IM.Grid, a Grid Computing Approach for Image Mining of High Throughput-High Content Screening

Hongkee Moon, Auguste Genovesio

IM.Grid, a Grid Computing Approach for Image Mining of High Throughput-High Content Screening

HongKee Moon and Auguste Genovesio
Image Mining Group
Institut Pasteur Korea
39-1, Hawolgok-dong, Seongbuk-gu, 136-791, Seoul, South Korea
moon@ip-korea.org, agenoves@ip-korea.org

Abstract

Image processing and analysis has become essential for both cell biology research and drug discovery since the advent of High Content Screening (HCS) technologies. In this context, the Grid technology is a good opportunity to solve intensive computing problems with large data set. In addition, the exploitation of the Grid is not a simple task for many users because it is difficult to use the Grid in practical fields. Another important issue is to provide a simple way to use of Grid resources. In this paper, we present IM.Grid, a grid computing extension of our in-house image analysis software called IM (Image Mining) providing capabilities to simultaneously access visual data located on NAS (Network-Attached Storage) and extract knowledge from the raw information by customizable image processing pipeline in a parallel way. A user makes a plug-in designing own image mining pipeline using specific built-in image processing libraries. Then, the plug-in becomes an actual processing unit when Grid starts to analyze multiple images retrieving them from the NAS at a time. The user receives output results as fast as numbers of computational grids are available. We apply this method to reduce the image processing and analysis time of cell biological images for drug discovery within High Throughput-High Content Screening (HT-HCS) context. Because the processing time grows dramatically as the image size becomes huge due to many factors like multi-channel, high resolution and so on. To deal with these constraints, we propose a high-performance computing environment on .NET framework that helps to improve productivity not only in developing phases but also in HT-HCS platforms.

1 Introduction

The HCS consists of running a microscopy experiment under thousands different conditions (drugs, siRNA, concentration, etc) in a fully automated manner. The automation of an experiment is made possible during a phase called assay development where the biological assay and the image analysis algorithms are set up jointly by biologists and image analysts. Since crucial phase must provide strong evidences, the biological model and the image analysis taken together should be robust enough to produce higher throughput where visual inspection is impossible.

According to our knowledge, in practice, very few HCS have been successful in achieving more than a few tens of thousands experiments. In general, there are four important bottlenecks in the HCS context, which are the acquisition time, the storage space, the image analysis and the processing time. Therefore, we focused on making it feasible in practical HT-HCS. We have setup efficient fully-automated microscope acquisition platforms in a Bio Safety Level 3 environment, a centralized NAS server with the storage capacity of 60 Terabytes, and an image analysis research lab of about ten people. Therefore, the remaining issue turns out to be the performance of image analysis.

Basically, the large amount of high quality images produced from various kinds of microscope are to be analyzed in a rapid way for many reasons such as searching for new drugs, finding out optimal combination of chemical compounds, and experiments validation especially in the HT-HCS environment. It is highly demanding to handle high definition images which comprise multi-channel and 3D stack information along with time series. As the collected visual information becomes significant, the processing time increases dramatically due to newly developed microscopy technologies.

In this respect, Image Mining Group (IMG) is dedicated to develop high performance image processing software

called IM (Image Mining). IM is a client-based application written in C#, which enables biologists to discover semantically significant information from the microscopic images with help of plug-ins.

In this paper, we first review some related works in section 2 and describe our software environment in section 3. Then, in section 4, we present IM.Grid exploitation in a practical way and the implementation details of IM.Grid follows in section 5. In section 6, we provide an evaluation based on two different tests. Finally, we conclude this paper mentioning summary and future works.

2 Related works

A lot of examples of frameworks, tools and middlewares with the purpose of sharing and management of specific biological information are presented in literature.

Globus Toolkit, gLite and Unicore are well-known grid middlewares [2, 6, 1]. Globus Toolkit supports Java and Unix platforms including Linux with C components. gLite is implemented in C on Scientific Linux while Unicore is based on Java.

caGrid is a grid infrastructure on which the cancer Biomedical Informatics Grid (caBIG^{TM}) project depends. The project links all contributors together in the cancer community to share data and knowledge from various data sources [7]. It provides the workflow design for scientists to develop their own application based on grids. BIRN-CC based on the Globus Toolkit provides infrastructure necessary to achieve distributed collaborations and data sharing in wide area networking [4]. Health-e-Child project developed on EGEE middleware which provides the virtual foundation for flexible, secure, coordinated sharing of distributed resource [3]. They are suitable for composing grid-enabled applications for image analysis. However, there is a limitation that decreases the scalability to extend image analysis algorithms with specific tools provided in Microsoft Windows based products.

None of the previous projects concerns the management of various commercial microscopic visual data and their processing in a parallel way. The middlewares provide robust grid computing environment. However, many commercial microscopic systems support Windows platforms. None of the open source middlewares has capability to use Dynamic-Link Library (DLL) files in order to access the Windows-based microscopic systems. We cope with these drawbacks by adding IM.Grid module to the image mining software.

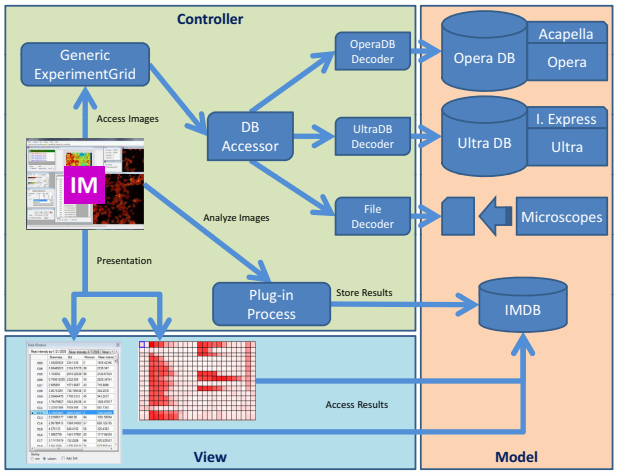

Figure 1. IM software core structure

3 The application environment

3.1 The Image Mining environment

IM is a 64-bit image analysis software developed by the Image Mining Group at Institut Pasteur Korea. IM helps biologists to find interesting and important factors from images taken in various microscopic imaging systems. Fundamentally, IM consists of core module, library part, and plug-ins implemented by computer scientists for use in diverse applications. The library enables the developers to implement plug-ins easily and accurately due to state-of-the-art techniques of image processing approaches and sophisticated computer algorithms. The core structure of the software follows the MVC (Model-View-Controller) model [5] and is maintained by engineers. Figure 1. shows IM software core structure where we can access any well-plate of biological experiments due to a generic well-plate representation called an *ExperimentGrid*.

Because of the large amount of projects, we adopted the plug-in based approach letting people develop plug-ins according to specific projects. This method gives a possibility to solve many problems in distributed computing environment because plug-ins are dynamically loaded in multiple processing units during runtime. Previously, a plug-in used to be a single instance where users had to analyze experiments in sequential mode even though the computer had multiple CPU cores with a large amount of memory. This means that the utilization of computer resources was much lower. Eventually, IM.Grid project started along with multi-threads support.

In addition, IM can access several microscope file format as Zeiss LSM or Leica^{TM}LEI but is also adapted to many screening platforms. That is, during a screening campaign,

a platform as Evotec Opera^{TM}or Molecular Device ImageXpress ULTRA^{TM}acquires and stores a massive amount of pictures from a given number of well-plates in their own database format on a centralized server. IM has the ability to load any well-plate of experiments whatever the platform. Therefore, it is possible to navigate with IM through experiments of any platform exactly in the same way. The format conversion is instantaneous and transparent to the end users. This was made possible owing to a generic conception of abstract loaders which enable to add a new platform DB decoder at any time. Multiple well-plates are handled in such a way that a screen of hundreds of well-plates can be sequentially processed.

3.2 A brief overview of the IM.Library

IM.Library provides an efficient implementation of the most common image processing and analysis methods. The library covers from basic filtering operations to sophisticated machine learning techniques. It has been implemented in C# and growing as the scientists validate state-of-the-art algorithms concerning specific projects. The library consists of classification, descriptor, filtering, mathematics, morphology and transforms. Thus, we can design an image mining pipeline by selecting components at a certain level of the analysis. They allows the development of image processing workflow simply by adding necessary modules. These algorithms are performed in sequential and in parallel according to the complexity of the problem domains and the size of visual information.

4 IM.Grid exploitation

The IM.Grid is implemented as an extension module to the conventional image mining software, achieving high performance values and flexible scalability. Firstly, we tried to tackle performance issues with multi-threads model. However, the multithreading has drawbacks when a number of high quality images are simultaneously loaded, which not only gives out-of-memory exceptions frequently, but consumes high network bandwidth. So, there was idle time if we try to open the images even in gigabit network environment.

In order to overcome these drawbacks, it was inevitable to develop a grid framework which takes into account the factors related to execute efficient image processing applications. The framework handles such large resources as the automated microscopic imaging systems produce high resolution images at great speed. IM.Grid has a crucial role to manage all the parallel operations communicating computational grids, and access the selected image data from the centralized storage from each grid through a gigabit network channel. In Figure 2 shows how IM.Grid is exploited in our context.

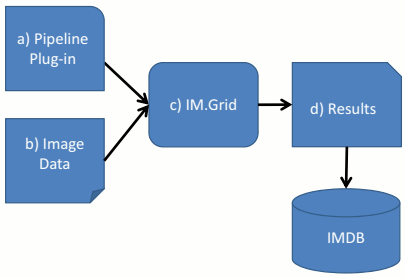

Figure 2. IM.Grid exploitation process

4.1 Image mining pipeline design and plug-in implementation

In terms of an actual image processing logic performed in the grid workers, developers can design an image mining pipeline, choosing some functions at each step of the flow, and define appropriate threshold values for the steps (see Fig. 2.a). For instance, if we want to make a procedure for finding statistical descriptor information from an image, we can design a pipeline including some steps as below.

1. Apply a segmentation algorithm
2. Construct graphs
3. Label 3D objects from the graph image
4. Get feature descriptors
5. Print the result

During the design phase, the users can check every output result at each step if they want to see them(see Figure 3). However, it is impossible to check the output of each step while grid operations are executed on a huge amount of input images in parallel. Therefore, it is necessary to validate the pipeline process before running them in the grids. In the plug-in implementation, the users make a grid-enabled plug-in DLL file embedding the process logic of the designed pipeline and IM.Grid APIs.

Sometimes, image analysts are involved in some special cases such as microscopic images are distorted according to unintentional artifacts. They can handle these problems making elaborate plug-ins adopting effective algorithms in IM.Library through this process.

4.2 Image data selection

All the image data are stored in centralized NAS server through a trunk of multiple gigabit lines providing high

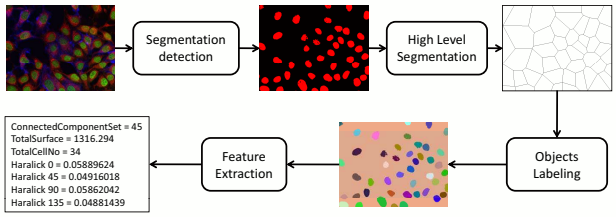

Figure 3. Pipeline design and output diagram

speed access and better reliability. Therefore, every client computer in our intranet can easily open the necessary images by specifying uniform naming convention(UNC) format after the microscopic imaging systems produce pictures. This is a key idea how to handle a large number of images over the network especially in the grid environment.

Users can just browse experiment folders having specific well plates or micro arrays, and select them just by several steps, since the framework provides a general and consistent way to access all the heterogeneous types of data sources (see Fig. 2.b). We developed an abstract concept for them. *ExperimentGrid* provides generic functions of accessing actual items in well plate and micro array, which has just UNC information of the remote images. Each item is corresponding to *Experiment* class. In the case of Evotec Opera^{TM}, a well-plate information can be retrieved by parsing the particular XML data file and making an *ExperimentGrid* instance, sending the meta information of the well-plate to grids in order to load actual pictures before image mining operations.

4.3 IM.Grid execution

To execute image mining exploiting IM.Grid, there are two requirements. One is to make a plug-in pipeline module and the other is to define input objects to be processed (see Fig. 2.c).

Regarding the first requirement, we made a plug-in by designing an image mining pipeline mentioned in section 4.1. When the grid operations start, each grid automatically checks if there are the latest version of dependencies of the plug-in module. If local version is different, IM.Grid distributes the plug-in DLL across the grids. The overall flow of process is shown in Figure 4.

In terms of the second condition, the selected image data is provided for parallel operations. In this respect, we have to consider task granularity. Basically, IM.Grid allows user-defined ways of selecting input types which can be an image or a set of images. If a plug-in takes heavy computational time for processing a single image, image based input type is recommended. But, image set based input method can be effective when fast processing plug-ins are used, removing overheads of loading images stored in a single file as a stack of pictures.

The following actions are sequentially performed when the grid operation begins.

1. Make specific number of threads of the plug-in
2. Create IM.Grid client instance in each thread
3. Send the plug-in binary and UNC of image data to the workers
4. Grid workers execute the tasks by loading images from the NAS
5. Received the final output results from the grid

4.4 Result manipulation

In the grid pipeline module, task completion event handlers should be implemented since output results are asynchronously delivered from the grids when parallel operations are finished (see Fig. 2.d). The defined event handlers are fired at the point of receiving the results successfully. The output type can be any kind of data which should be serializable or primitive. Once all the results are collected, table type results are stored in IM database which allows users to retrieve it at any time.

5 Behind the IM.Grid

Both screening platform independence and plug-in oriented architecture are the strengths of IM, allowing to decrease the development time of IM.Grid. Each plug-in module can be multiple instances when we create multi threads to run on different images. To make the screening platform independent, IM gives a dynamic binding method for accessing actual image data. This is a basic requirement for the implementation of distributed system to load the images on remote machines.

Given this context, the two next sections describe two approaches we implemented to increase the speed of a screen.

5.1 Multithreading model

As a natural way to increase the performance, IM was designed as a multithreaded program where a plug-in developed is able to run for several different experiments in parallel without any further modification of the plug-in. A thread pool with waiting slots is implemented for scheduling multiple threads as each thread manages the life-cycle of a plug-in instance, allowing a FCFS(First Come, First Served) manner by control of the number of active threads.

Although this method has proved good performances, it was still mainly insufficient to handle the huge size of image data and long processing time required by the computation of a large screening assay.

We extended this principle to a grid of computers located on the network. In this respect, multi threads are responsible for processing jobs with communication of the grid in a parallel way, allowing an efficient method to manage the network I/O idle time.

5.2 Grid computing model

The method of making copies of the plug-in instance for multithreading is also useful for developing a grid computing approach, which makes more efficient collaboration between a client PC and the computational grids.

There are three parts with different roles in IM.Grid. The *Grid Entity* represents an available CPU located on the network. It simply receives and performs a computational tasks and returns results. It also updates its status to the *Grid Service*. The status consists of current computational workload and availability on the network.

The *Grid Service* is a web service which centralizes the status of all registered *Grid Entity* on the network. It can be interrogated at any time by the IM core and return a fast answer giving the status of all the network. The *Grid Entity* sends its status to the *Grid Service* every 1 minute.

The *Grid Client* sends a plug-in copy to an available *Grid Entity* along with a UNC information of the image to be processed and receives the processed result.

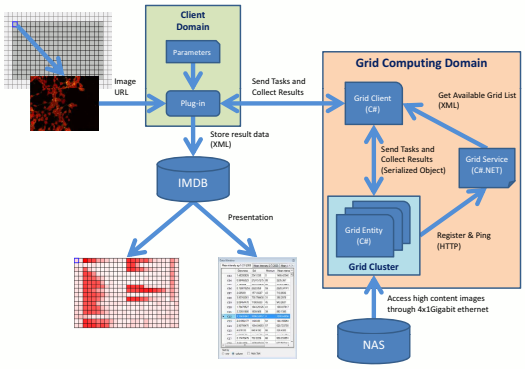

Figure 4. IM.Grid architecture

A *Grid Client* is initialized with the number of available *Grid Entities* and submits successively all queued jobs to its *Grid Entities* until completion of all jobs. A job is made of a serialized stream object of the plug-in instance and optionally the UNC information of the pictures to be processed. The *Grid Entity* receives the job, de-serializes the instance, processes the pictures with it, and sends back the results to the *Grid Client*.

6 Performance evaluation

To evaluate the performance of our methods, we carry out two different tests in batch and grid computing modes. The first test is a computation test which requires high processing power whereas the second focuses on heavy network usage with a simple processing of image data in different environments of 1Gbps and 54Mbps wireless. Both tests in this section were successively performed on the two following conditions:

- in batch processing on two 3.0GHz Intel Xeon processors with Hyper-Threading enabled, 2GB of memory and a gigabit Ethernet connection.
- on a cluster of 10 machines of the same specifications above. Totally, 40 worker nodes are activated since each machine has two CPUs with Hyper-Threading.

6.1 Computation test

We tested the plug-in that is implemented in Figure 3. Each well has 15 pictures(one picture has 3 channels with 649x506 pixels). A benchmark is performed from 10 to 90 wells. In batch process mode, the image mining process takes a picture at a time. Thus, it takes much longer time compared with the grid computing approach, which takes advantages of 40 worker nodes controlled by the 40-threads. 40 worker nodes load different images and analyze them in parallel. Figure 5. shows the performance gap between batch and grid mode.

6.2 Data transfer test

In HT-HCS context, it is also important to emphasize the efficacy of a grid computing approach regarding the data transfer. A screening campaign is about 500 Gb of pictures which, in the grid computing approach, do not have to flow through the client computer, but are directly accessed by the computers of the grid. To illustrate this, we created a very simple plug-in computing the mean intensity of a picture. Figure 6 shows the result when running this plug-in on a 384 well-plate in which a well has 4 images and one image has 2 channels with 672x507 pixels. The actual size of a well is about 2.6MB. The mean intensity plug-in is simply made for accessing every pixel of input images. Thus, the processing time is trivial whereas the loading time is more significant. We have almost same performance results on grid mode regardless of network types. However, in batch mode, the image loading time affected the processing time, which

takes 25 times longer on wireless network and 3 times more even on gigabit network. Consequently, we do not have to take care of network bandwidth when we process HT-HCS screening assay in the grid environment. Because the grid provides high computation power and fast network access no matter that a client computer is located in.

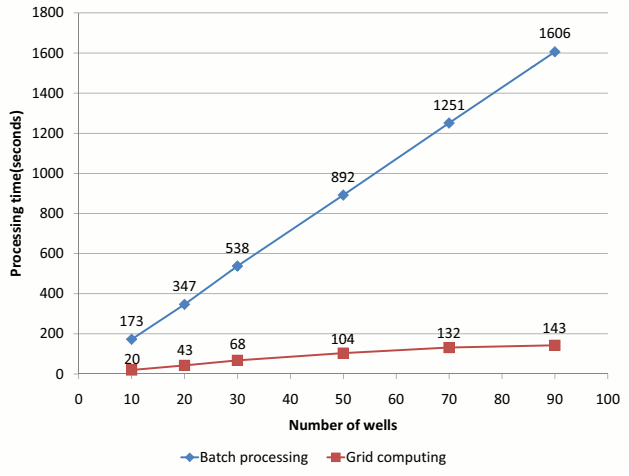

Figure 5. Computation test

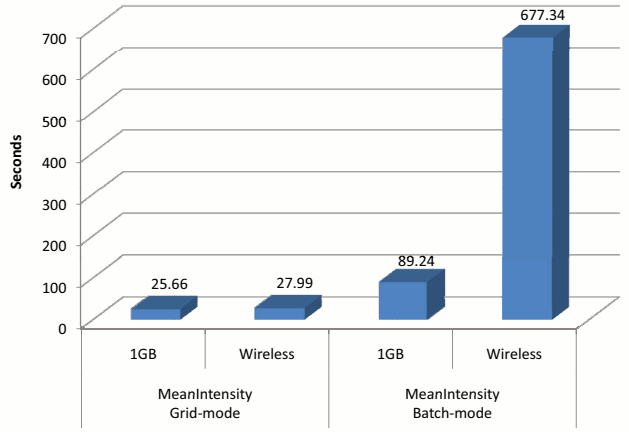

Figure 6. Data transfer test

7 Conclusion & future works

The Grid computing is becoming the focus of attention in not only scientific fields but commercial entities in order to take opportunity from highly profitable discovery. However, it is a difficult task in terms of the integration of a grid extension on the conventional software as well as the exploitation. In this paper, we presented IM.Grid which is the grid computing extension of the image mining software, and the exploitation through the customization of image mining pipeline in HT-HCS context. In the evaluation test, we showed the computation time could be highly reduced on account of the grid computing approach.

Typically, a sophisticated plug-in detecting hundreds of objects in a picture and computing a set of descriptors on each of them, which takes a few seconds per image, is very slow in batch processing mode. This architecture has offered a high performance computing environment to the scientists of the Image Mining Group developing plug-ins for biological assays since the time allowed for the analysis of a single picture was significantly increased.

Furthermore, optimization of tuning between grids and threads should be investigated as the computational grids lead to various overheads including client-server communication and big object transmission. In addition, we continue to improve the grid architecture for the integration of conventional grid systems as well as making a GUI based pipeline design system.

References

[1] D. Erwin et al. UNICORE a Grid computing environment. *Concurrency and Computation: Practice and Experience*, 14(13-15):1395–1410, 2002.

[2] I. Foster and C. Kesselman. Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, 11(2):115, 1997.

[3] J. Freund, D. Comaniciu, Y. Ioannis, P. Liu, R. McClatchey, E. Morley-Fletcher, X. Pennec, G. Pongiglione, and X. ZHOU. Health-e-Child: An Integrated Biomedical Platform for Grid-Based Paediatric Applications. *4th International HealthGrid conference*, 120:259–272, 2006.

[4] J. Grethe, C. Baru, A. Gupta, M. James, B. Ludaescher, M. Martone, P. Papadopoulos, S. Peltier, A. Rajasekar, and S. Santini. Biomedical informatics research network: building a national collaboratory to hasten the derivation of new understanding and treatment of disease. *Stud Health Technol Inform*, 112:100–9, 2005.

[5] G. Krasner and S. Pope. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988.

[6] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.

[7] J. Saltz, S. Oster, S. Hastings, S. Langella, T. Kurc, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag, and P. Covitz. caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, 22(15):1910, 2006.