hostlist Documentation

Release 0.0.1

Christopher Moussa

May 12, 2023

Basics

py-hostlist processes slurm-style hostlist strings and can return those strings in manipulated fashion.

Get py-hostlist from the github repository:

```
$ git clone https://github.com/llnl/py-hostlist.git
$ cd py-hostlist
```

From here, build and install the package on your machine:

```
$ python setup.py sdist bdist_wheel
$ pip install dist/py_hostlist-0.0.1.dev0-py2.py3-none-any.whl
```

Now you can start using it!

```
$ hostlist -h
```

CHAPTER 1

Features

This is a high-level overview of features that make up py-hostlist, a Python implementation of a hostlist manager.

1.1 Using Regular Expressions

py-hostlist utilizes the re package to look for certain types of hostlist strings to manipulate. Once it matches with a hostlist string, it breaks up the expression into certain control groups with which it can perform a number of operations on. In general, input hostlist strings are broken up into three control groups:

1. the host name
2. the range of nodes contained by the host
3. any suffixes or domains appended to the hostlist

1.2 String Manipulation

After an input string is matched, the control groups are cast to string variables, where they can be stripped of characters such as brackets, dashes, and commas in order to perform necessary operations, such as expanding or compressing a list.

1.3 Different Input Types

py-hostlist is flexible with the types of input passed into its methods. Both lists and strings can be passed into each of the operations that it supports. For example, the compress method accepts the following list input:

```
['node1','node2','node3','node4']
```

as well as a simple string:

```
'node1,node2,node3,node4'
```

Both will return the following: `node[1-4]`

py-hostlist achieves this functionality by checking the type of input before it attempts to do any manipulations/operations.

CHAPTER 2

Basic Usage

You can use the command line to process your hostlist strings by using the following command:

```
python cla_hostlist.py <method> <args>...
```

Here is a list of all of the methods available:

-h, --help Display this message.

-q, --quiet Quiet output (exit non-zero if empty hostlist)

-d, --delimiters Set output delimiter (default = ",")

-c, --count Print the number of hosts

-s, --size Output at most N hosts (-N for last N hosts)

-e, --expand Expand a compressed hostlist

-a, --abbreviate Compress an expanded hostlist

-t, --tighten Return a hostlist string

-m, --minus Subtract all HOSTLIST args from first HOSTLIST

-i, --intersection Intersection of all HOSTLIST args

-x, --exclude Exclude all HOSTLIST args from first HOSTLIST

-X, --xor Symmetric difference of all HOSTLIST args

-u, --union Union of all HOSTLIST arguments

-n, --nth Output the host at index N

-R, --remove Remove all occurences of NODE from HOSTLIST

-S, --sort Return sorted HOSTLIST

-F, --find Output position of HOST in result HOSTLIST

For example, to execute the expand function displayed above, users can run the following:

```
python cla_hostlist.py -e node[1-4]
```

This will return `node1,node2,node3,node4`.

CHAPTER 3

Method Reference

This is a high-level overview of the methods that make up py-hostlist.

3.1 Methods

`expand(nodelist)`

Parameters: nodelist **(str)** - The hostlist string.

Returns: An expanded hostlist string.

Description: expand takes in a hostlist string and returns a list of individual hostnames. For example, the input string `node[1-4]` will return `node1,node2,node3,node4`. The expand method will return the suffix string in its final expansion, as well as prepend any leading zeros found in the input string. Multiple ranges can be specified within brackets of a cluster like so:

```
node[1-4,6-10,19].suffix.com
```

Multiple clusters to be expanded can also be specified in an input string by separating the clusters with a comma followed by a space. An example below:

```
node1-[1-4], node2-[5-9].suffix.com
```

`compress_range(nodelist)`

Parameters: nodelist **(str)** or **(list)** - The expanded hostlist string.

Returns: A compressed hostlist string.

Description: compress_range takes in a hostlist list string and returns an ordered hostlist with a range. For example, the input list `['node1','node2','node3','node4']` will return `node[1-4]`. The compress_range method can also recognize multiple ranges.

compress_range can also recognize a string input. Going back to the example above, the following input will also be recognized: `'node1,node2,node3,node4'`. This will also return `node[1-4]`.

`compress(nodelist)`

Parameters: nodelist **(str)** - The hostlist string.

Returns: An ordered hostlist string.

Description: compress takes in a hostlist list string and returns an ordered hotlist string. For example, the input string `['node1','node2','node3','node4']` will return `[node1,node2,node3,node4]`.

`diff(nodelist1, nodelist2)`

Parameters: nodelist1 **(str)** or **(list)** - The hostlist string to be subtracted from.

following nodelists... **(str)** or **(list)**: The other hostlist strings.

Returns: A remaining hostlist string resulting from subtracting the following nodelists from nodelist1.

Description: diff will subtract elements in all following nodelists from nodelist1 and return a remaining hostlist. It accepts both string and list inputs.

`intersect(*arg)`

Parameters: hostlist strings **(str)** or **(list)** - Any number of nodelists to be intersected.

Returns: An intersecting hostlist string from all hostlist args.

Description: intersect will return a list of intersection nodes given n lists of nodes. It will sort the nodes in ascending order upon returning.

`union_nodes(*arg)`

Parameters: hostlist strings **(str)** or **(list)** - Any number of nodelists to be combined.

Returns: A union hostlist string from all hostlist args.

Description: union will return the union between n lists of nodes. It will sort the nodes in ascending order upon returning.

`nth(nodelist, n)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

n **(int)** - The index to search.

Returns: The host at the specified index.

Description: nth takes in two parameters: a hostlist string (similar to expand()'s parameter) and an index *n*. It will return the *nth* node in that range.

`find(nodelist, node)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

node **(str)** - The host to be searched inside of the hostlist string.

Returns: The position of the host within the hostlist string.

Description: find will return the position of the node in the input nodelist.

`count(nodelist)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

Returns: The number of nodes in the hostlist string.

Description: count will print the number of hosts in the nodelist. The input can accept a hostlist that is already expanded or one that contains ranges. For example, the input `node[1-5]` will return `5`.

`remove_node(nodelist, node)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

node **(str)** - The node to be removed.

Returns: The resulting hostlist upon deletion.

Description: remove_node() will remove all occurences of *node* in the nodelist. The input can accept a hostlist that is already expanded or one that contains ranges.

`delimiter(nodelist, d)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

d **(str)** - The custom delimiter.

Returns: The resulting hostlist string with its custom delimiter.

Description: delimiter() will take the hostlist string and output it with the specified delimiter *d*, which can be any string.

`size_hostlist(nodelist, N)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

N **(int)** - The number of hosts to print.

Returns: The resulting hostlist string with custom size.

Description: This method will print at most *N* hosts from the hostlist input. If a negative *N* is passed in, the output will consist of the last N hosts from the hostlist input.

`xor(*arg)`

Parameters: hostlist strings **(str)** or **(list)** - Any number of nodelists to be combined.

Returns: The resulting xor list.

Description: xor() takes the symmetric difference of an arbitrary number of hostlists passed in.

`exclude(*arg)`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

node **(str)** - The node to be excluded.

Returns: The resulting hostlist string without the nodes specified.

Description: exclude() will return a hostlist that excludes any nodes specified after the first argument, which is the original hostlist. Each node to be excluded must be passed in one at a time as separate arguments.

`quiet(nodelist=[])`

Parameters: nodelist **(str)** or **(list)** - The hostlist string.

Returns: None or non-zero output if an empty hostlist is passed in.

Description: returns quiet output for a hostlist input. It will exit non-zero if there is an empty hostlist passed in.

3.2 Helper Methods

`append_hostname(machine_name, num_list)`

Parameters: machine_name **(str)** - The name of the cluster.

num_list **(list)** - The list of nodes to be appended to the cluster name.

Returns: A hostlist string with the hostname and node numbers.

Description: append_hostname takes in two parameters: the name of the machine and its range of nodes; it is a helper method that will append the machine name (the host) to the node numbers it contains.

`sort_nodes(nodelist)`

Parameters: nodelist **(str)** - The hostlist string.

Returns: The hostlist string in ascending order.

Description: sort_nodes takes in a list of nodes; it is a helper method that will return a sorted string of those nodes in ascending order.

CHAPTER 4

py-hostlist Man Page

4.1 SYNOPSIS

```
python cla_hostlist.py [OPTION] ARGS
```

4.2 DESCRIPTION

py-hostlist is a hostlist utility implemented in Python. It uses regular expressions to manipulate hostlists and perform logic functions between different types of hostlists.

4.3 OPTIONS

-h, --help
Display this message.

-q, --quiet
Quiet output (exit non-zero if empty hostlist).

-d, --delimiters
Set output delimiter (default = ",").

-c, --count
Print the number of hosts.

-s, --size
Output at most N hosts (-N for last N hosts).

-e, --expand
Expand a compressed hostlist.

-a, --abbreviate
Compress an expanded hostlist.

-t, --tighten
Return a hostlist string.

-m, --minus
Subtract all HOSTLIST args from first HOSTLIST.

-i, --intersection
Intersection of all HOSTLIST args.

-x, --exclude
Exclude all HOSTLIST args from first HOSTLIST.

-X, --xor
Symmetric difference of all HOSTLIST args.

-u, --union
Union of all HOSTLIST arguments.

-n, --nth
Output the host at index N.

-R, --remove
Remove all occurences of NODE from HOSTLIST.

-S, --sort
Return a sorted HOSTLIST.

-F, --find
Output position of HOST in result HOSTLIST.

4.4 RESTRICTIONS

For most of the functions, hostlists can be input as any of the following three formats:

- `foo1,foo2,foo3,foo4,foo5`
- `foo[1-5]`
- `[foo1,foo2,foo3,foo4,foo5]`

4.5 EXAMPLES

1. To expand a hostlist:

   ```
   python cla_hostlist.py -e foo[1-5]
   ```

2. To set a custom delimiter:

   ```
   python cla_hostlist.py -d [DELIMITER] foo[1-5]
   ```

3. To see the first N hosts:

   ```
   python cla_hostlist.py -s [N] foo[1-5]
   ```

4. To exclude a node from a hostlist:

   ```
   python cla_hostlist.py -x foo[1-5] [EXCLUDED NODE] [EXCLUDED NODE]...
   ```

5. To find the nth host in a hostlist:

   ```
   python cla_hostlist.py -n [N] foo[1-5]
   ```

6. To remove all occurences of a node from a hostlist:

   ```
   python cla_hostlist.py -R [NODE] foo[1-5]
   ```

7. To find the position of a specific node:

   ```
   python cla_hostlist.py -F [NODE] foo[1-50]
   ```

The py-hostlist source code and all documentation may be downloaded from <https://github.com/llnl/py-hostlist.git>

CHAPTER 5

Contribution Guide

This guide is intended for developers or administrators who want to contribute a new feature or bugfix to py-hostlist. It assumes that you have at least some familiarity with Git VCS and GitHub. The guide will show a few examples of contributing workflows and discuss the granularity of pull-requests (PRs). It will also discuss the tests your PR must pass in order to be accepted into py-hostlist.

First, what is a PR? Quoting Bitbucket's tutorials:

> Pull requests are a mechanism for a developer to notify team members that they have **completed a feature**. The pull request is more than just a notification—it's a dedicated forum for discussing the proposed feature.

Important is **completed feature**. The changes one proposes in a PR should correspond to one feature/bugfix/extension/etc. One can create PRs with changes relevant to different ideas, however reviewing such PRs becomes tedious and error prone. If possible, try to follow the **one-PR-one-package/feature** rule.

5.1 Continuous Integration

py-hostlist uses Travis CI for Continuous Integration testing. This means that every time you submit a pull request, a series of tests will be run to make sure you didn't accidentally introduce any bugs into py-hostlist. **Your PR will not be accepted until it passes all of these tests.** While you can certainly wait for the results of these tests after submitting a PR, we recommend that you run them locally to speed up the review process.

Note: Oftentimes, Travis will fail for reasons other than a problem with your PR. For example, apt-get, pip, or homebrew will fail to download one of the dependencies for the test suite, or a transient bug will cause the unit tests to timeout. If Travis fails, click the "Details" link and click on the test(s) that is failing. If it doesn't look like it is failing for reasons related to your PR, you have two options. If you have write permissions for the py-hostlist repository, you should see a "Restart job" button on the right-hand side. If not, you can close and reopen your PR to rerun all of the tests. If the same test keeps failing, there may be a problem with your PR. If you notice that every recent PR is failing with the same error message, it may be that Travis is down or one of py-hostlist's dependencies put out a new release that is causing problems. If this is the case, please file an issue.

If you take a look in `py-hostlist/.travis.yml`, you'll notice that we test against Python 2.7, and 3.3-3.7 on macOS. We currently perform unit testing:

Unit tests ensure that core py-hostlist features like expand or compress_range are working as expected. If your PR only adds new packages or modifies existing ones, there's very little chance that your changes could cause the unit tests to fail. However, if you make changes to py-hostlist's core libraries, you should run the unit tests to make sure you didn't break anything.

To run the unit tests, use:

```
$ python py-hostlist/unittest_hostlist.py
```

It should only take a few seconds to complete. If you know you are only modifying a single feature, you can run a single unit test at a time:

```
$ python py-hostlist/unittest_hostlist.py TestHostlistMethods.test_expand
```

5.2 Git Workflows

py-hostlist is still in the alpha stages of development. Most of our users run off of the develop branch, and fixes and new features are constantly being merged. So how do you keep up-to-date with upstream while maintaining your own local differences and contributing PRs to py-hostlist?

5.2.1 Branching

The easiest way to contribute a pull request is to make all of your changes on new branches. Make sure your `develop` is up-to-date and create a new branch off of it:

```
$ git checkout develop
$ git pull upstream develop
$ git branch <descriptive_branch_name>
$ git checkout <descriptive_branch_name>
```

Here we assume that the local `develop` branch tracks the upstream develop branch of py-hostlist. This is not a requirement and you could also do the same with remote branches. But for some it is more convenient to have a local branch that tracks upstream.

Normally we prefer that commits pertaining to a package `<package-name>` have a message `<package-name>: descriptive message`. It is important to add descriptive message so that others, who might be looking at your changes later would understand the rationale behind them.

Now, you can make your changes while keeping the `develop` branch pure. Edit a few files and commit them by running:

```
$ git add <files_to_be_part_of_the_commit>
$ git commit --message <descriptive_message_of_this_particular_commit>
```

Next, push it to your remote fork and create a PR:

```
$ git push origin <descriptive_branch_name> --set-upstream
```

GitHub provides a tutorial on how to file a pull request. When you send the request, make `develop` the destination branch.

CHAPTER 6

Developer Guide

This guide is intended for people who want to work on py-hostlist itself.

6.1 Overview

py-hostlist is designed with two separate roles in mind:

1. **Users**, who need to use the hostlist tool without knowing all of the details about how it is built.
2. **Developers** who work on py-hostlist, add new features, and try to make the jobs of users easier.

As you might expect, there are many types of users with different levels of sophistication, and py-hostlist is designed to accommodate both simple and complex use cases for this tool. A user who only knows that he needs a small task, like expanding a hostlist, should be able to type something simple like `python cla_hostlist.py -e foo[1-4]` and get a result in return.

6.2 Directory Structure

Here is a high level view of py-hostlist's directory structure:

```
dist/                       <- packaged build

docs/
  Makefile
  conf.py
  index.rst
  make.bat
  build/                    <- HTML pages
  source/                   <- ReadTheDocs pages
    .doctrees/

hostlist/
```

(continues on next page)

(continued from previous page)

```
__init__.py
cla_hostlist.py           <- command-line arguments
hostlist.py               <- main features and methods
unittest_hostlist.py      <- unit tests for hostlist.py
```

6.3 Code Structure

For an overview of the various Python modules in py-hostlist, please see the Method Reference section.

CHAPTER 7

Adding Unit Tests

This guide is intended for anyone who wants to add unit tests for any new features added to py-hostlist.

7.1 Python's Unit Test Structure

First, some background on Python's unit test structure. py-hostlist utilizes the unittest unit testing framework in order to test its methods. All of the methods consist of testing the equality of two strings: the *expected* value and the *actual* value. An example:

```
def test_expand(self):
    expected = 'quartz4,quartz5,quartz6,quartz7,quartz8'
    test = hl.expand('quartz[4-8]')
    self.assertEqual(test, expected)
```

`def test_expand(self)` defines the method name. In this case, we are just testing the expand method; hence, the name `test_expand`. If we were to test something specific about expand, such as expanding with multiple ranges, than the method name should reflect it, i.e. something like `test_expand_multi_range`.

Note: All of the test methods used must start with the word `test`. This informs the test runner about which methods represent tests.

`expected = 'quartz4,quartz5,quartz6,quartz7,quartz8'` is the expected value we expect to get from running the `expand` method.

`test = hl.expand('quartz[4-8]')` stores a string in `test` resulting from calling hostlist's `expand` method on the hostlist `quartz[4-8]`.

Note: The hostlists passed into hostlist's methods are *strings*. Although the command-line tool does not require quotes around its arguments, the unit test file does.

`self.assertEqual(test, expected)` compares the variables `expected` and `test` and ensures that they are equal.

7.2 How to Add a Unit Test to unittest_hostlist

In order to add a unit test to py-hostlist, use the following steps:

1. Give your test method a name which: **1)** includes the name of the Python method you are testing, and **2)** clearly describes what it is testing.
2. Define the expected value you are looking to get and store it in the variable `expected`.
3. Call the method from within py-hostlist using `hl.<your_method>(<value>)` and store it in the variable `test`.
4. Call `self.assertEqual(test, expected)` to compare the two values. If the test fails, it will most likely be because the strings are not equal, in which case you will see the string returned by both variables to compare.

7.3 Running Your Unit Tests

When you push your new test methods to GitHub, Travis CI will automatically run the unit test script to check for successes or failures. However, to run your test methods locally, just run the following command from the `hostlist` directory:

```
python unittest_hostlist.py
```

If you just want to run your specific test method, you can use the following command:

```
python unittest_hostlist.py TestHostlistMethods.<your_method_name>
```

CHAPTER 8

py-hostlist package

8.1 Submodules

8.2 hostlist.cla_hostlist module

hostlist.cla_hostlist.**main**()

hostlist.cla_hostlist.**msg**(*name=None*)

8.3 hostlist.hostlist module

A slurm-style hostlist processor.

hostlist.hostlist.**append_hostname**(*machine_name*, *num_list*)
Helper method to append the hostname to node numbers.

Parameters

- **machine_name** – The name of the cluster.
- **num_list** – The list of nodes to be appended to the cluster name.

Returns A hostlist string with the hostname and node numbers.

hostlist.hostlist.**compress**(*nodelist*)
compress will return a hostlist string given a list of hostnames.

Param nodelist: The hostlist string.

Returns The hostlist string.

hostlist.hostlist.**compress_range**(*nodelist*)
compress_range will return a compressed hostlist string given a list of hostnames.

Param nodelist: The expanded hostlist string.

Returns The compressed hostlist string.

hostlist.hostlist.**count**(*nodelist*)
count returns the number of hosts.

Param nodelist: The hostlist string.

Returns The number of nodes in the hostlist string.

hostlist.hostlist.**delimiter**(*nodelist*, *d*)
delimiter sets the output delimiter (default = ",")

Param nodelist: The hostlist string.

Param d: The delimiter.

Returns The resulting hostlist string with custom delimiter.

hostlist.hostlist.**diff**(**arg*)
diff will subtract elements in all subsequent lists from list 1 and return the remainder.

Param nodelist1: The hostlist string to be subtracted from.

Param following nodelists: The other hostlist strings.

Returns The remainding list from subtracting the two original lists.

hostlist.hostlist.**exclude**(**arg*)
excludes all HOSTLIST args from first HOSTLIST

Param nodelist: The hostlist string.

Param node: The node to be excluded.

Returns The resulting hostlist string without the nodes specified.

hostlist.hostlist.**expand**(*nodelist*)
expand takes in a compressed hostlist string and returns all hosts listed.

Param nodelist: The hostlist string.

Returns The expanded hostlist string.

hostlist.hostlist.**filter_python**(*nodelist*)
TODO: filter maps Python code over all hosts in result HOSTLIST

Param nodelist: The hostlist string.

hostlist.hostlist.**find**(*nodelist*, *node*)
find outputs the position of the node in the nodelist passed in.

Param nodelist: The hostlist string.

Param node: The host to be searched inside of the hostlist string.

Returns The position of the host within the hostlist string.

hostlist.hostlist.**intersect**(**arg*)
Given references to n lists, intersect return a list of intersecting nodes.

Param nodelist: Any number of nodelists to be intersected.

Returns The resulting intersected list.

hostlist.hostlist.**nth**(*nodelist*, *n*)
nth returns the nth node from a list of nodes.

Param nodelist: The hostlist string.

Param n: The index desired.

Returns The host at the specified index.

hostlist.hostlist.**quiet**(*nodelist=[]*)
quiet will return quiet output (or exit non-zero if there is an empty hostlist)

Param nodelist: The hostlist string.

hostlist.hostlist.**remove_node**(*nodelist*, *node*)
removes a node from a passed in hostlist.

Param nodelist: The hostlist string.

Param node: The node to be removed.

Returns The resulting hostlist upon deletion.

hostlist.hostlist.**size_hostlist**(*nodelist*, *N*)
size will output at most N hosts (-N for last N hosts)

Param nodelist: The hostlist string.

Param N: the number of hosts to print.

Returns The resulting hostlist string with custom size.

hostlist.hostlist.**sort_nodes**(*nodelist*)
sort_nodes is a helper method that sorts the nodes in ascending order.

Parameters **nodelist** – The hostlist string.

Returns The hostlist string in ascending order.

hostlist.hostlist.**union_nodes**(**arg*)
union_nodes returns the union between n lists of nodes.

Param nodelist: Any number of nodelists to be combined.

Returns The resulting unioned list.

hostlist.hostlist.**xor**(**arg*)
xor returns the symmetric difference between n lists of nodes.

Param nodelist: Any number of nodelists to be xor.

Returns The resulting xor list.

8.4 hostlist.unittest_hostlist module

class hostlist.unittest_hostlist.**TestHostlistMethods**(*methodName='runTest'*)
Bases: unittest.case.TestCase

test_compress()

test_compress_as_string()

test_compress_range_as_string()

test_compress_range_mixed()

test_compress_range_simple()

test_compress_range_suffix()

test_compress_range_with_hyphen()

test_count()

test_count_as_list()

test_count_multi_ranges()

test_count_with_expand()

test_delimiter_as_list()

test_delimiter_as_string()

test_delimiter_with_expand()

test_diff()

test_diff_as_string()

test_diff_multiple()

test_diff_scr()

test_diff_with_expand()

test_exclude_as_list()

test_exclude_as_string()

test_exclude_with_expand()

test_expand()

test_expand_leading_zeros()

test_expand_mixed_range()

test_expand_multi_range()

test_expand_only_commas()

test_expand_prefix_and_suffix()

test_expand_scr()

test_find()

test_find_as_list()

test_find_as_string()

test_find_doesnt_exist()

test_instersect_as_string()

test_intersect_multiple()

test_intersect_scr()

test_intersect_simple()

test_intersect_with_expand()

test_nth()

test_nth_as_list()

test_nth_as_str()

test_nth_doesnt_exist()

test_quiet()

test_quiet_as_list()

test_quiet_empty()

test_quiet_with_expand()

test_remove_node()

test_remove_node_as_list()

test_remove_node_with_expand()

test_size_as_list()

test_size_as_string()

test_size_backwards()

test_size_with_expand()

test_size_with_expand_backwards()

test_sort()

test_sort_as_string()

test_union_as_string()

test_union_multiple()

test_union_simple()

test_union_with_expand()

test_xor_as_list()

test_xor_as_string()

test_xor_with_expand()

8.5 Module contents

A slurm-style hostlist processor.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

Symbols

A

C

D

E

F

H

U

X