IPSJ Transactions on Advanced Computing Systems Vol. 1 No. 2 193–206 (Aug. 2008)

Regular Paper

Experimental Study of Distributed SWAP-GA Models on the Grid

Shamim Akhter,[†1] Kiyoshi Osawa,[†1]
Motokazu Nishimura[†1] and Kento Aida[†1,†2]

Recently, the phenomenon which quantifies the agricultural and water management practices from remote sensing (RS) imagery, has been adapted to help the policy makers and farm/water managers to make better operational decisions. SWAP-GA is a combined model of the SWAP (Soil Water Atmosphere and Plant) crop model and the Remote Sensing (RS) data assimilation technique, which is optimized by Genetic Algorithm (GA). However, to run the SWAP-GA model on a single PC requires a massive amount of processing time. Based on the above observation, distributed or parallel computing can be a preeminent and convincing solution. At present, Multi-cluster Grids have emerged as the most popular type of distributed computing system. However, the performances of different parallelization methodologies on the Grid have not been discussed thoroughly. This paper presents the implementation of the SWAP-GA application and discusses its impact on the performance of parallelization methods.

1. Introduction

The demand for efficient and effective agricultural products monitoring systems is increasing. Agricultural researchers try to analyze various pieces of information about crops in order to take measures when they have problems. In particular, when an on-going experiment covers a large area such as a country, Remote Sensing (RS) plays a vital role by providing useful information over large areas. However, some information, or crop parameters, is not visible through RS images such as sowing dates, cropping intensity, growth, stress, etc. Crop models can help to solve this problem. The Crop models calculate missing information by analyzing crop information with real fields' experimental data.

†1 Tokyo Institute of Technology
†2 National Institute of Informatics

Ines and Honda [1] developed an assimilation scheme of the SWAP (Soil, Water, Atmosphere, Plant) crop model with RS data using Genetic Algorithm (GA). Similar works by Ines [2] and Srinuandee [3] used some remotely sensed information combined with a binary GA [4] and SWAP model for optimizing soil hydraulic parameters. Later, a real coded GA [5] was applied by Chemin, et al. [6]. Furthermore, Chemin worked with the SWAP-MultiGA model (Modified SWAP-GA) [7] and that was successfully implemented with a new hypothesis to assimilate RS evapotranspiration (ETa) data for satellite images observed by ASTER [8] and MODIS [9]. However, these researches have a problem in their practicality, that is, they require a huge amount of processing time.

Grid computing receives much attention as a powerful scheme to solve large scale problems with remote sensing data, and there are several works to develop applications on the Grid [17)–19)]. These works use a simple parallelization method, e.g. distribution of computations for pixels, and the impact on performance by the parallelization method has not been discussed thoroughly. The performance improvement of SWAP is discussed in Refs. 12), 13), and the work includes parallelization methods of SWAP-GA. However, the discussion is limited to parallelization approaches only, and their performance analysis is still missing.

This paper presents the implementation of the SWAP-GA application on the Grid and discusses the impact on the performance of parallelization methods. We implemented three parallelization methods, pixel distribution, population distribution, and hierarchical distribution, and compared the performances through the experiments on the Grid testbed. These methods use GridRPC 14) as the programming framework but ways of task distribution are different. The results show that the Pixel Distribution model exhibits the lowest communication overhead and shows the best performance on some settings of the Grid. However, the Hierarchical Distribution model is also a preferable model to employ more parallelism in the SWAP-GA application. We also present an analytical model to estimate the running time of the application. The model helps users to choose a parallelization method that is suited to the Grid environment. The experimental results show the accuracy of the analytical model. Our application runs with GRASS, which is an analysis tool for geographic resources commonly used in RS

community [15]. It can be easily extended to other applications that analyze RS data processed by GRASS.

The rest of the paper is organized as follows: Section 2 gives the background of the work. Section 3 presents three parallelization methods and their implementation. Section 4 shows the experimental results and discusses the performance. Finally, Section 5 concludes the work of this paper and outlines the future work.

2. Background

This section outlines the related work and our target application, or SWAP-GA. The programming tool used for the implementation is also introduced.

2.1 Related Work

Distributed or parallel computing combined with remote sensing is now a prominent and promising field for geosciences, environmental sciences, agricultural sciences and others. There are many on going projects [10),11)] to process remote sensing data with distributed computing and to create databases from them [16]. Distributed approaches have also been applied in some remote sensing image processing applications, such as terrain feature extraction [17], image classification [18], radiometric and geometric correction [19] over different parallel computing platforms. The overall implementation strategies of remotely sensed image processing on the grid environment have been discussed in Ref. 20).

On the other hand, agriculture researches are also developing rapidly. They require both remote sensing and distributed computing [21),22)]. Ninomiya [23] proposed a distributed decision support system to store agricultural data and provide agriculture related information over the grid distributed platform. IWMI [24] has the vision to improve water and land resource management all over the world.

SWAP-GA is an application that merges the agriculture, and remote sensing issues together. SWAP-GA is an assimilation scheme of the SWAP model with RS image data using GA. Another model, SWAP-PEST, provides the same functionality as SWAP-GA, and is successfully implemented in Refs. 25) and 26). While these works present approaches to run their applications on the Grid, discussions are limited to those of simple parallelization hypotheses. Also, some work focuses on methodologies to create a distributed database [23].

The work presented in Ref. 12) discusses multiple parallelization methods for the SWAP-GA application; however, the discussion is limited to the performance on a single cluster with MPI [31] implementation. The work used synthetic data only and has not yet been tested in real field experimental conditions.

Thus, the performance impact of parallelization methods on the Grid is still unknown. In particular, the performance is significantly affected by task distribution methods on the Grid, and developers of RS applications need to solve the problem of task distribution, or how to distribute tasks among multiple clusters on the Grid. This paper gives experimental results with its analytical model to solve this problem.

For the issue of practicality, experiments with real RS data are necessary to see the performance in the real world. The experiments in this paper run the SWAP-GA application with the real data collected in Thailand on the real Grid testbed. Moreover, interoperability between the application and existing RS software is also necessary to improve practicality. Although users need to manually extract RS data from some databases in the existing work in Refs. 3) and 7), the application presented in this paper is implemented with the interface for the GRASS tool [15].

2.2 SWAP-GA Framework

The goal of the SWAP-GA method is to calculate missing information about the target crop in remote sensing data using Genetic Algorithm. In this scheme, actual evapotanspiration (ETa), the water use of a particular crop at a given time, is computed both by directly analyzing remote sensing data (satellite image data) and by the indirect method, or the SWAP model.

ETa computed from remote sensing data, SatETa, can be obtained by an open source software for Geographical Information System, GRASS GIS [15]. ETa computed from the SWAP model, SwapETa, is derived by solving the Pernman-Monteith equation [27]. Initially, SatETa and SwapETa do not match, because some required parameters for the SWAP model, e.g., a ground water level, a cropping season time extent, a date for the emergence of crops, and irrigation scheduling, are missing. Genetic Algorithm is used to find the missing parameters in the SWAP-GA.

Figure 1 shows an overview of the SWAP-GA. Inside GA, every unknown parameter is treated as an individual gene and a population is constructed with

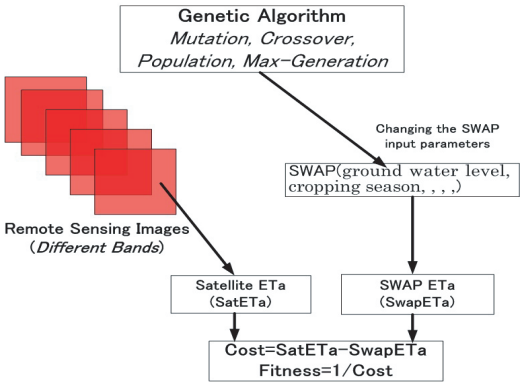


Fig. 1 SWAP-GA model structure diagram.

a set of genes. Each population provides a cost value ($|SatETa - SwapETa|$) and transforms that to a fitness value as 1/cost. Higher Fitness for a population indicates good assimilation and has a better chance of surviving for the next generation. The next generation's parameters will be set according to mutation and crossover rules. One pixel's assimilation requires one to several hundred population evaluations. The populations set will be evaluated by the predefined number of iterations, called the maximum generation (Max-Generation).

The real Remote Sensing images used in this paper are collected from MODIS (Moderate Resolution Imaging Spectroradiometer) and the images are processed from January 1st 2002 to May 1st 2002. MODIS is a radiometer developed by NASA and it is installed on the satellite. Each image has 2,401 (row) x 3,608 (column) pixels, each of which has 500 m spatial resolution and covers 250,000 m^2 area.

In this study, the performance of the SWAP-GA application is investigated with the real data of two crops per year in a large continuous rice field in the Suphanburi province, which is located in the Central Plain of Thailand. The total area of this province is 5,358 square kilometers that maps around 21,432 pixels of a 500 meter resolution image.

We used the RS data of 15 pixels extracted from the MODIS image of the Suphanburi province. Due to the limitation of available input parameters in a large area, e.g., data of two crops per year and soil properties for rice, we leave the experiment with larger data for our future work.

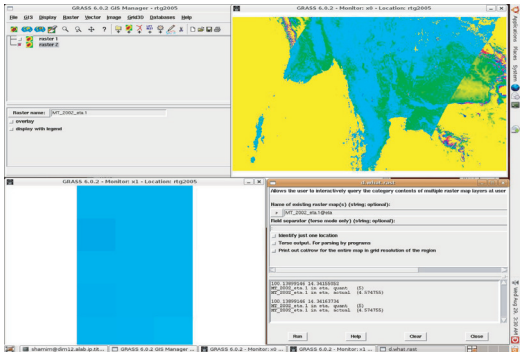


Fig. 2 RS image processing through GRASS.

Figure 2, presents the GRASS working monitor, which consists of the full RS image, image of 15 pixels, zooming and tracing the appropriate co-ordinate values extracted from the specific longitude and latitude value (up to down and left to right manner).

2.3 GridRPC and Ninf-G

The GridRPC is one of the familiar programming models for a Grid application based on the Remote Procedure Call (RPC) mechanism. It is middleware that provides remote library access and task parallel programming over the Grid environment. An application program using the GridRPC consists of a client program and server programs. When the client program invokes an RPC, the corresponding server program runs on a remote machine. GridRPC has now been supported by many well known grid programming systems such as Ninf-G [28] and Netsolve [29]. Ninf-G is implemented on the top of the Globus Toolkit [30]. Nowadays, it is one of the most popular programming platforms for developing master-worker (client-server) applications on the Grid. From the user's point of view, this is just an ordinary software library, where all Grid functionalities will be achieved by C type function calling methods. The SWAP-GA application is designed to run on parallel computing platforms in the master-worker manner. Thus, Ninf-G is chosen by on account of the above benefits.

3. Design and Implementation of Distributed SWAP-GA

Figure 3 shows the mechanism of SWAP-GA. The full SWAP-GA executable module is made with RSImageAccess, GASWAP, and Evaluation sub-modules. The RSImageAccess module is the main module, where the program starts. This module extracts the pixel value (SatETa) and the date for each image from the GRASS environment, and it calls this GASWAP module. The GASWAP module is able to run GA and completes the assimilation process. The Evaluation module runs the SWAP executable for each population and sends the SwapETa values to the GASWAP module. An example with 15 pixels image, 60 populations and 10 generations gives a clear view of calling procedures inside the SWAP-GA model. Particularly, in this case, the RSImageAccess module will call the GASWAP module 15 times (one for each pixel). For each pixel, the GASWAP module first internally executes the SWAP executable 60 times to initialize the ETa value for every population and then calls the Evaluation module 10 times (one for each generation). For each generation, the Evaluation module executes the SWAP executable 60 times and produces SwapETa values for 60 populations. A high demand of parallel computing is called for inside the whole SWAP-GA module. Three different strategies are applied to work SWAP-GA in a parallel manner.

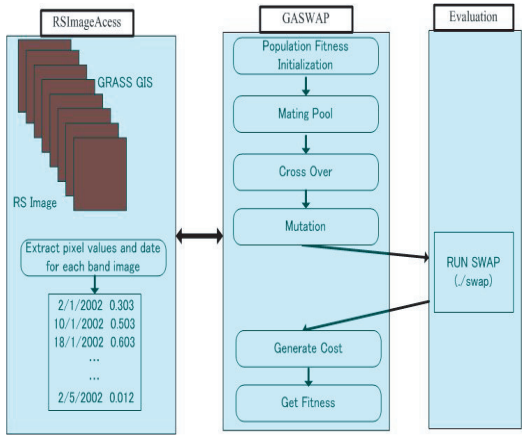


Fig. 3 SWAP-GA working modules.

The strategies are presented by the following models: i) the Pixel Distribution model, ii) the Population Distribution model, iii) the Hierarchical Distribution model.

3.1 The Pixel Distribution Model

A "pixel" is the smallest dimension of an RS image which indicates the image resolution as DN (Digital Number) value. The RSImageAccess module extracts the pixel values and sends them to the GASWAP module for processing. Computations for pixels are independent, thus they are distributed to multiple computers. This approach is called the Pixel Distribution model. In this paper's experiment, the Pixel Distribution model is implemented on the Grid using Ninf-G. The Master-worker paradigm is used for parallelization in the distributed SWAP-GA model. Inside the master node, the RSImageAccess module works, whereas in the worker nodes assimilation procedures (GASWAP and Evaluation modules) run. **Figure 4** illustrates a mechanism to run SWAP-GA with the Pixel Distribution model. The RSImageAccess module, running as an Ninf-G client, dispatches a set of pixels to the remote PC cluster. In the remote PC cluster, the GASWAP module and the Evaluation module, running as a Ninf-G server, perform a computation for each pixel, where the computations inside the PC cluster are distributed among computing nodes through the local batch scheduler in FCFS manner.

The Pixel Distribution model has the smallest overhead of the three models,

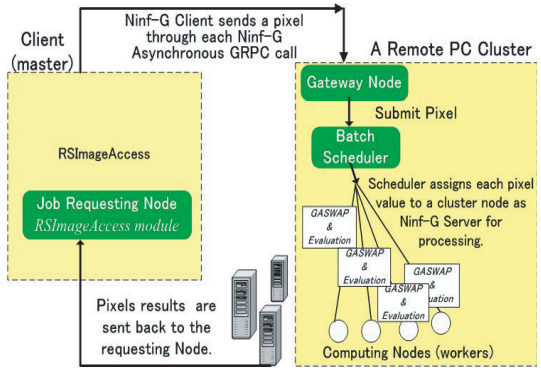


Fig. 4 The pixel distribution model.

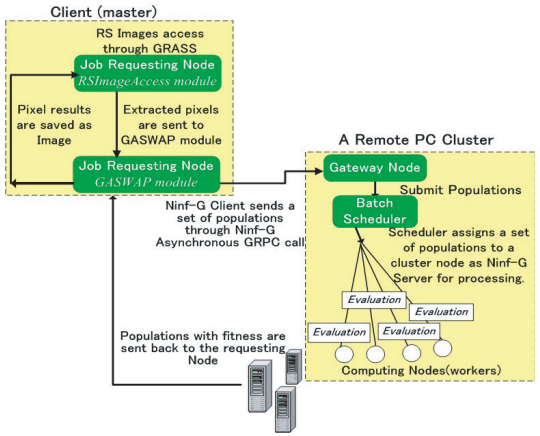


Fig. 5 The population distribution model.

and we expect an efficient parallel execution. However, parallelism exploited by this model is limited to the number of pixels in the input image.

3.2 The Population Distribution Model

The second approach is the Population Distribution model. An evaluation procedure is called for each population to execute the SWAP executable. Thus, the Evaluation module (to evaluate population) can be distributed through Ninf-G. Here, the master node runs both the RSImageAccess and the GASWAP module, and worker nodes only run the Evaluation module. **Figure 5** shows a mechanism to run SWAP-GA with the Population Distribution model. The GASWAP module, running as a Ninf-G client, dispatches a set of populations to the remote PC cluster. In the remote PC cluster, the Evaluation module, running as a Ninf-G server, performs an evaluation for each population. The computation inside the PC clusters is distributed among computing nodes through the local batch scheduler.

The advantage of the Population Distribution model is that it exploits maximum parallelism, which is defined by the population number. The drawback of this model is its large communication overhead. The drawback might significantly degrade performance on the Grid with a low network performance. However, we include the evaluation of this model in this paper, because it is worth showing

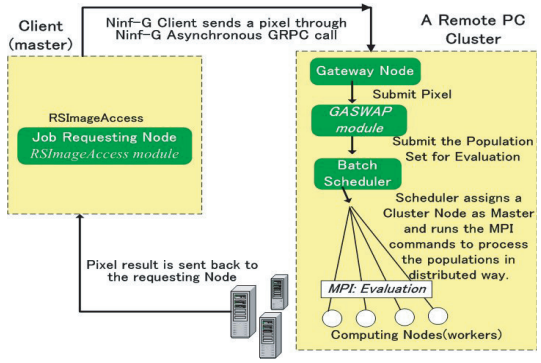


Fig. 6 The hierarchical distribution model.

the performance comparison with other models.

3.3 The Hierarchical Distribution Model

So far, the above two approaches are running with Ninf-G, whereas this third approach is implemented with combined Ninf-G and MPI and called the Hierarchical Distribution model. The idea of the Hierarchical Distribution model is to distribute the computation of pixels and populations in a hierarchical way. **Figure 6** presents a mechanism to run SWAP-GA with the Hierarchical Distribution model. Here, the master node runs the RSImageAccess module and distributes a pixel to the gateway node, or the master node of the remote PC clusters with Ninf-G to invoke the GASWAP module. After getting the pixel value, the master node (GASWAP) dispatches the populations to worker nodes (inside cluster) using MPI to run the Evaluation modules.

One of the insights behind the Hierarchical Distribution model is that we obtain the advantages of the Pixel Distribution model and the Population Distribution model. We expect that the Hierarchical Distribution model efficiently dispatches pixels to multiple PC clusters with a low communication overhead and employs more parallelism among populations in each PC cluster.

4. Experiments

The proposed distributed SWAP-GA models are implemented on the Grid testbed composed of the computer resources presented in **Table 1**. In the table,

Table 1 The Real Grid testbed.

Machine	Specification
GK, Titech	Pentium III 1 GHz (Single CPU, Not SMP), Red Hat Linux 7.1, RAM 256 MB, Globus 4.0.5, Ninf-G 4.2.1, MPICH 1.2.6, GRASS 6.0.2
Blade cluster, Titech, (Yokohama)	32 Nodes, each Node has dual Pentium III, 1.4 GHz, Red Hat Linux 7.1, RAM: 512 MB, Globus 4.0.5, Ninf-G 4.2.1, MPICH 1.2.4, Local scheduler: SGE, Throughput from GK: about10 MB/sec (measured)
F32 cluster,AIST, (Tsukuba)	260 Nodes (divided into 4 parts), each Node has dual Xeon 3.06 GHz, Red Hat Linux 8.0, RAM: 4 GB, Gigabit Ethernet, Globus 4.0.3, Ninf-G 4.2.0, MPICH 1.2.6, Local scheduler: SGE, Throughput from GK:about 6.7 MB/sec (measured)
Kuruwa cluster, NII, (Tokyo)	9 nodes (1 node is server and 8 nodes are workers), All nodes have dual AMD dual core Opteron 2.4 GHz, Cent OS 5 Linux 2.6, RAM: 4 GB (2 GB x2),Globus 4.0.5, Ninf-G 4.2.1, Gigabit Ether net, MPICH: 1.2.7, Local scheduler:SGE, Throughput from GK: about10 MB/sec (measured)

GK is used as the client node, or the requesting node, throughout the experiments. It is also used as the gateway node, or the master node, of the Blade cluster when a job is submitted to the Blade cluster. The Blade cluster is dedicated for this research experiment, while the Kuruwa and F32 clusters are shared with the external workload. In this section, we first present the experimental results to run SWAP-GA on a single site and analyze the results. Then, we show the results on multiple sites. All single site experiments with distributed SWAP-GA models were conducted with the data of 15 pixels, 60 populations and 10 generations.

4.1 Results in a Single Site

Figure 7 shows the running time of the SWAP-GA application using a single PC cluster, or the Blade cluster. Here, Computing Nodes in the x axis indicate the number of computing nodes in the Blade cluster. The number does not include the client node. The Population Distribution model exhibits the worst

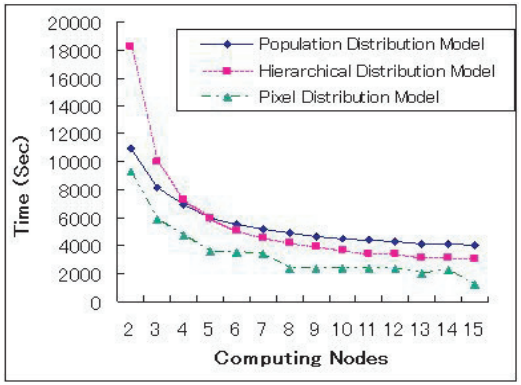


Fig. 7 The running time curves on the blade cluster.

performance of the three distribution models due to its overhead to frequently invoke Ninf-G calls for populations. Typically, the client PC performs slower than the server PC. GK is also used as a client machine and it performs a slower SWAP evaluation time (2.73 sec) than any Blade cluster computing nodes (1.8 sec). In particular, with the Population Distribution model, it is one of the reasons for creating a degrading impact on the performance. However, the Hierarchical Distribution model makes the SWAP-GA evaluation environment more generic and server oriented. Whereas, all SWAP evaluation will be executed in the cluster nodes.

Additionally, the Hierarchical Distribution model reduces the number of Ninf-G calls and increases the performance compared with the Population Distribution model. In this model, pixels are dispatched to master nodes in the cluster sites by Ninf-G and the populations are distributed among computing nodes in the PC cluster through MPI. To execute this model, the number of required computing nodes in a PC cluster is at least two, one master node for dispatching jobs and the other for the computing node for executing jobs. The time curve going down highlights that the parallel version works well and the total running time performs better than the Population Distribution model.

Although, the Hierarchical Distribution model decreases the total running time more than the Population Distribution model, the parallel performance is not in

Table 2 Running time of SWAP-GA on the blade cluster.

Strategy	GK Node
Serial	27,026(sec)

Strategy	# of Computing Nodes in Blade Cluster		
	1	2	15
The Population Distribution Model	19,745 (sec)	10,941 (sec)	4,031 (sec)
The Hierarchical Distribution Model	-	18,216 (sec)	3,152 (sec)
The Pixel Distribution Model	17,382 (sec)	9,276 (sec)	1,218 (sec)

such a desirable state. To improve the parallel performance more, it is necessary to decrease the communication overhead and increase the workload in computing nodes. To fulfill these purposes, the Pixel Distribution model performs best.

The total running time of the Pixel Distribution model at Computing Nodes 5, 6, and 7 is approximately the same because they took the same amount of time to complete the whole jobs. For five nodes, 15 pixels are distributed among five nodes and each node has a balanced workload (three pixels per process). However, for Computing Nodes 7, the 15 pixels are distributed among seven nodes and except one node (which processes three pixels), other six nodes process two pixels. In both cases the master node needs to wait for approximately the same amount of time to complete three pixels serially.

4.2 Discussion of the Results in a Single Site

Table 2 presents the running time comparison of the SWAP-GA serial model (running on the GK node) with the parallel SWAP-GA models with Computing Nodes 1, 2 and 15 on the Blade cluster. With 15 nodes, the performance of parallel models is improved. According to the distribution models, the Pixel Distribution model performs better than others. In the Pixel Distribution model, the dispatched workload (one whole pixel evaluation) is bigger and the communication overhead is hidden through the computing workload. Whereas, in the Population Distribution model, Ninf-G calls happen frequently (once to evaluate the assigned populations set for each generation) and the workload to the computing nodes is not sufficient to gain the efficient parallelism.

Additionally, Ninf-G takes some time for each RPC to establish and to close the session with computing nodes. On the other hand, to reduce the Ninf-G session establishment cost, the Hierarchical Distribution model is presented where (inside the cluster) MPI reduces the session establishment cost (that was taken by Ninf-G) and the performance is improved. However, the performance of the Hierarchical model is not superior to the Pixel Distribution model. The same number of Ninf-G calls are conducted in both the Pixel Distribution and the Hierarchical Distribution models. However, the Hierarchical model takes some time for MPI communication.

Table 3 Estimated serial running time.

Measured Running Time (sec)	Estimated Running Time (sec)
27,026	27,027

4.3 Running Time Estimation for Single Site

In order to estimate the running time of the parallel SWAP-GA models on different settings, we created the preliminary analytical models to estimate the running time in a single site. Those models will help users to trace the parallel behavior, e.g., speed up and system accuracy, of the application and compare it with a real situation. In addition, the analytical model will help users to choose the most suitable strategy according to the available dataset.

4.3.1 Estimation of Serial Running Time

The serial running time of SWAP-GA is derived by the formula (1). The number of pixels (NPixel), the number of populations (NPop), the number of generations (NGen) and one SWAP evaluation time (SWAPt) are the major components. We conducted a preliminary experiment to see the major components.

$$\mathrm{NPixel} \cdot \mathrm{NPop} \cdot \mathrm{SWAPt} \cdot (\mathrm{NGen} + 1)\,[\mathrm{sec}] \quad (1)$$

The Total Workload used in the experiment is composed of 15 pixels, 60 populations and 10 generations. GK takes 2.73 sec to evaluate each SWAP evaluation. **Table 3** shows the comparison between the estimated running time by the formula (1) and measured running time on the Blade cluster for serial SWAP-GA. According to Table 3, the accuracy of the equation is highly acceptable.

Increasing unknown parameters (number of genes) of SWAP-GA will append

Table 4 Parameters for running time estimation.

Term	
Npixel	number of pixels
NPop	number of populations
NGen	number of generations
SWAPt	one SWAP evaluation time
NSlave	number of slaves
TTime	total running time
TExe	total computation time
TCom	total communication time
TPixele	total time to extract pixel information from image
MPI_c	time for a single MPI communication
NG_c	time for a single Ninf-G communication
G_t	GRAM invoke time

a large multi-dimensional search space and obtain a larger time to assimilate. Additional population and generation numbers may require the solving of such problems. However, it will also have an additional effect on the SWAP running time. As SWAPt holds a single SWAP evaluation time, the number of genes is not considered as components to form the analytical model.

4.3.2 Estimation of Parallel Running Time

We applied the same mathematical formulation for the Hierarchical Distribution model (implemented in a single site). The estimated running time for the Hierarchical Distribution model is derived by the formulae (2)–(4). **Table 4** shows the meanings of variables in the formulae. The total running time is the sum of the total computation time and the total communication time. The former is calculated by summing the computation time of the sequential part, the time of the parallel part, and the time to extract pixel information from the target image. The latter is the sum of MPI communication time inside the PC cluster and communication time to invoke Ninf-G calls. Our preliminary experiment on the Blade cluster shows that SWAPt = 1.8 sec, MPI_c = 0.1 sec and NG_c = 0.6 sec. Usually, in the Blade cluster the GRAM invoke time (G_t) is from 2 sec to 6 sec. In our experiment we assumed G_t = 3 sec as a constant. TPixele, the total time to extract pixel information from the image, is calculated by multiplying the total number of pixels with one pixel extraction time (0.13334 sec).

The Hierarchical Distribution model is a combined model of the Pixel and Population Distribution models. So, it is also possible to implement the same running time equation for both the Population Distribution and the Pixel Distribution models. In the case of the Population Distribution model, to calculate the total time (TTime), TExe will be calculated by using the same equation as formula (3) with gateway node SWAPt for the sequential part. The communication time (TCom) will be calculated by using formula (5), which assumes that the client node sends a set of all population at one Ninf-G invocation. The running time for the Pixel Distribution model is derived by the following formulae (6) and (7). In formula (6) one extra generation is added because of the population initialization (sequential part in formula (3)).

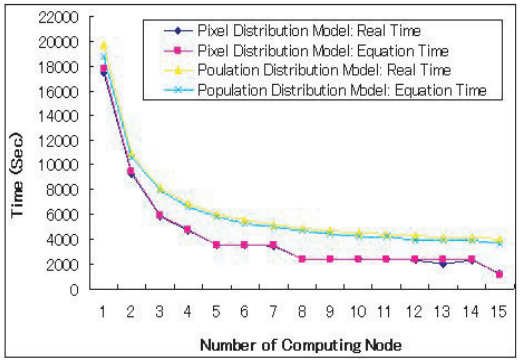


Fig. 8 Comparison between estimated and measured running time on the Blade cluster: the pixel and the population distribution model.

Figure 8 presents the experimental results to show the good accuracy of the running time estimation for both the Pixel and the Population Distribution model. **Figure 9** compares the estimated running time with the measured time on the Blade cluster for different settings presented in **Table 5** and shows that the results of the estimation model match the measured results.

$$\mathrm{TTime} = \mathrm{TExe} + \mathrm{TCom}\,[\mathrm{sec}] \tag{2}$$

$$\mathrm{TExe} = \mathrm{NPixel} \cdot \mathrm{NPop} \cdot \mathrm{SWAPt} + \mathrm{NPixel} \cdot \left\lceil \frac{\mathrm{NPop}}{\mathrm{NSlave}} \right\rceil \cdot \mathrm{NGen} \cdot \mathrm{SWAPt} + \mathrm{TPixele}\,[\mathrm{sec}] \tag{3}$$

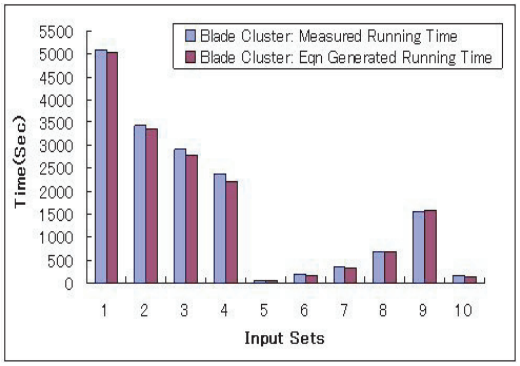


Fig. 9 The hierarchical distribution model equation time performance in the Blade cluster.

Table 5 Experimental setting for estimation of parallel running time: The hierarchical distribution model.

InputSet	NPixel	NGen	NPop	Computing Node
1	15	10	60	6
2	15	10	60	11
3	15	10	60	16
4	15	10	60	31
5	1	10	10	6
6	1	10	30	6
7	1	10	60	6
8	1	10	120	6
9	1	30	120	6
10	1	30	10	6

$$\mathrm{TCom} = \mathrm{NPixel} \cdot \mathrm{NGen} \cdot \left\lceil \frac{\mathrm{NPop}}{\mathrm{NSlave}} \right\rceil \cdot \mathrm{MPI_c} + \mathrm{NPixel} \cdot \mathrm{NG_c} + \mathrm{G_t}\,[\mathrm{sec}] \quad (4)$$

$$\mathrm{TCom} = \mathrm{NPixel} \cdot \mathrm{NGen} \cdot \mathrm{NG_c} + \mathrm{G_t}\,[\mathrm{sec}] \quad (5)$$

$$\mathrm{TExe} = \left\lceil \frac{\mathrm{NPixel}}{\mathrm{NSlave}} \right\rceil \cdot \mathrm{SWAPt} \cdot \mathrm{NPop} \cdot (1 + \mathrm{NGen}) + \mathrm{TPixele}\,[\mathrm{sec}] \quad (6)$$

$$\mathrm{TCom} = \left\lceil \frac{\mathrm{NPixel}}{\mathrm{NSlave}} \right\rceil \cdot \mathrm{NG_c} + \mathrm{G_t}\,[\mathrm{sec}] \quad (7)$$

4.4 Experiments on Multiple Sites (Emulated Grid Testbed and Real Grid Testbed)

The results in section 4.3 show that the Pixel stribution model and the Hierarchical Distribution model are effective. Firstly, we implemented both models on the Emulated Grid testbed 32) with multiple gateway nodes (g1, g2, g3, g4). The Blade cluster's computing nodes (16 nodes) are partitioned into four and dedicated to each gateway node. Another node g0 has been used as client to submit Ninf-G job invocation to the gateway nodes. The gateway nodes are configured similar to the GK node. Secondly, both models are implemented in the Real Grid testbed with multiple PC clusters (Blade, F32 and Kuruwa clusters). For both cases we compared the running times between the Pixel and Hierarchical Distribution model.

In our analytical model, we assume that $\mathrm{G_t}$ includes waiting time in a local batch queue. This assumption works well in the discussion for the single site, or the Blade cluster, because the Blade cluster is dedicated to our experiment and waiting time in the local batch queue is negligible. The assumption has a problem when we use clusters shared with the external workload, or the Kuruwa and F32 clusters. However, accurate estimation of waiting time in a batch queue is still an open issue; we will leave this issue for future work. In this section, we defined $\mathrm{G_t}$ for both clusters as follows:

We determined that $\mathrm{G_t}$ in the F32 cluster is from 8 sec to 20 sec and in the Kuruwa cluster is from 4 sec to 12 sec. In our case, we took $\mathrm{G_t} = 8$ sec for the F32 cluster and $\mathrm{G_t} = 6$ sec for the Kuruwa cluster as the constant value. SWAPt = 1.5 sec in the F32 cluster and SWAPt = 1.2 sec in the Kuruwa cluster. $\mathrm{MPI_c}$ and $\mathrm{NG_c}$ are assumed the same as the Blade cluster. SWAPt in the F32 cluster is a bit slower compared with its speed ratio with the Blade cluster. One of the reasons is that the F32 cluster CPUs are not always dedicated to a particular job. CPUs are shared with other jobs too. That decreases the time performance of SWAPt. In addition, SWAP evaluation requires many I/O operations as well as Disk R/W operations and those take most of its time when processing. So the processor's clock speed will not always make desirable speed up in SWAPt.

Table 6 Execution time on multiple sites: Emulated Grid.

NPixel	# of Computing Nodes				Pixel Model Total Time	Hierarchical Model Total Time
	g1	g2	g3	g4	(sec)	(sec)
	1	0	0	0	1,169	-
1	2	0	0	0	-	1,188
	4	0	0	0	-	472
4	4	0	0	0	1,226	-
	4	4	4	4	-	512
8	4	4	0	0	1,206	-
	4	4	4	4	-	975
16	4	4	4	4	1,218	1,809
32	4	4	4	4	2,448	3,569
64	4	4	4	4	4,870	7,037

4.4.1 Discussion of the Results on Multiple Sites

The experimental results on the Emulated Grid testbed were conducted with 20 generations and 30 populations (presented in **Table 6**). The pixel number (NPixel) is taken from the same RS images (section 2.2) with Synthetic field data. From Table 6, when the pixel number is one then the Pixel Distribution model can only run with one computing node. Whereas, the Hierarchical Distribution model performs better with four computing nodes. However, with two computing nodes (one master and one slave) the Hierarchical Distribution model does not perform better than the Pixel Distribution model because of the additional MPI communication overhead. It is clear from the table that the Hierarchical model performs well, when the pixel number is less than the computing nodes number.

Table 7 presents the experimental results on the real Grid testbed with the Hierarchical Distribution and the Pixel Distribution model with 15 pixels, 10 generations and 60 populations. The best performance for the Pixel Distribution model was achieved in the single site experiment (1,378 sec). However, the Grid with more CPU power makes the Hierarchical model performance (922 sec) better than the best performance of the Pixel Distribution model.

The results in Table 6 and Table 7 highlight the major drawback of the Pixel Distribution model, when the pixel amount is diminutive compared to the computing nodes number. For this particular workload (with 15 pixels, 10 generations and 60 populations) more than 15 nodes will not create any advanced effects on the Pixel Distribution model whereas the door is open for the Hierarchical model to use more than 15 computing nodes (at most 60 nodes in each cluster).

Table 7 Execution time on multiple sites: Real Grid.

Execution Strategy	# of Computing Nodes			Total Time
	Blade Cluster	F32 Cluster	Kuruwa Cluster	(sec)
	10	5	-	3,189
The	5	10	-	3,329
Hierarchical	21	31	-	1,193
Distribution	16	16	8	1,004
Model	16	21	8	922
	0	15	-	3,519
The Pixel	5	10	-	2,986
Distribution	10	5	-	2,750
Model	15	0	-	1,378

However, the Hierarchical Distribution model performance greatly depends on the number of clusters as well as the number of computing nodes in each cluster. So, when the cluster number is equal to the pixel number and the computing node number inside each cluster is equal to the population number, it may provide the best performance for the Hierarchical Distribution model.

$$\left\lceil \frac{\text{NPixel}}{\text{N\_Cluster}} \right\rceil \cdot \left\lceil \frac{\text{NPop}}{\text{Avg\_NSlave}} \right\rceil < \left\lceil \frac{\text{NPixel}}{\text{T\_NSlave}} \right\rceil \cdot \text{NPop} \quad (8)$$

In our rough estimate, the Hierarchical Distribution model performs better than the Pixel Distribution model when formula (8) is satisfied. In formula (8), N_Cluster is the number of available clusters, Avg_NSlave is the average number of available worker nodes in each cluster and T_Nslave is the total number of available worker nodes.

4.4.2 Time Estimation of the Hierarchical Distribution Model on Real Grid

The total running time of the Hierarchical Distribution model on the Grid, T (sec), can be estimated by the maximum evaluation time required in the available clusters.

$$T = \mathrm{Max}\left[\sum_{i=1}^{nc} CT_i \left(\begin{array}{l} NPixel_i, NGen, NPop, \\ NSlaves_i, NGc_i, MPIc_i, \\ SWAPt_i, Gt_i, TPixele \end{array} \right) \right]$$

$$\mathrm{Speed_i} = \frac{1}{\mathrm{CT_i^1}} \,[\mathrm{pixel/sec}] \tag{9}$$

$$\mathrm{NPixel_i} = \frac{\mathrm{TotalPixel}}{\sum_{k=1}^{n} \mathrm{Speed_k}} \cdot \mathrm{Speed_i} \,[\mathrm{pixel}] \tag{10}$$

Here, nc is the total number of available clusters, $\mathrm{NPixel_i}$ is the number of pixels evaluated on $\mathrm{i^{th}}$ cluster, $\mathrm{NSlaves_i}$ is the available computing nodes in $\mathrm{i^{th}}$ cluster, $\mathrm{SWAPt_i}$ is one SWAP execution time. $\mathrm{NGc_i}$, $\mathrm{MPIc_i}$ are the communication time and $\mathrm{Gt_i}$ is the GRAM invoke time inside $\mathrm{i^{th}}$ cluster. NGen, NPop and TPixele will be the same for every cluster. $\mathrm{CT_i}$ is the running time calculation function for $\mathrm{i^{th}}$ cluster and $\mathrm{CT_i}$ value will be calculated by adding formulae (3) and (4). For each individual cluster, all the above parameter values must be estimated beforehand, as we did for the Hierarchical Distribution model single site (Blade cluster) experiments. However, the challenging part of the $\mathrm{CT_i}$ function is to compute the $\mathrm{NPixel_i}$ for the $\mathrm{i^{th}}$ cluster. Formula (10) is formed to generate the $\mathrm{NPixel_i}$. In formula (9), $\mathrm{CT_i^1}$ is evaluated from $\mathrm{CT_i}$ function with $\mathrm{NPixel_i} = 1$ and other given parameters value. **Table 8** presents the estimated running time with $\mathrm{NPixel_i}$ derived from formula (10) and the measured running time on multiple sites (real Grid).

Because $\mathrm{NPixel_i}$ derived from (10) is not an integer, we rounded the value as shown in Table 8, where the fractional part of $\mathrm{NPixel_i}$, greater than 0.4 will be rounded to the next integer or other wise discarded. However, the 7^{th} row in Table 8, is a slightly special situation, as each value is a candidate for rounding to the next integer. However, the situation can be tackled by rounding $\mathrm{NPixel_i}$ to the next integer according to the priority basis from the highest $\mathrm{Speed_i}$ to the lowest $\mathrm{Speed_i}$ cluster. The workload used in these experiments is 15 pixels, 60 populations and 10 generations.

For the most part, the results in Table 8 show that the accuracy of the estimation model exhibits acceptable performance (90% accuracy) except on the 5^{th} row. The results on the 5^{th} row exhibit the limitation of our analytical model. Our model does not assume waiting time in a batch queue. However, the assumption significantly degrades the accuracy where PC clusters are shared with the external workload. So, the estimation of queue waiting time is still an open question. We leave this issue for our future research.

Table 8 Comparison between the estimated and measured running time on multiple sites: the hierarchical distribution model.

BN	FN	KN	Actual NPixel_i			RT	ET	Estimated NPixel_i		
			B	F	K			B	F	K
31	31	0	7	8	0	1,126	1,006	6.8	8.2	0
16	16	0	7	8	0	1,354	1,298	6.8	8.1	0
21	21	0	7	8	0	1,229	1,136	6.8	8.2	0
21	31	0	6	9	0	1,193	1,062	6.4	8.6	0
16	31	0	6	9	0	1,500	1,110	5.9	9.0	0
11	11	8	5	5	5	1,088	1,117	4.5	5.4	5.2
16	16	8	5	6	4	1,004	936	4.8	5.7	4.5
16	21	8	5	6	4	922	927	4.6	6.1	4.3

BN=#computing nodes on Blade cluster(B)
FN=#computing nodes on F32 cluster(F)
KN=#computing nodes on Kuruwa cluster(K)
RT=Real running time
ET=Estimated running time

5. Conclusion and Future Work

This paper presented the implementation of SWAP-GA on the Grid and discussed the impact of parallelization on the performance. Three different implementation strategies for the SWAP-GA model were successfully developed using Ninf-G GridRPC framework on the Grid. Increasing the computing nodes number improves the performance of the parallel SWAP-GA models. The Pixel Distribution model and the Hierarchical Distribution model performances improve by providing more computational power with respect to pixel and population number. While the Pixel Distribution model exhibits the lowest communication overhead and shows the best performance on some settings, the Hierarchical Distribution model is also a preferable model to implement SWAP-GA in Grid. The situation, where the Hierarchical Distribution will perform better than the Pixel

Distribution model can be traced by formula (8). Additionally, the Hierarchical Distribution model is capable of utilizing all the available resources inside a Grid testbed.

This paper also presented the analytical model to estimate the running time of SWAP-GA, and the experimental results show the accuracy of the model. The current model assumes that computing resources are dedicated or are lightly loaded, i.e., the model does not assume waiting time in a batch queue. Estimation of queuing time is an issue that should be solved, and we leave this issue for future work.

The web based portal for SWAP-GA on a distributed platform is required and it will be developed in the near future. Furthermore, GA can also be replaced by any probabilistic calculative model such as MCMC (Markov Chain Monte Carlo).

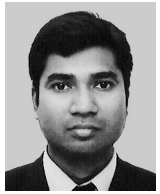
Acknowledgments The authors would like to acknowledge Dr. Yann Chemin (IRRI, Philippine) and Dr. Kiyoshi Honda (AIT, Thailand) for their advice and diligent support on porting the serial SWAP-GA model successfully. The authors would also like to thank to Advanced Institute of Science and Technology for supporting access to their F32 cluster.

References

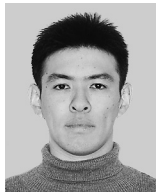
1) Honda, K. and Ines, A.V.M.: Genetic Algorithms in Quantifying Water Management and Agricultural Practices at the Sub-Pixel Level, *Proc. 6th International Conference on Hydroinformatics*, Vol.2, pp.1319–1325 (2004).
2) Ines, A.V.M.: Improved Crop Production Integrating GIS and Genetic Algorithms, PhD Thesis, AIT, Bangkok, Thailand. AIT Diss No.WM-02-01, 2002.
3) Srinuandee, P.: Data Assimilation in SWAP Model based on a NDVI-LAI Relationship obtained by Field Survey, Master's thesis, RS&GIS FoS, Asian Institute of Technology, Khlong Luang, Thailand (2005).
4) Michalewicz, Z.: *Genetic Algorithms + Data Structures= Evolution Programs*, 3rd Ed. Rev. and extended ed, Springer, USA (1996).
5) Goldberg, D.E. and Deb, K.: *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms*, Foundations of Genetic Algorithms, Morgan Kaufman, San Mateo, Calif., 69–93 (1989).
6) Chemin, Y., Honda, K. and Ines, A.V.M.: Genetic algorithm for assimilating remotely sensed evapotranspiration data using a soil-water- atmosphere — plant model. Implementation issues, *Proc. Open Source Free Software GIS — GRASS users conference* (2004). http://gisws.media.osaka-cu.ac.jp/grass04/viewpaper.php?id=27.
7) Chemin, Y. and Honda, K.: Spatio-temporal Fusion of Rice Actual Evapotranspiration with Genetic Algorithms and an Agro-hydrological Model, *IEEE Transactions on Geoscience and Remote Sensing*, In Print (Accepted on 22 May 2006) (2006).
8) ASTER Image Webpage (http://asterweb.jpl.nasa.gov/)
9) MODIS Image Webpage (http://modis.gsfc.nasa.gov/)
10) Global Earth Observation Grid (GEO Grid), http://www.geogrid.org/
11) GOEN (The Geosciences Network) http://www.geongrid.org/
12) Akhter, S., Honda, K., Chemin, Y. and Uthayopas, P.: Exploring Strategies for Parallel Computing of RS Data Assimilation with SWAP-GA, *Journal of Computer Science*, Vol.3, No.1, pp.47–50 (2007).
13) Akhter, S., Jangjaimon, I., Chemin, Y., Uthayopas, P. and Honda, K.: Development of a GRIDRPC tool for Satellite Images Parallel Data Assimilation in Agricultural Monitoring, *International Journal of Geoinformatics*, ISSN 1686-6576, Vol.2 No.3, (2006).
14) Nakada, H., Matsuoka, S., Seymour, K. and Dongarra,J.: GridRPC: A Remote Procedure Call API for Grid Computing, *GWD-I (Informational), Advanced Programming Models Research Group*, http://www.eece.unm.edu/?apm/docs/APM GridRPC0702.pdf, July 2002.
15) Grass Development Team, 2005. Grass GIS Webpage, http://grass.itc.it/.
16) Dong, S. and Hu, Q.: Building remote sensing database on grid, Geoscience and Remote Sensing Symposium, *IGARSS '05 Proceedings*, *IEEE International*, pp.25–29, Vol.1, p.3 pp.-ISBN: 0-7803-9050-4 (2005).
17) Rokos, D.Kl. and Armstrong, M.P.: Experiments in the identification and extraction of terrain features using a PC-based parallel computer, *Photogrammetric Engineering and remote Sensing*, Vol.64, No.2, pp.135–142 (1998).
18) Petrie, G.M., Dippold, C., Fann, G., Jones, D., Jurrus, E., Moon, B. and Perrine, K.: Distributed Computing Approach for remote Sensing Data, *Proc. 34th Symposium on. the Interface*, Montreal, Quebec, Canada.
19) Yang, C.T., Chang, C.L., Hung, C.C. and Wu, F.: Using a Beowulf cluster for a remote sensing application, *2nd Asian Conference on remote Sensing proceeding*, Singapore, 5–9 November (2001).
20) Shen, Z., Luo, J., Huang, G., Ming, D., Ma, W. and Sheng, H.: Distributed computing model for processing remotely sensed images based on grid computing, *Information Sciences*, Vol.177, Issue 2, 15, pp.504–518 (Jan. 2007), Online version available at www.sciencedirect.com.
21) MARS (Monitoring Agriculture through Remote Sensing techniques), http://www.marsop.info/scripts/mars.exe?page=bg
22) Agricultural Geo-Referenced Information System (AGIS), http://www.agis.agric.za/agisweb/agis.html.

23) Ninomiya, S., Laurenson, M. and Kiura, T.: Network Computing for Agricultural Information Systems-GRID for Agricultural Decision Support, http://www.google.com/searchhl=en&q=Network+COmputing+for+Agricultural+Information+Systems.
24) International Water Management Institute (IWMI), http://www.iwmi.cgiar.org/.
25) Dorji, M.: Integration of SWAP model and SEBAL for evaluation of on — farm irrigation scheduling with minimum field data, Enschede, ITC, 100 p (2003).
26) Jhorar, R.K.1, Bastiaanssen, W.G.M., Feddes, R.A. and Van Dam, J.C.: Inversely estimating soil hydraulic functions using evapotranspiration fluxes:, *Journal of Hydrology*, Elsevier Publication, Vol.258, No.1, pp.198–213(16), 28 February (2002).
27) Van Dam, J.C., Huygen, J., Wesseling, J.G., Feddes, R.A., Kabat, P., Van Waslum, P.E.V., Groenendjik, P. and Van Diepen, C.A.: Theory of SWAP version 2.0: Simulation of water flow and plant growth in the Soil-Water-Atmosphere-Plant environment, Technical Document 45, Wageningen Agricultural University and DLO Winand Staring Centre, The Netherlands (1997).
28) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumarn, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC -based Programming Middleware for Grid Computing, *Journal of Grid Computing*, Vol.1, pp.41–51, (2003).
29) Casanova, H. and Dongarra, J.: Netsolve: A Network Server for Solving Computational Science Problems, *Proc. Super Computing '96*, (1996).
30) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit (1997). http://www.globus.org
31) Message Passing Interface (MPI), (2007). http://www-unix.mcs.anl.gov/mpi/
32) Kasai, T., Nishimura, M., Maeda, T., Osawa, K. and Aida, K.: Network Emulation for Evaluation of Grid Applications, *IPSJ Trans. on Advanced Computing Systems*, Vol.48, No.SIG13(ACS19), pp.145–155 (2007) (in Japanese).

(Received January 29, 2008)
(Accepted May 5, 2008)



Shamim Akhter received his B.S. and M.S. degrees from the American International University Bangladesh (AIUB) in 2002 and the Asian Institute of Technology in 2005 respectively. He joined the Department of Computer Science in AIUB as a faculty member in 2002 and since 2005 he has been an assistant professor. He was a research associate at the RS and GIS FoS, Asian Institute of Technology in 2005. He is now pursuing his Ph.D. degree at the Department of Information Processing, Tokyo Institute of Technology. His current research interests are distributing remote sensing and agriculture applications on parallel or distributed computing, evolutionary algorithms and models for their parallelization. He is a member of IPSJ.



Kiyoshi Osawa received his B.E. degree from the Tokyo Institute of Technology in 1998 and his M.S. degree from the University of Tokyo in 2001. He is now a Post Doctorate Researcher at the Global Scientific Information and Computing Center in Tokyo Tech. His research interests are numerical calculation, parallel computing, and sabermetrics. He is a member of IPSJ, ORSJ, and JSIAM.

Motokazu Nishimura received his B.S. degree from Kanagawa University in 2006 and M.E. degree from the Tokyo Institute of Technology in 2008. His research was on the Virtual and Emulated Grid Testbed. He is now an employee at IBM Japan Systems Engineering Co., Ltd.

Kento Aida received his B.E., M.E., and Dr. Eng. degrees from Waseda University in 1990, 1992, 1997, respectively. He became a research associate at Waseda University in 1992. He joined the Tokyo Institute of Technology and became a research scientist at the Department of Mathematical and Computing Sciences in 1997, an assistant professor at the Department of Computational Intelligence and Systems Science in 1999, and an associate professor at the Department of Information Processing in 2003, respectively. He was a researcher at PRESTO, Japan Science and Technology Agency (JST) from 2001 through 2005. He was a research scholar at the Information and Computer Sciences Department, University of Hawaii in 2007. He is now a professor at the National Institute of Informatics and a visiting professor at the Department of Information Processing, Tokyo Institute of Technology from 2007. His research interests are parallelcomputing, Grid computing and scheduling. He is a member of IEICE, IEEJ, ACM and IEEE-CS.