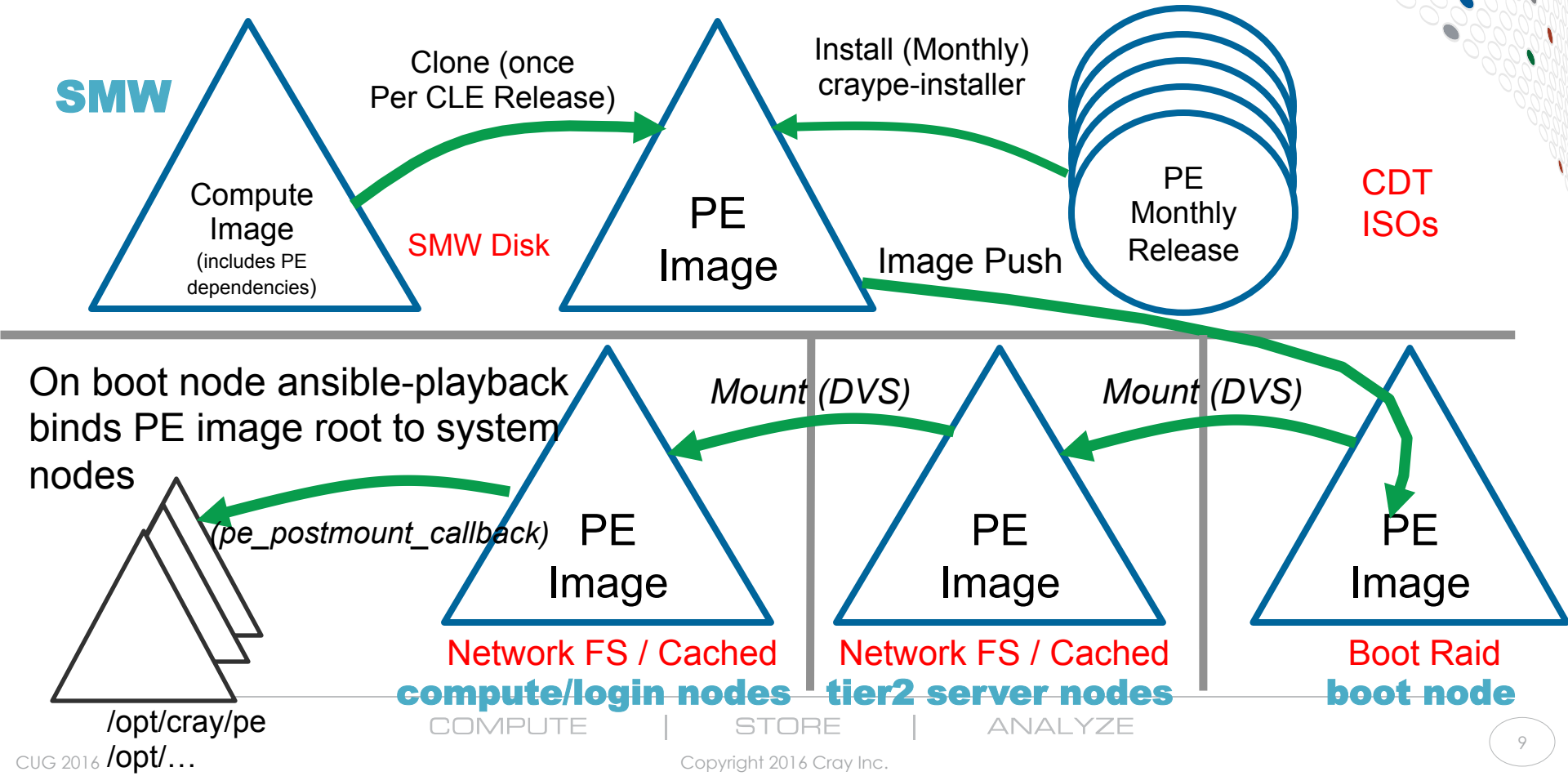
- Once installation has completed (including PE third party software), PE image root pushed to the boot node:

```
smw# image push –dest boot <PE image root>
```



Boot Raid
boot node

CLE 6.0 PE Management



Installation of Third Party PE Software

- Performed on SMW by using `chroot` to access the PE root image
- PGI Compiler example:

```
smw#  cp pgi-16.4.0-*.rpm <PE root image>/var/tmp
smw#  chroot <PE root image> rpm -ivh /var/tmp/
pgi-16.4.0-*.rpm
```

- Intel compiler -- not system agnostic, needs access to install system's /dev file. Workaround:

```
smw # mount --bind /dev <PE root image>/dev
```

CLE 6.0 Shell Initialization (rc) Files to Load Programming Environment Modulefiles

- Typically site administrators will initialize `/etc/bash.bashrc.local` and `/etc/csh.cshrc.local` to automatically load programming environment moduefiles.
- In CLE 6.0, the file `/etc/opt/cray/pe/admin-pe/site-config` is used to specify the modulefiles to be loaded when a user logs in
- Supports bash (sh), csh, tcsh, zsh, ksh (lksh, mksh, pdksh)
- Task performed by `/opt/cray/pe/bin/setup_shell_rcs.sh`

site-config example

```
$ cat /etc/opt/cray/pe/admin-pe/site-config
# Defines the Programming Environment modules
# that will automatically be loaded.
  module add PrgEnv-cray
  module add atp
  module add cray-mpich
  module add craype-haswell
  module add perftools-base
  module add forge
  module add slurm
$
```

CLE 6.0 PrgEnv modulefile features

- **PrgEnv modulefiles (i.e PrgEnv-cray, PrgEnv-intel, …) now released in CDT and not CLE 6.0**
  - CLE 5.2 wil continue used PrgEnv modulefiles released with CLE 5.2
- **SITE_MODULE_NAMES environment variable**
  - User specified modulefiles to be swapped during a PrgEnv module swap
- **cdt modulefiles**
  - Specify modules from a specific CDT release
- **Modules –S substring search**

SITE_MODULE_NAMES

- Add modulefile name to the SITE_MODULE_NAMES environment variable results in the modulefile being unloaded and loaded during a module swap of PrgEnv
- Automatically done for Cray Programming Libraries that are loaded
- Insures that environment variables are set correctly for the compiler that is loaded
- craype compiler drivers also check which compiler, network target, and CPU target that is loaded for Cray Programming Environment libraries

SITE_MODULE_NAMES example

```
$ module load PrgEnv-cray
$ module load cray-netcdf cray-tpsl boost
$ export SITE_MODULE_NAMES=boost
$ module show PrgEnv-gnu 2>&1 | grep swap
module          swap craype/2.5.4
module          swap cray-mpich cray-mpich/7.3.3
module          swap cray-hdf5 cray-hdf5/1.8.16
module          swap cray-tpsl cray-tpsl/16.03.1
module          swap boost boost/1.59.0
$
```

cdt modulefiles

- Introduced in CLE 6.0
- modulefile for each Cray Developer toolkit (CDT) release
- Instructs module command to use software components from a specific CDT release
- Effectively changes the set of default modulefiles

cdt modulefile example

```
$ module load cdt/16.3
$ module load cray-mpich
$ module list 2>&1 | grep mpich
 22) cray-mpich/7.3.2
$ module load cdt/16.4
$ module swap cray-mpich cray-mpich
$ module list 2>&1 | grep mpich
 22) cray-mpich/7.3.3
$
```

Module command substring search

- Introduced in modules 3.2.10.4 (CDT 16.04)
- Available for CLE 5.2

```
$ module avail trilinos
$ module avail -S trilinos

----------- /opt/cray/modulefiles -----------
cray-trilinos/12.2.1.0(default)
$
```

GNU Autoconf -- configure

- Version 2.63 (available on CLE 5.2) has Cray specific fixes/ CLE 6.0 has version 2.69.
- Autoconf generated configure scripts often run slowly on Cray systems
  - Many Cray systems default to static linking
  - Several large networking (i.e. ugni) and programming environment (i.e cray-libsci) libraries are linked
- The configure '-C' option creates a config.cache file. Subsequent executions using the '`--config-cache`' option will avoid many of the configure tests.
- Some applications support the configure directive '`cross_compiling=yes`', may resolve issue where build machine does not match target machine. Becoming more supported, build for ARM is a big reason for this situtation.

CMake 3.5

- Developed by Kitware Inc.
- In March 2016 released CMake 3.5.0:

  "*The 3.5 release introduced a new platform file to increase the compatibility of CMake with the Cray Linux Environment (CLE). This file allows CMake to cross-compile code in the CLE to target compute nodes.*"

- `-DCMAKE_SYSTEM_NAME=CrayLinuxEnvironment` is specified on the the CMake command line
- Enables Cmake to use appropriate build settings with the PrgEnv modulefiles loaded

Downloading and building CMake

- CMake 3.5.1 that was downloaded from cmake.org.
- CC and CXX environment variables were intentionally not set, so the build process defaults to directly calling the gcc and g++ compilers.

```
$ tar xzvf cmake-3.5.1.tar.gz
$ cd cmake-3.5.1
$ module load gcc  # load a current version of GCC
$ export PE_INSTALL=<Installation directory for PE tools>
$ mkdir –p $PE_INSTALL/cmake/3.5.1
$ ./configure ––prefix=$PE_INSTALL/cmake/3.5.1
$ gmake install
```

Creating a CMake modulefile

```
$ module load craypkg-gen
$ craypkg-gen -m $PE_INSTALL/cmake/3.5.1
$ module use $PE_INSTALL/modulefiles
$ module load cmake
$ cmake --version
cmake version 3.5.1

CMake suite maintained and supported by Kitware
(kitware.com/cmake).
$
```

Using craypkg-gen to integrate third party software with Cray Programming Environment

- ***Custom Product Integration and the Cray Programming Environment* -- CUG 2015**
- **Creates modulefiles**
  - Intel compiler
  - PGI compiler downloaded from PGI (not Cray)
- **Generate pkg-config (*.pc) files**
  - Allows integration with the CrayPE compiler drivers
- **Create an RPM of the software**

Boost C++ Library Example

- Open source C++ library
- Has a wide variety of libraries
- Available at **boost.org**
- Use craypkg-gen to:
  - Create pkgconfig files
  - Create modulefile
  - Create an RPM

Building Boost


```
$ tar xzvf boost_1_59_0.tar.gz
$ cd boost_1_59_0
$ export CC=cc
$ export CXX=CC
$ export PE_INSTALL=<PE_Installation Directory>
$ ./bootstrap.sh --prefix=$PE_INSTALL/boost/1.59.0/
CRAY –-without-libraries=python cflags="-hgnu -h
ipa0" cxxflags="-hgnu -hipa0”
$ ./b2 toolset=cray link=static     #static
$ ./b2 toolset=cray.                #dynamic
$ ./b2 toolset=cray install
$
```

Creating pkgconfig_files for boost

- craypkg-gen '-p' option
- Create pkgconfig/<*library-name*>.pc files in the software's library directories
- Modulefile sets env. variable to point to these files
- CrayPE compiler drivers use them to set compiler options to find appropriate header files and libraries

```
$ module load craypkg-gen
$ craypkg-gen -p $PE_INSTALL/boost/1.59.0/CRAY
```

Creating boost modulefile

- craypkg-gen '-m' option
- Initializes environment variables such as $PATH and $MANPATH
- Sets env. variables to point to the pkgconfig files
- Creates a set_default script
  - Used to make modulefile the default version

```
$ module load craypkg-gen
$ craypkg-gen -m $PE_INSTALL/boost/1.59.0
```

Building a boost RPM

- craypkg-gen '-r' option
- RPM package can be installed on other systems
- Can be used to transfer software from a user's local directory to a system directory
- '-prefix' option used to specify destination directory

```
$ module load craypkg-gen
$ craypkg-gen -r $PE_INSTALL/boost/1.59.0
 –prefix=/opt/local
$
```

boost modulefile and craype integration

```
$ module load PrgEnv-cray
$ module swap craype-network-aries craype-network-none
$ module unload cray-mpich cray-libsci
$ module use $PE_INSTALL/modulefiles
$ module load boost
$ cat main.c
int main() { }
$ cc -c -craype-verbose main.c
driver.cc -hcpu=ivybridge -hstatic -D__CRAY_IVYBRIDGE -D__CRAYXT_COMPUTE_LINUX_TARGET
-hnetwork=none -c main.c -Wl,--rpath=/opt/cray/cce/8.5.0/craylibs/x86-64
-hlast_user_arg -nostdinc -ibase-compiler /opt/cray/cce/8.5.0/CC/x86-64/
compiler_include_base -isystem /opt/cray/cce/8.5.0/craylibs/x86-64/include
-I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include
-I/opt/gcc/4.8.1/snos/lib/gcc/x86_64-suse-linux/4.8.1/include-fixed -isystem /usr/
include -ugcc_base=/opt/gcc/4.8.1/snos -uno_driver_libs
-I$PE_INSTALL/boost/1.59.0/CRAY/include -I/opt/cray/rca/1.0.0-2.0502.60530.1.62.ari/
include -I/opt/cray/cce/8.5.0/craylibs/x86-64/pkgconfig/..//include -I/opt/cray/krca/
1.0.0-2.0502.63139.4.31.ari/include
-I/opt/cray-hss-devel/7.2.0/include
$
```

boost modulefile and craype integration

- **Assuming that a GCC version of Boost 1.59.0 was also built and installed into $PE_INSTALL/boost/1.59.0/GNU, then a module swap to PrgEnv-gnu will result in the CC compiler driver automatically using the GCC version of the Boost library:**

```
$ module swap PrgEnv-cray PrgEnv-gnu  2>/dev/error
$ module unload cray-libsci
$ cc -c -craype-verbose main.c
gcc -march=corei7-avx -static -D__CRAY_IVYBRIDGE -
D__CRAYXT_COMPUTE_LINUX_TARGET
-upthread_mutex_destroy -D__TARGET_LINUX__ -c main.c
-I$PE_INSTALL/boost/1.59.0/GNU/include
$
```

mpi4py example

- Allow python programs to use MPI library
- Can be downloaded from pypi.python.org
- Build instructions provided in the CUG paper
- To create mpi4py modulefile:

```
# craypkg-gen -m $PE_INSTALL/mpi4py/2.0.0
```

- Add following line to the mpi4py modulefile:

```
prepend-path PYTHONPATH $PREFIX/lib64/python2.7/site-packages
```

mpi4py example


```
$ cat test.py
from mpi4py import MPI
import os
import glob
COMM = MPI.COMM_WORLD
irank = COMM.Get_rank()
print 'Hello world from rank', irank
$ module use $PE_INSTALL/modulefiles
$ module load mpi4py
$ aprun -n 8 python test.py
Hello world from rank 2
Hello world from rank 4
Hello world from rank 5
Hello world from rank 6
Hello world from rank 7
Hello world from rank 3
Hello world from rank 0
Hello world from rank 1
Application 8209638 resources: utime ~0s, stime ~0s, Rss ~9892, inblocks
~4282, outblocks ~12
$
```

Future Opportunities

- **Craype-installer**
  - Ability to remove older PE releases
  - Finer granularity on selecting products to install or uninstall
- **PrgEnv modulefiles**
  - Snapshot/Restore feature – ability to create user customized PrgEnv modulefiles
- **Address Autoconf Issues**
  - Faster linker (gold,LLVM lld)
  - Automatically detect configure scenario

Summary

- **CLE 6.0 changes to Cray Programing Environment**
  - Overview of installation
  - CLE 6.0 shell initialization files
- **Enhancements to CLE 6.0 PrgEnv modulefiles**
  - SITE_MODULE_NAMES
  - cdt modulefile
- **Recent Topics in porting and integrating third party software with Cray PE**
  - Issues with porting and GNU Autoconf
  - Cmake 3.5 compatibility with CLE
  - craypkg-gen examples

Q&A
Suggestions!
Feedback!

Geir Johansen
geir@cray.com

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.: 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.