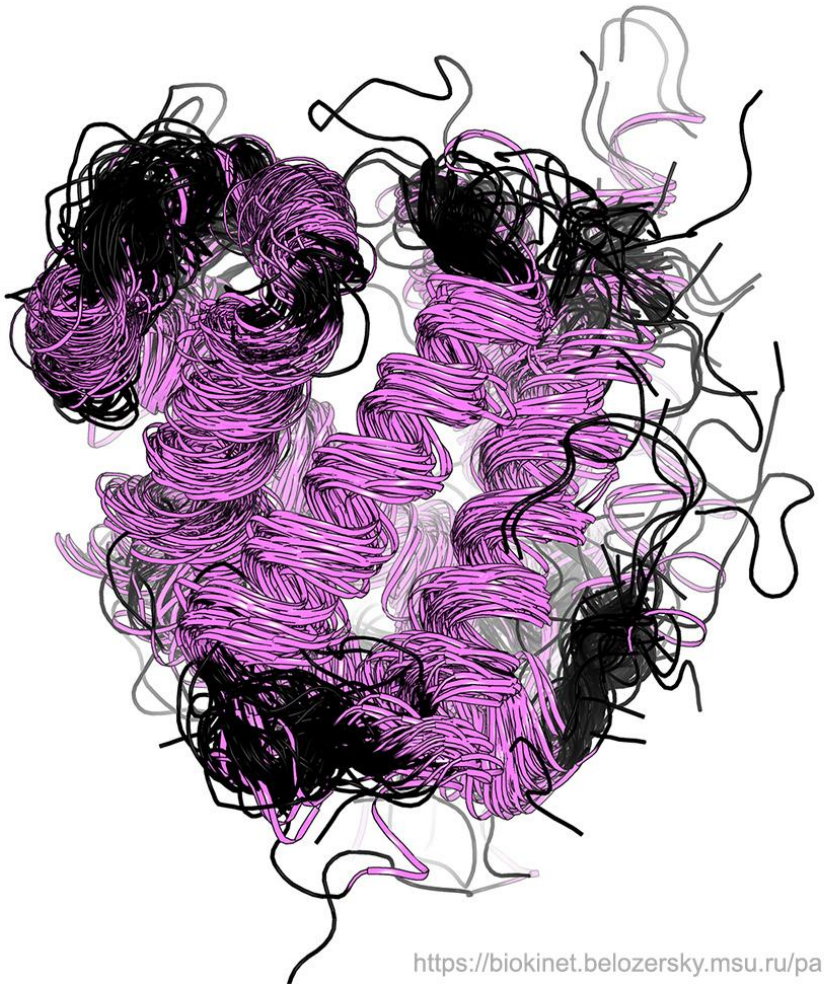
parMATT

Parallel multiple alignment with translations and twists for distributed-memory systems

The User Manual



June 28th, 2017

Contents

Abstract

Protein structure is more conserved throughout the evolution compared to sequence. Currently, there are more than 135000 structures in the Protein Data Bank (PDB) and growing. The availability of this information gives new opportunities for comparative bioinformatic analysis of remote evolutionary relatives which have lost sequence similarity during natural selection and specialization from a common ancestor but preserved a common structural core. The parMATT is a parallel implementation of a popular algorithm MATT (Multiple Alignment with Translations and Twists) and is intended for distributed-memory systems, i.e., computing clusters and supercomputers hosting memory-independent computing nodes. The parMATT can significantly accelerate the time-consuming process of building a structural alignment from a large collection of 3D-models of homologous proteins. The program is available at https://biokinet.belozersky.msu.ru/parmatt.

Introduction

The parMATT (https://biokinet.belozersky.msu.ru/parmatt) was developed at the Lomonosov Moscow State University based on the MATT (http://matt.cs.tufts.edu/) algorithm and source code. The parMATT was designed to accommodate computationally hard tasks, e.g. to align hundreds of homologous protein structures, by implementing the Multiple Parallel Interface (MPI). The parMATT inherits the bioinformatic part (the algorithm, the input and the output formats) from the MATT source code. With a few exceptions, all options and environmental variables available in MATT also work in parMATT. The advantage of the parMATT over MATT is the ability to run on multiple nodes (multiple CPUs) of multiprocessor computer systems. The parMATT can significantly accelerate the time-consuming process of building a structural alignment from a large collection of 3D-models of homologous proteins.

This User Manual focuses on the multiprocessor-related features as well as changes to the input options and variables which have been introduced in parMATT. Implementation of parMATT in the laboratory practice as well as guidelines for collecting large data sets of 3D-models of homologous proteins will be also discussed.

For a full list of options and features of the “Multiple alignment with translations and twists” algorithm please refer to the MATT’s user manual and the web-page.

Prerequisites

parMATT was developed for multiprocessor computer systems with distributed memory. It implements MPI for communication between processors (nodes) and pthreads to utilize multiple cores within each processor (node). Therefore, to achieve the best performance of parMATT you should use a computing cluster or a supercomputer hosting *memory-independent computing nodes*, i.e. powerful computers with multiple CPUs. If you want to run a multiple structural alignment on a single desktop CPU with multiple computing *cores* (i.e., *shared-memory computing system*) you could use parMATT or compile the original MATT sources with the openMP support (see installation instructions for the MATT program at http://matt.cs.tufts.edu/).

To compile the parMATT binary from the source code you need a Linux computer system with MPI environment and pthreads support. This means that you would normally need an MPI compiler (e.g. Intel MPI or openMPI) and a C compiler (e.g., Intel *icc* or GNU *gcc*). You can install the free GNU tools openMPI (https://www.open-mpi.org/) and GCC (https://gcc.gnu.org) – both packages should be available from a developers software repository on any Linux distribution (e.g., in openSuSE you can install both packages in YaST). Or go for the all-in-one Intel Parallel Composer XE/Intel Parallel Studio XE pack which is commercial, though non-commercial license terms are available for particular cases.

The current implementation of parMATT has been tested on, but is not limited to, the following software environment: a cluster with CentOS Linux 7.1.1503, OpenMPI 2.0.1 and Intel *icc* compiler version 15.0.3.

Compilation

After the **prerequisites** have been met enter the project folder and run:

```
make
```

If the compilation has been successful the executable binary file would be placed in the `'./bin'` folder.

parMATT's options and variables

The parMATT supports the same list of options (i.e., command line arguments) as the original MATT program.

Special characteristics of parMATT are:

- The '`-d`' option for display status is not supported;
- The '`-p`' option for partial alignment is disabled by default (i.e., `-p0`);
- The '`-t`' option specifies the number of pthreads ***t*** to be spawned *on each node* to process the subtasks (i.e., build the alignment). Additional pthreads will spawned for the administrative purposes - one pthread for communication on each slave node and two pthreads, one for communication and one for tasks' scattering, on the master node. E.g. if you run parMATT with '`-t` ***4***' parameter than 4+1 pthreads would be spawned on each slave node (4 for the alignments and 1 for communication) and 4+2 pthreads would be spawned on the master node (4 for the alignments, 1 for communication, and 1 for tasks' scattering). The default is '`-t` 1'.

Recommendations on selecting the '`-t` ***t***' number of pthreads: ***t*** should be set equal to the number of physical cores on your nodes. E.g., to run parMATT on multiple nodes, where each node hosts an Intel Core i7 CPU with ***4*** physical cores (8 logical threads), you should use '`-t` ***4***'. Or to run parMATT on multiple nodes, where each node hosts an Intel Xeon CPU E5-2697 v3 CPU (***14*** physical cores or 28 threads), you should use '`-t` ***14***'.

parMATT supports the same list of environment variables as the original MATT program (i.e., `MATT_SEARCH_PATH`, `MATT_PDB_PATH`, `MATT_PARAMS`). To set a variable use the `export` feature of the shell:

```
export VAR="VALUE"
```

Then you can check if the assignment worked:

```
echo $VAR
```

A new environment variable `MATT_LOG_LVL` is introduced to control the amount of data to be displayed in the standard output of parMATT. The variable is intended for developers and is not needed for the 'scientific' use. The variable does not influence the results or the format of the output alignments produced by the parMATT.

The possible values are:

- `MATT_LOG_LVL 0` - log only timings for pairwise alignments and iterative part of the algorithm;
- `MATT_LOG_LVL 1` - all above plus the information about tasks sent;
- `MATT_LOG_LVL 2` - all above plus timings for each alignment and barrier wait time (default).

To set the variable use the `export` feature of the shell:

```
export MATT_LOG_LVL="0"
```

Then you can check if the assignment worked:

```
echo $MATT_LOG_LVL
```

The parMATT input

There are two ways of providing the list of input PDB files to parMATT/MATT. The first one is via a *list* file – plain text file stating the paths to each PDB on a separate line:

```
sbatch -N 8 ompi parMatt -L input.list -t 14 -o output
```

The second way is to provide each PDB path as a separate command line argument:

```
sbatch -N 8 ompi parMatt file1.pdb file2.pdb ... -t 14 -o output
sbatch -N 8 ompi parMatt *.pdb -t 14 -o output
sbatch -N 8 ompi parMatt `find ./ -name *.pdb` -t 14 -o output
```

The second way may appear more convenient, but it is impractical. The parMATT was designed to accommodate computationally hard tasks, e.g. to align large data sets of protein structures. The paths to a large number of PDB files will most likely exceed the default maximum length of command line arguments allowed by the shell. Therefore, we recommend the *list*-file based input when using the parMATT. You can prepare the *list* file using the command:

```
find /path/to/pdbs -name "*.pdb" > list.file
```

Running parMATT

The difference between running the original MATT on a local computer and running parMATT on a computing cluster/supercomputer is that

(1) patMATT has to be launched as an MPI program by the appropriate MPI utility, and

(2) the '`-t` ***t***' parameter must be set manually for the particular hardware.

This may sound complicated, but it's not. You just need to add a few MPI-related "words" as a prefix to the regular command, and you need to learn the number of physical cores in the CPUs used in your computing cluster/supercomputer from its administrator or by exploring the vendor's site (the CPU model can be found in the `/proc/cpuinfo` file). Once you know the command and the number of physical cores in your CPU model, running parMATT will be as easy as running any other program on your local computer.

To run the parMATT you need to execute the '`bin/parMatt`' binary in the MPI environment and provide the '-t ***t***' parameter. The exact command will depend on the particular setup of your computing system. A few examples are provided below.

Launch parMATT locally on 4 physical cores of a single Desktop CPU:

```
/path/to/parMatt -t 4 -L input.list -o output
```

Please note that the advantage of the parMATT over MATT is the ability to run on multiple nodes (multiple CPUs). Thus, local execution on a single CPU should be considered for evaluation/testing purposes only.

Launch parMATT on 8 nodes (i.e., 8 CPUs), 14 physical cores on each node, using the *mpirun*:

```
mpirun -np 8 /path/to/parMatt -t 14 -L input.list -o output
```

Launch parMATT compiled with OpenMPI on 8 nodes (i.e., 8 CPUs), 14 physical cores on each node, using the Slurm Workload Manager (*sbatch* and *ompi* script):

```
sbatch -N 8 ompi /path/to/parMatt -t 14 -L input.list -o output
```

Launch parMATT compiled with Intel MPI on 8 nodes (i.e., 8 CPUs), 14 physical cores on each node, using the Slurm Workload Manager (*sbatch* and *impi* script):

```
sbatch -N 8 impi /path/to/parMatt -t 14 -L input.list -o output
```

Launch parMATT on 8 nodes (i.e., 8 CPUs), 14 physical cores on each node, using the Cleo Workload Manager:

```
cleo-submit -np 8 /path/to/parMatt -t 14 -L input.list -o output
```

If you do not know how to launch a program on multiple nodes (i.e., CPUs) of your computing cluster or supercomputer you should contact the administrator.

The parMATT output

The following files are produced by parMATT/MATT on successful completion:

- the **coordinate representation of a multiple structural alignment**, i.e., a PDB file with aligned coordinates of all 3D-models from the input;
- the **sequence representation of a multiple structural alignment**, i.e., a sequence alignment file in FASTA format;
- a **text file with a summary** of the input PDBs (the pairwise comparison tree) and the output superimposition (number of residues in the core alignment, RMSD of the core alignment, the MATT raw score and the sequence representation of the alignment in the PHYLIP format);
- a **Rasmol script** to highlight aligned residues.

The parMATT example

The input and output files for this example can be downloaded from the parMATT's page at https://biokinet.belozersky.msu.ru/parmatt

The input. The input set for this example is based on the **3.20.20.80** superfamily (Glycosidases) according to the CATH classification. The non-redundant set contains 275 protein structures in the PDB format with an average length of 348 amino acids. Download file **3.20.20.80_input.tar.gz** with the input set from the web page.

Execution. Extract the input data from the downloaded archive, enter the input folder, create the output folder, and run the parMATT program:

```
tar xzf 3.20.20.80_input.tar.gz
cd input
mkdir output
mpirun -np 8 /path/to/parMatt -t 14 -L 3.20.20.80.list -o output/3.20.20.80
```

Note, that the last command (i.e., the execution of the parMATT) will start parMATT on 8 nodes (i.e., 8 CPUs), assuming there are 14 physical cores on each node. The number of nodes and threads, as well as the general syntax of the command depends on the hardware and software setup on your particular computing cluster or supercomputer. The actual command may be different in your case. See the corresponding chapter of this tutorial for more information about running parMATT. To run this example on a single local Desktop CPU with 4 physical cores you can use the following command:

```
/path/to/parMatt -t 4 -L input.list -o output
```

Note, that the 3.20.20.80.list *list* file in the input folder should contain valid paths to the PDB files. To correct the paths you may use the `sed` editor which is available by default in most Linux distributions:

```
sed "s#./#${PWD}/#" 3.20.20.80.list -i
```

This command will substitute all '`./`' for the full path to the current folder (i.e., output of the pwd shell command) and the '`/`' path separator character.

Running times. The amount of time required to calculate the alignment of the provided set will, of course, depend on the power of computing system implemented for the task, however will take hours by the order of magnitude:

- **1** x Intel Core i7-4790 CPU 3.60GHz (4 physical cores) - 5.76 hours;

- **1** x Intel Xeon CPU E5-2697 v3 CPU 2.60GHz (14 physical cores) - 2.55 hours;
- **16** x Intel Xeon CPU E5-2697 v3 CPU 2.60GHz (14 physical cores) - 0.34 hours;

Output. Four files will be created on successful completion:

- The `3.20.20.80.pdb` file with the coordinate representation of a multiple structural alignment, i.e., a PDB file with aligned coordinates of all 3D-models from the input;
- The `3.20.20.80.fasta` file with the sequence representation of a multiple structural alignment, i.e., a sequence alignment file in FASTA format;
- The `3.20.20.80.txt` file with a summary of the input PDBs (the pairwise comparison tree) and the output superimposition (number of residues in the core alignment, RMSD of the core alignment, the MATT raw score and the sequence representation of the alignment in the PHYLIP format);
- The `3.20.20.80.spt` file with a Rasmol script to highlight aligned residues.

Download file **3.20.20.80_output.tar.gz** with the parMATT output for this example from the web page.

Collecting a set of 3D-models of homologous proteins

The increasing number of protein structures in public databases provides new opportunities for comparative bioinformatic analysis of remote evolutionary relatives which have lost sequence similarity during natural selection and specialization from a common ancestor but preserved a common structural core. The parMATT can significantly accelerate the time-consuming process of building a structural alignment from a large collection of 3D-models of homologous proteins.

In general, there are three ways of collecting a set of homologous proteins characterized by diverse functional properties within a common structural organization.

- **Analysis of the literature.** Some protein superfamilies are well studied (e.g., the alpha/beta hydrolase superfamily) and numerous scientific publications are available describing particular families and corresponding members with different functions. The respective three-dimensional models of these proteins can be manually collected from the PDB database. The advantage of collecting your PDB set by hand is that it will be based on experimental research and the corresponding functional annotation will be available for each included protein. The downside is that you can miss rare proteins whose properties have not been studied yet or were only poorly studied, as well as miss out recently added protein structures.
- **Structure similarity search.** Bioinformatic approach to collecting a set of remote homologs is based on detecting a significant structural similarity between the user-defined query protein and every protein structure available in the PDB. Selection of a query protein depends on the particular task and research objective. It can be the target protein selected for the further experimental design, or the most studied member of the superfamily, etc. The **PDBeFold server** (http://www.ebi.ac.uk/msd-srv/ssm/) is available on-line and provides easy and intuitive interface to search for structure similarities in the PDB database. The advantage of collecting your PDB set by bioinformatic analysis is that all information available up to the date in public databases will be taken into account, including poorly studied and recently added proteins. The downside is that this process usually requires some knowledge of bioinformatics to collect and postprocess the set (e.g., remove redundant entries).
- **Mustguseal server**. **Mustguseal** is a bioinformatic protocol designed to build alignments of protein families and superfamilies, and a platform (a web-server) to provide a user-friendly web-based interface to the Mustguseal protocol through the World Wide Web (https://biokinet.belozersky.msu.ru/mustguseal).

Automatic collection and filtering of a non-redundant set of remote homologs by structure similarity search in the PDB database is part of the Protocol. Mustguseal can be used to collect a large set of proteins with diverse functions within a common structural core. This set can be downloaded by the user and aligned using the parMATT.

Implementation of parMATT in the laboratory practice

Comparative bioinformatics is the cornerstone of computational approaches to understanding the sequence-structure-function relationship in proteins. Multiple sequence comparisons became a common tool of such analysis. Protein structure is more conserved throughout the evolution compared to sequence. It is therefore expected that three-dimensional alignment will provide more significant clues to protein function, properties and evolution than sequence alignment alone. Bioinformatic analysis of conserved and variable positions in large alignments of protein families/superfamilies can help with understanding of protein mechanisms, engineering enzymes with improved properties for practical application, and designing novel modulators of the activity of wild type proteins. The following three publications discuss these issues in more detail:

Suplatov, D., Kirilin, E., & Švedas, V. (2016). Bioinformatic Analysis of Protein Families to Select Function-Related Variable Positions. In Understanding Enzymes: Function, Design, Engineering, and Analysis (pp. 351-385) *Ed. Allan Svendsen.* Pan Stanford.

Suplatov, D., Voevodin, V., & Švedas, V. (2015). Robust enzyme design: Bioinformatic tools for improved protein stability. *Biotechnology journal*, 10(3), 344-355.

Suplatov, D., & Švedas, V. (2015). Study of functional and allosteric sites in protein superfamilies. *Acta Naturae*, 7(4), 27, 34-45.

The output of the parMATT is compatible with Modes 2 and 3 of the **Mustguseal server**. Currently, the server has a limitation of at most 32 protein structures to be aligned automatically when building the core structural alignment in Mode 1. The structural alignment of a representative set of homologous proteins is the core of the Multiple Structure-Guided Sequence Alignment built by the Mustguseal. It is important that proteins in this core structural alignment represent the desired diversity among the protein families of interest. The user may wish to include more structures in the core structural alignment and the parMATT can significantly accelerate this process on a local computing cluster/supercomputer. The corresponding user-built core structural alignment can be then submitted back to the Mustguseal server to build a large structure-guided sequence alignment of the corresponding superfamily.

The parMATT license

parMatt is licensed under the GNU public license version 2.0.

Citing parMATT

If you find parMATT or its results useful please cite our work:

Shegai M., et al. (2017) parMATT: Parallel multiple alignment with translations and twists for distributed-memory systems, *submitted*

The parMATT is based on the MATT algorithm and code:

Menke, M., Berger, B., & Cowen, L. (2008). Matt: local flexibility aids protein multiple structure alignment. *PLoS Comput Biol.*, 4(1), e10.