Radiation Protection Dosimetry (2014), pp. 1–10 doi:10.1093/rpd/ncu133

AUTOMATING DICENTRIC CHROMOSOME DETECTION FROM CYTOGENETIC BIODOSIMETRY DATA

Peter K. Rogan[1,*], Yanxin Li[1], Asanka Wickramasinghe[1], Akila Subasinghe[1], Natasha Caminsky[1], Wahab Khan[1], Jagath Samarabandu[1], Ruth Wilkins[2], Farrah Flegal[3] and Joan H. Knoll[1]
[1]University of Western Ontario, 1151 Richmond Street, London, ON, Canada N6A 3K7
[2]Health Canada, 775 Brookfield Road, PL 6303B, Ottawa, ON, Canada K1A 1C1
[3]Atomic Energy of Canada Ltd., STN 51, Bldg 513, Chalk River, ON, Canada K0J 1J0

*Corresponding author: progan@uwo.ca

We present a prototype software system with sufficient capacity and speed to estimate radiation exposures in a mass casualty event by counting dicentric chromosomes (DCs) in metaphase cells from many individuals. Top-ranked metaphase cell images are segmented by classifying and defining chromosomes with an active contour gradient vector field (GVF) and by determining centromere locations along the centreline. The centreline is extracted by discrete curve evolution (DCE) skeleton branch pruning and curve interpolation. Centromere detection minimises the global width and DAPI-staining intensity profiles along the centreline. A second centromere is identified by reapplying this procedure after masking the first. Dicentrics can be identified from features that capture width and intensity profile characteristics as well as local shape features of the object contour at candidate pixel locations. The correct location of the centromere is also refined in chromosomes with sister chromatid separation. The overall algorithm has both high sensitivity (85 %) and specificity (94 %). Results are independent of the shape and structure of chromosomes in different cells, or the laboratory preparation protocol followed. The prototype software was recoded in C++/OpenCV; image processing was accelerated by data and task parallelisation with Message Passaging Interface and Intel Threading Building Blocks and an asynchronous non-blocking I/O strategy. Relative to a serial process, metaphase ranking, GVF and DCE are, respectively, 100 and 300-fold faster on an 8-core desktop and 64-core cluster computers. The software was then ported to a 1024-core supercomputer, which processed 200 metaphase images each from 1025 specimens in 1.4 h.

INTRODUCTION

Cytogenetic biodosimetry has been validated for accurate estimation of radiation exposures[1]. It involves determining the actual frequency of dicentric chromosomes (DCs) in a set of metaphase cells and compares these with established standards with known exposures to compute exposure. DC analysis presents a number of bottlenecks in processing patient samples and interpreting results that impede processing the large volumes of samples expected in the event of a mass casualty[2]. Metaphase chromosomes from thousands of patients need to be analysed in a short assessment and treatment window. Results are needed quickly to assess large numbers of exposed, suspected and worried individuals.

Automating detection of normal and aberrant chromosome identification presents a number of unique challenges[3]. Cytogenetic biodosimetry studies based on DCs are labour intensive for large numbers of patient samples and require special expertise. Microscope images of metaphase chromosomes are variable in morphology, and different laboratories use different preparative methods resulting in their inconsistent appearance. Overlapping/touching chromosomes confound detection of DC. Sister chromatid separation present in some preparations affects the accuracy of image analysis. Finally, DCs are much rarer than normal chromosomes. The identification of DCs thus requires the development of rapid and accurate image-processing techniques for detecting these abnormalities regardless of differences in morphology from different sources.

There are several steps in cytogenetic biodosimetry processes (in the cytogenetic laboratory and in the analysis of laboratory-derived data), which limit the rate at which patient samples can be processed. The present study describes the current state of our efforts to accelerate the interpretation of cytogenetic image data. Our previous work has involved development, implementation and testing of novel image segmentation algorithms used in different aspects of automated dicentric analysis[4, 5]. Here, we integrate these different components into prototype software to identify and count DCs present in sets of images from DAPI-stained metaphase cells previously obtained from irradiated blood from one or more individuals.

Both the traditional dicentric chromosome analysis (DCA) undertaken by cytogenetic experts in reference laboratories and the automated Desktop version that we developed (ADCI Desktop) are challenged to process thousands of samples within a clinically relevant timeframe (i.e. hours to days). For this reason, dedicated, accurate, and rapid image-processing procedures are required to recognise the key morphological features in chromosomes that are diagnostic for

DCs[4, 5]. Furthermore, several of the system modules require considerable time to perform the computations necessary to extract these features, in part, because of the highly variable morphology of chromosomes. It is also difficult to find faster algorithms with the requisite accuracy to produce useful measurements of radiation exposure based on DCA. As parallel computing hardware such as multicore processors and computing clusters has become widespread, a feasible approach to accelerate ADCI is by parallelisation of iterative elements in existing algorithms. The need to handle thousands of samples within hours motivated the development of integrated software (ADCI Cluster) that exploits many processors in a high-performance computing environment to efficiently accelerate DCA. We developed a set of strategies to accelerate both ADCI Desktop and ADCI Cluster from different perspectives.

METHODS

Image segmentation procedures

A set of algorithms was developed in MATLAB to automatically and accurately locate centromeres[4, 5]. Briefly, graphical and cytogenetic features were extracted from images and used to construct a classification system to select the optimal images[6, 7]. Using a training set previously categorised by cytogeneticists, metaphase cells are classified as either well separated (nice), having a high level of overlapped chromosomes or overspread. Objects in images below or exceeding the size of chromosomes are masked. Metaphases are classified within the sample into categories and ranked with each category. After discarding overspread metaphase cells with an incomplete complement of chromosomes, nice or overlapped metaphases are ranked in decreasing order of their scores.

The chromosome classification module determines whether the input "blob" is a single chromosome or a chromosomal cluster with two or more chromosomes in overlap. A variation of the algorithm proposed by Rizvandi *et al.*[8] generates and prunes a coarse centreline for an input blob, and counts the number of conjoined parts of the centreline. If more than one conjoined part is found, the blob is a cluster of multiple chromosomes. Clusters of multiple chromosomes can produce false-positive DC assignments. A procedure for separating touching and overlapping chromosomes is being implemented[8]; however, at present, ADCI selects only well separated, single chromosomes for further processing.

Gradient vector flow (GVF) contour extraction is used to produce a descriptive outline of the input chromosome. The GVF active contour is an energy minimisation function based on the edge map of the chromosome object[9, 10]. Active contours (or snakes, in computer vision) are curves that can move under the effect of internal energy from the shape of the curve itself and the effect of external energy from the pixel data in the image. For chromosome boundary determination, the initial curve for GVF snake can be flexible. The GVF snake encourages convergence to boundaries with concave morphologies, which is very common in chromosome contours.

A linear approximation of the path of each chromosome is obtained from a one-pixel wide centreline that traces the length. Discrete curve evolution (DCE) is used, a minimum polygon is obtained in the DCE module for long chromosomes, which are then pruned. Centrelines of short chromosomes are obtained by medial axis thinning[5]. A centreline consisting of discrete or continuous points is obtained for every chromosome. In order to get a curve that exactly represents the chromosome centreline, cubic spline interpolation is applied to both axes to connect these points. The centreline obtained after interpolation contains a long stem and two short twig branches. The shortest twig is pruned to obtain the resultant centreline spanning the length of the chromosome.

To detect centromeres, the current system takes a set of lateral projections orthogonal to the centreline to generate a trellis-like structure that samples the width and staining intensity of the chromosome at regular intervals along its length. Intensity information is weighted based on a Gaussian distribution; and the width profile is obtained from the previous GVF snake result. These profiles are combined and the global minimum is obtained as the first centromere position. This location is masked and the process is repeated to obtain the position of the second centromere, provided that a second global minimum is present on the same chromosome.

Software integration

The algorithm prototypes coded and tested with MATLAB have been converted to an equivalent version of ADCI written in C++ using the QT and OpenCV set of image-processing libraries. MATLAB was not available for all hardware compute clusters on which ADCI was implemented, such as for the IBM BG/Q and the Symmetric Computing platforms. Developing software in MATLAB also creates licensing complexities that have been circumvented by recoding in C++, using OpenCV and QTlibraries. Finally, a significant improvement in performance is obtained by recoding. The current ADCI system comprised six functional modules: metaphase ranking (ranking), chromosome classification (classifying), gradient vector flow contour extraction (GVF), DCE, centreline interpolation (interpolation) and centromere detection (centromere). Additional modules under development improve the accuracy of DC analysis (sister chromatid separation with integrated intensity Laplacian and chromosome separation[11]). Figure 1 shows the flow

chart of the ADCI system, and Figure 2 gives a visualised example of the ADCI process. Two versions of ADCI, ADCI Desktop and ADCI Cluster, were developed for interactive use in desktops and handling mass casualty events in high-performance computing clusters, respectively. ADCI Desktop has a graphical user interface (GUI) module and ADCI Cluster has a scheduling module. The application also logs runtime errors, and classifies them as fatal, ordinary or warnings for debugging; the GUI displays errors in segmentation, centreline and dicentric routines visually for chromosomes that have been discarded during processing.

Parallelisation

Under optimal conditions, the increased speed brought about by a parallel algorithm can be linearly inverse of the number of computing units available. Most parallel algorithms are unable to achieve this level of performance, because of overhead from interprocess communication, latency due to scheduling and constraints on the specific hardware used. Applications can parallelise data and task processing. In data parallelism, data items are distributed to different computing units and processed at the same time. In task parallelism, computation on a single data object is subdivided into multiple tasks that are handled concurrently.

Performance estimates for unit tests were based on repeated random selection of a typical patient sample consisting of 200–250 metaphase cell images and sets of 46 chromosomes, from a larger repository of 1500 different cell images obtained from irradiated (3 Gy) blood. In the cluster and supercomputer applications, large datasets were initially generated containing between 1025 and 18 694 patient samples before processing.

Cluster and supercomputer software application

Parallelisation of the C++/OpenCV ADCI was developed with Intel Threaded Building Blocks (TBB), MPI or OpenMP, depending on the system used. Elements of the most time-consuming functions in ADCI were themselves parallelised.

Chromosomes were identified by binary connected component labelling (referred to as binary image labeling: 6,7). By profiling the ranking module, binary image labelling uses 47 % time of total processing time. Since most metaphase images selected by the ranking module are 'nice' images, chromosomes occur as a well-separated distribution with regular morphologies. This limits the number of preliminary connected components in an image. In high ranked metaphase images, the number of total preliminary connected components is effectively a constant. The parallel binary labelling algorithm divides it into several sub-images. The sub-images are labelled by recursively calling the parallel binary image labelling method. Labelled sub-images are combined to form a labelled complete image. In parallel ADCI, during the first pass, a metaphase image is divided into two sub-images in horizontal direction, and sub-images are divided recursively in the same way. This occurs until the size of sub-images fulfills the stopping condition, which is termed, the ending sub-images. Each ending sub-image completes its first scan serially, returning a local equivalence set and the corresponding labelled ending sub-image. This process can be executed in parallel for all ending sub-images.

To solve the energy function in GVF module, circulant tri-diagonal matrices have to be repeatedly inverted. These matrices are circulant, sparse but not strictly tri-diagonal, whose general form is expressed as

$$M = \begin{bmatrix} b & c & \cdots & a \\ a & b & c & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ c & \cdots & a & b \end{bmatrix},$$

where a, b and c are usually non-zero and all other cells are zero. Inversion of a circulant matrix also results in a circulant matrix. Matrix inversion can be simplified to solve the linear system represented by the matrix. As the matrices in GVF are always sparse tri-diagonal, a fast solution, the Thomas algorithm[12], can be used to solve the linear system. Implementation, also called odd-even reduction[13], requires two sets of operations. In the forward substitution phase, even rows in the matrix are eliminated by substituting the row above and the row below with elements in the current row. This process is repeated until only the first and the last rows remain. After solving this two-row matrix, unknown cells are solved in back substitution phase in the reverse order of the forward substitution phase. Every row in the matrix is only eliminated once in forward substitution and every unknown is calculated once in back substitution. The elimination of even rows in forward substitution and solving of unknowns in back substitution are executed in parallel.

RESULTS

Visualisation of image segmentation results

ADCI's graphical user interface provides results of intermediate and final stages of image processing for the metaphase ranking and chromosomal analyses steps in the process. Typical results are presented in Figure 2.

Expert evaluation of ADCI

Experts in cytogenetic biodosimetry at University of Western Ontario, Health Canada and Atomic Energy of Canada processed images from radiation-exposed

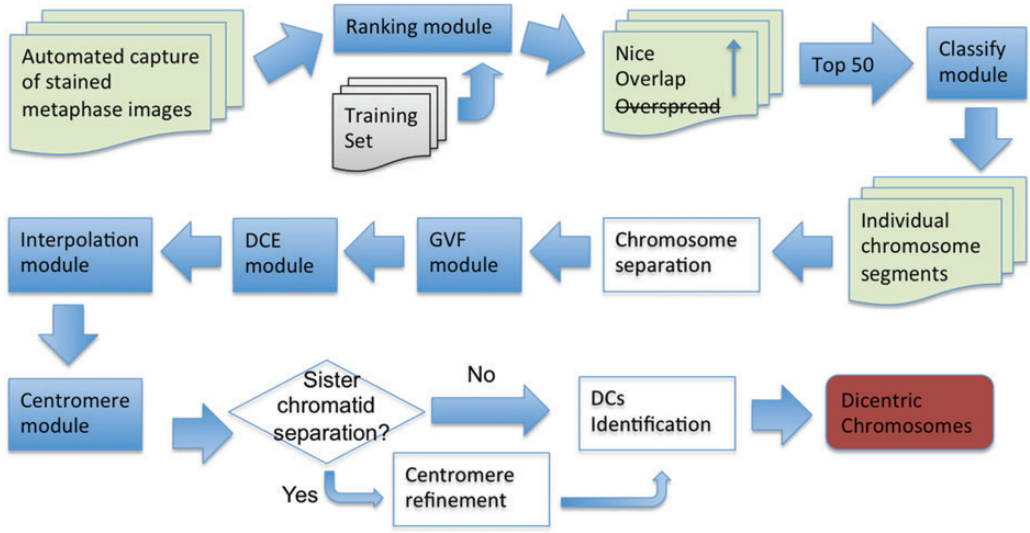

Figure 1. Flow chart of ADCI system. Light shaded shapes depict data in ADCI. Shading represents functional modules. In the box indicating three categories of metaphases, the strikethrough indicates that the 'overspread' images are discarded and the arrow indicates the rank: 'nice' metaphases have higher ranks than those with a significant number of 'overlapped' chromosomes.

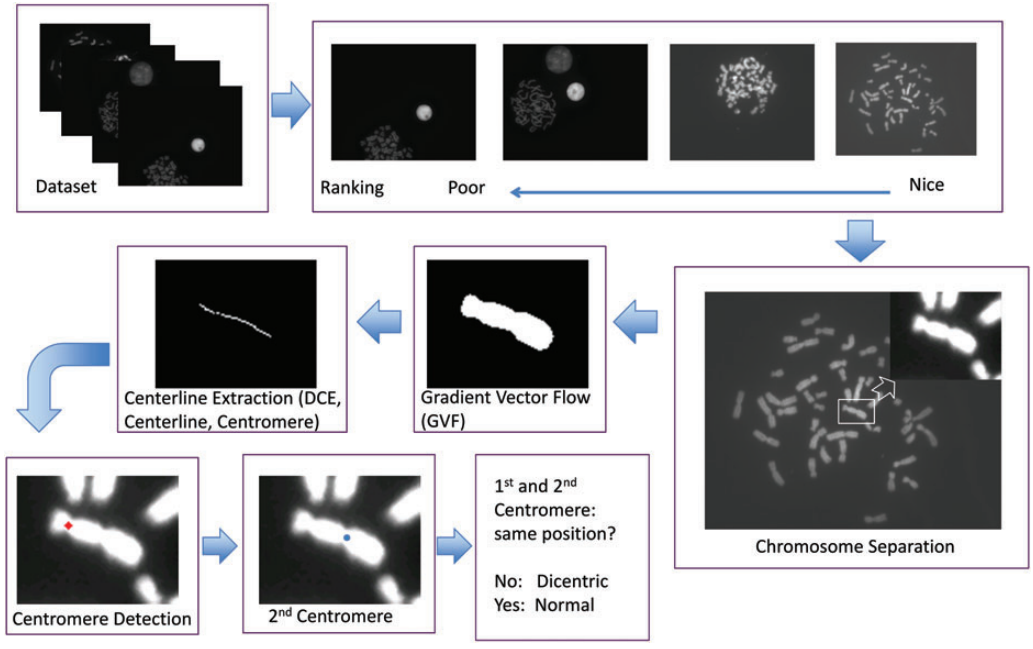

Figure 2. Example of outputs produced by the functional software modules. The default parameter for the ranking module selects the best 50 images of all DAPI-stained metaphases in a sample[7], though any number of ranked images can be processed. For each selected image, the classifying module segments individual chromosomes and classifies each into a single chromosome class or chromosome cluster class. Single chromosomes are analysed by a series of segmentation processes[4,5].

samples with the Desktop ADCI software. Although all steps are integrated in a single application, the results of each step (Figure 2) are visualised once sample processing is complete. The system can process either single or multiple samples. For ranking, ADCI currently uses a pre-trained set of image data to classify new sets of images; however, custom training sets may be substituted. After metaphase ranking is completed, the number of top-ranked cells is selected for subsequent image processing. Individual chromosomes are then extracted by binary thresholding of the objects in each image (threshold values can be manually optimised). Over-thresholding separates touching chromosomes into distinct objects. However, it can also result in chromosome fragmentation. Under-thresholding has the opposite effect. It promotes clustering of multiple chromosomes. An area threshold is applied to filter out interphase nuclei and groups of several overlapping chromosomes. Overlapping or touching chromosome pairs are detected[14], and can be either eliminated or included in subsequent steps by specifying the size of the window circumscribing a blob. Selected blobs (predominantly individual chromosomes) are then segmented to define accurate chromosome boundaries and the chromosome centreline is extracted. Although segmentation and centreline interpolation parameters can be altered, these values are already optimised for chromosomes. The location of the first centromere of each chromosome, determined from the global width and intensity profiles, is masked prior to searching for the second centromere, if present. The default masking area, a percentage of chromosome area, can be modified, as well as the normalised distance between the two centromeres. If the distance between two centromeres exceeds this minimum value, that chromosome will be labelled as dicentric. A summary of the analysis of eight slides analysed by two experts is shown in Figure 3. The software has a high false-positive rate, compared with ground truth; however, both had comparable sensitivity of detecting known DCs.

Parallelisation results

ADCI has been developed, modified and tested on desktop, cluster and supercomputer computing systems. The most significant performance improvements resulted from data parallelisation. In the complete ADCI, task parallelism is applied only when

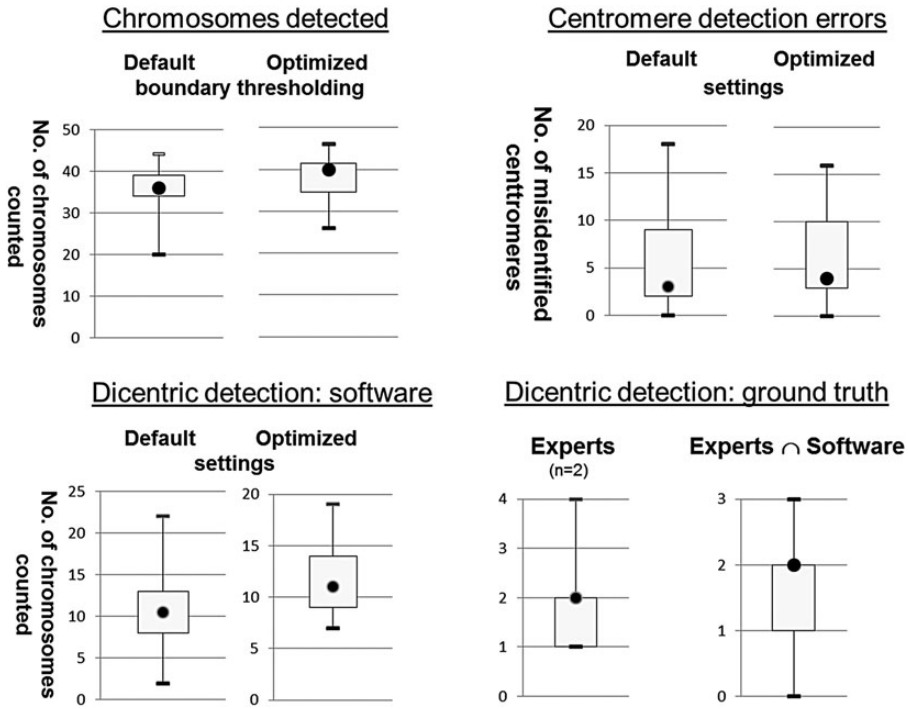

Figure 3. Results of ADCI image analysis for 3Gy exposure. Default and optimised refer to settings for chromosome boundary thresholding used in chromosome separation and the minimum distance between centromeres and radius of first centromere mask. Each chromosome was analysed by ADCI and independently by two experts. In the lower right panels, the *y* axis indicates the number of chromosomes counted.

there is sufficient residual parallel computing capacity remaining after data parallelisation has taken place. Our previous studies suggested that separate code parallelisation of these components would significantly expedite their performance[15].

Task parallelisation

The time requirement of parallel binary image labelling is approximately reciprocal to the number of processors available, and depends on the number of ending sub-images. Parallelised binary image labelling on an 8-core desktop was found to reduce the labelling time by 72.3 % of its serial time requirement after dividing each image into up to 32 subimages. This improved the overall speed of ranking of all metaphase images in a sample by 1.25-fold (8.5 vs. 10.8 s).

The GVF module, with the implementation of parallelised odd-even matrix reduction on an 8-core desktop was shown to be 3.4 times faster than the serial GVF module. From unit tests, parallel odd-even reduction is able to limit the increased computation required by the growing size of matrices used in GVF.

Desktop parallelisation

Initially, data parallelism was tested on an 8-core desktop with metaphase ranking module (200 images, the GVF and centreline-based modules: DCE, spline interpolation and centromere detection). Using 8 processors on the ranking, GVF and centreline-based modules increased speeds by 4.8-, 7.6- and 5-fold, respectively, relative to a single processor. The GVF module approximately reached the theoretical optimal speedup, which is 8-fold. In contrast, the ranking module is unable to perform data parallelism throughout, due to differences between individual metaphase images. The performance of the centreline-based modules was also affected by unbalanced workloads. The fully functional ADCI Desktop was also tested on the same hardware. Average time to process a sample (250–300 images) with a serial ADCI Desktop is 163.27 s, whereas the parallelised software required 43.44 s, approximately one-fourth of the time for the same sample. Overhead in the parallel version is increased due to file loading and result committing operations.

The processing speed of individual metaphase images can vary within- and between-patient samples for several reasons. Chromosomes are of unequal length and some require more time to process. Additionally, among metaphase images, there may be several chromosomes that touch or overlap each other, forming a chromosome cluster. Processing a chromosome cluster is slower than that required for an individual chromosome. This results in an unequal work distribution among the hardware threads for different metaphase cells. When dividing the iteration space in a parallel loop, multi-thread parallel platforms like OpenMP and TBB split the work on the basis of the iteration index. However in our case, the workload is biased based on the size and distribution of the chromosomes in an image, neither of which is predictable until image processing begins.

Compute cluster parallelisation

We also tested ADCI with mid-size cluster computing systems. Performance of metaphase ranking for varying numbers of processors (4 to 64) was assessed on a 1.5-Tb shared memory system running TBB. Processing speed plateaued at 40 CPUs, taking on average 0.65 s per sample and 3.38 h for all samples. This was probably due to either workload imbalance, overhead or that the implementation may not have been optimised for this system architecture. An MPI parallelised version of the ranking module was also run on 1000 samples with an 8-node (8 cores/node, 48 Gb RAM) computer cluster. Sample processing was balanced by workload sharing between processors. The total time to complete all MPI processes was 434 s, which was limited by the slowest MPI process. The fastest process required 101 s, and the average time per sample was 0.43 s.

Input/Output (I/O) latency was found to be a significant bottleneck in throughput. Loading metaphase images delayed the total time of the parallel ranking module. The time spent on loading all images in a sample could be 10-fold higher than the actual time required for image processing. An asynchronous I/O library was implemented to overlap I/O with real image processing; however, the improvement was, at best, modest (5.8 % faster).

Supercomputer parallelisation

ADCI Cluster was tested with samples on IBM Blue Gene/Q hardware. The software version was specifically modified for Blue Gene/Q because of the unique requirements of its PowerPC-based hardware architecture. The computing units in BG/Q comprised computing cards, containing a single chip with a 16-core 1.6 GHz 64-bit PowerPC A2 processor and 16 GB of RAM. A computing card in BG/Q is a computing node with 64 logic cores and 16 GB memory. The BG/Q internal network topology has faster inter-processor communication speed relative to other systems. A group of 64 computing nodes is associated with a single I/O node. Therefore, nodes in BG/Q do not compete for the same file I/O system, as they do in other cluster systems.

A total of 64 nodes containing 1024 physical cores in BG/Q were utilised to process 1025 samples. The I/O bottleneck in sample processing was mitigated by

combining all images in a sample into a single metaphase grid file (BigTIFF format). An MPI process was created for every computing node to process samples in serial. In total, 64 MPI processes were run in parallel. Inside a node, 64 OpenMP threads executed ranking of a sample in parallel. After the ranking is completed for a sample, 50 OpenMP threads are issued to process the top 50 metaphases in parallel. The chunk size was set to four samples. Figure 4 shows the deployment of ADCI on BG/Q.

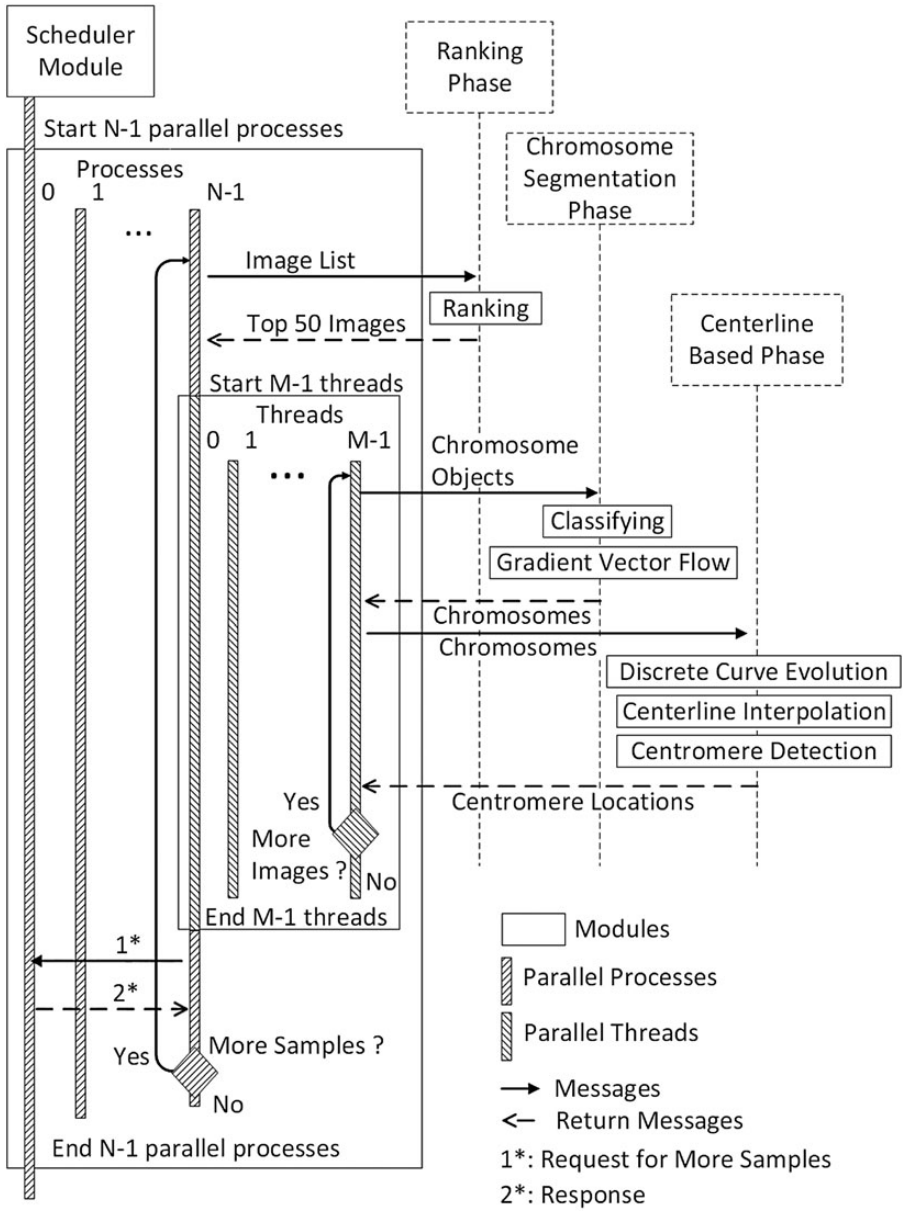

Figure 4. Sequential diagram showing logic architecture of ADCI Cluster. Scheduling module directs N processes, each with M threads, which are issued in parallel, and orchestrates coordination of the six functional modules using hardware resources, which form the major elements in ADCI: Ranking (ranking module), chromosome segmentation (classifying and GVF modules) and the centreline-based phase (DCE, interpolation and centromere detection modules).

Table 1. Multi-sample experiment performance on BG/Q.

Parameter	File loading	Ranking module	Chrom. modules	Complete ADCI
Data scale	1025 tiff files	1025 tiffs *(250–300 images)	1025 tiffs * (250–300 images) * (46 chrom)	1025 tiff files
Parallel mode	64 MPI processes	64 MPI processes * 64 threads	64 MPI processes * 50 threads	
Cumulative time: all samples	4 h 45 m 25 s	19 h 24 m 47 s	44 h 44 m 7 s	68 h 54 m 19 s
Maximum time per sample	18 s	1 m 12 s	5 m 33 s	7 m
Minimum time per sample	16 s	1 m 7 s	2 m 16 s	3 m 41 s
Average time per sample	16 s	1 m 8 s	2 m 37 s	4 m 2 s
Standard deviation of time	0.48 s	0.73 s	53.6 s	53.8 s

Table 2. Performance of ADCI versions on different hardware platforms.

Software	Time per sample			Accuracy[a]
	MATLAB	C++/OpenCV		
		Serial	8-core	
Image ranking	266 s	10.7 s	4.1 s	~98 %
Chromosome-based modules	3540 s	145 s	38.2 s	96.6 % Normal; 85 % DCs
Sister chromatid separation	NA	NA	NA	93.1 %
Complete ADCI	63.4 m	2.6 m	0.7 m	
	Time to process 1000 samples			
	MATLAB	Serial	8-core	BG/Q
Complete ADCI	44 d	1.8 d	11.7 h	1.4 h

[a]Compared with ground truth expert cytogeneticists[1, 2]; m, minutes; h, hours; s, seconds; d, days.

ADCI-BG/Q performs three major steps (based on parallelisation modes) when executing sample analysis. These include TIFF file loading, ranking and chromosome-based steps such as the chromosome classification, GVF, DCE, interpolation and centromere modules. The Tiff file loading step was parallelised with MPI at a sample level. The ranking and chromosome-based steps were parallelised with MPI at a sample level and with OpenMP at a metaphase level. In Table 1, a TIFF file represents a metaphase grid file containing all metaphases in a sample. The accumulated time sums the time spent on all samples for each step. It gives the total workload for processing all samples. The maximum and minimum time per sample helps to understand the workload distribution among samples at each step. The loading and ranking processing times for a sample were very consistent, as indicated by a relatively small standard deviation.

A problem in inaccurate centromere detection specific to ADCI BG/Q was identified during conversion of the program, and has been traced to incompatibility between OpenCV and Tiff library implementation on the PowerPC architecture. This issue has been resolved. A large standard deviation in the speed of the chromosome-based processing steps signifies that the workload was unbalanced between samples and emphasises the importance of scheduling workload among processors. The total processing time for 1025 samples with ADCI-BG/Q was the longest time for any node, which was 5090 s (1 h 24 m 50 s). With load balancing, most nodes processed 16 samples. However, some nodes processed 20 or 12 samples, one chunk more or less than the median number.

SUMMARY

The current version of ADCI includes image ranking and chromosome-based modules. Modules for chromosome and sister chromatid separation are under development and the final ADCI may include these modules. The performance of these components and ADCI are summarised in Table 2. Regarding performance, the ranking and chromosome-based modules resulted in an ~25-fold increase in speed from conversion of MATLAB code to serial C++. Converting MATLAB to parallel C++ for an 8-core desktop computer resulted in 64-fold and 92-fold increased speed for the ranking and chromosome-based modules, respectively. The complete C++ ADCI software application is on average 3.7-times faster when on an 8-core desktop, and 30-fold on BG/Q. This allows for the 8-core parallel ADCI Desktop and ADCI BG/Q to process ~1000 samples in 12 and 1.5 h, respectively.

It is encouraging to see the impact of parallelisation on the time to process cytogenetic biodosimetry data, as this could potentially fulfill the performance benchmarks required in a mass casualty. Because this approach is scalable, sourcing of additional hardware capacity might be expected to fulfill the processing requirements for any event where many individuals are concerned about exposure to clinically significant radiation levels. In theory, this strategy should be able to achieve the necessary throughput for data analysis to effectively triage those patients in need of scarce medical resources, within clinically relevant diagnostic and treatment windows over a wide range of exposures.

ACKNOWLEDGEMENTS

We thank Symmetric Computing, SHARCNET (Compute Canada) and SOSCIP for access to cluster computing facilities. We also acknowledges support from Canadian Foundation for Innovation (P.K.R., J.H.M.K.) and Canada Research Chairs Secretariat (P.K.R). P.K.R. and J.H.M.K. are founders of Cytognomix, Inc., which holds US Pat. 8,605,981 and other pending patents related to methods described in this paper.

FUNDING

Supported by the Western Innovation Fund (University of Western Ontario), Natural Sciences and Engineering Research Council of Canada. Funding from the Pilot Project Program of the Dartmouth Physically-Based Biodosimetry Center for Medical Countermeasures Against Radiation [to P.K.R.] via the National Institutes of Health [U19AI091173 to H.M. Swartz].

REFERENCES

1. Blakely, W. F., Salter, C. A. and Prasanna, P. G. S. *Early-response biological dosimetry-recommended countermeasure enhancements for mass-casualty radiological incidents and terrorism*. Health Phys. **89**, 494–504 (2005).
2. Alexander, G. A. *et al. BiodosEPR-2006 meeting: acute dosimetry consensus committee recommendations on biodosimetry applications in events involving uses of radiation by terrorists and radiation accidents*. Radiat. Meas. **42**, 972–996 (2007).
3. Piper, J., Baldock, R., Towers, S. and Rutovitz, D. *Towards a knowledge-based chromosome analysis system*. Automation of Cytogenetics, pp. 275–293 (1989).
4. Arachchige, A. S., Samarabandu, J., Knoll, J., Khan, W. and Rogan, P. K. *An accurate image processing algorithm for detecting FISH probe locations relative to chromosome landmarks on DAPI stained metaphase chromosome images*. In: IEEE Canadian Conference on Computer and Robot Vision, Ottawa, Ontario, pp. 223–230 (2010).
5. Arachchige, A. S., Samarabandu, J., Knoll, J., Khan, W. and Rogan, P. K. *An image processing algorithm for accurate extraction of the centerline from human metaphase chromosomes*. In: IEEE International Conference on Image Processing, Hong Kong, pp. 3613–3616 (2010).
6. Kobayashi, T., Shyu, C-R., He, L., Rogan, P. K. and Knoll, J. *Content and classification based ranking algorithm for metaphase chromosome images*. In: IEEE Conference on Multimedia Imaging, Taipei (2004).
7. Yanala, P., Lu, T., El-Ghussein, F., Zhao, C., Medhi, D., Wang, Y-P., Knopp, J., Knoll, J. H. and Rogan, P. K. *Automated detection of metaphase chromosomes for FISH and routine cytogenetics*. In: American Society of Human Genetics Meeting, Toronto, p. 195 (2004).
8. Rizvandi, N. B., Pizurica, A., Rooms, F. and Philips, W. *Skeleton Analysis of Population Images for Detection of Iso-lated and Overlapped Nematode C. elegans*. In: Proc. 16th European Signal Processing Conference, Switzerland, pp. 2972–2975 (2008).
9. Xu, C. and Prince, J. L. *Snakes, shapes, and gradient vector flow*. IEEE Transac. Image Processing **7**, 359–369 (1998).
10. Canny, J. *A computational approach to edge detection*. IEEE Transac. Pattern Anal. Mach. Intellig. **6**, 679–698 (1986).
11. Subasinghe, A. A., Samarabandu, J., Knoll, J. H. and Rogan, P. K. *Intensity Integrated Laplacian Based Thickness Measurement for Detecting Human Metaphase Chromosome Centromere Location*. IEEE Trans. Biomed. Eng. (2013). doi:10.1109/TBME.2013.2248008.
12. Thomas, L. H. *Elliptic problems in linear differential equations over a network*. Watson Laboratory Report. Columbia University (1949).
13. Stone, H. S. *Parallel tridiagonal equation solvers*. ACM Transac. Math. Software (TOMS). **1**, 289–307 (1975).
14. Ranjan, R., Subasinghe, A., Samarabandu, J., Rogan, P. K. and Knoll, J. H. M. *Automatic detection of pale path and overlaps in chromosome images using adaptive search technique and re-thresholding*. In: Proceedings of International Conference on Computer Vision Theory and Applications, Rome, pp. 462–466 (2012).

15. Li, Y., Wickramasinghe, A., Subasinghe, A., Samarabandu, J., Knoll, J. H., Wilkins, R., Flegal, R. and Rogan, P. K. *Towards large scale automated interpretation of cytogenetic biodosimetry data*. In: IEEE 6th Annual International conference on Automation for Sustainability, Beijing, pp. 30–35 (2012).