RAVEN Installation, Documentation and Workflow

RAVEN workshop

2015-Apr-26

PSA 2015 - April 26th 2015, Sun Valley (ID)

www.inl.gov

Idaho National Laboratory

Presentation Overview

- Ways of Getting Raven (Release or hpcgitlab)?
- Documentation (Wiki and PDF User Manual)
- Installing Release and Running
- Advanced Installation and Development

Obtaining RAVEN Overview

- Access to RAVEN (requires license)
- Access to INL HPC (requires approval, optional)
- Supported Systems:
  - Mac OSX Yosemite or Mavericks* (* requires MOOSE)
  - Linux (Fedora 21 and Ubuntu 14.4 and newer, Virtual machines okay)
- RAVEN: Release or hpcgitlab?
  - Release: Simpler, but less up to date, harder to develop with
  - INL's HPC gitlab: Up to date, easier to develop with, harder to setup

Manual and other Fine Documents

- We send a PDF manual with the release.
- The wiki contains a precreated version:

  https://hpcgitlab.inl.gov/idaholab/raven/wikis/raven_user_manual.pdf
- The wiki contains various documentation:
  - https://hpcgitlab.inl.gov/idaholab/raven/wikis/home
- Advanced: The manual is in raven/doc/user_manual and there is a Makefile that can be used to get `pdflatex` to generate it

Package Variations

- Source code
  - Example: `raven_framework_tag_number_5_source.tar.gz`
  - Contains the source code for RAVEN, CROW and needed MOOSE
- RAVEN libraries only
  - Example: `raven_libs_framework_only_OSX_10.10.3_tag_number_5.dmg`
  - Contains the RAVEN libraries needed for OSX
- Complete RAVEN package
  - Example: `raven_libs_framework_and_crow_OSX_10.10.3_tag_number_4.dmg`
  - Contains both the RAVEN libraries and RAVEN and CROW.

Package Installation

- OSX Yosemite
  - Install XQuartz and XCode command line tools
  - Install raven_libs_framework_and_crow….dmg
    - (Control click .pkg -> Open With -> Installer)
- Linux (Ubuntu 14.4 and newer or Fedora 21)
  - Use apt-get or yum to install needed dependencies (see manual for command line)
  - untar the source code
    - `tar -xvzf raven_framework_*_source.tar.gz`
  - Build crow
    - `cd trunk/crow; ./crow_build`
  - Test
    - `cd ../raven; ./run_tests --re=framework --skip-config-checks`

Running RAVEN

- RAVEN uses the Driver.py file, and takes xml input files on the command line
  - `python framework/Driver.py tests/framework/test_Grid_Sampler.xml`
- Alternatively the raven_framework script can be used
  - `raven_framework tests/framework/test_Grid_Sampler.xml`
  - This can be added to the path, or linked to somewhere that is in the path (This is automatically done in the OSX Yosemite installer):
  - `PATH="$PATH:/path/to/raven"`
- The files are xml files, and can be edited by any text editor such as TextEdit on OSX or gedit on Linux.

Development Installation and Workflows

- Libraries
- Source code (from git)
- Compiling
- Patch workflow

Installation – RAVEN Dependencies

- RAVEN requires various python libraries to run (such as numpy and scipy)
- MOOSE is used by RAVEN and is available at http://www.mooseframework.org
- OSX - Install XQuartz and XCode command line tools
  - OSX Mavericks
    - Install MOOSE's `mavericks-environment.pkg`, then
    - Use miniconda to install scikit-learn
  - OSX Yosemite
    - Install MOOSE `yosemite-environment.pkg`, or
    - install RAVEN libraries only package
- Linux
  - Use `apt-get` or `yum` to install libraries
- (For non-supported OS's there is a script `raven_libs_script.sh` that might help)

Installation - Git

- Git is the revision control system used for RAVEN (and CROW and MOOSE …)
- It stores the current version of the software and past revisions.
- Git stores various branches, which have different information
  - (Example: One branch can be for developing a new feature)
- Resource for learning Git: Pro Git book, available online at: http://git-scm.com/book/en/v2
- Quick command reference:
  - `git clone <url> #Initializes the local repository`
  - `git pull #Updates the local repository`
  - `git checkout <branch> #Switches between branches`
  - `git add, git commit, git push, #some of the commands needed to do development with git.`

Installation – Getting Source Code from Git

- Initial setup:
  - Either set up MOOSE following http://mooseframework.org/getting-started/
  - or only get the MOOSE source code
    - `git clone https://github.com/idaholab/moose.git`
    - `cd moose`
    - `git submodule init libmesh`
    - `git submodule update libmesh`
    - `cd ..`
  - Get CROW and RAVEN from Git repository:
    - `git clone git@hpcgitlab.inl.gov:idaholab/crow.git`
    - `git clone git@hpcgitlab.inl.gov:idaholab/raven.git`

Directory Tree

- The directory tree should look something like this (most of the folders are not shown):
- projects
  - moose
    - libmesh
  - crow
  - raven
  - bison (optional)
  - relap-7 (optional)
  - …

Installation – Compiling Source Code

- Either the script can be used, or python setup can be used
  - `cd crow`
  - `./crow_build`
- or Python setup
  - `cd crow`
  - `python setup.py build_ext build install --install-platlib=`pwd`/install`
- Then test:
  - `cd ../raven`
  - `./run_tests --re=framework --skip-config-checks`

Updating Software

- The basics of updating the software are:
  - `cd crow`
  - `git pull`
  - `./crow_build`
  - `cd ../raven`
  - `git pull`
- MOOSE and libmesh can also be updated if needed.

Development Workflow - Part 1

- Create ticket in hpcgitlab
  - https://hpcgitlab.inl.gov/idaholab/raven/issues
- Create branch to work on
  - `git checkout -b  cogljj/fix_coffee`
- Do the development
- Tell git about new and changed files
  - `git add -u directory`
  - `git add new_file.c`
- Commit the files (and add a comment)
  - `git commit`
- Push the files upstream
  - `git push --set-upstream origin cogljj/fix_coffee`
  - `git push #every time after the first time`

Development Workflow - Part 2

- Run through checklist on your own code
  - https://hpcgitlab.inl.gov/idaholab/raven/wikis/Developer_Information#checklist-for-merging
- Create the merge request
  - https://hpcgitlab.inl.gov/idaholab/raven/merge_requests/new
  - Note: if the merge request cannot be automatically done, usually a git merge is required:
    - `git fetch; git merge origin/devel`
- A second developer will review the checklist, and either accept the merge request, or request fixes
- After the merge is accepted, the regression tests are automatically run, and if they pass, devel is merged with master

Questions?