SEVENTH FRAMEWORK PROGRAMME
Research Infrastructure

FP7-INFRASTRUCTURES-2010-2 – INFRA-2010-1.2.3:
Virtual Research Communities

Combination of Collaborative Project and Coordination and Support Actions (CP - CSA)

LinkSCEEM-2

Linking Scientific Computing in Europe and the Eastern Mediterranean – Phase 2

Grant Agreement Number: RI-261600

D8.6
Pilot data processing tools and training

Final

Version:	1.0
Author(s):	Zubair Nawaz, SESAME
Date:	05/02/2015

Project and Deliverable Information Sheet

LinkSCEEM Project	**Project Ref. №: RI-261600**	
	Project Title: LinkSCEEM-2	
	Project Web Site: http://linksceem.eu	
	Deliverable ID: < **D8.6**>	
	Deliverable Nature: <DOC_TYPE: Report>	
	Deliverable Level: PU	**Contractual Date of Delivery:** 28 / 02 / 2015
		Actual Date of Delivery: 17 / 03 / 2015
	EC Project Officer: Leonardo Flores Anover	

* - The dissemination level are indicated as follows: **PU** – Public, **PP** – Restricted to other participants (including the Commission Services), **RE** – Restricted to a group specified by the consortium (including the Commission Services). **CO** – Confidential, only for members of the consortium (including the Commission Services).

Document Control Sheet

Document	**Title:** <**Pilot data processing tools and training material**>	
	ID: <**D8.6**>	
	Version: <0.7>	**Status:** Final
	Available at: http://www.eniac.cyi.ac	
	Software Tool: Microsoft Word 2010	
	File(s): LinkSCEEM-deliverable-D5.4-Year-4-Draft	
Authorship	**Written by:**	Zubair Nawaz (Z.N.)
	Contributors:	WP Leaders and Participants
	Reviewed by:	Mostafa Zoubi, SESAME Salman Matalgah, SESAME Claudio Ferrero, ESRF Stelios Erotokritou, CyI
	Approved by:	PMO

Document Status Sheet

Version	**Date**	**Status**	**Comments**
0.1	19/05/2014	Draft	First version
0.2	1/08/2014	Draft	Corrections proposed by Z.N. and Claudio Summary by Mostafa Zoubi & Salman Matalgah
0.3	5/08/2014	Draft	Review Mostafa Zoubi
0.4	6/08/2014	Draft	Executive Summary by Mostafa Zoubi & Salman Matalgah
0.5	7/08/2014	Draft	Review by Mostafa Zoubi & Salman Matalgah
0.6	25/8/2014	Draft	Review by Stelios - CyI
1.0	05/2/2015	Final draft	Final review by Mostafa Zoubi & Salman Matalgah
1.0	04/03/2015	accepted	accepted by PMO

Document Keywords

Keywords:	LinkSCEEM-2, Computational Science, Data Analysis, Heterogeneous HPC, GPGPU, OpenCL

Table of Contents

References and Applicable Documents

[1] http://www.linksceem.eu
[2] http://www.prace-project.eu
[3] LinkSCEEM-2 Project Execution Plan
[4] N. Canestrari, D. Karkoulis, M. Sánchez del Río (2011) SHADOW3-API : the application programming interface for the ray tracing code SHADOW. SPIE 8141
[5] (Poster) J. Kieffer, D. Karkoulis, V.A. Solé & M. Sánchez del Río (2011) PyFAI … yet another azimuthal integration tool. ESRF SSD 2011
[6] J. J. Rehr and R. C. Albers (2000) *Theoretical Approaches to X-ray Absorption Fine Structure*, Rev. Mod. Phys. 72, 621.
[7] I. D. Brown and D. Altermatt (1985) Bond-valence parameters obtained from a systematic analysis of the Inorganic Crystal Structure Database, Acta Cryst. B41, 244-247.
[8] L. Pauling (1975) Maximum-valence radii of transition metals. Proc Natl Acad Sci U S A. 1975 Oct; 72(10):3799–3801.
[9] http://en.wikipedia.org

Executive Summary

SESAME and ESRF were closely collaborating on WP12 to port new programs for Synchrotron-light Data-Analysis to GPUs using OpenCL. These efforts led and reflected on huge improvements and achieved a decrease on the latency on Synchrotron-light data processing and analysis for faster off-line and on-line data analysis, execution and simulation.

This deliverable "D8.6: Pilot data processing tools and training" aims to link, integrate, utilize and install the developed Synchrotron-light applications from WP12 (D12.3) - namely PyFAI, PyHST and CoNBoVal-XAS, on LinkSCEEM HPC facilities and resources.

The outcome of this deliverable is to guide and show the designated end-user how to install and run the three developed synchrotron applications. Practical examples and raw data are also included for these three applications; demonstrate in detail the easy use of such scientific synchrotron tools.

In order to adapt these scientific applications for Cy-Tera, the synchrotron applications are now installed and tested on Cy-Tera system, while ESRF and SESAME are offering full support and public raw data for testing purposes, from a user perspective an analysis study can be done utilizing the integrated LinkSCEEM resources. A full guide, how-to and training material for these applications have been prepared and published on LinkSCEEM training portal website the main achieved objected is to offer training opportunities and demonstration on synchrotron data analysis for regional researchers, communities and synchrotron scientists.

The training material is publicly available under http://supercomputing.cyi.ac.cy/additional/Tutorials/SynchrotronRadiationApplications/ and a full guide for installation is also presented on the LinkSCEEM-2 website under *Developed Codes/Software* (http://www.linksceem.eu/ls2/developed-codes-software/synchrotron-software.html).

1 Identified programs and installation guidelines

1.1 PyFAI : Fast Azimuthal Integration in Python

PyFAI performs fast Azimuthal Integration on powder diffraction images. For online data analysis, where acquisition can reach up to 60 images per second, other azimuthal integration software is lacking. PyFAI is fast, since its performance demanding parts have been written in Cython and OpenCL. The code written in OpenCL can take advantage from hardware that contains multi/many cores or GPUs.

1.1.1 Installation

PyFAI is publicly available on the ESRF's public web server.

http://forge.epn-campus.eu/projects/azimuthal/files

For the installation process, one must download the latest tar or zip file and then unzip it.

```
tar xvzf pyFAI-0.9.4.tar.gz
```

or

```
unzip pyFAI-0.9.4.zip
```

All the files are unpacked to `pyFAI-0.9.4`.

Before installation, one needs to resolve the following dependencies:

```
1. python 2.6 or 2.7
2. python-numpy
3. python-scipy
4. python-matplotlib
5. python-dev
6. python-fabio
7. pyOpenCL (optional)
```

If Ubuntu or Debian based distributions are used, then the first five items above can be easily installed using the following command:

```
sudo apt-get install python-numpy python-scipy python-matplotlib
python-dev
```

It is also recommended to install the latest version of fabio from the following website

http://sourceforge.net/projects/fable/files/fabio/

Download the latest tar and then unzip it.

```
tar xvzf fabio-0.1.4.tar.gz
```

All the files are unpacked to fabio-0.1.4. Then the following should be executed:

```
cd fabio-0.1.4
```

and install fabio with

```
python setup.py install
```

If pyFAI is to make use of a graphic card, please install pyopencl from:

http://mathema.tician.de/software/pyopencl

After all the dependencies are installed, one must go to the directory where pyFAI is unzipped.

```
cd pyFAI-0.9.4
```

This must be built and tested. In order to test pyFAI test images must be downloaded. A proxy configuration such as the following may need to be setup as follows:

```
export http_proxy=http://proxy.site.org:3128

sudo python setup.py build test
```

and install pyFAI using:

```
python setup.py install
```

The development version of the pyFAI can be downloaded and installed as follows:

```
git clone https://github.com/kif/pyFAI.git
cd pyFAI
sudo python setup.py build test install
```

1.1.2 Running a Small Example

While running the test, the installer module of pyFAI downloads test images and setups which can be used as examples. These are all located in the test/testimages folder.

Example of calibration:

Consider the image named “Pilatus1M.edf” which contains the Debye-Scherrer rings of Silver Behenate, a calibrant used in small angle scattering experiments. It was taken using a Pilatus1M detector from Dectris with a wavelength of 1 Angstrom. To perform the calibration one must run:

```
pyFAI-calib -w 1 -D Pilatus1M -c AgBh -r Pilatus1M.edf
```

This command line defines the detector (-D or –detector), the wavelength (-w or –wavelength) and the calibrant. The “-r” option is specific to module based pixel detectors to recover the missing signal hidden by gaps between the modules.

When the image appears, right click on the inner-most ring, then on the second inner-most ring. Come back to the terminal and press enter to assign rings to the corresponding d-spacing of the calibrant. Type “0” enter then “1” enter.

Once roughly calibrated (check that the lines freshly drawn overlay pretty well with the image rings), ask for a recalibration procedure by typing: “recalib 4”. Four rings will be extracted and calibrated. To quit the program, type “done”. The calibration has been saved in the “Pilatus1M.poni” file which can be subsequently used for azimuthal integration.

<u>Azimuthal integration</u>
To integrate a set of images, use the pyFAI-integrate tool and provide the list of files to process. A window pops up letting the user choose the geometry by loading a “PONI-file” or to modify all parameters by hand. There are check boxes to select all kinds of corrections: dark-current, flat-field, polarization, and also restrictions on the radial/azimuthal range.

Note: After performing a calibration operation in the test image directory, subsequent tests are likely to fail. Remove all calibration files before running the tests again.

1.2 PyHST : High Speed Tomography in Python

Since fast two-dimensional detectors are used in computed tomography, the data acquisition is much faster than the reconstruction. PyHST performs high speed tomography reconstruction by adopting a distributed and pipelined architecture. GPUs are used as compute co-processors to reconstruct slices, while the CPUs prepare the next ones. The code is written in C, Python and CUDA. Only NVidia GPUs supporting CUDA can be used for performance enhancement.

1.2.1 Installation

In order to run PyHST a CUDA capable GPU is required. Source files for PyHST can be retrieved from the following address using Git:

```
git clone git://forge.epn-campus.eu/repositories/pyhst2
```

The source file will be downloaded to pyhst2 directory.

The following dependencies need to be resolved before installation:

1. CUDA
2. Open-mpi
3. mpi4py
4. FFTW3
5. Hd5
6. h5py
7. pyMCA
8. python-fabio

After installing CUDA, the other dependencies can be installed using the following command:

```
sudo apt-get install libfftw3-dev libfftw3-doc openmpi-bin openmpi-
checkpoint openmpi-common libopenmpi-dev libhdf5-serial-dev python-
mpi4py python-h5py pymca
```

After moving to pyhst2 directory, run the following line command to install pyhst2:

```
source compila
```

1.2.2 Running a Small Example

Set the paths before running the pyhst2.

```
export PATH=~/packages/bin:$PATH
export PYTHONPATH=~/packages/lib/python2.7/site-packages/
```

Go to the example crayon directory and then run the following command:

```
pyhst2_xxxx pyhst2.par hostname,0
```

where xxxx is the build number and hostname is the hostname of your computer. The file pyhst2.par defines all the experiment parameters. The directory testdata contains the input files. The aforementioned command will generate files with names res2_xxxx.edf. The results can be viewed with the following command:

```
pymca -f res2_xxxx.edf
```

1.3 CoNBoVal-XAS : Coordination Number Bond Valence for X-Ray Absorption Spectroscopy

1.3.1 Introduction

Nowadays, experiments and simulation are complementary methods to achieve satisfying results. If the results obtained only from experiments are limited by different factors due to experimental set up, the simulations don't have such restrictions. Thus, the user can extrapolate the results from the simulation, if the results from the simulations are matched 80 – 90% with the experimental results in the same region. Therefore, experiments and simulations go side by side.
For the aforementioned reasons, the X-ray absorption Spectroscopy (XAS) technique is not only based on experiments, but also based on simulations using results from different modelling methods and codes such as Density Functional Theory (DFT), Molecular Dynamics (MD) or any structural modelling method. The idea is to use results from any structural simulation – model - as inputs for simulating X-ray Absorption Fine Structure (XAFS) experiment using a code called FEFF developed by Rehr's group at University of Washington [6].

CoNBoValXAS is a program that calculates the **Co**ordination **N**umber, **Bo**und **Val**ence and generates input files for **XAS** simulation. Results from the structural model simulated using different simulation codes are used as input for the program. The idea is based on calculating the interatomic distances around a given type of atoms in the structure. Based on distances around the query atoms the Coordination number is calculated using two parameters: i) a flexible cut-off distance and ii) selection of type of the first neighbouring atom. Both parameters are defined by the user. Thereafter, the coordination number is used to calculate the bound valence values around the query atoms based on Brown & Altermatt [7] and Pauling [8] approaches. Both coordination number and bound valence are key results to validate the plausibility of the structure in simulated models. Moreover, input files generated in order to be used by FEFF code are widely known between XAS community for simulating XAFS.

1.3.2 Problem Statement

Solid is one of the four fundamental states of matter (the others being liquid, gas, and plasma). It is characterized by structural rigidity and resistance to changes of shape or volume. The atoms in a solid are tightly bound to each other, either in a regular geometric lattice (crystalline) or irregularly (amorphous).
Generally, in theoretical simulations (DFT, Molecular Dynamics, …), one has to start looking at the simplest representation of the problem in order to enlarge the simulation to complex phenomena such as adsorption, radiation damage, erosions, etc. In solids, the **crystalline structure** offers an excellent start to enter the simulation parameters.
Crystal structures may be conveniently specified by describing the arrangement within the solid of a small representative group of atoms or molecules, called the 'unit cell.' By multiplying identical unit cells in three directions, the location of all the particles in the crystal is determined.
A solid structure in any theoretical simulation (MD, DFT …) is represented by spatial coordinates to define the position of each atom in the structure. Therefore, the user is provided

with only the position of the atoms in the 3d space. Spectroscopy simulation by FEFF is interested in the radial distribution function (RDF) around a given atom called absorber. This means that from the structural simulation results a query atom (type of atom) is selected and then looks for all the atoms within the neighbourhood of certain radius for all the occurrences of the query atom. The objective is to repeat the process for all the atoms of the same type in the structure which come to a scenario when the queried atom is at the edge or the corner of the box of the crystal as shown in Figure 1. In this case, the neighbouring atoms may exist in adjacent crystals. In that case, atom position data of the crystal needs to be replicated after unit translation in that direction. The query atom can be adjacent to any of the edge or corner of the 3d box, therefore the position data for a crystal needs to be replicated in each 3d direction after unit translation as shown in Figure 2. After extrapolation, the number of atoms increases to 27 times the original, which also requires the same number in memory.

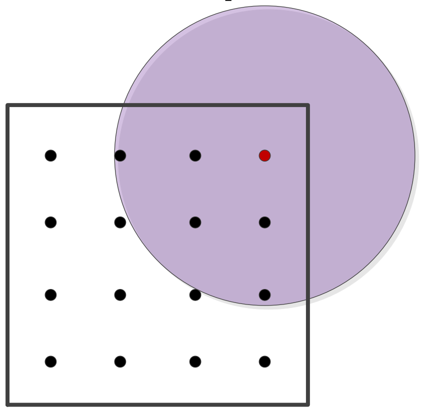
Figure 1: *Query atom in the corner of the box of the crystal*

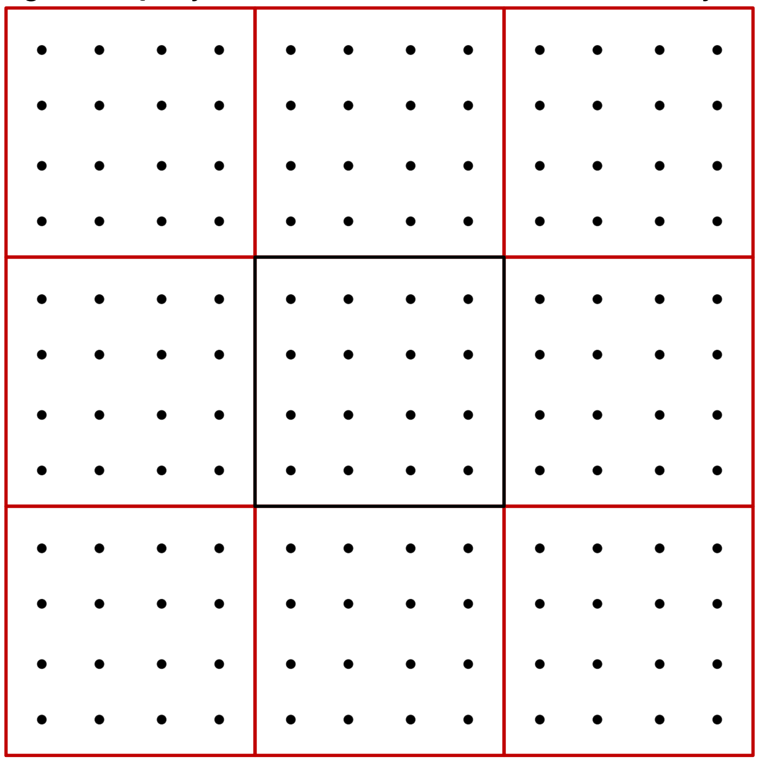
Figure 2: *Query crystal along with the adjacent crystals*

The naive implementation of finding all the atoms in a certain radius of the query atom, tries to find distance of every atom in the data set and then shows the atoms which are within a certain radius. Let n denote the total number of atoms in the extended data set and k the number of the query type atom. In this case the naive implementation requires O (kn) time.

1.3.3 Efficient solution to the problem

This problem is a very a well-known problem in computational geometry and it is known as the fixed radius neighbourhood problem. This can cause a serious problem for validating the structural simulated model as well as for generating input files for XAS simulation especially on the borders of the box. The problem has been skirted in two ways as follows:
The first method considers the symmetry of the starting crystalline model, with the size of the extrapolated atoms by 8 times instead of 27 times, by extrapolating only half of the total box on the edges and a quarter on the corners as shown in Figure 3.

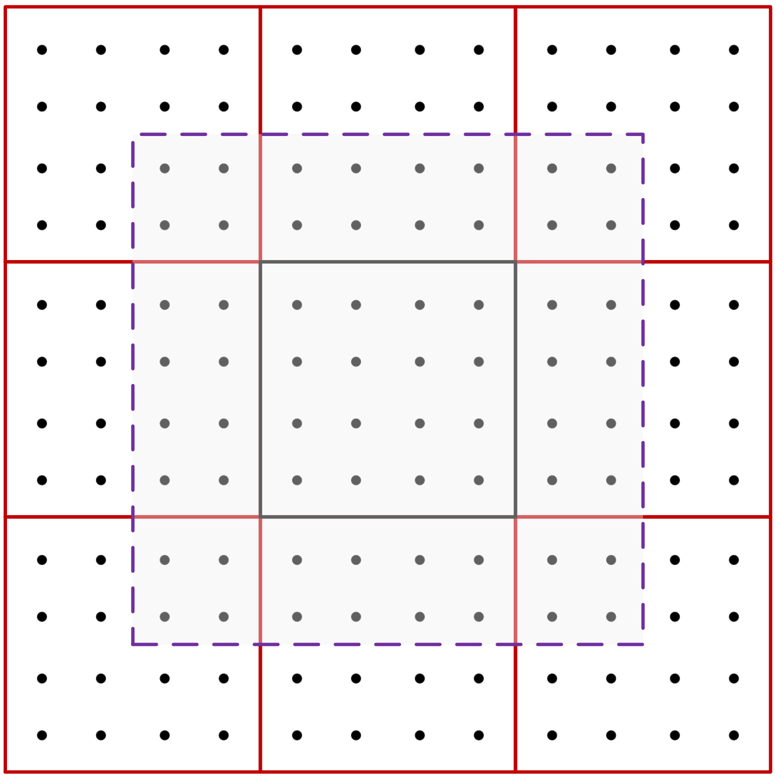

Figure 3: *Blue dotted line defines the new boundary*

Secondly, in computational geometry, the fixed radius nearest neighbourhood is solved by a very efficient data structure called k-d tree. In this implementation, the position of the atoms is read once into a k-d tree in linear time of the size (n) of the data. Then, the fixed radius nearest neighbourhood query can be answered in O(log n) time. In this way, the time for fixed radius nearest neighbourhood for all (k) the query atoms is O(n+k log n).

1.3.4 Implementation and Results

For fixed radius nearest neighbourhood, a library called ANN (Approximate Nearest Neighbour, http://www.cs.umd.edu/~mount/ANN) was used. This is an efficient implementation based on k-d tree data structure. Our implementation requires only 6 seconds to find fixed radius nearest neighbourhood problem for a total of 130,000 atoms, in which there are around 30,000 query atoms. It reads a file of 9MB as input and outputs around 10 MB at the same time. Earlier implementation used to take a few hours for the same job.

1.3.5 Installation

Source files for CoNBoVal-XAS can be retrieved from the following address using Git:

```
https://github.com/znawaz/CoNBoVal-XAS
```

To build the CoNBoVal-XAS, one needs to download the ANN (Approximate Nearest Neighbors) library from http://www.cs.umd.edu/~mount/ANN. Download the latest zip file and unzip in a directory. Move it to that directory.

Compile ANN with the following command:

```
make linux-g++
```

ANN_HOME then needs to be set in the environment variable. In Linux, this can be set as follows:

```
export ANN_HOME=[path to ann library]/ann_1.1.2
```

For example, on my own system it is

```
export ANN_HOME=/home/zubair/softwares/ann_1.1.2
```

If one wants to keep this environment variable permanently, copy the above command into the configuration file .bashrc.

To build the executable, type in from the root directory of CoNBoVal-XAS the following command

```
make
```

1.3.6 Running a Small Example

The program is invoked as follows

```
CoNBoVal-XAS [-i inputfile] [-d prefix] [-b bondValence]
```

where:

`inputfile` name of the input file containing atomic positions (default = "MD-data")

`prefix` prefix name of the output files containing distances (default = "feff")

`bondValence` suffix name of the output file that contains bond valence (default = "_bondVal.txt")

Results are stored in `prefixnnnnn.inp`, `prefix_bondValence.txt` and `prefix_feffscript.make` files, where `nnnnn` represents the line number of the query atom in the list and `bondValence` is the suffix name of the Bond Valence file as given in the argument. The file `prefixnnnnn.inp` is the input to `feff`. The file `prefix_bondValence.txt` includes the coordination number and the bond valence of the coordination atom. The last file `prefix_feffscript.make` contains the Linux bash commands to prepare the input file `prefixnnnnn.inp` for feff.

Example:

To run this demo use:

```
CoNBoVal-XAS -i MD-data -d feff -b bond-valence.txt
```

or

```
CoNBoVal-XAS
```

The command line help can be obtained with the following:

```
CoNBoVal-XAS --help
```

2 Conclusion

This specific deliverable which is focusing on related Synchrotron-light code development and mainly to produce three advanced applications (PyFAI, PyHST and CoNBoVal-XAS) has been successfully done by the workgroup (SESAME and ESRF). The first installation was done in-house on SESAME computing nodes which equipped with NVIDIA – K20 GPU, in the next step these codes were exported to Cy-Tera HPC resources. The development has been tested and validated successfully using public synchrotron raw data offered by ESRF as an operational Synchrotron-light lab. Today synchrotron applications are accessible on Cy-Tera with all required documentations (full guide, how-to and training material) to regional synchrotron scientists.

On the other hand, SESAME has achieved a success in identifying code problem on the X-ray Absorption Fine Structure (XAFS) and X-ray Fluorescence (XRF) beamlines software where a new code has been developed and now available with huge performance improvement – the code name is abbreviated as CoNBoValXAS (Coordination Number, Bound Valence and generates input files for XAS).

As result of LinkSCEEM Project and the strong partnership with the regional research user communities such as Synchrotron-light users namely SESAME and ESRF, LinkSCEEM resources today are able to execute a live demonstration on the synchrotron data analysis using LinkSCEEM and Cy-Tera platforms.