LHSG-CT-2003-503420

BioXHIT

A European integrated project to provide a highly effective technology platform for Structural Genomics.

Life Sciences, Genomics and Biotechnology for Health

WP5: De5.2.9 Investigate integration of tracking database/dbCCP4i with other PX applications

Due date of deliverable: **31.06.2007**
Actual submission date: **17.07.2007**

Start date of project: **1.1.2004** **Duration: 60 months**

Organisation name of lead contractor for this deliverable: CCLRC-CCP4
Daresbury Laboratory, Warrington WA4 4AD UK **Author** Peter Briggs

Investigation into integration of the Tracking Database with PX application software (BIOXHIT deliverable 5.2.9)

Peter Briggs, CCP4

1 Introduction

This report describes the results of investigation into the possibility of integrating the tracking database system into other protein crystallography (PX) application software.

The MrBUMP automated Molecular Replacement package developed by Ronan Keegan and Martyn Winn was used as a pilot study to investigate the issues and assess the feasibility of using the system in other systems. MrBUMP is typical of automated software pipelines – it runs many different programs (dependent on protocols selected by the user at the start), analysing the results and making decisions internally, and presenting its final results to the user.

MrBUMP seemed to be a good choice for the pilot study as the developers had already identified issues with the presentation to the end user of both the procedure and the final results. The end user faced a challenge when reviewing and evaluating these results, and typically needed to filter through many different output files and directories. Using the tracking database to store information about the individual steps within the MrBUMP run would enable users to monitor and review the progress and result from the run using the visualiser tools already developed.

2 Background to the MrBUMP Package

MrBUMP is a framework that automates the determination of macromolecular structures via the Molecular Replacement method (MR). In MR, a trial model based on a known structure is used as a template for extracting the structural information of the target from the experimental reflection data. The success of the MR procedures is critically dependent upon the choice of trial model, and so the approach used in MrBUMP emphasises the generation of a variety of search models.

In good cases MrBUMP will give a single clear solution to an MR problem; in other cases it will suggest a number of likely search models that can be investigated manually by the scientists – typically around 10-15 search models, although it could be as much as ten times that. In preparation for molecular replacement each model structure also undergoes rounds of processing by various programs such as side-chain pruning based on sequence similarity between the target and template. The modifications are then carried out using various CCP4 programs such as Chainsaw or Molrep. This can result in a large number of different intermediate and final output files that can easily overwhelm the end user.

The MrBUMP web site is at http://www.ccp4.ac.uk/MrBUMP/

3 Implementation of the Tracking Database into MrBUMP

The following sections outline the objectives, design decisions, and the practical and technical details of incorporating the tracking database into the MrBUMP package.

3.1 Objectives for incorporating the tracking database into MrBUMP

The aims of the MrBUMP developers in integrating the tracking database system were to obtain the following benefits:

- Present the results of a run in a more intuitive graphical form
- Make it easier for the user to find and examine a particular file associated with the processing of each search model
- Allow real time monitoring of the progress of a run

3.2 Implementation Decisions

In order to use the system the MrBUMP developers first needed to make decisions about how to represent a MrBUMP run within the tracking database. It was decided:

- Each run would be represented by a new project within the system

- A job-node would represent stages within a run that would be determined by the pipeline program, for example in a typical run of MrBUMP job-nodes would represent any or all of the following set of steps:

    - Processing of the input target data.
    - Carrying out the various searches (sequence, secondary structure, domain, multimer)
    - Multiple alignment of target with search model sequences.
    - Downloading search model PDB files.
    - Preparing search models (side-chain prunings, superposition of structures, etc.)
    - Molecular replacement.
    - Refinement of molecular replacement results.
    - Phase improvement.

    Each of these steps can involve the use of several CCP4 or third-party programs and can occur multiple times depending on the number of search models used, so it makes more sense to represent the steps as nodes rather than the individual jobs. This helps to provide a more intuitive and informative view of what is happening in the pipeline.

While this interpretation of job-nodes in MrBUMP differs from the original conception of jobs within CCP4i (where each job is a distinct run of a CCP4i task or script), the database tracking system is flexible enough to allow these different interpretations to co-exist. Ultimately as far as the tracking system is concerned, a job-node represents the fact that “something happened”, and that “something” has various attributes such as a name, a description, and a set of associated input and output files.

3.3 Practical and Technical Details for the Implementation

The integration of the tracking database system took place as part of the developments in MrBUMP version 0.4.0.

3.3.1 Communicating with the database handler

Since MrBUMP is implemented in the Python programming language all interactions with the database system could be performed using the Python database client API module “dbClientAPI.py”. The general procedure is:

- Establish a connection with the database handler
- Create a new project to represent the MrBUMP run
- Add new jobs to the project to represent each stage of the run
- Set the data associated with the job, for example name, status, title and associated logs and input and output files.

The following code fragments show how these operations are implemented in the MrBUMP code:

1. **Including the client API functions:**

   import dbClientAPI

2. **Establishing a connection to the database handler**
   This is done by creating a HandlerConnection instance, which is then used for subsequent communications with the database:

   ```
   conn = dbClientAPI.HandlerConnection(‘mrbump’,True)
   ```

   The connection persists until the client closes it.

3. **Creating a new project**
   This requires a unique project name and the path of an associated directory:

   ```
   conn.CreateProject("mrbump_1","/home/user/mrbump_1")
   ```

4. **Adding a job to the project**
   This returns a new job id number:

   ```
   jobid = conn.NewJob("mrbump_1")
   ```

5. **Set data for the new job (title, status, …)**

   ```
   conn.SetTitle("mrbump_1",jobid,"Refine mr coordinates")
   conn.SetStatus("mrbump_1",jobid,"FINISHED")
   ```

6. **Associate input and output files with the job**

   ```
   conn.SetLogfile("mrbump_1",,jobid,"10_refmac5.log")
   conn.AddInputFile("mrbump_1”,,jobid,"abc.mtz")
   conn.AddOutputFile("mrbump_1",,jobid,"molrep1_refmac1.pdb")
   ```

7. **Closing the project and the connection**

This step isn't essential as the database handler can deal with the connection being terminated unexpectedly, however it is good practice:

```
conn.CloseProject("myproject")
conn.HandlerDisconnect()
```

3.3.2 Issues with the database handler and CCP4i

MrBUMP is typically run via a graphical interface that is part of the CCP4's "CCP4i" interface, and this integration work uncovered a problem with using the tracking database system – by default the tracking database uses the same underlying files for data storage as CCP4i, however while the tracking database can cope with sharing these resources the current version of CCP4 (1.4) cannot.

In the short term this limitation was circumvented by providing a modified version of the tracking database code which uses a "parallel" system of database files, which is not shared with CCP4i. This will be addressed by future changes to CCP4i and to the database system.

3.3.3 Launching the database visualiser

The last part of the integration process was to provide a way to view the MrBUMP jobs by running the dbviewer visualiser application. This was done in two ways:

- An option in the MrBUMP graphical interface launched an instance of the viewer at the moment the run began. This can then be used for real-time monitoring of the run's progress - a steps complete and their status changes, or new steps are added, the dbviewer updates its display accordingly.
- A "launcher" task was incorporated into CCP4i to start the viewer at any time, to review a run that was completed previously or to monitor the progress of an ongoing run.

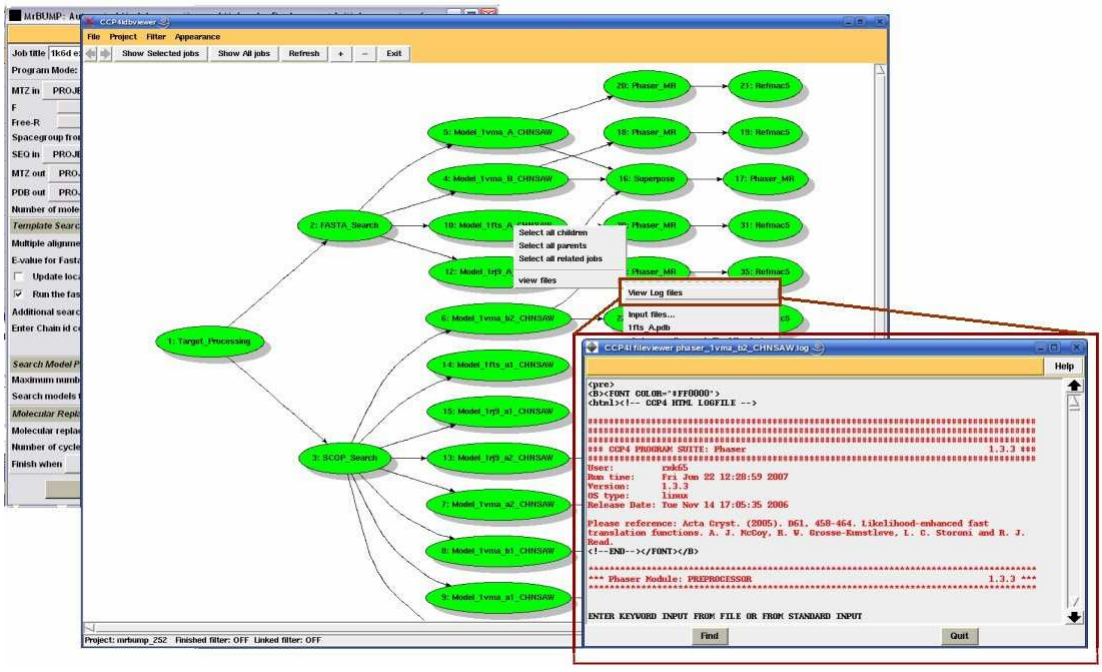

Figure 1: An example of the viewer window showing the jobs in a MrBUMP run.

3.4 Outcomes

MrBUMP version 0.4.0 was released publicly in July 2007 and incorporated the tracking database system and the viewer. It is reported on the MrBUMP website: http://www.ccp4.ac.uk/MrBUMP/#dbviewer
It has also been described in a CCP4 Newsletter article: *"Project Tracking System for Automated Structure Solution Software Pipelines"* (Wanjuan Yang, Ronan Keegan and Peter Briggs, CCP4 Newsletter on Protein Crystallography 46, August 2007): http://www.ccp4.ac.uk/newsletters/newsletter46/articles/project-tracking.html

Implementation of the database system in this version of MrBUMP has helped achieve the aims of making it easier for users to monitor, review and understand the results of the program.

The pilot project also identified a number of features in the system that were missing at the time but which would be useful to incorporate in future, specifically:

- The facility to make a MrBUMP run as a job in an existing CCP4i project, and then to add the stages of the run as "subjobs" within that job.
- The facility to store "logical links" between jobs – currently the links are inferred from the transfer of data files.
- The facility to associate arbitrary application-specific data with jobs (for example, to indicate the quality of a molecular replacement solution) and provide a mechanism to display that data in the viewer.

These additional features would also be of use to other PX software applications that wished to make use of the tracking database system.

4 Conclusion

The integration of the tracking database system into the MrBUMP package has demonstrated that the tracking database can be incorporated into an existing PX software application with relative ease. The use of the tracking database system means the progress and outcomes of an automated pipeline can be made much clearer to the end user, as well as improving accessibility to output files and other data. In addition the database viewer can be used as a monitor to view the real-time progress of the pipeline.

At the same time, the pilot study has also identified a number of issues that still remain to be resolved in order to increase the usefulness of the system to automated pipelines. These are being addressed in continuing developments of the tracking database system, with the intention of applying them both to future releases of MrBUMP and to other automated pipeline systems.