ScalaLife

Scalable Software Services for Life Science

Standards for data handling

Ramon Goni, Rossen Apostolov, Magnus Lundborg, Christoph Bernau, Ferdinand Jamitzky, Erwin Laure, Erik Lindhal, Pau Andrio, Yolanda Becerra, Modesto Orozco, Josep Lluís Gelpi

Efficient data management, quick and fast interaction with the computer and flexibility in access to computer resources are in many fields of life sciences at least as important as total theoretical peak power.

Source: European Exascale Software Initiative

1. Background

Computer simulation is today the so-called third methodology, next to theory and experimentation. Molecular simulation is widely used for many biological problems, for example the discovery of new drugs through the so-called structure-based drug design or the identification of protein-protein interactions [1]. There are numerous simulation techniques that can be applied, amongst which molecular dynamics (MD) is widely regarded as the most rigorous and powerful one [2]. Every year, tens of thousands of scientific publications cite the use of this technique, and the size of MD projects is likely to increase by the end of the decade (see Figure 1). Unfortunately, MD, like other simulation techniques, suffers from heavy computational time and production of large raw data. The amount of data necessary to represent a rigid biomolecule (~100Kb) can increase 5 orders of magnitude (~10Gb) when simulated. The MD community is demanding standards for data management and storage as it is currently struggling with large files, in application-specific formats, spread over a large number of fragmented research laboratories. Advances in simulation data management should provide a valuable help in the day-to-day research activity and support future grand challenges. ScalaLife (Scalable Software services for Life Sciences) is a European initiative that launched a cross-disciplinary Competence Center (http://www.scalalife.eu) for life sciences. ScalaLife performed code optimization, benchmarking on different architectures for simulation application, as well as designed new standards for efficient MD data storage and interoperability. We present in this paper a set of proposed standards and best practices developed during the ScalaLife project.

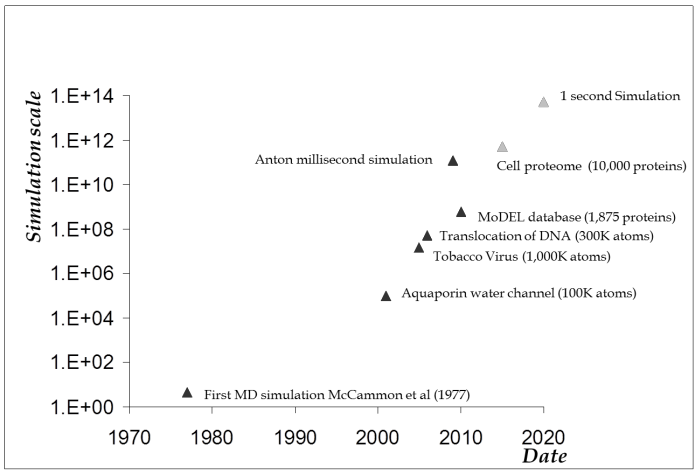

Figure 1. Expected exponential growth of biomolecular simulation. The Y-axis represents the simulation scale calculated multiplying simulation length (in nanoseconds) with size (in number of atoms). From PRACE Scientific Case 2012-2020.

Molecular Dynamics Data Overview

Data derived from Molecular Dynamics experiments may be structured in 4 blocks: i) simulation metadata, ii) pre-processing data, iii) trajectory data and iv) analysis/post-processing data (see Figure 2 for a general schema). The simulation metadata is a heterogeneous set of information that describes, e.g. the biological system and the parameters of the simulation experiment. Metadata can include the molecular descriptions (name, id, organism etc.), conditions of the simulation (solvent, temperature, etc.), the application and version used (GROMACS, Amber11, etc.), the parameters of the simulation (time step, force field, etc.) and links to external sources of information (gene information, protein domains, ligands, etc.). Pre-processing data may be considered part of the trajectory itself, and strongly depends on the simulation packages. Normally preprocessing data includes the original system, the system prepared (modifications on the structure, filling of gaps, solvent, ions, etc.) and an initial short simulation to equilibrate the system.

The trajectory information provides the necessary data on a series of recorded snapshots. The user defines the time step to record snapshots of the trajectories, which are normally some orders of magnitude longer than the MD integration

time step. For each snapshot the minimum information of the system is the atom coordinates. Atom information (atom number, type, etc.) will normally be located in external topology files that can be complemented with atom connectivity information. As explained below, some data formats support to store extra information on every time step, like velocities and forces. This information is not only important to track simulation flow, but mandatory if we wish to re-start a simulation from a specific time-point. Finally a simulation can be complemented with a set of post-processing quality controls and analysis results. Post-processing information can include all-atom or backbone root-mean square difference (RMSD), system flexibility (B-factor, PCA) or identification of functional sites (cavities, channels, etc.).

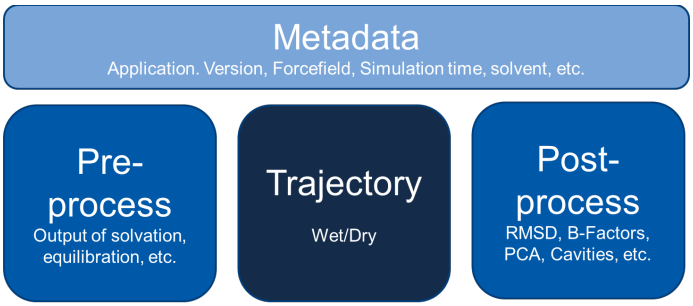

Figure 2: General schema of MD data

One of the main tasks of ScalaLife is to propose new open standards for efficient Molecular Dynamics data storage and transmission. The data standard presented in this paper is divided in two layers. The top layer, that we call the Unified Molecular Modeling (see chapter 3), is a human-readable and self-describing file format designed to store the simulation metadata. The second layer is embedded in the first layer. We call the standard of the second layer, the Trajectory File Format (see below) and it is designed to store the trajectory information with high compression.

2. Trajectory File Format (TNG)

Many of the widely used software packages for MD simulations adopt different formats for storing the trajectory information. Analysis of this information using third party post-processing tools requires implementation of support for those different formats. Moreover, some of them do not have the capabilities to efficiently store the data.

NAMD (http://www.ks.uiuc.edu/Research/namd) and AMBER (http://ambermd.org), along with GROMACS (http://www.gromacs.org) [11], are amongst the three most popular MD applications. NAMD stores trajectories in the DCD format, which represents binary FORTRAN files with atom coordinates (optionally velocities and forces). The format does not have compression and does not support storage of additional data types. The files are not transportable between big-endian and little-endian computer architectures, and require additional tools for conversion. As the developers point out "the exact format of these files is very ugly but supported by a wide range of analysis and display programs."

(http://www.ks.uiuc.edu/Research/namd/2.6/ug/node13.html)

AMBER uses a file format that is based on NetCDF (Network Common Data Form) developed by Unidata (http://www.unidata.ucar.edu/software/netcdf). The format defines a set of conventions to be applied to NetCDF libraries, which are designed for representation of arbitrary array-based data and have bindings for many languages such as C, C++, Fortran (F77 and F90), Java, Python etc. with read/write support. Although the AMBER format offers more advantages over DCD in terms of portability, extensibility and compatibility, the NetCDF library is rather large (~5MB compressed distribution, compared to e.g. ~10MB GROMACS, ~8MB NAMD, ~4.5 AMBER) and bundling it with software packages will increase substantially their footprint. The new library that we have developed has a size of ~150KB even including examples files. Moreover, although NetCDF has support for a wide range of architectures, potential porting issues cannot be reliably offloaded to the upstream developers. GROMACS uses two kinds of formats for storage of output data. TRR is a full-precision portable-binary data format, while XTC is a reduced-precision format. The latter is available as part of the xdrfile library (ftp://ftp.gromacs.org/pub/contrib/xdrfile-1.1.1.tar.gz). The XTC format offers very good compression of the data and is portable. However, it has many drawbacks such as no possibility to store arbitrary user- or meta-data, no indexing for fast searches, no parallel I/O etc.

To overcome the problems present in the current approaches discussed, we developed a new container type format (TNG) that fulfills the following requirements:

- Architecture independent

- Different data types
- Temporal compression of data similar to multimedia formats [3]
- Digital signatures
- Extended meta-data with full information about simulated systems and conditions
- Random access
- Parallel I/O
- Plugin-like architecture for different compression algorithms

To fulfill the above requirements, TNG has been developed along the following specifications as released in version 1 of the format.

General specifications

General specifications are given below; others are described in the relevant block sections.

- The file contains a number of blocks.
- The order of the blocks follows the order specified in the section "Description of blocks".
- All integer and floating-point (floats and doubles) values can be stored either as big or little endians. Conversions to and from the native format of the computer are performed during reading and writing of numerical fields. Floating-point values are stored in the IEEE 754 format.
- Integer values are stored as 64-bit values, whereas floating-point values can be stored as either floats (32 bits) or doubles (64 bits).
- MD5 hashes are used to verify the integrity of the data.
- Strings are limited to a max length of 1023 bytes and are terminated by a null character ('\000'). If longer text data must be saved a data block containing multiple entries of general (character/string) data can be used.
- If a trajectory converter program encounters errors during reading a block which format is not recognized, the block can be written out as binary object without modification.
- No compression (in version 1) for general data streams such as integers, floating-point numbers, particle indices etc.

Blocks

Each block contains the following fields as a header:

- Size of the header (integer)
- Size of the block contents (except header) (integer)
- Block type identifier (integer)
- MD5 Hash (16 characters, if they are all "\0" characters the MD5 checks are disabled)
- Name (string)
- Version of the block (allows addition of more fields in the future to existing blocks, although old fields should never be removed, to allow older readers read new files) (integer)

Description of blocks

Each has a unique integer identifier and a matching "name".

- Info block (1) "GENERAL INFO" (required)
- Molecules block (2) "MOLECULES" (optional)
- Trajectory frame set block (3) "TRAJECTORY FRAME SET" (required) (multiple "trajectory frame sets" are allowed)
    - Trajectory particle mapping block (4) "PARTICLE MAPPING" (required if there are trajectory frames blocks) (multiple particle mapping blocks with corresponding trajectory frames blocks are allowed, e.g. to allow parallel writes of different atom sets)
    - Data blocks:
        - Box shape block (10000) "BOX SHAPE" (optional)
        - Positions, block (10001) "POSITIONS" (optional)
        - Velocities, block (10002) "VELOCITIES" (optional)
        - Forces, block (10003) "FORCES" (optional)
        - Partial charges block (10004) "PARTIAL CHARGES" (optional)
        - Other custom blocks, both non-trajectory and trajectory blocks, each with unique id & name (See Supplementary file 1 for detailed specifications of the block contents).

    If the data of the data blocks change from one frame to another they are located inside

frame sets, otherwise they are placed before the frame sets (after particle mapping blocks if there is a relevant particle mapping).

Data blocks can be used to store whatever data is needed. Data blocks with IDs in the range 10000 to 10999 are reserved for standard data (such as box shape, positions, velocities etc.), whereas IDs from 11000 and above can be used for any kind of user data.

The format is implemented and released as a standalone library with APIs in C, C++ and FORTRAN. The library provides an API that is divided into a low-level and a high-level set of functions that can be referred to as a low-level and a high-level API. The low-level API exposes all of the available functionality of the format and gives developers fine-grained control, whereas the high-level functions aim for simplicity and hide the complexity behind sensible default settings, which makes them more suitable for novice users. The high-level functions provide the user with the most important functions, but can be complemented with the more advanced low-level functions (see Supplementary File 2).

The TNG format will replace, in GROMACS, the currently used XTC and TRR formats. Our efforts are towards its adoption by the wider community. As part of the library we provide converters to/from other commonly used file types such as NetCDF.

The TNG library is available at Scalalife's Competence Center: http://www.scalalife.eu/content/binary-trajectory-file-format. Converter tools from/to TNG are also available to download from the Competence Center (see manual in Supplementary File 3)

3. Data Transmission

Unified Molecular Modeling (UMM)

Although the general aim is to minimize the transmission of data, keeping raw trajectories in their original sources and performing also analysis in their location, at some point data should be transmitted. To achieve both compact storage and universally useful formats, we need a very flexible and hierarchical standard format that can store metadata using consensus ontology and able to wrap the trajectory information encoded in an efficient binary format. To make

Text BOX 1: Example of simulation encoding using UMMv1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<simulation>
   <boxX>0.0</boxX>
   <boxY>0.0</boxY>
   <boxZ>0.0</boxZ>
   <clIons>0</clIons>
   <cutoff>0.0</cutoff>
   <description>PHOSPHOLIPASE A2 (PLA2) </description>
   <ewald>0</ewald>
   <firstIonAtom>0</firstIonAtom>
   <firstIonResidue>0</firstIonResidue>
   <firstLigandAtom>0</firstLigandAtom>
   <firstLigandResidue>0</firstLigandResidue>
   <firstSolventAtom>0</firstSolventAtom>
   <firstSolventResidue>0</firstSolventResidue>
   <forceField>GROMOS 53</forceField>
   <integrationTime>0.0</integrationTime>
   <ligandAtoms>0</ligandAtoms>
   <ligandResidues>0</ligandResidues>
   <naIons>0</naIons>
   <pbc>0</pbc>
   <pdbId>1A3D</pdbId>
   <program>GROMACS</program>
   <programVersion>3.3</programVersion>
   <saltConcentration>0.0</saltConcentration>
   <samplingTime>2</samplingTime>
   <shake>0</shake>
   <simulationTime>10000</simulationTime>
   <soluteAtoms>0</soluteAtoms>
   <soluteCharge>0.0</soluteCharge>
   <soluteMolecules>0</soluteMolecules>
   <soluteResidues>0</soluteResidues>
   <solvent>T4P</solvent>
   <solventAtoms>0</solventAtoms>
   <solventResidues>0</solventResidues>
   <source>RSCB PDB</source>
   <statusComment>Simulation finished 10ns</statusComment>
   <temperature>0.0</temperature>
   <totalAtoms>0</totalAtoms>
   <totalCharge>0.0</totalCharge>
   <totalIons>0</totalIons>
   <totalMolecules>0</totalMolecules>
   <totalResidues>0</totalResidues>
   <trajType>WET</trajType>
   <volume>0.0</volume>
   <waters>0</waters>
</simulation>
```

the format more general, we have decided to separate it into three largely orthogonal parts (but these are frequently expressed together in a single XML file) that handle i) the description of the fundamental interaction force field parameters, ii) the description of the molecules in the system, and iii) the job metadata such as the number of steps, algorithms used interaction settings such as cut-off or shapes, and not least information about how this particular file has been generated by other tools.

The UMM (Unified Molecular Modeling) file format specifies descriptions of molecular systems and associated simulation procedures. The format uses a simplified XML-like syntax that aims to be human readable. The file is processed by a converter script in order to generate a standards-compliant XML file that will be used as actual input. The latter is more "computer-readable" and can be easily processed by standard XML libraries. The standard aims to be flexible and adaptable to various configuration scenarios. UMM has been designed to make it easy to include support by other software packages and thus facilitate adoption. The format will allow inclusion of the history of the trajectory (job submission). The simulation setup and workflows should be easily shared with other users.

The specification of UMMv1 XML schema is implemented in an XSD file (available at ScalaLife Competence Center). XSD provides the syntax and defines a way in which elements and attributes can be represented in an XML document. It also advocates that the given XML document should be of a specific format and specific data type. The first version of UMM-XSD file sets common definition of high-level simulation information (see text BOX1 and Supplementary File 4), allowing future expansions to application-specific fields, providing end-users with the freedom for reaching low levels of granularity.

Lossy compression format (PCAZIP)

Although for some specific tasks full raw trajectories including solvent are needed most analysis would not require this level of detail. In particular, analysis related to structure flexibility can be performed with a reduced set of data if selected carefully. The use of principal component analysis allows recovering in the major components the essential movements of the structure (big movement) discarding high frequency movements that do not influence the overall dynamic behavior of the structure (vibrations). It is important to remark that using this approach it will not be possible to recover the original information but could help to reduce the size of the simulations. Using 90% accuracy, the size of dry trajectories can be reduced to 5-10% while keeping the major determinants of structure flexibility. The application pcasuite implements this principle and is available at the ScalaLife Competence Center (http://www.scalalife.eu/content/pcasuite)

4. MD Data Storage

The impact of technological advances and high-performance computing solutions in Life Sciences is evident. The international collaborative efforts in structural genomics and proteomics have resulted in an exponential growth in the amount of biological data leading mostly to an increasing amount of information based on genomics, structural and functional annotations. Many of these data are organized in databases and public repositories that have fostered research in biomedical sciences. The European Bioinformatics Institute (EMBL-EBI) hosts many of the publicly available data repositories of genomic, proteomic and protein structure data. These include the European Nucleotide Archive (ENA), UniProt and PDB. Some of these repositories, such as UniProt and PDB, are maintained in collaboration with the international partners. The data in these repositories are manually curated for high quality annotations and are also supplemented by automated annotation efforts. Over the last decade, efforts to integrate data from various archives have led to greater understanding of biological systems. This has also resulted in advances in computational techniques for handling various types of biological data.

However, in recent years we have experienced an explosion of molecular simulation data, which are not archived in a systematic and well-documented manner. This is greatly hampering our capability to analyze and process such data. Even with current computational capabilities, the lack of common repositories and the different formats are complicating the efficient use of existing data. There are some examples of incipient molecular simulations repositories. Two initial attempts, BioSimGrid [4] and P-Found [5] provided software infrastructure to build biosimulation databases. They both shared the philosophy of having a central repository of trajectories that would allow obtaining a comprehensive view of biomolecular structure. At the time when these projects were started, computer power was still limited, and hence, both projects were conceived as open databases where their users would provide

trajectories. Unfortunately, this approach was less effective than expected. Two additional projects reported a major release in 2010: Dynameomics [6] and MoDEL [7]. In these projects, besides of the software infrastructure, a significant amount of trajectories for proteins was reported. No equivalent initiatives in the nucleic acid simulation have been reported, although the ABC-Consortium created a simulation database oriented to characterize DNA flexibility [8]. The database provides 50-100 ns trajectories of 39 different DNA oligomers, which cover all possible nearest neighbor sequence environments for the 10 possible DNA base steps.

Database infrastructure

There is a major necessity for a data storage system (archiving and management) that can provide the research community with an efficient environment to deposit, share and access data, particularly adapted to the characteristics of biological simulations data. There are different systems that can be adapted to MD data, among them relational databases (MySQL or Microsoft SQL server), distributed file systems (Hodoop System and Amazon) and other non-relational databases (Apache Cassandra). Ideally the database infrastructure should be an efficient environment to deposit, share and access data (all trajectory or part of it), particularly adapted to the characteristics of biological simulations data and considering a dynamic and distributed multi-datacenter environments. Unfortunately, there is no single existing data system fulfilling these requirements. Although non-relational models will be an interesting choice in the near future, the authors believe that nowadays, relational databases are the right choice for all those applications with high requirements of consistency that need transactional behavior (ACID properties) and have a clear data scheme [9].

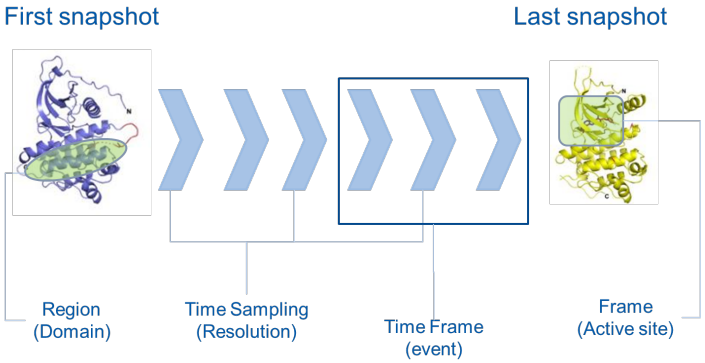

Figure 3: flexible ways to access trajectory data.

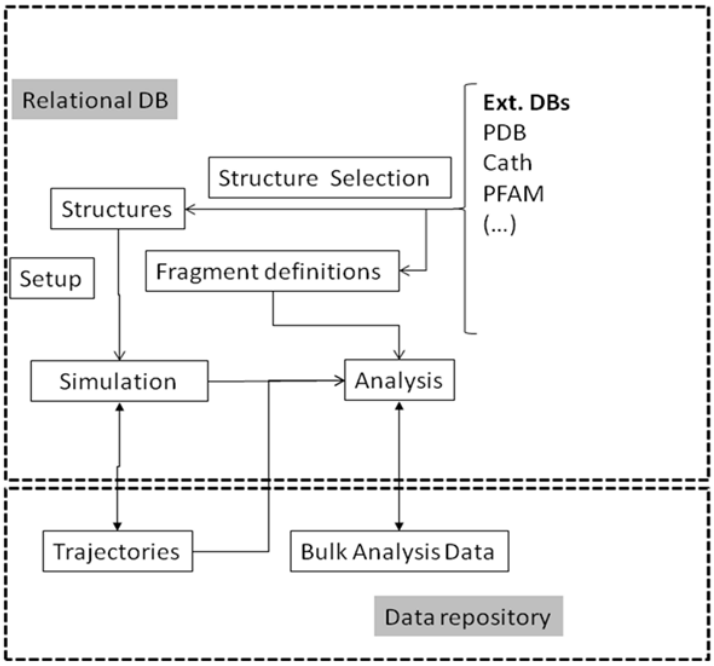

Figure 4. MoDEL schema of the relational database with trajectories stored in indexed files

Storing trajectories in a relational database

When designing the relational database for MD data, the first decision should be if: i) trajectories should be stored in the database or ii) store only metadatata and keep references to trajectory files. BioSimGrid and Dynameomics used the first strategy and every atom of every time frame is stored in the database. This approach has very interesting advantages and may gradually adopted in the future as it provides a very flexible way to access information (like fragments of the simulation or different time steps, see Figure 3). Unfortunately nowadays this advantage could not be fully exploited, as there is a lack of analysis tools compatible with such an approach. Another weak point of this strategy is the scalability issues for large molecular systems or large time scales. For these reasons ScalaLife defined the data storage standards on the second strategy (trajectory files accessible through the database), which is also adopted by the iBIOMES system [10] and MoDEL database (see Figure 4).

Table 1. List of data elements. First column is the main area, second column is the name of the data-element and third column is the list of parameters that define the element.

SIMULATION DETAILS	Simulation	simulationId, idCode, version, programVersion, programType, forceFieldId, solventId, date, status, statusComment, simulationTime, saltConcentration, boxType, boxX, boxY, boxZ, volume, waters, totalAtoms, totalResidues, totalMolecules, temperature, integrationTime, samplingTime, totalCharge, soluteAtoms, soluteResidues, soluteMolecules, soluteCharge, shake, cutoff, PBC, ewald, ligandAtoms, firstLigandAtom, ligandResidues, firstLigandResidue, totalIons, nalons, cIons, firstIonAtom, firstIonResidue, solventAtoms, firstSolventAtom, solventResidues, firstSolventResidue, moviePath, movieSize
	Program	programType, name, description
	Solvent	solventId, name, description
	ForceField	forceFieldId, name, description
	SetupOption	simulationId, idMon, number, type
	Monomer	idMon, sourceData, name, numberAtoms, synonym, formula, molecularWeight, charge
	SimulationResidueMapping	simulationId, resName, resSeq, pdbElementId
REFERENCE STRUCTURES	Entries	idCode, title, ascDate, compound, source, authors, resolution, expType, type, hetnam, expTypeClase, keywords, taxId
	PDBElement	pdbElementId, idCode, model
	Residue	pdbElementId, resName, chainId, resSeq, iCode, resNum, chainSeq, idCode,
	Atom	pdbElementId, residueId, serial, name, altLoc, element, xcoor, ycoor, zcoor
	entriesResidueMapping	idCode, resName, resSeq, pdbElementId
ANALYSIS PERFORMED	AnalysisSet_Simulation	analysisSetId, simulationId
	AnalysisSet_entries	analysisSetId, idCode
	TimeSlice	timeSliceId, initialSnapshot, lastSnapshot
	AnalysisSet_TimeSlice	analysisSetId, timeSliceId
	AnalysisSet	analysisSetId, fragmentSetId, fileSystemPath, status, GQ, LQ
	AnalysisList	analysisListId, name
	AnalysisSet_AnalysisList	analysisSetId, analysisListId
	FragmentSet	fragmentSetId, fragmentId, description
	Xref_FragmentSet	XrefDB_idDB, fragmentSetId
	XrefDB	idDB, name, description
	Fragment	fragmentId, initialResidue, lastResidue, description, nres, sequence
	Xref_Fragment	XrefDB_idDB, fragmentId
	GeneOntologyFragment	geneOntologyId, fragmentId, evidence, description
	Analysis	analysisId, parentAnalysis, name, type, title, description
	Result	resultId, analysisSetId, analysisId, floatValue, stringValue, status
	1DPosition	matrixId, x, floatValue, stringValue
	2DPosition	matrixId, x, y, floatValue, stringValue
	Matrix	matrixId, resultId
	RawData	rawDataId, pdbElementId, resultId, floatValue, stringValue, instant

The UMM-MoDEL Database Schema

We analyzed the minimum data-elements per area and we studied how they need to relate between them. For example one requirement is that one amino acid of the protein structure has one or more atoms, but one atom belongs to only one amino acid, if any. Based on the UMM standard and our previous experience with the MoDEL database we designed the UMM-MoDEL general schema with the minimum information per data-element (parameters). The proposed standard to store MD data in a relational database follows an organization in three blocks: Simulation details, Reference structure and Analyses performed (see Table 1).

Table 2. Descriptors used to check for potential problems in protein MD simulations and recommended values: absolute and relative RMSd, the TM-scorermsd (Zhang & Skolnick 2004) the radii of gyration and solvent accessible surface (SAS), number of native contacts, and the secondary structure. Trajectories failing to fulfil two (or more) global and/or one (or more) local criteria are labeled with warnings in the database and, although they are accessible, interpretation should be made with care. ($) Values referred to the total number of residues in the protein. (%)Values compared with the experimental model. (&) To reduce thermal noise, a comparison is made here between the first and last ns of trajectory.

Global descriptor	Threshold	Local descriptor	Threshold
RMSd	< 8 Å	lost native contacts[%]	< 30%
relative RMSd[$]	< 0.2 Å/res	sec struct. changes[&]	< 10%
Tmscore	< 2.5 Å		
relative Tmscore[$]	< 0.06 Å/res		
relative Rgyr[$]	< 0.4 Å/res		
relative SAS[$]	< 100 Å^2/res		

The UMM-MoDEL software stack is available for download at Scalalife's Competence Center http://www.scalalife.eu/content/umm_model

Data Curation

MD is based on a simplified physical model that can generate wrong simulations. Errors in simulations are mainly related to: i) incorrect decisions during the set-up, particularly wrong ionic states, poorly placed solvent or wrong description of the ligand; ii) errors in the equilibration and heating procedure; iii) technical problems along equilibrated trajectory (problems with SHAKE, extreme velocities, thermal coupling, etc.) and iv) force-field problems.

Other reasons for wrong simulations can be due to local uncertainties in the experimental models and variations of environmental conditions (for example: different pH, different ionic strength or protein concentration). It is highly recommended to inspect trajectories before their storage and future analysis. This will allow us to recognize errors derived from technical factors (set-up, equilibration, heating, integration, coupling). Expert-supervised review is always the best approach to curate MD data, but according to our experience there is a set of descriptors that can be automatically checked in order to determine if a protein simulation is correct. Table 2 summarizes the list of descriptors and the recommended values according to our experience [7].

5. Outreach

ScalaLife is keeping open discussion channels with MD application developers for the adoption of the proposed standards. VMD/NAMD and AMBER developers showed an interest to implement support for TNG files in future releases. Meanwhile, we implemented an API for TNG input and output, used for converting TNG files to other formats (see Supplementary file 3). We are also working in set of application interfaces compatible with UMM/TNG standard: MDWeb (a web interface to prepare, launch and analyse MD), Copernicus (a software interface, which can be used for preparing and running MD simulations distributed over computers in a network) and Nafflex (a web interface to launch MD simulations of DNA molecules) among other on-going projects. During the development of the UMM and TNG formats we were contacted by the developers of Mosaic (MOlecular SimulAtion Interchange Conventions, https://bitbucket.org/molsim/mosaic/wiki/Home). The goals of the Mosaic project align very well with our work and we are working towards future collaborative work.

6. Conclusions

We present in this paper ScalaLife's proposed data standards and best practices for Molecular Dynamics data management. Our proposed data standard is divided in two layers: the XML UMM format and the embedded TNG file. TNG is a high-compressed and robust file format that allows non-sequential access to trajectory information. TNG will be adopted by GROMACS from the next major release. UMM is implemented in XSD and sets the common definition of high-level simulation information. We also implemented UMM-MoDEL, a database software bundle to manage MD simulation data using UMM/TNG standards. ScalaLife is working for the adoption of the standards developing open I/O APIs, converters and interfaces for Life Sciences community.

7. Supplementary Information

- **Supplementary file 1** (PDF): TNG specifications of the block contents: http://www.scalalife.eu/system/files/SupplementaryFile1.pdf
- **Supplementary file 2** (PDF): TNG low-level functions: http://www.scalalife.eu/system/files/SupplementaryFile2.pdf
- **Supplementary file 3** (PDF): TNG converter manual: http://www.scalalife.eu/system/files/SupplementaryFile3.pdf
- **Supplementary file 4** (PDF): UMM user guide: http://www.scalalife.eu/system/files/SupplementaryFile4.pdf

8. References

[1] Teague S.J. Implications of protein flexibility for drug discovery, Nat. Rev. Drug Disc. 2003, 2, 527.

[2] Karplus M., Kuriyan J. Molecular dynamics and protein function. Proc. Natl. Acad. Sci. USA 2005, 102, 6679.

[3] Spångberg D., Larsson D.S., van der Spoel D. Trajectory NG: portable, compressed, general molecular dynamics trajectories. J. Mol. Model. 2011, 17, 2669-2685.

[4] Tai K., Murdock S., Wu B., Ng M.H., Johnston S., Fangohr H., Cox S.J., Jeffreys P., Essex J.W., Sansom M.S. BioSimGrid: towards a worldwide repository for biomolecular simulations. Org. Biomol. Chem. 2004, 2, 3219-3221

[5] Silva C.G., Ostropytskyy V., Loureiro-Ferreira N., Berrar D., Dubitzky W., Brito R.M.M. P-found: The protein folding and unfolding simulation repository, IEEE, 2006.

[6] van der Kamp M.W., Schaeffer R.D., Jonsson A.L., Scouras A.D., Simms A.M., Toofanny R.D., Benson N.C., Anderson P.C., Merkley E.D., Rysavy S., Bromley D., Beck D.A., Daggett V. Dynameomics: a comprehensive database of protein dynamics. Structure 2010, 18, 423-435.

[7] Meyer T., D'Abramo M., Hospital A., Rueda M., Ferrer-Costa C., Pérez A., Carrillo O., Camps J., Fenollosa C., Repchevsky D., Gelpí J.L., Orozco M. MoDEL (Molecular Dynamics Extended Library): a database of atomistic molecular dynamics trajectories. Structure. 2010, 18, 1399-1409

[8] Dixit S.B., Beveridge D.L., Case D.A., Cheatham III T.E., Giudice E., Lankas F., Lavery R., Maddocks J.H., Osman R., Sklenar H., Thayer K.M., Varnai P. Molecular dynamics simulations of the 136 unique tetranucleotide sequences of DNA oligonucleotides. II: sequence context effects on the dynamical structures of the 10 unique dinucleotide steps. Biophys. J. 2005, 89, 3721-3740.

[9] Codd E.F., Codd S.B., Salley C.T. E. F. Codd and Associates. Providing OLAP On-Line Analytical Processing to User-Analysts: An IT Mandate. 1993.

[10] Thibault J.C., Facelli J.C., Cheatham T.E. III. iBIOMES: Managing and Sharing Biomolecular in a Distributed Environment. J. Chem. Inf. Model. 2013, 53, 726–736.

[11] Pronk, S., Páll, S., Schulz, R., Larsson, P., Bjelkmar, P., Apostolov, R., Shirts, M. R., et al. (2013). "GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit." Bioinformatics (Oxford, England), 29(7), 845–854. doi:10.1093/bioinformatics/btt055

To receive ScalaLife updates register to the mailing list through http://www.scalalife.eu/contact or send an email to scalalife@pdc.kth.se .

More information you can find at: www.scalalife.eu