USE OF THE CELL ACCELERATOR PLATFORM FOR SYNCHROTRON DATA ANALYSIS

J. Qin, M. A. Bauer, N. S. McIntryre
The University of Western Ontario, WSC-143, UWO, London, ON. N6A 5B7, Canada.

Abstract

The analysis of synchrotron-based Polychromatic X-ray Microdiffraction (PXM) data has been used by scientists and engineers to understand elastic and plastic strains in materials on a micro or nano scale. Such experiments generate hundreds or thousands of images where the analysis of each image often entails intensive computations- a challenging task. As well, in the past, the speed of such computations has made it difficult to obtain feedback on the experimental results in near real time. This has constrained researchers from making critical decisions on direction subsequent experiments should take based on the results in hand. In order to improve the analysis performance of PXM images, we have investigated the use of parallel analysis schemes. This paper reports on the design and implementation of accelerated PXM analysis software. It has been developed on IBM PowerXCell 8i processors and Intel quad-core Xeon processors. A substantial improvement in processing speed has been obtained to the extent that it should be possible to obtain results at the same rate as they are produced on the VESPERS beamline at the Canadian Light Source (CLS) Synchrotron (~1 Hz).

INTRODUCTION

The development of high-energy PXM as a non-destructive method to determine elastic and plastic strains has been ongoing for the past decade [1-5]. The data generated in PXM experiments can consist of a large number of 2D digital images. Once these images have been generated from an experiment, ideally, it is expected that data can be processed at a same speed level as data is collected.

There are three major procedures involved in PXM data analysis, including peak searching, indexing and strain calculation. Briefly, *peak searching* attempts to extract useful information about intensity points (peaks) from an image to be used as input for the next two procedures. The *indexing* procedure takes the output from the *peak searching* procedure and generates the structural information about the sample material, e.g. the orientation of a crystalline lattice plane from which a diffraction spot is generated. Based on the indexing results and peak information, the strain analysis procedure then produces strain tensors in the sample. Based on the indexing results and strain tensor information, an orientation map and a strain map can be generated for the entire scanned area from which all PXM data were collected.

There are some existing software packages for PXM data analysis, such as the 3D X-ray Micro-diffraction Analysis Software Package at APS in Chicago which was developed at ORNL[6], and X-ray Micro-diffraction Analysis Software (XMAS) at ALS in Berkeley[7]. The common feature of these two packages is that they both are Windows-based software with a frontend interface implemented in Interactive Data Language (IDL) [8] and some backend procedures implemented in Fortran. Both can process a large amount of PXM data sequentially, i.e., step by step and one by one in sequence. This is a very time consuming process, and it usually takes days to finish processing a set of data collected from one PXM experiment. However, synchrotron time is valuable and it is often difficult to get a scheduled beam time. Data analysis using existing software means that researchers must complete the analysis following their time on the synchrotron. Faster analysis could help researchers make decisions on subsequent experiments during their synchrotron session and gain significant insight into the materials that they are studying.

In this paper, we introduce the development of an accelerated software for PXM data analysis, so called Fast Online X-ray Micro-diffraction Analysis Software (FOXMAS). It has been developed on a Cell accelerator platform comprised of Intel and IBM Cell processors. The software developed and the system it runs on makes it possible for PXM data to be processed in "near-real time", that is, nearly as fast as it is being produced. A description of the platform, the development approach, some performance evaluations, conclusions and future work are reported.

CELL ACCELERATOR PLATFORM

The target Cell accelerator platform, called *Prickly*, is one of the clusters in SHARCNET [9]. It is a heterogeneous High Performance Computing (HPC) system consisting of one head node for hosting user logins and a chassis with 12 Linux cluster blades providing total 160 computing cores. Among the 12 blades, four blades are Intel blades and the other eight are IBM Cell blades. On each of the Intel blades, there are *two* quad-core Xeon E5420 processors running at 2.5GHz with 8GB of memory. Each of the Cell blades contains *two* PowerXCell 8i processors, so called Cell processors, running at 3.2GHz with 16GB of memory. Blade interconnection is achieved through Gigabit Ethernet.

Unlike traditional multi-core processors which are homogenous, such as those on Intel blades, the Cell processor itself has heterogonous multi-cores [10]. It employs two types of cores optimized for different kind of tasks. Each Cell processor has nine cores, i.e. *one*

*Work is part of Science Studio project supported by CANARIE http://www.canarie.ca/

PowerPC Processor Element (PPE) and *eight* Synergistic Processor Elements (SPEs). The PPE is just a traditional 64-bit Power processor and acts as a large-scale processor core to run the operating system and performs control-intensive tasks. In contrast, the SPEs are much simpler, but devote more resources to perform computationally intensive tasks. Since each Cell blade has two Cell processors, in total, there are *sixteen* SPEs on each Cell blade. The sixteen SPEs are independent, 128-bit vector processors. Each SPE has its own local storage (256KB) for instructions and data. The SPE access to the memory is achieved through its Direct Memory Access (DMA) controller. The DMA can work concurrently with SPE executions, which hides the latency caused by memory accesses.

The Element Interconnect Bus (EIB) provides four 128-bit data transmission channels for the inter-communication among PPE, SPEs, main memory and I/O. It can support up to 307GB/s bandwidth between any two bus units. Therefore, with EIB, each SPE can not only work alone, but also be chained together to perform data processing with an intensive workload, such as stream processing.

While the Cell's special architecture offers many advantages for high performance computations, the architecture also makes programming on Cell more difficult.

DEVELOPMENT APPROACH

The goal of this development is to port the PXM data analysis software onto the target Cell accelerator platform to achieve an accelerated performance.

There are two major challenges involved in this porting process. First, the exiting software was written in IDL with some backend procedures written in Fortran. Our target Cell platform Prickly can only support programs mainly in C/C++. The software has to be rewritten into C in order to make it run on Cell.

Another challenge is to program on the Cell. To make use of all those advanced features provided by the Cell, especially the computation power provided by those SPEs, programming on Cell is a challenging. As each SPE has its own local store for holding instructions and input/output data, data needs to be moved back and forth between the local store and the main memory with explicit DMA commands. Because of the limited space (256k) for a local store on SPEs, only tasks that fit can be considered, otherwise, an advanced overlay management needs to be used.

There were two objectives in developing the PXM data analysis application. First, we wanted to create an implementation of the three major analysis procedures where the processing tasks were pipelined in order to accelerate the processing of a PXM image. Second, we wanted an implementation so that multiple PXM images could also be processed in parallel.

To further improve the processing speed on a single image, we want to identify the performance "bottleneck" of the entire process and then target an implementation on Cell around that "critical" part. Our measurements on a sequential version of the analysis code indicated that more than 80% of the processing time was spent in the peak searching procedure; therefore, it was initially targeted as the "critical" computation to be considered for porting to the Cell.

The peak searching procedure involves finding a threshold, blob searching, and curve fittings on each of the blobs. Among all three subtasks in peak searching, curve fitting is the most intensive one. During curve fitting on a blob, it applies two 1-D fittings (i.e., one for the X direction and one for the Y direction) and one 2-D fitting for a box area around each blob. Fig. 1 illustrates blobs identified in a PXM image with a certain intensity threshold. Each fitting process actually entails solving a multi-variable, non-linear least square minimization problem. It involves iterations to update the state of corresponding variables continually until certain criteria

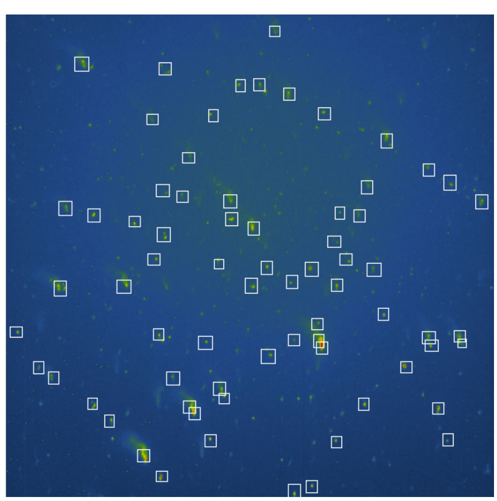

Figure 1: An PXM image with identified blobs that need curve fittings

are met. Specifically, the 1-D fitting involves solving four variables; these results become the initial states for the 2-D fitting. In turn, the 2-D fitting involves solving for six variables. The existing software carries out the curve fittings sequentially for each of the blobs in an image; this is very time consuming and becomes the bottleneck of the entire PXM data analysis.

Considering the computational power of a Cell's SPE, with a limited local store, it works well for a process with relatively small size but needs to run many times. Fortunately, the curve fitting is applied to each blob, which is in a relatively smaller area than the entire image area. The computation of the fitting process is also relatively intense and needs to be applied to every blob in an image. Therefore, the curve fitting process was selected as the processing task for the Cell's SPEs. After a collection of blobs has been identified, the fitting process can be done on the Cell's multiple SPEs in parallel, i.e. multiple blobs can be fitted by multiple SPEs simultaneously.

To analyze a large set of PXM images, we considered a computational approach involving processing multiple images in parallel and doing curve fitting on multiple blobs in parallel for each image during peak searching. The design of our parallel PXM data analysis program is illustrated in Fig. 2.

Using Fig. 2 as a guide the processing proceeds as follows. Initially, n images are loaded to be processed in parallel. Each of these images is initially processed on one of Cell's PPEs in order to identify blobs – potential regions having peaks. A list of blobs is produced for each image. Curve fitting is then done on each list of blobs on Cell's SPEs in parallel. The processing of the blobs from an image results in a list of possible peaks. The list of peaks is then passed to the indexing computation which results in index data and orientation maps based on index data. The index data is also used in the strain computation which produces a strain results and maps based on strain data. The indexing computation and strain computations are done on the Intel blades.

The design illustrated in Fig. 2 that has been implemented and deployed on Prickly, FOXMAS, has a web interface for job submission and online result visualization, that are not discussed in this paper. Because of the limited length, this paper only focused on the development of the accelerated data analysis.

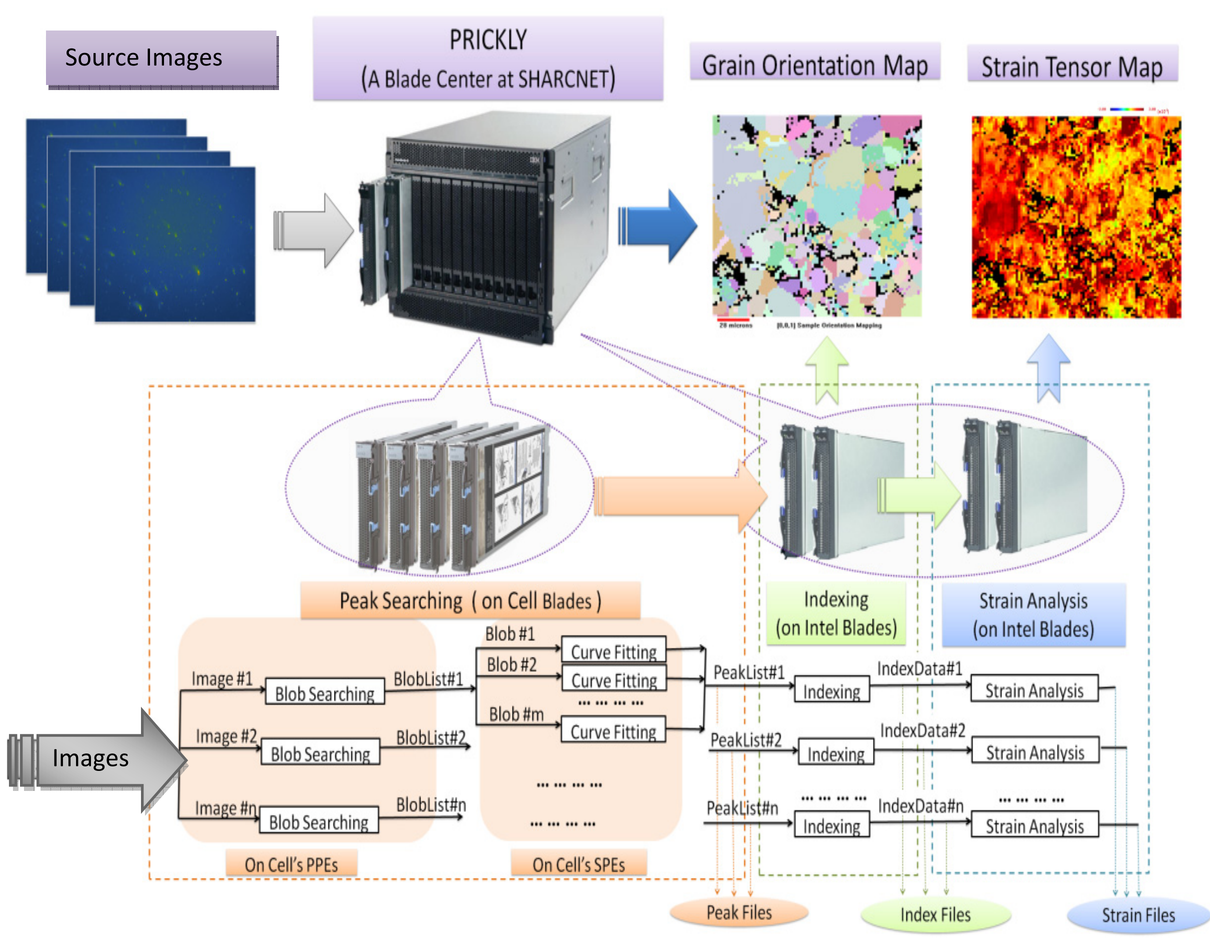

Figure 2: The configuration of high performance PXM data analysis on Prickly

PERFORMANCE EVALUATION

We measured the time needed for processing different sets of images sequentially with the original IDL software. We also measured the performance of FOXMAS, i.e., the average processing speed with various settings, including the number of parallel pipelines and number of SPEs used on each image. The speedup of each test case is compared to the speed of the IDL software.

PXM images could have different sizes depending on the type and setting of a CCD detector. Larger size images tend to provide more information with a trade-off of a heavier analysis workload. In this evaluation, we examined two different image sizes. One set of images were collected from APS, each of which has 1042X1042 pixels and is about 2MB/image. Another set of images were from CLS, each of which has 2084X2084 pixels and is about 8MB/image. Using the original IDL software on a desktop machine, the average processing speed for APS images is about *4.31* sec./image and for CLS images is about *14.36* sec./image.

Prickly has total of 4 Intel blades and 8 Cell blades, we examined the performance of data analysis on *one* pair of Cell-Intel blades and on *multiple* pairs of Cell-Intel blades. As described in Fig. 2, peak searching is done on the Cell blade; while indexing and strain analysis are done on the Intel blade. Since each Cell blade has a total of 16 SPEs, if n images are processed in parallel, i.e. n pipelines, and m SPEs are allocated for each image or each pipeline, m and n are constrained to be values such that $m \times n = 16$. Different combinations of m and n were tested. For the workload on the Intel blade, if n pipelines are initiated for processing n images in parallel, n processes are created on the Intel blade, and each of these n processes works on the indexing and strain analysis on one of n images. The operating system takes care of workload distribution among the *eight* cores on the Intel blade. The speedup of each test case is compared to the desktop speed using IDL software. Tables 1 and 2 present the measured results on one pair of computation nodes on Prickly.

Table 1: Results of processing APS images on *one* pair of Cell-Intel nodes on Prickly

Images in parallel (pipelines)	**Number of SPEs for each image**	**Average Speed (sec./image)**	**Speedup vs. IDL (times)**
1	16	0.63	6.84
2	8	0.43	10.02
4	4	0.35	12.31
8	2	0.26	16.58
16	1	0.22	19.59

The results presented in Table 1 illustrate that for processing images of the size of the APS images, the more images that are processed in parallel, the better throughput, i.e. processing 16 images resulted in average speed 0.22 sec./ image and speedup of 19.59 times compared to IDL software.

Table 2: Results of processing CLS images on *one* pair of Cell-Intel nodes on Prickly

Images in parallel (pipelines)	**Number of SPEs for each image**	**Average speed (sec./image)**	**Speedup vs. IDL (times)**
1	16	2.84	5.06
2	8	1.81	7.93
4	4	1.55	9.26
8	2	1.67	8.60
16	1	1.68	8.55

In processing the larger size images, i.e. those from CLS, the results of Table 2 suggest that processing 4 images in parallel and with 4 SPEs allocated for each image can produce the best throughput, i.e. an average speed 1.55 sec./image and about 9.26 times of speedup compared to IDL software. The processing of multiple images in parallel was achieved through multi-process programming, while the parallel blob fitting on the Cell was achieved through multi-threaded programming. In general, process creation cost is much larger than thread creation cost. The overall performance gain of a parallel application is dependent on balancing the computational workload and the trade-off in setting up the parallel processing elements. The result of reducing the number of SPEs allocated for each image, i.e., to 2 or 1 in Table 2, in lieu of having more parallel pipelines to process more images in parallel is not sufficient to overcome the blob processing done on each of the larger images within the SPEs. Consequently, using 4 pipelines and 4 SPEs results in the best performance for images of this size.

Table 3: Results of processing APS images on *multiple* pairs of Cell-Intel nodes on Prickly

Pair(s) of Cell-Intel nodes	**Images in parallel (pipelines)**	**Average speed (sec./image)**	**Speedup vs. IDL (times)**
1	16	0.22	19.59
2	32	0.14	30.78
3	48	0.07	61.57
4	64	0.07	61.57

To examine the performance of using *multiple* pairs of Cell-Intel nodes on Prickly, based on the results presented in Tables 1 and 2, *1* SPE per APS image and *4* SPEs per CLS image were used. Tables 3 and 4 present the results. The results illustrate that the performance of PXM data analysis has been boosted significantly when more computational resources are used. For the smaller sized images collected at APS, as presented in Table 3, when 48 processing pipelines were setup on *three* pairs of Cell-Intel blades, the average processing speed can reach as high as 0.07 sec./image, which is 61.57 times speedup compared with 4.31sec./image of using existing IDL software on a desktop machine. For larger size images collected at CLS, when 16 processing pipelines were setup on *four* pairs of Cell-Intel blades, the average processing speed (see Table 4) can reach as high as 0.59 sec./image, which is 24.34 times speedup compared with 14.36 seconds/image when using the IDL software on a desktop machine.

Table 4: Results of processing CLS images on *multiple* pairs of Cell-Intel nodes on Prickly

Pair(s) of Cell-Intel nodes	Images in parallel (pipelines)	Average speed (sec./image)	Speedup vs. IDL (times)
1	4	1.55	9.26
2	8	0.96	14.96
3	12	0.70	20.51
4	16	0.59	24.34

Notably, the measured *speedup* from both sets of experiments presented in Tables 3 and 4 do not result in a linear improvement as more nodes are added. One of the factors affecting the speedup is the increased overhead of in setting up more pipelines exchanging information across the two types of nodes on Prickly. Another factor that affects achieving a linear speedup is the data transfer and communication cost of the Gigabit Ethernet; as more processes are added there is an increase in communication.

The goal of this project is to make use of such an accelerated data analysis for a synchrotron beamline to achieve a real time experiment and data analysis. The Cell platform Prickly is located at The University of Western Ontario in London Ontario. A synchrotron beamline, such as CLS is located in Saskatoon Saskatchewan. To achieve a real time PXM experiment and data analysis, data collected at CLS needs to be transferred to UWO in an ultra high speed. CANARIE's cross country lightpath network can provide such an ultra high speed data transmission. By using CANARIE's dedicated lightpath we are able to complete such a scenario.

A preliminary functional test has been measured for such a scenario. It included a procedure of sending a set of total 100 PXM images (about 8MB/image) from CLS to UWO, then getting processed on SHARCNET's Prickly at UWO, and presenting final results at an FTP site for users to download. It only took around 4 min. to complete the entire procedure. In specific, it took about 2 min. for data transmission from CLS server to UWO server through the lightpath, and less than 1 min. for data transmission from UWO server to SHARCNET's Prickly through UWO's intro-network. It only took about 1 min. to finish the data analysis on Prickly and send the analysis results back to UWO server for users to download. Even thought there are still rooms for refinement, the performance is quite promising for a real time experiment.

CONCLUSION AND FUTURE WORK

In this paper we reported the development of an accelerated PXM data analysis, FOXMAS, on a Cell accelerator platform, i.e. the cluster Prickly on SHARCNET. Using the computation power of Prickly, especially the Cell processors, FOXMAS can achieve up to 60 times faster than a desktop performance of using original IDL software package, depending on the size of images and the number of computation nodes used on Prickly. Combined with CANARIE's dedicated lightpath for data transmission, the promising performance makes it possible to process the data at the same high rate as it is produced at the synchrotron (CLS).

Future work for our next step is to implement the function of data transmission/analysis at the same time as it has been collected *during* a synchrotron experiment, i.e. a real time data collection and analysis. This is currently underway using the VESPERS beamline at the CLS. Such a model could also be adapted to other synchrotron and HPC facilities.

ACKNOWLEDGEMENTS

FOXMAS was developed based on source code of IDL packages from APS [6] and XMAS from ALS [7]. We are grateful for the useful document from Dr. Nobumichi Tamura of ALS in helping us to understand the analysis procedures involved in PXM data analysis. Thanks to Dr. M.L. Suominen Fuller and Ph.D. student Jing Chao at UWO for their great help on the validation of results produced by FOXMAS. Thanks to Dong Liu at CLS for his collaborated work in the measurement of data transportation from CLS to UWO through the dedicated lightpath.

REFERENCES

[1] J.S. Chung and G. E. Ice, *Journal of Applied Physics*, 86 (1999) 5249-5255

[2] G. E. Ice and B. C. Larson, *Advanced Engineering Materials,* 2(2000) 643-646

[3] B. C. Larson, W. Yang, G. E. Ice, J. D. Budai and J. Z. Tischler, *Nature,* 415(2002) 887-890.

[4] M. L. Suominen Fuller, R. J. Klassen, N. S. McIntyre, A. R. Gerson, S. Ramamurthy, P. J. King and W. Liu, *Journal of Nuclear Materials*,374 (2008) 482-487

[5] J. Chao, A. Mark, M.L. Suominen Fuller, N. S. McIntyre, R. A. Holt, R.J. Klassen and W. Liu, *Material Science and Engineering,* A 524 (2009) 20-27

[6] PXM data analysis at APS : http://www.aps.anl.gov/Sectors/33_34/microdiff/downloads/

[7] PXM data analysis at ALS: http://xraysweb.lbl.gov/microdif/user_resources.htm

[8] K. P. Bowman, *An Introduciton to Programming with IDL: Interactive Data Language*, Elsevier Inc. 2006

[9] SHARCNET website: https://www.sharcnet.ca/

[10] M. Scarpino, *Programming the Cell Processor: For Games, Graphics, and Computation,* Printice Hall, 2008