Eigene Software

Software kompilieren

Da nicht jedes Programm vom HPC-Team optimiert werden kann, da es beispielsweise nur in Binär-Form vorliegt, reicht es, wenn man die Binär-Dateien in sein Homeverzeichnis kopiert. Da diese auf allen Rechenknoten unter demselben Pfad zur verfügung steht, kann man somit das Programm in seinen Job-Scripten verwenden.

Muss jedoch Software kompiliert werden sind mehrere Dinge zu beachten. Alle Komponenten im Cluster laufen auf Intel-Prozessoren mit x86_64 jedoch aus unterschiedlichen Generationen. Wenn man nun sein Programm auf hpc-login kompiliert, dann sollte man keine besonderen Optimierungen verwenden und auch nicht auf spezielle Chip-Erweiterungen setzen. Da diese möglicherweise nicht auf den anderen Systemen zur Verfügung stehen.

Um ein Programm genau auf die Hardware der Rechenknoten zu optimieren empfieht es sich somit einen Job zu schreiben der das Programm kompiliert.

Falls dabei der Intel-Compiler (module load intel) verwendet wird, sollten im ersten Schritt die folgenden Compiler-Optionen gesetzt werden.

Intel-Compiler-Flags
`-O2 -xHost`

Weiter Optimierungs-Optionen finden sich auf der Intel-Webseite.

Falls der GCC verwendet wird, sollten die folgenden Compiler-Optionen gesetzt werden.

GCC-Flags
`-O2 -march=native -mtune=native`

Mit diesen Flags sollte das Programm einen großteil der möglichen Optimierungen für das System beinhalten und sicher laufen.

Nachdem das Programm kompiliert wurde kann es im Homeverzeichnis gespeichert werden und für weitere Jobs verwendet werden.

⚠ Jobs ohne Angabe einer Architektur können auf der UV2000 oder den Ivybridge-Knoten starten oder auch gemischt bei parallelen Jobs. Es empfiehlt sich also eine Erkennung dafür in das Job-Script zu packen und abhängig vom System ein anderes Binary zu starten.

Eigenes Module-File anlegen

Es besteht die Möglichkeit eigene Module-Files anzulegen um weitere Software innerhalb eines Jobs zu laden. Dazu muss das Modulefile nach ~/.modules abgelegt werden. Am einfachsten ist es, wenn man folgendes Template kopiert und die Variablen am Anfang anpasst.

```
#%Module########
##
# Load dependencies
set reqs "hpc_basis"
source /software/modules/common/common.tcl

set softwareName "Test-Software"
set softwareVersion "1.0"
# installation base folder of software
set softwareInstallFolder [resolvePath /software/$softwareName/$softwareVersion/REPLACE]

set msg "Sets environment variables for $softwareName $softwareVersion"

# Can be modified
```

```
set bin_dir "$softwareInstallFolder/bin"
set includes "$softwareInstallFolder/include"
set libs "$softwareInstallFolder/lib"
set uses_python "1"

# Should not be changed

proc ModulesHelp { } {
    global hpc_basis_version

    puts stderr "\t$msg"
}

module-whatis "$msg"

if { ! [ file isdirectory $softwareInstallFolder ] } {
    puts stderr "\n!!!ERROR!!!\n"
    puts stderr "$softwareName not available for architecture ${::env(ARCH)}!"
    puts stderr "$softwareInstallFolder does not exist"
    exit
}

if {[string compare $bin_dir ""] != 0} {
    prepend-path PATH $bin_dir
}

if {[string compare $libs ""] != 0} {
    prepend-path LIBRARY_PATH $libs
    prepend-path LD_LIBRARY_PATH $libs
}

if {[string compare $includes ""] != 0} {
    prepend-path INCLUDE $includes
    prepend-path C_INCLUDE_PATH $includes
    prepend-path CPLUS_INCLUDE_PATH $includes
    prepend-path INCLUDE_PATH $includes
    prepend-path LD_INCLUDE_PATH $includes
}

if {[string compare $uses_python ""] != 0} {
        set p_version "python${::env(PYTHON_VERSION)}"
        set p_libdir [string range $p_version 0 8]
         prepend-path PYTHONPATH $softwareInstallFolder/lib/$p_libdir/site-packages
}
# Additional Paths
# prepend-path PATH $softwareInstallFolder/scripts


if { [ module-info mode load ] &&  ! [ catch {exec tty -s} ] } {
    puts stderr "\n  $softwareName ($softwareVersion)"
    puts stderr "  |----------------------------------------------------------------~o"
    if {[string compare $bin_dir ""] != 0} {
        puts stderr "    binaries  : $bin_dir"
    }
    if {[string compare $libs ""] != 0} {
        puts stderr "    libs       : $libs"
    }
    if {[string compare $includes ""] != 0} {
        puts stderr "    includes  : $includes"
    }
}
```