18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Multicore-Based High Performance Image Analysis for Batch Processing in Drug Discovery

Wang, D.D. [1], **D. Bourke** [1], **L. Domanski** [1] **and P. Vallotton** [1]

[1] *Biotech Imaging, CSIRO Mathematical and Information Sciences*
Locked Bag 17, North Ryde, NSW 1670, Australia
Email: dadong.wang@csiro.au

Abstract: Automated image analysis allows drug companies to use High Content Screening (HCS) to measure subtle but important changes in fluorescently labeled cells, both rapidly and accurately. Researchers using HCS imaging systems generate thousands of images from which they measure millions of parameters. Some images can be very "dense", with hundreds of cells and complicated cell morphology. It may take several hours or even days to process the images generated from a single experiment which may be unacceptable for the workflows in laboratories. The large image datasets and fast turnaround requirement has made the efficient batch processing of the massive amount of images a challenging task.

With the enormous progress in computing power, computers with multi-core CPUs are now becoming standard. The multi-core shift presents unprecedented opportunities for researchers to deal with large datasets with acceptable computing performance. Unfortunately, having a dual core CPU does not speedup one's application automatically. In fact, most applications use just a single core and see no speed improvements when run on a multi-core machine. It is still a challenge to develop parallel algorithms that address the issues of task concurrency and data parallelism, and that actually take advantage of those multiple processors. In this paper, some efforts have been made to develop a multi-core based high performance solution to speedup the batch processing for micro-well-plate-based HCS.

The proposed solution employs an automatic parallelization engine which automatically dispatches the batch processing tasks. The engine automatically detects the number of processors available on the computer on which our High Content Analysis (HCA) software runs, and then creates equal number of work threads to process images in parallel. Therefore, the solution can automatically scale to additional cores and future multi-core processors. Loop parallelization mechanism has been implemented to assign different images to different processors so that multiple images are processed at the same time. Upon completion of image processing on a processor, a new image will be automatically assigned to the processor. This process continues until all images are processed. For individual images, the processing is sequential and highly data dependent. This one-image-per-processor protocol can simplify the parallelization for data dependent computation problems and minimize the development effort in migrating sequential image analysis algorithms to a parallel form. Flow control is employed in the parallel batch processing to coordinate the processors' access to shared resources such as database and Graphical User Interface (GUI) components. At the end of individual image processing, all features extracted from the image are piped into a structured database for more sophisticated data analysis. To improve the database operation performance, some database manipulations, such as multiple data records insertion, have been optimized to maximize the batch processing throughput. To verify the proposed solution, a full plate of images, with 96 wells and 6 images per well, have been screened. The experimental results are validated and evaluated by comparing the performance of the proposed approach with the conventional batch processing. With the proposed approach on a quad-core machine, the batch processing time for neuron body detection has been reduced to 38% of the original, and 46% for neurite analysis. The results show that the proposed solution can significantly increase the throughput of batch processing, improve the workflow in HCS laboratories, and make a difference in terms of cost and quality in drug development. The proposed solution can also be used in other data and compute-intensive applications.

Keywords: *High Performance Computing, Data Intensive Computing, Parallel Processing, Image Analysis, Batch processing, High Content Screening*

1. INTRODUCTION

HCA is emerging as one of the fastest growing sectors in drug discovery and development. It represents the convergence between cell-based assays, high-resolution imaging, and advanced image processing and analysis (Liszewski, 2008). HCA systems achieve high throughput by rapidly capturing and processing the images of each well from entire micro-well plates. Each well in these plates contains cells which have been labeled to detect changes in morphology produced by some interaction, such as addition of a candidate drug compound or gene knockout experiments. The ideal image processing time should be within the same time frame as the image capturing. However, the image processing time may be much longer than industry's expectations even on a high-end computer. Therefore, high throughput image processing becomes a challenging task due to the data intensive computing and the fast turnaround required.

High Performance Computing (HPC) comes from parallelism, fast-dense circuitry, and packaging technology (Bell and Gray, 2001). Over the last decade, several research studies have been conducted in the application of HPC to image processing (e.g. Kikinis et al., 1998; Beynon et al., 2001; Cambazoglu et al, 2007; Rao et al., 2007). Most of this research focuses on the usage of HPC infrastructure or distributed processing in an application driven and research environment. The solutions presented in these studies are based on supercomputers and computer clusters. In general, the parallel image analysis algorithms have not been sufficiently developed and investigated in a HPC context (Rao et al., 2007). As most of HCA systems are not based on supercomputing facilities, studies on high performance HCA solutions based on multi-core computers are more practical, and have potential to make a difference in terms of cost and quality in drug discovery.

With the low-cost commercial HPC components becoming more common such as multi-core CPUs and GPGPUs (General-Purpose computation on Graphics Processing Units), nearly every computer on the market has a processor of at least two cores. Multi-core processors offer opportunities for tremendous speed improvement. However, a dual-core processor does not double a computer's performance. This is because few applications take full advantage of this new technology and see performance gains when run on a multi-core computer. The many-core shift has triggered some efforts towards developing parallel image analysis algorithms that take advantage of these powerful processors to achieve fast execution. Trease et al introduced a high-performance hybrid multi-core processing framework for processing videos and images (Trease et al., 2008). Hartley et al illustrated a cooperative parallelization approach when multiple CPU sockets, multiple GPUs and multiple cluster nodes coexist (Hartley et al., 2008). Literature reviews have revealed that few cases have been reported in the past that explore the multicore-based HCA systems for drug discovery. In this paper, we will present an enabling solution to show how to make the most of one's multi-core computer to achieve high performance in high throughput image processing in HCS.

The rest of the paper is organized as follows: Section 2 introduces the data and computation intensive problem in HCA. In Section 3, we present the proposed high performance solutions focusing on its structure and implementation. The experimental results are demonstrated in Section 4, followed by discussions on the limitations and conditions of the proposed solution and conclusions.

2. CHALLENGE IN HIGH CONTENT IMAGE ANALYSIS

Researchers attempt to make sense of massive amount of image data through HCS. For example, to identify compounds effect on neurite outgrowth, several micro-well-plate experiments need to be carried out. Eventually, hundreds of thousands of microscope images are generated. Well- and even cell-based features are extracted from the images using HCA software packages. As some dense images may have over 1000 cells with complicated cell structures, processing massive amounts of such images in a timely manner becomes difficult. This section will use neurite outgrowth analysis as an example to provide some background information on the challenge.

2.1. Large Image Datasets

When screening for neurite outgrowth, multiple plates of images may be produced. Each micro-well plate can have 384 wells, and 12 images may be sampled from each well. This will produce 4608 images from each plate. If 5 plates of images are generated, there will be 23040 images to be processed. If each image has three channels (RGB), with one channel showing the labeled nuclei, one showing the labeled neurons and neurites, and one showing additional information such as labeled proteins in some sub-cellular compartments, the number of images to be processed will be 69120. A typical image has a dimension of 1280 x 1280 pixels. It turns out that the HCS for neurite outgrowth analysis is a data intensive computation process.

2.2. Compute-Intensive Analysis

In neurite analysis, some images may be very "dense", containing over 1000 neurons. As shown in Figure 1, many neurons may be clumped together, and some neurons may have complicated neurite structures. This will increase the complexity of analysis. From Figure 1, one can see that many neurite branches overlap. Generating neurite branch-level features and assigning the correct neurites to the corresponding neurons becomes a challenging task.

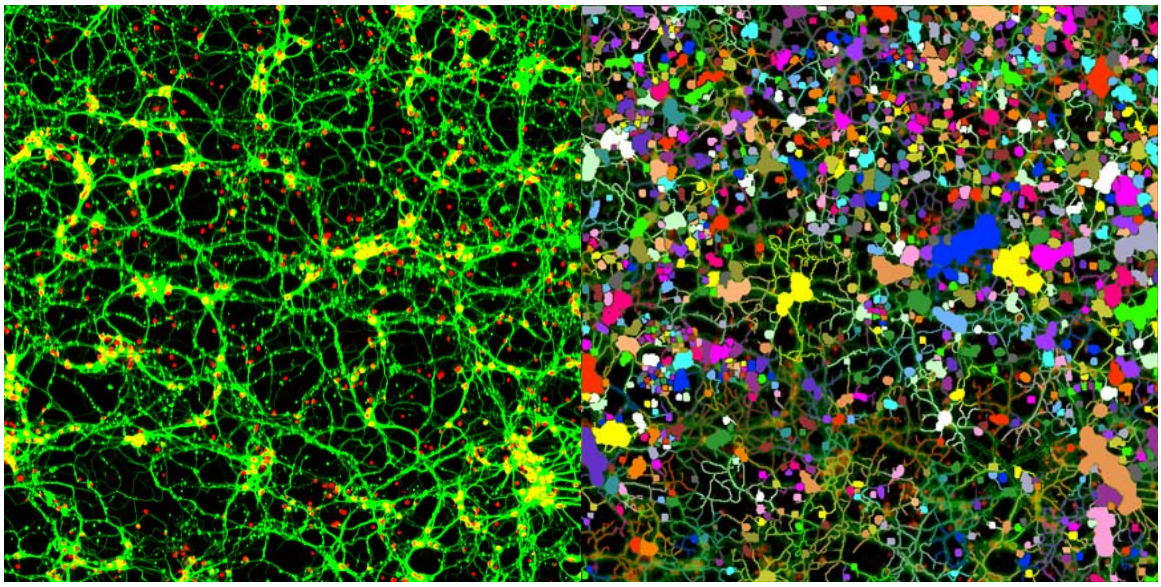

Figure 1. A typical dense image with many "clumped" neurons and overlapped neurite structures. Left image acquired with IN Cell Analyzer 3000 shows a high degree of neurite branching complexity. Images courtesy of Marjo Simonen, Novartis Institutes for BioMedical Research. Right image displays the segmented neurons and neurites using CSIRO HCA-Vision software package.

To analyse neurite outgrowth, the following measurements are required:

- Cell based measurements. The cell-based measurement includes 34 parameters such as cell area and perimeter, maximum and mean intensity, total neurite length, max neurite length, max and mean branch layer, number of branch points, number of roots, number of segments, number of extremities, neurite field area, neurite area, max and mean intensity of neurite etc.
- Image-wide summary statistics. This includes the number of cells in the image, total and average neurite length, total and average number of segments, average longest neurite from a cell, total and average number of roots, total and average number of extreme neurite, total and average number of branch points, average branching layers etc.
- Well-based summary for each plate. The plate summary is comprised of normalized features for each well. All features extracted from the images sampled from the same well are averaged to produce the well-based normalized statistics.

The above image analysis demands considerable computation. Therefore, batch processing thousands of images is a computation intensive task, taking hours even days for a single experiment. The following section will address how to speedup the batch processing with multi-core based high performance image computing.

3. HIGH PERFROMANCE IMAGE COMPUTING

Multi-core computers have a CPU with multiple cores combining two or more independent processors into a single package composed of a single integrated circuit (IC). However, disposing of two processors does not speedup one's application automatically. According to Amdahl's law (Lewis et al., 1992), the amount of performance gain from using a multi-core processor depends on the problem being solved and the algorithms used, as well as how they are implemented in software. Most application software packages rely on only a single core and see very limited speed improvements when run on a multi-core machine. This is because they

have not been designed to take advantage of the available parallelism. In this section, a multi-core based high performance solution for HCA will be addressed. It can enable the conventional HCA software to process images in parallel to achieve significant performance gain.

3.1. Sequential Neurite Analysis Procedure for a Single Image

As shown in Figure 2, the sequential neurite analysis involves three steps: (1) neuron body detection, (2) neurite detection, and (3) neurite analysis.

The neuron body detection aims at identifying and marking the neuron bodies. This includes smoothing the raw image to distribute the intensity more evenly within the neuron bodies; background correction to remove global trends in the background intensity; suppressing small structures such as neurites; intensity thresholding to detect neuron bodies; detecting nuclei to be used as masks to split touching neuron bodies; filtering neuron bodies when mixed cell types exist or applying certain cell selection criteria; producing neuron body detection label image and cell based measurements.

The neurite detection aims at detecting neurite structures (Sun et al., 2006). The detection procedure includes image smoothing to remove noise within the image; linear feature detection to segment neurite structures; removing small objects which are not of interest; closing gaps between detected neurite endpoints; and generating neurite skeleton image.

The neurite analysis is to trace neurites and to associate neurites with the corresponding neurons. The tracing consists of debarbing small neurites; thickening neuron bodies to connect neighboring neurites which would otherwise be disconnected from the neuron bodies and be removed as orphan neurites; removing small neurite trees which are not of interest; neurite tracing to generate tree statistics such as branching layers, primary, secondary and tertiary layer structures; and producing all tree analysis results.

Upon completion of neurite analysis for an image, all measurements are piped into a structured database for further investigation. The whole procedure, from loading the input image to saving results in the database, is highly sequential. In the processing sequence, the input image for each processing step is the result image of its previous step; and the result image of the process step is the input image of the next processing step. The following section will discuss how to speedup this sequential processing.

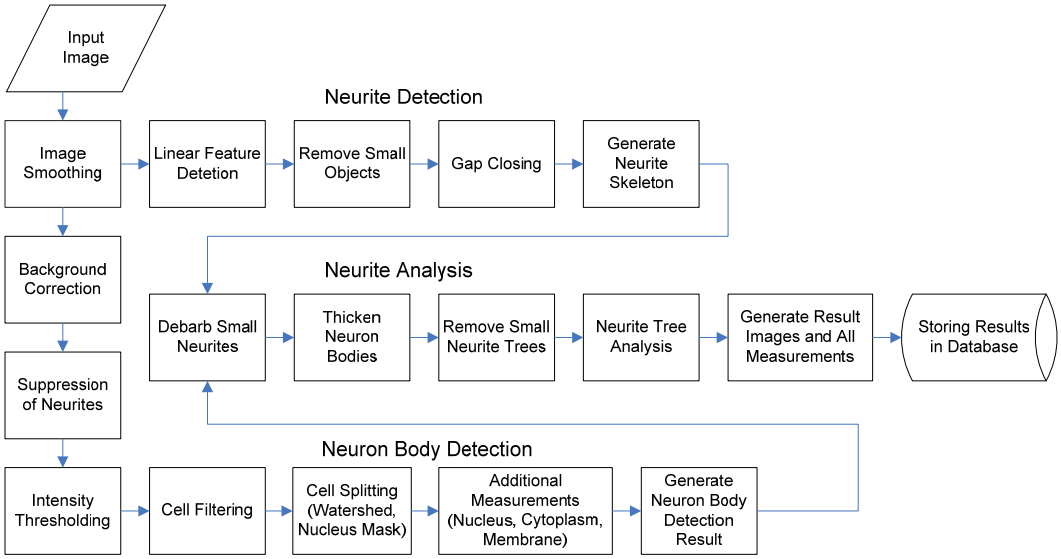

Figure 2. Sequential procedure of neurite analysis for a single image

3.2. Flowchart of Automatic Parallel Batch Processing

When batch processing neurite images, the sequential procedure outlined in Section 3.1 is repeated until all images have been processed. As the neurite analysis is highly data dependent, it is not an easy matter to parallelize the processing for each image across multi-core processors. However, if one image is assigned to one processor at a time, multiple images are assigned to multi-cores; the parallelization problem is

significantly simplified. Figure 3 uses a 4-core computer as an example to illustrate the flowchart of the proposed automated parallel batch processing.

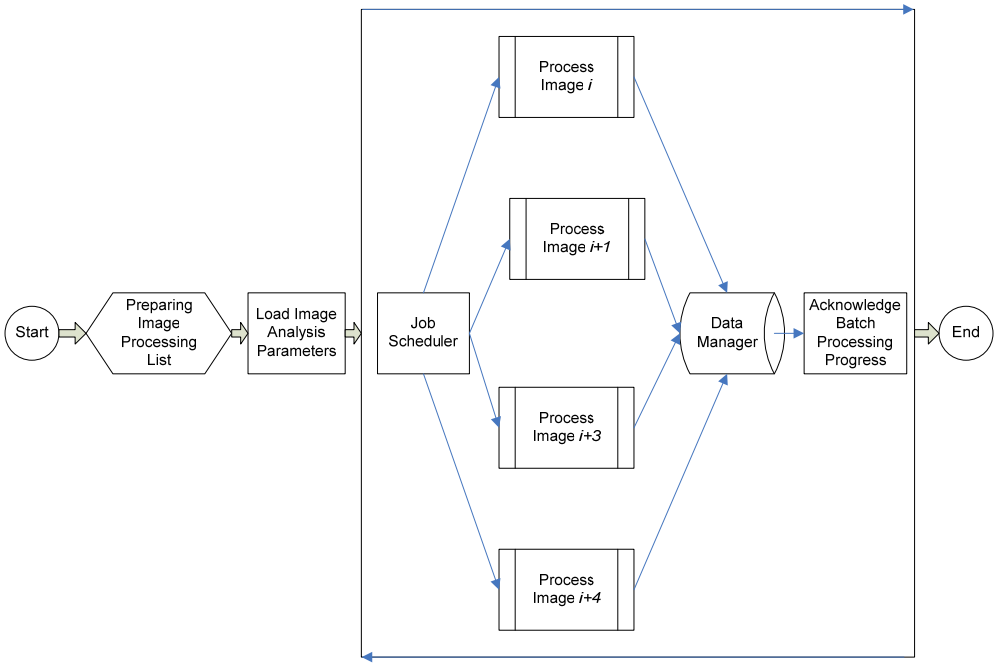

Figure 3. Flowchart of the Proposed Parallel batch processing

To implement the automatic parallelization, Microsoft .Net Parallel Extension is employed. It is a managed programming model for data parallelism, task parallelism, and coordination on parallel hardware unified by a job scheduler (Microsoft Parallel Computing Developer Center website, 2009). The parallel extension enables software developers to build multi-core capable applications using existing code and compilers. It provides library based support for introducing concurrency into applications. Task Parallel Library (TPL) provides imperative data and task parallelism; Coordinate Data Structure (CDS) contains lightweight and scalable thread-safe data structures and synchronization primitives, providing support for work coordination and managing shared state. The job scheduler is a robust, efficient, and scalable engine designed to use co-operative scheduling, and work-stealing algorithms to achieve fast, efficient, and maximum CPU utilization.

The Parallel Extension has a task manager that, by default, uses one worker thread per processor. Each worker thread has its own local task queue. Each worker usually just pushes new tasks onto its own queue and pops work whenever a task is done. To scale well on multiple processors, work-stealing techniques are applied to dynamically adapt and distribute work items over the worker threads (Blumofe et al., 1994). When a worker's local queue is empty, it looks for work itself and tries to "steal" work from the queues of other workers (Microsoft Technology Blogs: Parallel Programming with .Net, 2009). This procedure continues until all images are processed. With the parallelized batch processing, individual images are assigned to different processors automatically. Figure 3 shows the image assignment, data management, and the batch processing loops. Individual images are processed on different processors in parallel. The time spent in processing an image on individual processors depends on the complexity of individual images. For each image, the analysis results are piped into the database. As multi-processors can not access the database at the same time, flow control is incorporated in the parallel processing. When a processor is accessing the database, a "lock" is obtained and released when the database operation is completed. To minimize the wait time, the database operation is optimized to reduce the number of locks. This is addressed in the following section.

3.3. Optimization of Database Operations

When parallelizing the batch processing, we identified that the database manipulation represented a bottleneck. This is because multiple threads cannot access the same database at the same time. This can be resolved by applying a "lock" mechanism. Another issue is the optimization of database operations. From Section 2.2, we can see that some images have over 1000 neurons. At the end of the processing of each image, all cell-based measurements are grouped into cell-based records to be saved into the database. Each

record has 34 fields. Inserting cell-based records individually is time consuming. We have conducted some experiments on different computers to optimize the database operation performance. The computers include two high end computers, Dell T7400 with 4-cores and 4GB RAM, and Dell Xeon computer with 4 processors and 8GB RAM. The experimental results show that inserting different numbers of cell based records using one insert statement takes different amounts of time. The database inserting time is optimal for the two high end computers when saving about 20 records per insert statement. This is due to the fact that database connection, sending and parsing a query takes 5 - 7 times of the actual data insertion, depending on the row size.

4. EXPERIMENTS

We conducted experiments to evaluate the performance of the parallel batch processing on a high end computer with 4 Intel® Xeon™ 3.2Ghz processors and 8GB of RAM. Both neuron body detection and neurite analysis were tested for a 96-well plate of images with 6 images per well, altogether, 396 images. These images have a dimension of 1280 x 1280, and two channels, the first being the neuron body and neurite channel, and the other, the nucleus channel.

The test was conducted three times for both sequential and parallel processing. With the proposed parallel batch processing, significant performance improvement was achieved. Overall, the execution time of the parallel batch processing has been reduced to 38% of the original sequential batch processing for neuron body detection, and 46% for neurite analysis. Figure 4

shows the performance comparison of the sequential and parallel batch processing for neuron body detection and neurite analysis, respectively. The time difference among three different executions for parallel and sequential batch processing may be caused by other OS tasks running on the computer.

Figure 5 illustrates the CPU usage comparison between the sequential and parallel batch processing for neurite analysis. When batch processing images in sequential mode, the average CPU usage is about 25% of its full execution power, however, 100% usage is achieved with the proposed multi-core based parallel batch processing.

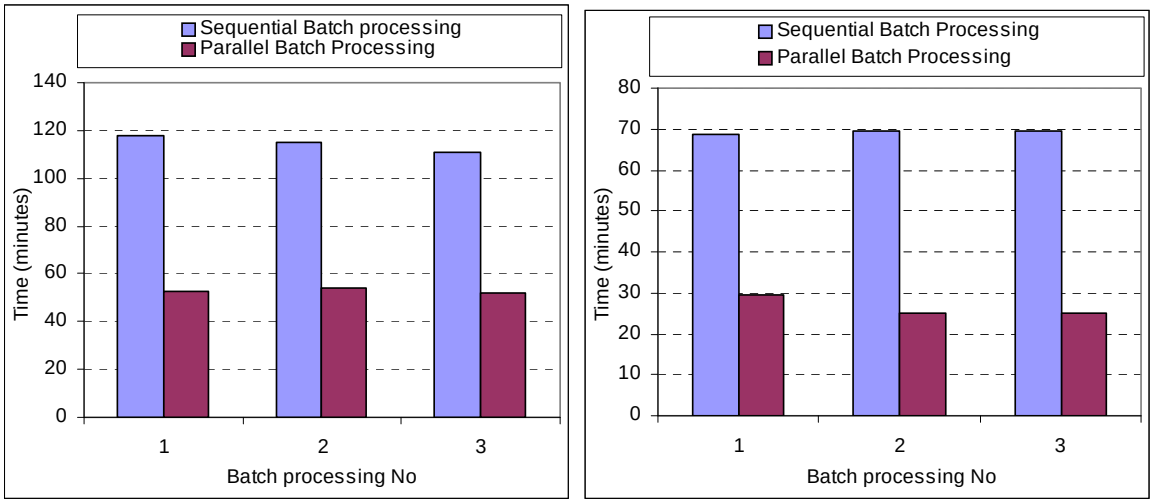

Figure 4. Execution time comparison of sequential and parallel batch processing for neuron body detection (left) and neurite analysis (right)

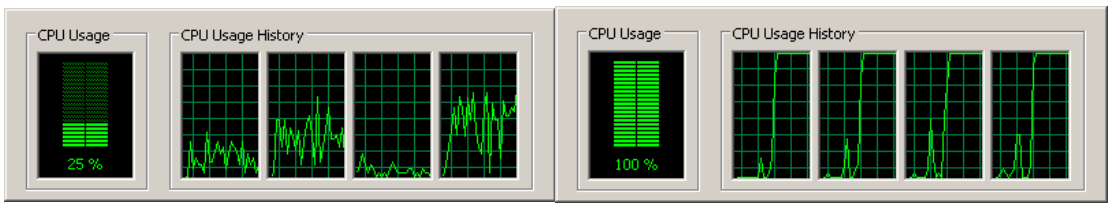

Figure 5. CPU usage charts. Left – CPU usage with sequential batch processing approach; Right – CPU usage with the proposed multi-core based parallel batch processing.

5. DISCUSSION AND CONCLUSIONS

In this paper, we have presented a multi-core based batch processing solution which can significantly improve performance in HCS applications. The solution can automatically scale to additional cores and future multi-core processors. By identifying the bottleneck of the batch processing and implementing a parallel image analysis procedure, we have established a solid and efficient HCS platform. The experimental results show considerable speedup with our proposed solution, compared to the conventional sequential approach. The increased throughput allows biologists to conduct large scale high content screening experiments with massive datasets and within a reduced time frame.

The multi-core based solution has some limitations and conditions. First of all, all image processing routines have to be made thread safe. No global or static variables can be used in the routines, which depend only on the arguments passed in. No logical and data dependence is allowed among different work threads running on different cores. A "Lock" mechanism shall be applied when accessing shared resources such as file I/O, GUI components and databases. The amount of speedup achieved depends on how many cores are available, but is not strictly proportional to the number of cores.

The proposed solution can be applied to other data and compute-intensive applications as well. This can bring high performance to a single desktop computer and has the potential to make significant difference in the cost and quality of scientific computations and simulations.

ACKNOWLEDGMENTS

The authors would like to thank Marjo Simonen, Novartis Institutes for BioMedical Research, for permission to use the neurite outgrowth images presented in this paper.

REFERENCES

Bell, G., and Gray, J. (2001), High Performance Computing: Crays, Clusters, and Centers. What Next? *Technical Report*, Microsoft Research, August 2001.

Beynon, A., Catalyurek, K., Chang, C., Sussman, A., and Saltz, A. (2001), Distributed Processing of Very Large Datasets with DataCutter. *Parallel Computing*, 27(11): 1457-1478.

Blumofe, R., Leiserson, C. E. (1994), Scheduling Multithreaded Computations by Work Stealing, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 356-368.

Cambazoglu, B. B., Sertel, O., Kong, J., Saltz, J., Gurcan, M. N., and Catalyurek, U. V. (2007), Efficient Processing of Pathological Images using the Grid: Computer-Aided Prognosis on Neuronblastoma. *Proceedings of International Workshop on Challenges of Large Applications in Distributed Environments*, 35-41.

CSIRO HCA-Vision website: http://www.hca-vision.com.

Hartley, T., Catalyurek, U., and Ruiz, A. (2008), Biomedical Image Analysis on a Cooperative Cluster of GPUs and Multicores. *Proceedings of the 22nd annual international conference on Supercomputing*, 15–25.

Kikinis, R., Warfield, S., Westin, C. (1998), High Performance Computing in Medical Image Analysis at the Surgical Planning Laboratory, *High Performance Computing Asia'98*, 290-297.

Lewis, T. G., and El-Rewini, H. (1992), Introduction to Parallel Computing, *Prentice-Hall*, Inc. pp 31-32, 38-39.

Liszewski, K. (2008), High-Content Discovery Analysis, Genetic Engineering & Biotechnology News, 28(1).

Microsoft Parallel Computing Developer Center website: http://msdn.microsoft.com/en-us/concurrency/default.aspx.

Microsoft Technology Blogs*: Parallel Programming with .Net*, http://blogs.msdn.com/pfxteam/.

Rao, A., Cecchi, G., and Magnasco, M. (2007), High performance computing environment for multidimensional image analysis. *BMC Cell Biology*, 8, 1471-2121.

Sun, C., and Vallotton, P. (2006), Fast Linear Feature Detection Using Multiple Directional Non-Maximum Suppression, *Proceedings of the 18th International Conference on Pattern Recognition*, Volume 02, 288 – 291.

Trease, H., Farber, R., Wynne, A. and Trease L. (2008), High-Performance Video Content Analysis using Hybrid, Multi-Core Processors, *2008 IASTED Proceedings of Signal and Image Processing*, 632.

Vallotton, P., Lagerstrom, R., Sun, C., Buckley, M., Wang, D., Silva, M., Tan, S., and Gunnersen, J. (2008), Automated analysis of neurite branching in cultured cortical neurons using HCA-Vision, *Cytometry Part A*, Volume 71A Issue 10, 889 – 895.