2007－HPC－111 (23)
2007／8／2

Performance Evaluation of Distributed SWAP-GA Models with GridRPC

Shamim Akhter† Kiyoshi Osawa† Kento Aida††

†Tokyo Institute of Technology
†† National Institute of Informatics

Abstract 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). Still, to run the SWAP-GA model in 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, it is not an easy or straight forward task to port the SWAP-GA model in Grid environment. This paper discusses different strategies of implementing SWAP-GA model with GridRPC and presents their performance evaluation.

1. Introduction

Recently, a demand for efficient and effective agricultural products monitoring systems is increasing. Researchers of agriculture try to analyze various information about crops in order to take measures if they had some problems. Specially, when an on-going experiment covers 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. Crop models calculate missing information by analyzing crop information with real fields' experimental data.

Ines and Honda [1] developed an assimilation methodology 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. Afterward a real coded GA [5], was applied by Chemin et al. [6]. Furthermore, Chemin worked with the SWAP-MuliGA model (Modified SWAP-GA) [7] and that was successfully implemented with a new methodology to assimilate RS evapotranspiration (ETa) data for satellite images observed by ASTER (http://asterweb.jpl.nasa.gov/) and MODIS (http://modis.gsfc.nasa.gov/).

The problem of these methodologies is that they require a huge amount of processing times [8]. Grid Computing will be a solution for this circumstance. However, it is not an easy or straight forward job to implement SWAP-GA model on Grid environment. There is a variety of Grid implementation methodologies for SWAP-GA model on the Grid. However, their performances have not been discussed. This paper presents three implementation methods, the Pixel Distribution, the Population Distribution and the Hierarchical Distribution for SWAP-GA model on the Grid. These methodologies use GridRPC [9] as the programming framework but ways of task distribution are different. The experimental results show the advantage of GridRPC based distributed SWAP-GA models by evaluating their performance on time domain.

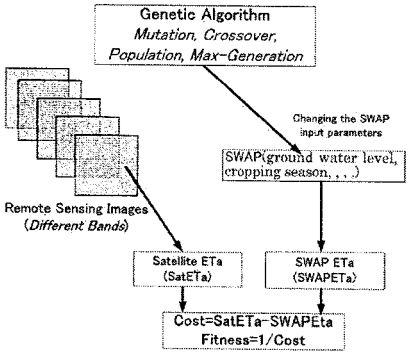

Fig.1. SWAP-GA Model Structure Diagram

2. Background

2.1 SWAP-GA Framework

SWAP-GA is an assimilation methodology of the SWAP model with RS images using GA. Fig.1 shows an overview of

SWAP-GA.

The data set used in this research is MODIS (Moderate Resolution Imaging Spectroradiometer) image of 1000x1000m pixel size (the smallest dimension of an Image), from 8-days composite standard products (surface reflectance at 500x500m, emissivity and land surface temperature at 1000x1000m). The time period of MODIS images processed is from January 1st, 2002 to May 1st, 2002. However, the SWAP-GA evaluation on full MODIS image is not possible due to the required field data constraint. In this research only 15 pixels values have been extracted from the real image.

Evapotranspiration (ETa) is one of the indicators of crop productivity and can be estimated from satellite remote sensing. GRASS GIS (Geographic Resources Analysis Support System) [10] is open source software for Geographical Information System (GIS). It has been used to process RS images and evaluated the ETa images [7]. The SWAP model can also produces ETa by solving the Penman-Monteith equation [11]. Additionally, the SWAP model requires the metrological data, the crop description data, and soil-hydrologic field data. Some parameters, e.g. ground water level and cropping season time extent of crop fields, can not be visible directly through RS images, while those parameters are the input for the SWAP model to generate the ETa value. The unknown parameters are searched by GA in order to assimilate ETa generated by the SWAP model (SwapETa) with ETa generated by RS data (SatETa).

Each population will provide a cost value (|SatETa-SwapETa|) and transport that to fitness value as 1/cost. Higher Fitness for a population indicates good assimilation and has a better chance to survive for the next generation. 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 a predefined number of iterations, called as maximum generation.

2.2 GridRPC

The GridRPC is one of the familiar programming models for a Grid Application based on Remote Procedure Call (RPC) mechanism tailored for Grid system or architecture. It is middleware that provides remote library access and task parallel programming over Grid environment. An application program using the GridRPC consists of a main code and stub codes. When the main program invokes an RPC, the corresponding stub program runs on a remote machine. The stub program must be prepared on the remote machine before the RPC is invoked or be shipped to the remote machine at the same time as the RPC invocation. GridRPC has now been supported by many well known grid programming runtime systems such as Ninf-G [12] and Netsolve [13]. Ninf-G is a reimplementation of the Ninf system [14] on top of the Globus Toolkit [15].

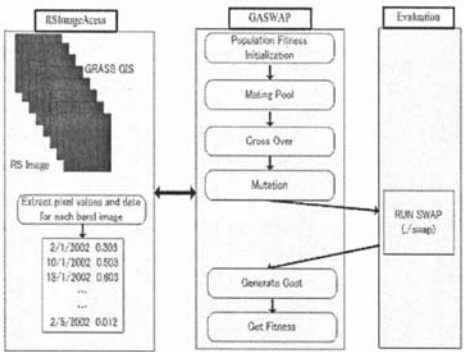

Fig.2. SWAP-GA Working Modules

3. Design and Implementation of Distributed SWAP-GA

The full SWAP-GA executable module is made with RSImageAccess, GASWAP, and Evaluation sub-modules (Fig.2). The RSImageAceess (main.c) module is the main module, where the program starts. This module extracts the pixel value (SatETa) and the date for each band image from GRASS environment, and it calls the GASWAP (gaswap.c) module. GASWAP module is accomplished to run GA and completes the assimilation process. The Evaluation module runs the SWAP executable (swap.exe) for each population and sends the SwapETa values to GASWAP module. An example with 15 pixels image, 60 populations and 10 generations will clear the calling system procedures inside SWAP-GA model. Particularly, in this case, the RSImageAcess module will call the GASWAP module 15 times. For each pixel, the GASWAP module will call the Evaluation module 10 times. For each generation, the Evaluation module executes the SWAP executable 60 times and produces SWAP ETa values for 60 populations.

High demand of distributing behavior is appealed inside the SWAP-GA module. Three different strategies are applied to work the SWAP-GA in distributed manner. These are, i) Pixel Distribution, ii) Population Distribution, iii) Hierarchical Distribution.

3.1 Pixel Distribution

A "pixel" is the smallest dimension of a RS image which indicates the image resolution as DN value. The RSImageAccess module extracts the pixel values sequentially

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 the experiment of this paper, 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) run (Fig.3).

3.2 Population Distribution

Another approach is the Population Distribution. An Evaluation procedure is called sequentially for each population to execute the swap executable. Thus, as like the above scenario only Evaluation module (to evaluate population) can be distributed again through Ninf-G programming platform. Here, the master node will run both the RSImageAccess and the GASWAP module and the worker nodes only run the Evaluation module (Fig. 4).

3.3 Hierarchical Distribution

So far, the above two approaches are running inside Ninf-G programming platform, whereas this third approach is implemented with combined Ninf-G and MPI [17] programming platforms and called as the Hierarchical Distribution model. The idea of the hierarchical distribution model is to distribute computation of pixles and populations is hierarchical way. Here, the master node (Grid Master) will run the RSImageAccess and with Ninf-G (remote procedure call) distribute the pixels to Cluster local master nodes to invoke the GASWAP services. After getting the pixel value, the Cluster master (GASWAP) dispatches the populations to the working nodes (inside Cluster) using MPI to evaluate the Evaluation module (Fig. 5).

4. Experiments

The proposed distributed SWAP-GA models are implemented on the Grid testbed composed of computer resources presented in Table 1. In Blade Cluster, GK serves as the gateway node and the requesting node. GK submits the requested job to the Cluster scheduler (PBS) through Ninf-G. Whereas, GK serves only as requesting node for F32 Cluster and submits the requested job to the Cluster local scheduler (SGE) through Ninf-G. In both Clusters, schedulers then dispatch the jobs to the computing nodes. All experiments with distributed SWAP-GA models were conducted with 15 pixels, 60 populations and 10 generations. According to the performance order (low to high), the models are presented.

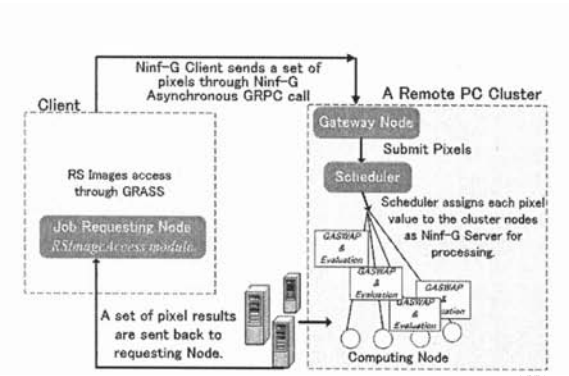

Fig. 3. Pixel Distribution Model

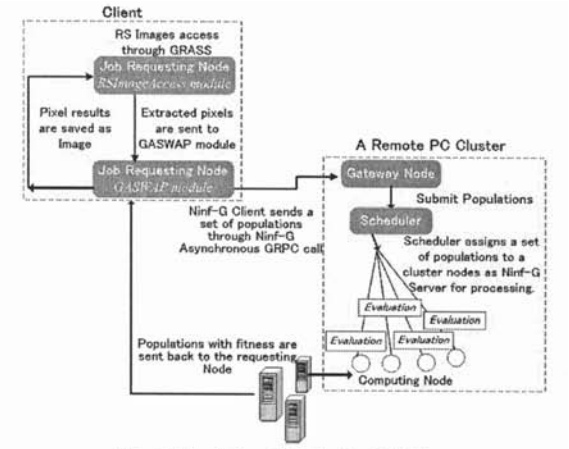

Fig. 4. Population Distribution Model

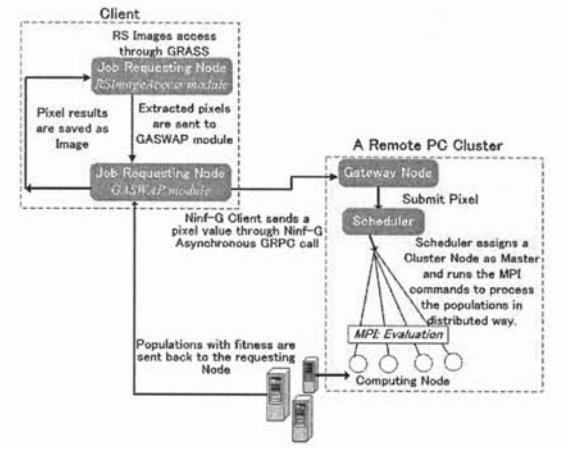

Fig. 5. Hierarchical Distribution Model

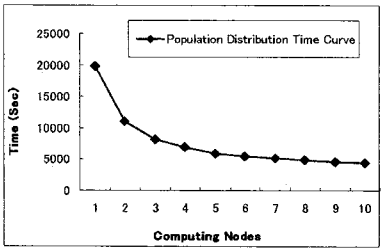

Fig. 6. The Result of Population Distribution Model

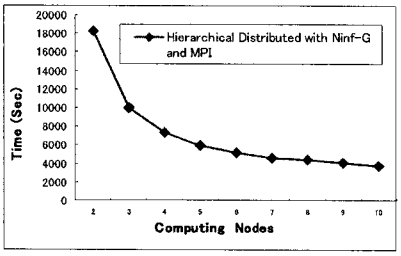

Fig.7. The Result of Hierarchical Distribution Model

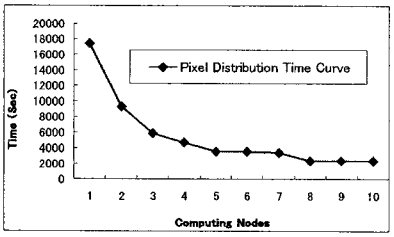

Fig.8. The Result of Pixel Distribution Model

4.1 Population Distribution in a Single Site

The running time curve with increasing computing nodes is generated from the Population Distribution model and is presented in Fig.6. Here, the Computing Nodes 1 highlights that only one node in Blade Cluster has been used to complete the Ninf-G job invocation. Whereas, Computing Nodes 3 means three nodes in Blade Cluster together will run the Ninf-G job invocation in parallel.

4.2 Hierarchical Distribution in a Single Site

To reduce the number of Ninf-G calls and increase the performance of Population Distribution model, this combined model (Ninf-G and MPI) has been implemented. Here, the pixels are distributed to master nodes in Cluster sites by Ninf-G and the populations are distributed among Cluster nodes through MPI. The total running time curve of the Hierarchical

Table 1: The Grid testbed

Terms	Specification
GK (Gateway Node), Titech	Pentium III 1GHz (Single CPU, Not SMP), Red Hat Linux 7.1, RAM: 256MB GRASS 6.0.2
Blade Cluster, Titech	32 Nodes, each Node has dual Pentium III 1.4GHz, Red Hat Linux 7.1, RAM: 512 MB. Globus 4.0.3 Ninf-G 4.1.0 MPICH 1.2.7 Local scheduler: PBS
F32 Cluster, AIST	260 Nodes (divided into 4 partitions), each Node has dual Xeon 3.06 GHz, Linux 8.0, RAM: 4GB, Giga-bit Ethernet. Globus 4.0.3 Ninf-G 4.2.0 MPICH 1.2.6 Local scheduler: SGE

Distribution model with increasing computing nodes number is the concerning issue of Fig. 7. To execute this model, the required computing nodes numbers are at least two (One Cluster master for dispatching jobs and another is Cluster slave for executing jobs). The time curve is going down highlights that the parallel version is working well and the total run time is performing better than the Population Distribution model.

4.3 Pixel Distribution in a Single Site

So far, Hierarchical Distribution model decreases the total running time than the Population Distribution model. However, the parallel performance is not in a desirable state. To improve the parallel performance more, it is needed to decrease the communication overhead and increase the workload in computing nodes. To fulfill these purposes, the Pixel Distribution model has been implemented.

Figure 9 represents the total run time behavior of the Pixel Distribution model. In this figure, the time curve is generated by increasing the computing nodes numbers. In Fig. 9, total running time at Computing Nodes 5, 6, and 7 are approximately same because they took same amount of time to complete the whole job. For five nodes, 15 pixels are distributed among five nodes and each node has balance workload (three pixels to process). However, for Computing Nodes 7, the 15 pixels distributed among seven nodes and all nodes process two pixels except one

node, which will process three pixels. In both cases the master node needs to wait approximately the same amount of time as to complete three pixels serially.

Table 2: Runtime Chart of SWAP-GA models

Implementation Strategy	GK Node	Blade Cluster Computing Nodes		
		1	2	15
Serial SWAP-GA	27026 (sec)	–	–	–
Population Distribution	–	19745 (sec)	10941 (sec)	4031 (sec)
Hierarchical Distribution	–	–	18222 (sec)	3180 (sec)
Pixel Distribution	–	17382 (sec)	9276 (sec)	1218 (sec)

4.4 Discussion of the results in a Single Site

Table 2 presents the run time comparison chart of the SWAP-GA serial model (running in GK node) with the distributed SWAP-GA models with Computing Nodes 1, 2 and 15. With 15 nodes, the distributed models performance is improved. So, parallel SWAP-GA models are gained benefit from the point of time domain. According to the distributed models, the Pixel Distribution model is performing well than others. In the Pixel Distribution model, the dispatched workload (one whole pixel evaluation) is bigger and the communication workload is hidden through the working workload. Whereas, in the Population Distribution model, Ninf-G calls are happened regularly (one time 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 side, to reduce the Ninf-G session establishment cost, the Hierarchical Distribution model is presented where (inside Cluster) MPI reduces the session establishment cost (that was taken by Ninf-G) and the performance is improved. However, the performance of combined model is not superior to Pixel Distribution model. The same amounts of Ninf-G calls are conducted in both the Pixel Distribution and the Hierarchical model. However, the combined version takes some time to establish the MPI running environment. Additionally, the Cluster nodes are sometimes occupied for evaluating populations and the communication among Cluster nodes is repeatedly established (at each generation).

Table 3: Execution Time on Multiple Sites

Implementation Strategy	Blade Cluster Node Number	F32 Cluster Node Number	Total Runtime(sec)
Hierarchical Distribution	10	5	3189
	5	10	3329
	16	31	1500
Pixel Distribution	10	5	2750
	5	10	2986
	15	0	1378
	0	15	3519

4.5 Experiments on Multiple Sites

Additionally, increasing Computing Nodes may reduce runtime behavior of the Hierarchical Distribution model. With this supposition, the model is implemented on Grid Environment (Blade and F32 Clusters) and compared the running time with the Pixel Distribution model.

Table3 presents the experimental results on Grid environment with the Hierarchical Distribution and the Pixel Distribution model. Though, F32 is faster than Blade Cluster according to the specification. Nevertheless, due to the exterior workload and waiting time in the scheduler queue for resource allocation, F32 formulates slower performance. The best performance for the Pixel Distribution model has achieved in Single site experiment (1378 sec). However, the Grid with more CPU power provides the Hierarchical model performance (1500 sec) more closely with the best performance of the Pixel Distribution model. This highlights the major drawback of Pixel Distribution model, when the pixel amount is diminutive comparing to population 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 opened for the Hierarchical model to use more than 15 computing nodes (atmost 60 nodes).

5. Conclusion and Future Work

Three different implementation strategies for the SWAP-GA model were successfully developed and ported on Ninf-G GridRPC framework. Increasing the computing nodes number improves the performance of the SWAP-GA model. The Pixel Distribution model and the Hierarchical Distribution model performances improve in the Grid testbed by providing more computational power with respect to population number. The web based portal for SWAP-GA on distributed platform is required and it will be developed in near future. Furthermore, GA can also be replaced by any probabilistic calculative model such as MCMC (Markov Chain Monte Carlo).

Acknowledgment

The authors would like to acknowledge Dr. Yann Chemin for his advices and diligently support on porting serial SWAP-GA model successfully. Authors also like to thank to Advanced Institute of Science and Technology for supporting access to F32 Cluster.

References

[1] K. Honda and A.V..M. Ines, "Genetic Algorithms in Quantifying Water Management and Agricultural Practices at the Sub-Pixel Level", Proceedings of the 6th International Conference on Hydroinformatics, Volume 2, pp. 1319-1325, 2004.

[2] A.V..M. Ines, "Improved Crop Production Integrating GIS and Genetic Algorithms." *Doctoral Thesis.* 2002, AIT, Bangkok, Thailand. AIT Diss no. WM-02-01.

[3] P. Srinuandee, P., "Data Assimilation in SWAP Model based on a NDVI-LAI Relationship obtained by Field Survey", Msc. Thesis, 2005, Asian Institute of Technology, Khlong Luang, Thailand.

[4] Z. Michalewicz, "Genetic Algorithms + Data Structures= Evolution Programs". *3rd Ed. Rev. and extended ed. Springer.* USA, 1996.

[5] D. E. Goldberg and K. Deb. "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms". Foundations of Genetic Algorithms, Morgan Kaufman, San Mateo, Calif., 69-93, 1989

[6] Y. Chemin, K. Honda, and A.V.M. Ines, "Genetic algorithm for assimilating remotely sensed evapotransiration data using a soil–water– atmosphere – plant model. Implementation issues", Open Source Free Software GIS - GRASS users conference, http://gisws.media.osaka-cu.ac.jp/grass04/viewpaper.php?id=27, 2004

[7] Y. Chemin and K. Honda, "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] S. Akhter, K. Honda, Y. Chemin, and P. Uthayopas, "Exploring Strategies for Parallel Computing of RS Data Assimilation with SWAP-GA", Journal of Computer Science, 3(1): 47-50, 2007, 2007

[9] H. Nakada, S. Matsuoka, K. Seymour and J. Dongarra, "GridRPC: A Remote Procedure Call API for Grid Computing", July 2002, GWD-I (Informational), Advanced Programming Models Research Group, http://www.eece.unm.edu/~apm/docs/APM GridRPC 0702.pdf

[10] Grass Development Team, 2005. Grass GIS Webpage, Http://grass.itc.it/.

[11] J.C. Van Dam, J. Huygen, J.G Wesseling, R. A. Feddes, P. Kabat, P.E.V. Van Waslum, P. Groenendjik, and C.A. Van Diepen, "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.

[12] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumarn and S. Matsuoka,."Ninf-G: A Reference Implementation of RPC –based Programming Middleware for Grid Computing". Journal of Grid Computing, Vol.1, 41-51, 2003

[13] H. Casanova and J. Dongarra, "Netsolve: A Network Server for Solving Computational Science Problems". In proceedings of Super Computing'96, 1996.

[14] H. Nakada, S. Matsuoka, and S. Sekiguchi, "Design and Implementations of Ninf: towards a Global Computing Infrastructure", Future Generation Computing Systems, Metacomputing Issuse, 15(5-6):649-658, 1999.

[15] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, 1997, http://www.globus.org.

[16] Ninf-G Homepage, 2007, http://ninf.apgrid.org/

[17]Message Passing Interface (MPI), 2007, http://www-unix.mcs.anl.gov/mpi/