The ExaML v1.0.0 Manual

Introduction

Exascale Maximum Likelihood (ExaML) is a code for phylogenetic inference using MPI.

This code implements the popular RAxML search algorithm for maximum likelihood based inference of phylogenetic trees. It uses a radically new MPI parallelization approach that yields improved parallel efficiency, in particular on partitioned multi-gene or whole-genome datasets.

It is up to 4 times faster than RAxML-Light [1].

Similar to RAxML-Light, ExaML also implements checkpointing, SSE3, AVX vectorization and memory saving techniques. Thus, the usage philosophy is similar. To get started please first read the RAxML and RAxML-Light manuals. The current manual simply represents a quick start guide by example.

Support is provided via the RAxML google group: https://groups.google.com/forum/?hl=de&fromgroups#!forum/raxml. Please refrain from sending emails with questions directly to the authors of the code!

[1] A. Stamatakis, A.J. Aberer, C. Goll, S.A. Smith, S.A. Berger, F. Izquierdo-Carrasco: "RAxML-Light: A Tool for computing TeraByte Phylogenies", Bioinformatics 2012; doi: 10.1093/bioinformatics/bts309.

Compiling the code

This is actually a suite of tools to accomplish the task.

When you download the code there will be four directories:

```
examl
manual
parser
testData
```

Let's first compile the parser by typing:

```
cd parser
make -f Makefile.SSE3.gcc
```

This should produce an executable called `parser`

Next, let's compile ExaML. Note that you will require a MPI compiler to accomplish this (if you don't know what this is ask your local geeks). Type:

```
cd ../examl/
```

```
make -f Makefile.SSE3.gcc
```
→ this will compile the SSE3 vectorized version of the code, the binary will be called `examl`

If you have a recently purchased system you can also try to compile the AVX-vectorized version of ExaML (for performance and further details please refer to the on-line supplement of [1])

first remove the object files created when you compiled `examl`:

```
rm *.o
```
then compile the AVX version by typing:
```
make -f Makefile.AVX.gcc
```
this should produce a binary called `examl-AVX`

A Usage Example

I will walk you through a simple example of how to use the code on my laptop.

Initially we will have to transform an alignment file in relaxed phylip format (as used for RAxML) into a binary file format that can be read by ExaML. The main reason for this is to allow ExaML to read this file faster and not waste any valuable parallel computing time for this simple pre-processing task.

Help for the command line arguments of the parser can be obtained by typing:

```
./parser -h
```

parser
-s sequenceFileName
-n outputFileName
-m substitutionModel
[-c]
[-q]
[-h]

-m Type of Data:

For DNA data use: DNA
For AA data use: PROT

-c disable site pattern compression

-q Specify the file name which contains the assignment of models to alignment partitions for multiple models of substitution. For the syntax of this file please consult the manual.

-h Display this help message.

Now let's use the parser to transform the test file into a binary file by typing:

```
./parser -s ../testData/49 -m DNA -n 49.unpartitioned
```

So we are transforming an unpartitioned phylip file with 49 DNA sequences into a binary file. The parser directory will now contain a file called:

```
49.unpartitioned.binary
```

that can be read by ExaML.

If we want to partition the data, we will have to pass a standard RAxML partition file to the parser, e.g:

```
./parser -s ../testData/49 -q ../testData/49.model -m DNA -n 49.partitioned
```

This will generate a file called

```
49.unpartitioned.binary
```

Note that, every time you change the partition scheme, you will have to re-generate a binary alignment file that encodes the partitioning scheme!

Now, we are almost ready to start an ExaML run. However, you also need to provide a starting tree to ExaML (in a similar way as for RAxML-Light) by using, for instance Parsimonator or standard RAxML. More details are provided in the RAxML-Light manual.

We have included a starting tree in the testData directory, such that we can now run ExaML.

For this we will have to change into the ExaML directory by typing:

```
cd ../examl
```

On-line help regarding ExaML command line options can be obtained by typing:

```
./examl -h
```

which will yield the following output:

```
examl|examl-AVX
    -s binarySequenceFileName
    -n outputFileNames
    -m rateHeterogeneityModel
    -t userStartingTree|-R binaryCheckpointFile
    [-a]
    [-B numberOfMLtreesToSave]
    [-c numberOfCategories]
    [-D]
    [-e likelihoodEpsilon]
```

[-f d|o]
[-h]
[-i initialRearrangementSetting]
[-M]
[-Q]
[-S]
[-v]
[-w outputDirectory]

-a use the median for the discrete approximation of the GAMMA model of rate heterogeneity

DEFAULT: OFF

-B specify the number of best ML trees to save and print to file

-c Specify number of distinct rate catgories for ExaML when modelOfEvolution is set to GTRPSR
Individual per-site rates are categorized into numberOfCategories rate categories to accelerate computations.

DEFAULT: 25

-D ML search convergence criterion. This will break off ML searches if the relative Robinson-Foulds distance between the trees obtained from two consecutive lazy SPR cycles is smaller or equal to 1%. Usage recommended for very large datasets in terms of taxa. On trees with more than 500 taxa this will yield execution time improvements of approximately 50% while yielding only slightly worse trees.

DEFAULT: OFF

-e set model optimization precision in log likelihood units for final optimization of model parameters

DEFAULT: 0.1

-f select algorithm:

"-f d": new rapid hill-climbing

DEFAULT: ON

"-f o": old and slower rapid hill-climbing without heuristic cutoff

DEFAULT for "-f": new rapid hill climbing

-h Display this help message.

-i Initial rearrangement setting for the subsequent application of topological

changes phase

-m Model of rate heterogeneity

select "-m PSR" for the per-site rate category model (this used to be called CAT in RAxML)
select "-m GAMMA" for the gamma model of rate heterogeneity with 4 discrete rates

-M Switch on estimation of individual per-partition branch lengths. Only has effect when used in combination with "-q" Branch lengths for individual partitions will be printed to separate files. A weighted average of the branch lengths is computed by using the respective partition lengths.

DEFAULT: OFF

-n Specifies the name of the output file.

-Q Enable alternative data/load distribution algorithm for datasets with many partitions
In particular under PSR this can lead to parallel performance improvements of up to factor 10!

-R read in a binary checkpoint file called ExaML_binaryCheckpoint.RUN_ID_number

-s Specify the name of the BINARY alignment data file generated by the parser component

-S turn on memory saving option for gappy multi-gene alignments. For large and gappy datasets specify -S to save memory.
This will produce slightly different likelihood values, may be a bit slower but can reduce memory consumption from 70GB to 19GB on very large and gappy datasets

-t Specify a user starting tree file name in Newick format

-v Display version information

-w FULL (!) path to the directory into which ExaML shall write its output files

DEFAULT: current directory

To execute a simple run using two processors we type:

```
mpirun.openmpi -np 2 ./examl-AVX -s ../parser/49.unpartitioned.binary
-t ../testData/49.tree -m GAMMA -n T1
```

The output files and checkpoint files are analogous to those of RAxML-Light.

For a tree inference on a the partitioned dataset we type:

```
mpirun.openmpi -np 2 ./examl-AVX -s ../parser/49.partitioned.binary
-t ../testData/49.tree -m GAMMA -n T2
```

For better performance do not forget to experiment with the `-Q` and `-S` options!