by João F. Matias Rodrigues and Christian von Mering
Institute of Molecular Life Sciences
University of Zürich
Switzerland

HPC-CLUST v1.2 (February 2015)

Distributed hierarchical clustering of large number of pre-aligned sequences

Contents

1 Introduction

HPC-CLUST is a set of tools designed to cluster large numbers (1 million or more) pre-aligned nucleotide sequences. HPC-CLUST performs the clustering of sequences using the Hierarchical Clustering Algorithm (HCA). There are currently three different cluster metrics implemented: single-linkage, complete-linkage, and average-linkage. There are currently 4 sequence distance functions implemented, these are: - identity gap-gap counting as match - nogap gap-gap being ignored - nogap-single like nogap, but consecutive gap-nogap's count as a single mismatch - tamura distance is calculated with the knowledge that transitions are more likely than transversions

One advantage that HCA has over other algorithms is that instead of producing only the clustering at a given threshold, it produces the set of merges happening at each threshold. With this approach, the clusters can be very quickly computed for every threshold with little extra computations. This approach allows the plotting of the variation of number of clusters with clustering threshold without requiring the clustering to be run for each threshold independently.

Another advantage of the HPC-CLUST implementation is that the single, complete, and average linkage clusterings can be computed in a single run with little extra overhead.

2 INSTALLATION

2.1 Linux/Unix/MacOSX

To install HPC-CLUST on a linux, unix, or MacOSX simply type:

./configure make make install

In the directory where you unpacked the package contents. Alternatively, if you want the program to be installed to another location instead of the default system wide /usr/local/ directory, you can change the ./configure command to:

./configure –prefix=$HOME/usr

This would install the program binaries to a directory usr/bin inside your home directory (i.e.: $HOME/usr/bin/hpc-clust), after you type the command "make install".

3 HPC-CLUST, HPC-CLUST-MPI

Two programs are provided in the HPC-CLUST package.

When running on a single machine you should use HPC-CLUST. It is multi-threaded so it can make use of several processors if you run it on a multi-core computer.

HPC-CLUST-MPI is the distributed computing version of HPC-CLUST to be used on a cluster of computers. HPC-CLUST-MPI uses the MPI library to automatically setup the communication between master and slave instances.

4 USING HPC-CLUST

To use HPC-CLUST you must first prepare your alignment file. You can accomplish this using the INFERNAL aligner (http://infernal.janelia.org/), which aligns all sequences against a model consensus sequence. Any other alignment such as multiple sequence alignment will work as well, although, in practice, for very large numbers of sequences it becomes difficult to compute a multiple sequence alignment.

HPC-CLUST takes the alignment file in aligned fasta format or non-interleaved Stockholm format. It detects automatically the format based on the first character. It switches to fasta format if the first character is '>'.

An example of non-interleaved stockholm format is:
SEQUENCE_ID1 —ATGCAT—GCATGCATGC—-... ...AT–AATT
SEQUENCE_ID2 —ATCCAC—GCACGCATGC-AT-... ...AT–ATAA
...
...

An example of aligned fasta format is:
>SEQUENCE_ID1
—ATGCAT—GCATGCATGC—-... ...AT–AATT
>SEQUENCE_ID2
—ATCCAC—GCACGCATGC-AT-... ...AT–ATAA
...

An example of how to call the INFERNAL aligner such that it produces a file in the right format is: cmalign -1 –sub –matchonly -o alignedseqs.sto bacterial-ssu-model.cm unalignedseqs.fasta

This would produce a file with the aligned sequences "alignedseqs.sto", using the "bacterial-ssu-model.cm" from the unaligned sequences file "unalignedseqs.fasta"

If the input sequences can be found in the file "alignedseqs.sto". Then to cluster the sequences one simply has to run the following command:

hpc-clust -sl true alignedseqs.sto

This will cluster the sequences using single-linkage clustering. The output will be written to the file "alignedseqs.sto.sl"

To run all linkage methods at once:

hpc-clust -sl true -cl true -al true alignedseqs.sto

In this case, the single, complete and average linkage clusterings will be written to the "alignedseqs.sto.sl", "alignedseqs.sto.cl", and "alignedseqs.sto.al", respectively.

You can change the number of cores used by hpc-clust with the -nthreads <no_cores >argument. The name of the output file using the -ofile <output_filename >argument. The threshold at which distances should be kept is specified using the -t <threshold >argument. The lower this value, the more distances will be stored, and the lower the threshold until which the clustering can proceed. However, the more distances are stored, the higher the memory requirement.

5 USING HPC-CLUST-MPI

To run the distributed version of HPC-CLUST, one needs to have the MPI library installed before installing HPC-CLUST. If the MPI library is already set up, running HPC-CLUST-MPI can be accomplished simply with:

```
mpirun -n 10 hpc-clust-mpi -sl true alignedseqs.sto
```

This will start 10 instances of hpc-clust-mpi, (1 master + 9 slaves) and perform single linkage clustering on the "alignedseqs.sto" file. The results can be found in the "alignedseqs.sto.sl" file as explained in the "USING HPC-CLUST" section above.

The MPI version of HPC-CLUST is best used in conjunction with a queuing system such as the Sun Grid Engine system. In that case, if every instance of HPC-CLUST-MPI runs on a separate machine it can make use of the combined memory of all the machines.

The exact command for submitting a job to the queuing system will depend on your local cluster configuration. An example of the command to submit a job to run hpc-clust-mpi on 10 nodes would be:

```
qsub -b y -cwd -pe orte 10 mpirun -n 10 hpc-clust-mpi -sl true alignedseqs.sto
```

The speed of the computation and memory usage can be further optimized by having more than one thread per node, in this manner the sequence data is shared between the threads. You can increase the number of threads to use in each node by specifying the -nthreads <number_threads >argument to hpc-clust-mpi.

6 EXAMPLE MERGE FILE OUTPUT

```
# seqsfile: aligned-archaea-seqs.sto
# OTU_count Merge_distance Merged_OTU_id1 Merged_OTU_id2
51094 1.0 37 38
51093 1.0 37 12176
51092 1.0 37 33214
51091 1.0 42 79
51090 1.0 42 2734
51089 1.0 42 2746
51088 1.0 42 2747
```

```
51087 1.0 42 2748
51086 1.0 42 2749
51085 1.0 42 2750
51084 1.0 42 2751
...
```

In the merge file output, each line indicates a merging event. For each merge event, the number of OTUs existing after the merge, the identity distance at which the merge occurred, and the OTU ids are shown. When a merge event occurs, the new OTU will have the same id as the smallest OTU id merged. For example, if OTUs' 37 and 12176 are merged, the new OTU will have id 37.

7 OBTAINING CLUSTERS FROM THE MERGE RESULTS

Once the merge result files (.sl,.cl,.al) have been computed, you can produce the clusters at different thresholds by using hpc-clust with the "-makeotus" option.

```
hpc-clust -makeotus alignedseqs.fa alignedseqs.fa.sl 0.97
```

Would generate the clusters produced at the 97% identity threshold from the single linkage merge file "alignedseqs.fa.sl".
You can also produce a mothur compatible OTU list with the following command: `hpc-clust -makeotus_mothur alignedseqs.fa alignedseqs.fa.sl`

8 OBTAINING CLUSTER REPRESENTATIVES

After clustering and producing an OTU file at a certain threshold it is possible to obtain a fasta file containing a set of representatives for each OTU by using the -makereps option.

```
hpc-clust -makereps alignedseqs.fa alignedseqs.fa.sl.0.97.otu
```

For each OTU, the sequence chosen to be the representative is the one having the smallest average distance to all other sequences in the OTU.

9 FAQ

Frequently asked questions and other documentation can be found at http://meringlab.org/software/hpc-clust/faq.html

ACKNOWLEDGMENTS

Sebastian Schmidt

For further information contact: João F. Matias Rodrigues <joao.rodrigues@imls.uzh.ch>