Technical Report

Modular Implementation of Property Molecular Dynamics Procedures into *Pfound* Data Repository

pfound_11dez06.tgz

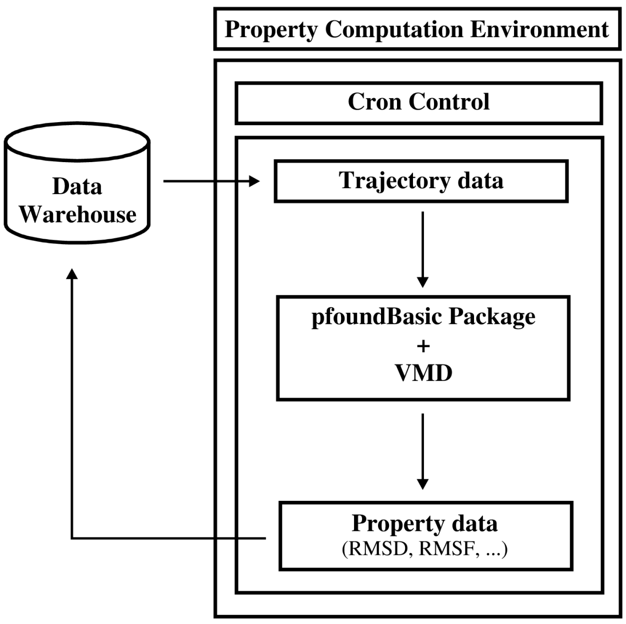
Aim

After uploading a trajectory to *P-found*, the data warehouse (DW) will store the relevant data in a specific location on behemoth. A cron job will be running every X minutes, in order to check if there are any trajectory files to be processed. Visual Molecular Dynamics program (VMD) will start to process the trajectories and calculate some properties for each available data set. A log file will be written with an successful/abort condition, which will inform the DW that data is available or not to be imported.

Data Repository Name

The DW will be known from now on, as *P-found* (**P**rotein **Fo**lding and **Un**folding Simulation **D**ata Repository).

The Link Scheme


Property Computation Environment
Cron Control
Data Warehouse
Trajectory data
pfoundBasic Package + VMD
Property data (RMSD, RMSF, ...)

Legend: Steps in the computation of molecular properties following uploading of simulation raw data (Silva *et al.,* 2006, we hope ;-)

Silva, C.G., Ostropytskyy, V., Loureiro-Ferreira, N., Berrar, D., Dubitzky, W., and Brito, R.M.M. ' P-found: a protein folding and unfolding simulation data repository' 2006. (Proceedings IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology – CIBCB 2006)

The System Architecture

Currently, the hardware supporting the DW is a 64-bit Intel Xeon dual processor server, with 4GB memory, 500GB hard drive, running Suse, and hosting Oracle Database 10g.

For the calculation of molecular properties, the software VMD (Humphrey *et al.,* 1996) runs on a 64-way Itanium Altix 3700 high-performace computer (behemoth) with 128GB of addressable memory and 8 TB of storage space (Silva *et al.,* 2006).

Humphrey, W., Dalke, A. and Schulten, K., `VMD – Visual Molecular Dynamics', J. Molec. Graphics 1996, 14.1, 33-38.

About P-*found*

As soon as a trajectory is uploaded into P-*found*, the DW will give a unique simulation ID to that experiment. Presently, two files are being requested to the user submiting a trajectory to the DW, namely a topology or structure file and a trajectory file. The relevant files will be then copied to a specific dir on behemoth. Supposing that the working directory is /data/pfound (== $pfound), files will be stored in $pfound/to-process. The basename of these files will be the corresponding simulation ID while maintaining the original extension, eg. 1013.dcd and 1013.psf, for a NAMD trajectory where 2 files were uploaded.

This is the working $pfound directory structure:

```
[guests@host-1-57 pfound]$ ls
history/  lib/  logs/  processed/  scripts/  to-process/

[guests@host-1-57 pfound]$ ls to-process/
1013.dcd  1013.psf
```

About Crontab

The cron daemon (Vixie cron) is used to execute scheduled commands. Job pfound.cron will run for example hourly. This job is a VMD script (written in Tcl + VMD builtin commands), that is going to check which files under $pfound/to-process are ready to be processed. If there is at least 1 trajectory to be processed, say simID 1013, that trajectory is moved to a newly created simID dir $pfound/processed/1013 and VMD will start then to calculate relevant properties, saving all output files inside simID dir.

This is the location of pfound.cron:

```
[guests@host-1-57 pfound]$ ls scripts/
pfound.cron*  start.vmd  top2psf.pl*
```

Example of the crontable content:

```
# checks files to be processed every hour; run vmd
0 * * * * vmd –dispdev text –eofexit < pathTo_pfound.cron > some_destination
```

About pfound.cron

This script will be run using the VMD Tcl interpreter (no need to have a tcl interpreter installed on the machine!!), and requires 3 external packages presently: La 1.0 and pfoundBasic 1.0, pfoundMultimedia 1.0

la1.0 provides a Linear Algebra and similar math package (Hume Integration Software).

PfoundBasic1.0 provides procedures to calculate some properties on the trajectories (homemade).
PfoundMultimedia provides all the required procedures to produce multimedia files.

The lib dir will be used to store any needed Tcl packages:

```
[guests@host-1-57 pfound]$ ls lib/
la1.0/  pfoundBasic1.0/  pfoundMultimedia1.0/
```

The cron job will start to check if all relevant directories/scripts are present on $pfound, in order to proceed with the analysis of the available simulations. It checks if files inside $pfound/to-process are being modified, prior moving them to $pfound/processed/simID and work on them. Once they are in $pfound/processed dir, all properties descriminated in the script start.vmd, will be calculated.

If there's an error during VMD execution, that error is caught in order to allow for pfound.cron to process all available simID's, and inform the DW which data is readily available for import. The simID under $pfound/processed that generated errors will be moved to $pfound/not-processed. After the debugging, relevant files (trajectory and topology) will be again moved to $pfound/to-process [still not implemented]

About start.vmd

Its a plain Tcl script implementing a procedure named calculateProperties, which calls a series of ::pfoundBasic:: procedures responsible for the analysis of every simulation uploaded to P-*found*. For example, the following command will calculate the gyration radius of simulation $dataBasename, using only protein backbone atoms. The resul of this calculation will be stored in file ${dataBasename}.rgyr :

```
set outExt rgyr
::pfoundBasic::calcRgyr "backbone" ${dataBasename}.$outExt
```

The script could be easily modified in order to add/modify/delete property calculations, accordingly to P-*found* objectives. If one wants to calculate the gyration radius of the protein using only the C_α atoms, then the command will be issued like:

```
::pfoundBasic::calcRgyr "name CA" ${dataBasename}.$outExt
```

The script start.vmd will be sourced from inside pfound.cron, only if there are trajectories available to be processed, using the following command line:

```
source $pfound/scripts/start.vmd
```

The procedure calculateProperties will be called like this:

```
set vmdCode [calculateProperties ${sim}.$topType ${sim}.$trajType]
```

where:

$sim	== simID basename being processed (eg. 1013)
$topType	== topology/coordinate file extension (eg. top, gro, psf ...)
$trajType	== trajectory file extension (eg. dcd, xtc ...)

The variable vmdCode will be == 0, if there's no catched error during the calculation of properties. In this case, a successful condition will be stored on a log file, otherwise, an abort condition will be issued.

About VMD

Check document LoadingScheme_CS_NF_060201.pdf, where is described the main reasons to choose VMD for processing trajectory data.

The cronjob pfound.cron will be run inside VMD, and will start to process data in a specific simID dir using the procedure calculateProperties. That simID dir already contains the relevant trajectory files needed by VMD to correctly load the data and work on it. VMD will first load a topology/coordinate file, in order to establish the atomic connectivities and atomic nomenclature (residue names, atom names, chain ID's, segment ID's). Then a trajectory (set of coordinates vs time), will be read using built-in VMD plugins. If VMD loads a coordinate file (with one set of coordinates; **carefull with this, it could contain more than one set of coordinates**) prior loading the trajectory, frame 0 will be deleted. This way, first frame (also called the reference frame) will come always from the trajectory.

Example of a simID dir structure, before the calculation of the properties:

```
ls  /data/pfound/processed/1013
1013.dcd        1013.psf
```

All the properties calculated by VMD will have as basename the simID, and will be created inside simID dir.

SimID dir content after having calculated the properties:

```
ls  /data/pfound/processed/1013
1013.dcd        1013.psf        1013.rgyr       1013.rgyr.png           1013.rmsd
        1013.rmsd.png           …
```

About log files

There are 2 directories where log files are going to be written.
Inside $pfound/history, 3 files are created, namely:

```
cron.history    -->     full debug file [have to work more on this one]
errors.history  -->     historical file containing all caught errors
status.history  -->     historical file with successful or abortion condition per simID
```

Inside $pfound/logs, a simid.success will be written if no errors are caught during property calculation. If an error condition is issued, then 2 files are written, simid.aborted and simid.debug. This last file should be sent by e-mail to the responsible for the property calculation.

!!
Examples of the output obtained:

```
guests@yosemite history]$ more cron.history
###---###
```

```
Cronjob started at Mon Jun 05 15:06:00 GMT 2006
HOSTNAME: yosemite
USER: guests

[ Checking directories/scripts ]
Work dirs/scripts checked

[ Checking simID's to process ]
Available files: 1.dcd 1.psf 2.dcd 2.psf
Possible simID's to be processed: 1 2

[ Testing simID's ]
        --> 1 <--
Seems eligible to proceed.
Creating working dir /home/guests/behemoth/processed/1
Checking if 1.dcd file is being modified
Moving 1.dcd to /home/guests/behemoth/processed/1
Checking if 1.psf file is being modified
Moving 1.psf to /home/guests/behemoth/processed/1
        --> 2 <--
Seems eligible to proceed.
Creating working dir /home/guests/behemoth/processed/2
Checking if 2.dcd file is being modified
Moving 2.dcd to /home/guests/behemoth/processed/2
Checking if 2.psf file is being modified
Moving 2.psf to /home/guests/behemoth/processed/2

SimId's to proceed: 1 2

[ Checking file extensions for simId 1 ]
1.dcd: identified as a trajectory file
1.psf: identified as a topology file
Checking if top/traj are uniquely defined ... ok

[ VMD executation stat on simId 1 ]

*** Tcl Trace start ***
Unable to load file '1.psf' using file type 'psf'.
    while executing
"mol new $topology waitfor all"
    (procedure "::pfoundBasic::loadTraj" line 12)
    invoked from within
"::pfoundBasic::loadTraj $topology $trajectory"
    ("eval" body line 1)
    invoked from within
"eval {::pfoundBasic::loadTraj $topology $trajectory}"
*** Tcl Trace end ***
```

```
[ Checking file extensions for simId 2 ]
2.dcd: identified as a trajectory file
2.psf: identified as a topology file
Checking if top/traj are uniquely defined ... ok

[ VMD executation stat on simId 2 ]
No catched errors!

Cronjob ended at Mon Jun 05 15:06:05 GMT 2006
###---###
```

```
[guests@yosemite history]$ more errors.history
*** Tcl Trace start ***
SimId   1       aborted
Mon Jun 05 15:06:05 GMT 2006
Unable to load file '1.psf' using file type 'psf'.
    while executing
"mol new $topology waitfor all"
    (procedure "::pfoundBasic::loadTraj" line 12)
    invoked from within
"::pfoundBasic::loadTraj $topology $trajectory"
    ("eval" body line 1)
    invoked from within
"eval {::pfoundBasic::loadTraj $topology $trajectory}"
*** Tcl Trace end ***

[guests@yosemite history]$ more status.history
SimId   1       aborted         Mon Jun 05 15:06:05 GMT 2006
SimId   2       successfull     Mon Jun 05 15:06:05 GMT 2006

[guests@yosemite behemoth]$ ls logs/
1.aborted  1.debug  2.success
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

P-found will check periodically under $pfound/logs, to see if there are any *.success files, to start the import of the calculated properties/graphs.
As soon as the DW finishes the import of the relevant data, it should warn VMD in order for this to delete the data [not yet implemented]. For example, It could write on $pfound/logs a file simid.done

On contrary, if it exists a simid.aborted file, an error log contained on simid.debug file will be sent to the administrator [not yet implemented].

About XY graphics & ascii matrices

The properties calculated on the trajectories, will be saved on ascii files, and manipulated in accordance to the format that most suites our purposes.

All xy results (like .rmsd, .rmsf …), are being converted to a PNG file (default) using xmgrace (gracebat).

Ascii-matrices (like .ss) will also be converted to a picture format. This is still under discussion. One of the proposals is to convert the ascii-matrices to XPM, and then use ImageMagick to convert to PNG, for example. If instead of using Tcl scripting to do this we use C, then , work with libpng to achieve the goal. Another suggestion it's to produce the matrices with grace. Already have a file example of a matrix, should not be that hard to implement this. The output file will also be a .png file.

Index of file extensions produced during a trajectory analysis

Extension	Short description	Output type
.dm	--> **d**istance **m**atrix	(ascii matrix)
.hb	--> number of **h**ydrogen **b**onds *vs* time	(ascii xy graphic)
.info	--> protein primary sequence, plus some other protein/trajectory **info**rmation	(ascii)
.nhb	--> number of **n**ative **h**ydrogen **b**onds *vs* time	(ascii xy graphic)
.nhbi	--> **n**ative **h**ydrogen **b**onds **i**dentification	(ascii)
.nhbm	--> **n**ative **h**ydrogen **b**onds **m**atrix (time persistence)	(ascii matrix)
.nc	--> number of **n**ative **c**ontacts *vs* time	(ascii n-xy graphic)
.rgyr	--> **r**adius of **gyr**ation *vs* time	(ascii xy graphic)
.rmsd	--> **r**oot **m**ean **s**quared **d**eviation *vs* time	(ascii xy graphic)
.rmsf	--> **r**oot **m**ean **s**quared **f**luctuation *vs* time	(ascii xy graphic)
.ss	--> residue **s**econdary **s**tructure *vs* time	(ascii matrix)
.ssc	--> distribution of res. *vs* secondary structure	(ascii n-xy)
.sasa_*	--> SASA files[1]	(ascii n-xy/matrix)
.X.png	--> PNG figure of property X (eg., **.rmsd.png**)	(figure)

[1]Property calculation still under development ;-) (Have to try vmd measure sasa command).

Note: only **bold** extensions files are presently being produced though the system is able to do it. The properties not being calculated, will be added in a future version to the DW.

After a successful condition, P-*found* will import the above listed files into BLOBs.

Implementation of Tcl packages (pfoundBasic – an example)

The procedures that calculate the properties are being written inside the pfoundBasic package. The namespace concept will also be used, in order to take care of problems with future variable/procedure names collision.

I adopted the following convention regarding procedure naming:
- procedures that begin with a capital letter --> used inside pfoundBasic scope
- procedures that begin with lower case letter --> used inside pfound.cron scope

The modularity of procedure/package/namespace Tcl structures, provides an easy way to implement new routines. I'm trying to follow the Tcl rules stated at:

- Namespaces and Packages: http://www.wjduquette.com/tcl/namespaces.html
- Pratical Programming in Tcl and Tk (4th ed., 2003): chapters 12 & 14
- Tcl Style Guide: http://purl.org/tcl/home/doc/styleGuide.pdf

Other sources of helpful information:
- http://www.tcl.tk/software/tcllib/
- http://wiki.tcl.tk/
- http://130.83.61.160/~robert/LinuxDoc/Programmieren/TclTK/DevGuide/cdrom/
- http://www.tcl.tk/cgi-bin/tct/tip/
- http://tcl.tk/man/tcl8.4/TclCmd/
- http://www.ks.uiuc.edu/Research/vmd/vmd-1.8.4/docs.html
- http://www.ks.uiuc.edu/Research/vmd/vmd-new/devel.html
- http://www.ks.uiuc.edu/Research/vmd/alpha/
- http://groups.google.com/group/comp.lang.tcl
- http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
- http://plasma-gate.weizmann.ac.il/Grace/
- http://koala.ilog.fr/lehors/xpm.html
- http://scv.bu.edu/SCV/Tutorials/ImageFiles/image101.html

Package procedures index

These are the currently exported procedures already written. Some of them need more debugging.

proc ::pfoundBasic::loadTraj {topology trajectory} {
    Loads trajectory based on the extension
proc ::pfoundBasic::infoMolecule {filename} {
    Get primary sequence (and other molecule relevant information; 2do)
proc ::pfoundBasic::calcRgyr {selection filename} {
    Calculates gyration radius
proc ::pfoundBasic::calcRmsd {selection filename} {
    Calculates root-mean-squared deviations
proc ::pfoundBasic::calcRmsf {selection filename start {end 1000} {step 1}} {
    Calculates root-mean-squared fluctuations
proc ::pfoundBasic::calcHb {output {selection "all"} {dist 3.0} {angle 20.0}} {
    Calculates the number of hydrogen bonds

proc ::pfoundBasic::calcHbNativeTrans {output {selection "all"} {dist 3.0} {angle 20.0}} {
Studies the persistence of native hydrogen bonds

proc ::pfoundBasic::calcSsTrans {filename selection} {
Calculates secondary structure

proc ::pfoundBasic::calcNatContacts {selection output {cutoff 4.2} {neighbours 4}} {
Calculates native contacts

proc ::pfoundBasic::distMap {selection out {fr 0}} {
Calculates a distance map

proc ::pfoundBasic::xyToGrace {infile {hdevice PNG}} {
Produces XY graphics using xmgr

proc ::pfoundBasic::timeFormat { } {
Time format

proc ::pfoundBasic::checkScripts {path} {
Checks existence of needed dirs and scripts

proc ::pfoundBasic::checkSims { } {
Which simId's are available to process?

proc ::pfoundBasic::createSomeDirs { } {
Creates some of the working dirs, if they don't exist (logs, processed, to-process)

proc ::pfoundBasic::createSimDir { } {
Checks files and creates a dir $processedPath/simId
for each SimId eligible to be processed by VMD, moving the relevant
trajectory files into that dir.

proc ::pfoundBasic::getExt {Dir} {
Gets file extension on working files

proc ::pfoundBasic::extId {sim item} {
Parses extension to file type: top || traj

proc ::pfoundBasic::extUniq { } {
Check if there's a topology||coordinates & trajectory by files extension

proc ::pfoundBasic::endCronTime { } {
Returns cron end time

proc ::pfoundBasic::calcSasa {output {selection protein} {zslice 0.1}} {
Solvent accessible surface area (using naccess).

proc ::pfoundBasic::genMovie {output ...} {
Generates a Movie. Its still under development.

proc ::pfoundBasic::calcSsCounts {output} {
Counts the number of residues per secondary structure motif

proc ::pfoundMultimedia::genMovie { bn mt mf rm } {
Main procedure

proc ::pfoundMultimedia::angleChanged { newval } {
Changes angle of rotation

proc ::pfoundMultimedia::trajstepChanged { newval } {
Determines which trajectory frames are going to be renderized

proc ::pfoundMultimedia::durationChanged { args } {
Movie duration for an animated gif

proc ::pfoundMultimedia::buildmovie {} {
proc ::pfoundMultimedia::testfilesystem {} {
    Test for file creation capability for work areas
proc ::pfoundMultimedia::genframes_rotation {} {
    Generates all frames for a rotation movie
proc ::pfoundMultimedia::genframes_trajectory {} {
    Generates all frames for a trajectory movie
proc ::pfoundMultimedia::renderframe { basefilename } {
    Render one frame using existing settings
proc ::pfoundMultimedia::ppmtompeg {} {
    Calls ImageMagick ppmtompeg
proc ::pfoundMultimedia::imgif {} {
    Calls NetPBM convert program
proc ::pfoundMultimedia::convertframes { newformat finalformat } {
proc ::pfoundMultimedia::smoothframes { } {
    Calls ImageMagick pnmsmooth
proc ::pfoundMultimedia::rescaleframes { } {
    Calls ImageMagick pnmscale
proc ::pfoundMultimedia::labelframes { } {
    Calls ImageMagick ppmlabel
proc ::pfoundMultimedia::compositeframes { } {
    Calls ImageMagick pnmcomp
proc ::pfoundMultimedia::cleanframes {} {
    Cleans all temporary files created
proc ::pfoundMultimedia::ChangeRep { } {
proc ::pfoundMultimedia::writePdb {output} {
    Writes a pdb file from the first frame of the trajectory
proc ::pfoundMultimedia::genSnapshot {} {
    In testing phase. Trying to incorporate orient1.01

<u>NACCESS compilation issues</u>

I was successfull compiling naccess on behemoth, no errors were reported.
Executing a test with xray.pdb (monomer from 1tta) I get the following:

```
nferreira@behemoth:/data/moldyn/nferreira/data/apagar>/data/moldyn/nferreira/opt/naccess2.
1.1/naccess xray.pdb
naccess: using defualt vdw.radii
naccess: using default STD FILE
startio: error in format
apparent state: unit 4 named xray.log
last format: (a,i,a,i,a)
lately writing sequential formatted external IO
Abort
```

I commented in the fortran code the write command that issues this error.
This was enough to surpass the error.

Nuno Loureiro-Ferreira 22may2006

Remarks - 1

One of the purposes of the DW is to store user's trajectory raw data. In principle, we won't change the raw data in any respect, namely, centering the protein on the simulation box, checking that the molecule is whole, etc. These things must be taken care by the user, though I think we should have some checking on these issues to.

Since behemoth is a cluster, we would get better performance on the calculation of the properties per simulation available, if we could distribute the work in the cluster.

Remarks – 2 **(060302 - chat with John Stone)**

Final remarks:

1) VMD can run on a cluster and distribute jobs to multiple nodes in order to concurrently analyze multiple trajectories

Suggestion of implementation:

- Start cron job (VMD) on a cluster node
- Cron job counts up the number of datasets to be analysed
- Cron job assigns a node ID to each job
- Loop over calls to start VMD instances on the appropriate nodes until we either run out of nodes or run out of jobs

If we run out of nodes then, wait until the first batch of jobs are done, then continue to assign jobs to nodes.

If we run out of jobs, than the cron job goes to sleep (exits)

This must be worked out with Olivier and John, but for the time being, it has low priority.

2) The number of files needed to be loaded by VMD, depend on the MD package that produced the trajectory. We only need to be sure if VMD has both structure and coordinate information. For example MMTK NetCDF file, contains both information in a single file, therefore only one file will be needed for VMD to correctly access/calculate data from the MMTK trajectory. Other programs like Amber/Charmm/NAMD use two files, at least that I know of.

Suggestion of implementation:

- We should write the code to work with a list of files

For the time being, I'll concentrate on trajectories produced by Gromacs, NAMD and Amber. These MD packs, need at least 2 files. Code is written is this way.

3) Add MMTK, LAMMPS and CPMD to our list of trajectory producers. Ask John for an protein trajectory example produced by each of these packs.

4) We must write the code in such a away that VMD can recognize the files being loaded, using the “type” flag (when using mol new and mol addfile cmds; if we wish to load files direclty when we issue the vmd command, then we must issue: vmd -parm7 file.parm7 -crdbox file.crd, for example).

Suggestion of implementation:

- When a user uploads data to the P-*found* data repository, it should give exact info about the files (namely the MD package/version, box information)
- That info will be stored as metadata, and will generate an index file description (this is going to be done by Vitaliy).

  Example of 1013.ndx:

SoftPack	AMBER7
BoxInfo	yes

- Cron job will read this index file, and parse to VMD the correct file type, using the “type” command flag

  For example, if the uploaded files were produced by AMBER7, with box information saved on the trajectory, then cron job will issue the following commands:

  ```
  mol new file.parm7 waitfor all
  or,
  mol new file.parm7 type parm7 waitfor all
  plus,
  mol addfile file.crd type crdbox waitfor all
  ```

  (VMD recognizes the .parm7 by its extension, because it's uniquely defined; but "au contraire", crd could come from a traj with or without box info)

  If the files were produced with Amber7 without box info, then:

  ```
  mol new file.parm7 waitfor all
        plus
  mol addfile file.crd waitfor all
  or, mol addfile file.crd type crd waitfor all
  ```

  (VMD by default recognizes .crd as being an Amber coordinates file without periodic box info)

This is an urgent issue but for now, files will be recognized by their extensions.

5) Treating PBC of uploaded trajectories, is definitly an important issue to be taken. Simulations with wrapped coordinates won't work well, for example with the RMSD and RMSF calculations, because the protein could be "broken".

Suggestion of implementation:

- Unwrap coordinates prior calculating the properties that we want.
- There's a script on the VMD-L that wrapps off coordinates of oxygen atoms. It could be a good example to write an unwrap procedure.
- There's also a pbcwrap procedure in the VMD scripts repository, which is just the opposite of we want to do. This script only works with the pair of files xst/dcd. I could write to Jan Saam (the author) and ask him if he has an wrapoff procedure already written saam@charite.de.

This is an urgent issue but for now, files being processed have whole proteins.

Remarks – 3 (060303 - chat with John Stone)

1) In order to generate movies with vmd in batch mode (-dispdev text), we should not use snapshots, but tachyon rendering (render comand).
2) **VMD is still not very stable using the Python interpreter. Tcl for the time being, is the best choice.** ;-)

Remarks – 4 (060308 – test on behemoth)

The xyToGrace proc was not working. Instead of calling exec gracebat ..., I called like : exec /full_path_to/gracebat, and it worked. I do not understand it, since gracebat is in the path. The same for naccess, plus, it gives an error stating that the file vdw.radii was not found. This doesn't happen on yosemite. Perhaps when calling a sub-process on a behemoth node, that node does not see the $PATH (among others) defined in the mother node.

This issue was resolved on 060527. At this moment, gracebat and naccess are being called by their full path name.

Remarks – 5 (060311 – about raster files)

I read some stuff about raster image formats (RIFs), like .xpm, .ppm ...
A raster image file is generally defined to be a rectangular array of regularly sampled values, known as pixels. Each pixel (picture element) has one or more numbers associated with it, generally specifying a color which the pixel should be displayed in.

I asked a question in VMD-L about if there are any scripts done and/or if the approach of converting an ascii matrix to a raster text file, is a good one. The rasterized file will be then converted to .png using convert (ImageMagick) or ppmtopng (NetPBM).

<u>Remarks – 6</u> (060313 – about raster files, part II)

<u>Question #1 posed on VMD-L (Nuno):</u>

Le Samedi 11 Mars 2006 20:05, Nuno R. L. Ferreira a écrit :
> Hi *
>
> I'm calculating some properties on a trajectory, using VMD in batch mode.
> Some of these properties are ascii matrices, like a Calpha distance map
> (matrix has real numbers) or secondary structure vs time (matrix contains
> stride code letters; nº rows == nº residues).
>
> How do I convert this plain matrices to an image?
> I though that the best way was to produce a raster file (like .ppm or
> .xpm), and then use ImageMagick or NetPBM programs to convert the text file
> to a more appropriate image format, like .png.
>
> To do that, I need some scripting.
> If I want to get a .ppm (or netpbm like) file, I need to convert each
> matrix entry to an RGB code. In the .xpm case, I need to convert the matrix
> entries to a code, which is then coupled to a color via a code on the
> header of the file.
>
> Do you guys ever tryed to do that? Is this a good approach?
>
> Best regards,
> N.

<u>Answer #1 to Q#1 (Nicolas):</u>

You should try Gnuplot. It is very easy to plot matrices with this
software. If I remember, the commands are :

```
set pm3d map
set palette gray
splot "yourfile.txt"
```

<u>Answer #2 to Q#1 (Jerome):</u>
I use ImageMagick as a very convenient way to do what you suggest. Here is the
part of my awk script that does the image writing. The array 'color' is used
to associate RGB values to my raw data (in the 'struct' array).

```
--------
        # Pipe PPM image data to convert to make a PNG
        conv = "convert - " name ".png"

        printf "P3 %i %i 255\n", NR, height * residues | conv

        # last residue at the top, first at the bottom
        for (r=residues; r>=1; r--) for (i=1; i<=height; i++) {
           for (t=1; t<=NR; t++) {
              print color[struct[t, r]] | conv
           }
        }

        close(conv)
--------
```

So I use the ASCII PPM format, but because of the piping capability of GNU awk, I don't even have to actually write the PPM file on disk.

Answer #3 to Q#1 (John Stone):

Both PPM and XPM would be very easy to generate within Tcl scripts.
Let me know if you need help with that.

Answer #4 to Q#1 (Axel):

i think your idea is pretty reasonable and straightforward to
implement. i'd suggest you try the .xpm format first, as it is
quite self-explanatory and easily scriptable from VMD.

Question #2 (Nuno):
I tryed quickly Nicolas suggestion (gnuplot) in a Ca distance matrix, and its reasonably simple to produce a figure. I think gnuplot won't plot matrices with letter entries (secondary structure information), though thats not a big issue since the original matrix could be converted to a matrix containing numbers.

I did not yet tryed to implement a Tcl script to produce outputs formated as xpm/.ppm.
The suggestion of Jerome seems to be cool (I like the pipe stuff).
I agree with axel, since we do not need to change the relevant property value calculated, if I use the .xpm format. If the choice was the .ppm format, I have to convert the property value to a RGB code. Jerome surpasses this issue by an array convertion scheme.

Jerome, did you perform any kind of convertion on your matrix prior saving the information to the array struct (like a descritization of the data?). Is your color array pre-defined, or is it defined in accordance with the range of values in the matrix?

But one question arises, dealing with this .xpm/.ppm (not an issue for
gnuplot). How am I going to incorporate the x-y tick labels/marks? By writing
some more pixels on the first column and last row of the outputed file? ;-)
Answer #1 to Q#2 (Jerome):

Actually, since we are trying to do the same thing, I thought it would be
simpler to share the whole stuff. I use a VMD script that outputs a file
containing one line per frame. It includes first the frame number, then the
list of structures for each residue:

```
0 C C T T T C C C H H H H H H H H H H G G H H H
1 C C C T T C C H H H H H H H H H H H G G H H T
2 ...
```

Then I apply the script stride2png, attached to this message. This script can
be tuned a bit, depending on the aspect ratio you want for the final picture.
I think everything inside is properly commented.

A great way to add 100% vector axes and legends around that, is to make an
empty graph with the right axes in xmgrace and export it as EPS. You can then
open it with inkscape (you need to have pstoedit installed to do that),
import the png picture and overlay it with the graph. The result can be saved
as SVG, PS or PDF to get a perfect rendering regardless of the resolution.

Remarks – 7 (0603?? – about raster files, part III)

John suggested to use libpng, if the code to convert an ascii matrix to a figure is written in C.

Remarks – 8 (060318 – about the reference frame)

When analysing a trajectory, the user must define for certain property calculations (eg. RMSD, native hydrogen-bonds) a reference frame. This reference frame usually is the first frame to be loaded into VMD. In order to simplify the upload process to the DW, a maximum of 2 files are allowed to be uploaded (a coordinates/topology file + trajectory).
The topology file is prefered over the coordinates file (check LoadingScheme_CS_NF....pdf). If the user chooses to upload a topology + trajectory, the first frame will come from the trajectory. If instead the user uploads a coordinates + trajectory, the first frame comes from the coordinate file. Presently, the code is written is such a way that it deletes the 1st frame if it comes from a file other than a regular trajectory data file, to be consistent with the case where we have a topology + trajectory. I think that the possibility of having a reference frame other then the first frame coming ftom the trajectory, should be discussed with the target scientific community.

Remarks – 9 (060320 – about the linking scheme)

The linking scheme presented in this document could be changed, accordingly to suggestions regarding performance issues.

Remarks – 10 (060529 – about creating a pfound user & other things)

I think we should create a new user named pfound on behemoth, in order to have consistency on how to deal with specific situations related only with the properties. The new dir /data/pfound should be created. Inside this dir VMD is going to work on the trajectories coming from the DW.

I also suggest to install vmd (pre-compiled version in a first approach) and naccess (as well as other programs needed by vmd) under /opt . I have then to change the paths on the scripts.

Remarks – 10 (29may06 - Notes on common molecular file formats)

VMD natively understands several popular molecular data file formats: PDB coordinate files, CHARMM, NAMD, and X-PLOR style PSF topology files, CHARMM, NAMD, and X-PLOR style DCD trajectory files, NAMD binary restart (coordinate) files, AMBER structure (PARM) and trajectory (CRD) files including both the old format and the new formats used by AMBER 7.0, and Gromacs (e.g. GRO, G96, XTC, TRR) structure and trajectory files. These files may contain some redundant information and can be loaded in different combinations.
PDB files contains data about atoms, residues, segment names, occupancy and beta factor, and one coordinate set. PSF and PARM files contain atoms, residues, segment names, residue types, atomic mass and charge, and the bond connectivity. VMD supports four file formats used by Gromacs: GRO, G96, TRR and XTC. GRO and G96 files contain structure information including atoms, residue and segment data, and one coordinate set. CRD, DCD, TRR and XTC files contain only coordinate data (frames). It should be noted that while PDB, GRO and G96 files were designed to contain only one coordinate set, multiple files can be concatenated into one larger file to create a makeshift trajectory file which can be loaded by VMD.
When VMD loads a file it requires information about atom names and coordinates and tries to fill in the rest. Since the PDB file contains all this information, it does not need to be loaded with any other data files. However, the PDB file doesn't contain the atom types, masses, and charges, so these are guessed or assigned default values. In particular, charges will be assigned a value of 0.0 if the file does not contain explicit charge information.
A PSF file does not contain coordinate information so it must be loaded along with a PDB or DCD file. If a PDB and PSF are given there is no missing data and VMD makes no assumptions. If a PSF and DCD are given then only the chain identifier and occupancy and beta values are missing so they are given a default value. A PARM file is similar to a PSF in that it too contains no coordinate information. It must be loaded along with a CRD trajectory file. If a PARM and CRD file are loaded together, then only the segname and chain ID for the atoms are left blank. A CRD or DCD file can be specified along with the PDB, in which case the PDB file will be read as normal, and then coordinate sets are read from the DCD or CRD until the end of the file is reached. Gromacs GRO and G96 files can be loaded on their own since they contain the necessary atom data and coordinates. They can also be loaded along with TRR and XTC files to obtain trajectory data. Additional coordinates from a PDB, CRD, or DCD file can be appended to the current coordinate set using the *Molecule File Browser* form.

Taken from VMD user guide (1.8.4 version)
http://www.ks.uiuc.edu/Research/vmd/current/ug/node21.html

Reamark 11 (VMD input file types - talk with John - 30may2006)

Mon 15:43:nunolf>John, from previous discussions and from mailing list, VMD recognises the files by their extension, but in some cases (eg. amber trajectory with box info), we must specify the type of the file
Mon 15:44:nunolf>like, mol addfile file.crd type crdbox waitfor all
Mon 15:47:nunolf>Could you point me to the vmd source file, where I can find all file types VMD can read?
Mon 15:50:nunolf>I want to have a list of those file types. As you suggested previously, we should read files whenever possible, by specifying their type.
16:33:johns>nunolf: there isn't a single source file which contains this information.
16:33:johns>The file extensions are actually defined in the plugins themselves.
16:33:johns>You can query the list of plugins and their filetypes using Tcl commands however.

```
[guests@yosemite guests]$ vmd_185 -dispdev text
Info) VMD for LINUX, version 1.8.5a3 (May 25, 2006)
[...snip...]
vmd > plugin list {mol file reader}
{{mol file reader} biomocca} {{mol file reader} cpmd} {{mol file reader} psf} {{mol file reader} pdb} {{mol file reader} dcd} {{mol file reader} gro} {{mol file reader} g96} {{mol file reader} trr} {{mol file reader} xtc} {{mol file reader} parm} {{mol file reader} crd} {{mol file reader} crdbox} {{mol file reader} namdbin} {{mol file reader} binpos} {{mol file reader} grasp} {{mol file reader} msms} {{mol file reader} stl} {{mol file reader} cube} {{mol file reader} edm} {{mol file reader} ccp4} {{mol file reader} DSN6} {{mol file reader} brix} {{mol file reader} plt} {{mol file reader} raster3d} {{mol file reader} parm7} {{mol file reader} rst7} {{mol file reader} tinker} {{mol file reader} uhbd} {{mol file reader} dlpolyhist} {{mol file reader} lammpstrj} {{mol file reader} xyz} {{mol file reader} cor} {{mol file reader} molden} {{mol file reader} delphibig} {{mol file reader} grd} {{mol file reader} situs} {{mol file reader} dx} {{mol file reader} spider} {{mol file reader} map} {{mol file reader} fld} {{mol file reader} fs} {{mol file reader} pqr} {{mol file reader} mol2} {{mol file reader} grid} {{mol file reader} car} {{mol file reader} mdf} {{mol file reader} gamess} {{mol file reader} xsf} {{mol file reader} bgf} {{mol file reader} xbgf} {{mol file reader} webpdb} {{mol file reader} netcdf}
```

Presently, pfound is asking for 2 files: a topology and a trajectory.
The following dictionaries must be changed:

TrajectoryFormatDict & TopologyFormatDict

I think that the entries in this two dicts. should be (for now) the following:

TrajectoryFormatDict entries	**File extension**	**VMDFileType**
AMBER coordinates	crd	crd

TrajectoryFormatDict entries	**File extension**	**VMDFileType**
AMBER coordinates with periodic box	crd	crdbox
CHARMM Coordinates	cor	cor
CHARMM,NAMD,XPLOR	dcd	dcd
GROMACS TRR	trr	trr
GROMACS Compressed XTC	xtc	xtc

TopologyFormatDict entries	**File extension**	**VMDFileType**
AMBER Parm	parm	parm
AMBER7 Parm	parm7	parm7
CHARMM, NAMD, XPLOR	psf	psf
GROMACS Gro	gro	gro
GROMACS G96	g96	g96
GROMACS Top	top	top
PDB	pdb	pdb

Remarks – 12 **(061211 – about vmd console)**

A way to learn the commands equivalent to the interactive operations in the vmd-GUI is to use the command: logfile console. To stop outputing to the text console use: logfile off

Remarks – 13 **(About Molecular Representation)**

Molecules are represented by *reps*, which are defined by four main parameters: the selection, the drawing method, the coloring method, and the material. The selection determines which part of the molecule is drawn, the drawing method defines which graphical representation is used, the coloring method gives the the color of each part of the representation, and the material determines the effects of lighting, shading, and transparency on the representation.

2DOList

- Scripts need more debugging
- Test with different types of trajectories (MD pack & proteins with one or more chain)
- Install VMD from source on behemoth (I'm using the pre-compiled version 1.8.5a3)
- Re-ordering of arguments on procs
- Initialize variables in each procedure
- Check if uploaded trajectory is wrapoff (protein must be whole)
- Read trajectory specifications (and other info) from an external file produced by P-*found*
  - *authors*
  - *program version*

    - *vmdFileType (traj & top)*
    - ...
- Load files to VMD using the “type” flag; check if VMD have topology + coordinates
- Think on how to deal with AMBER trajectories (with or without box information)
- Test measure sasa vmd command (compare with calcSasa)
- Distribute each simulation job to a different behemoth node
- Parse ascii matrix outputs to XPM format; convert to .png using ImageMagick. This is still under discussion.
- Improve xyToGrace procedure (study xmgr format)
- Delete selections/variables (prevent memory problems)
- How to deal with oligomeric proteins (multiple chain's)? Do we need to do major changes on the package scripting?

Monday, December 11, 2006
Nuno Ricardo Loureiro Ferreira