Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 00 (2010) 1–10
Procedia Computer Science

A General Model for the Generation and Scheduling of Parameter Sweep Experiments in Computational Grid Environments

Javier Díaz[a,*], Sebastián Reyes[a], Rosa M. Badia[b], Alfonso Niño[a], Camelia Muñoz-Caro[a]

[a] *Grupo de Química Computacional y Computación de Alto Rendimiento, Escuela Superior de Informática, Universidad de Castilla-La Mancha, Paseo de la Universidad 4, 13071, Ciudad Real, Spain.*

[b] *Barcelona Supercomputing Center and IIIA - Artificial Intelligence Research Institute CSIC - Spanish National Research Council, Jordi Girona, 29. E-08034 Barcelona. Spain*

Abstract

Parameter sweep experiments (PSE) involve several issues. In this work, we consider two of them: the generation of the parameters space and the scheduling of the associated tasks. Thus, we propose a general model to generate the parameters space of any PSE applying the Nested Summation Symbol operator. On the other hand, for the scheduling of these kinds of problems, we test an adaptive scheduling approach with fault tolerance. This approach has been implemented, using the DRMAA-C version for GridWay, to allocate tasks in a Grid environment. In the tests, the scheduler shows a good performance. Moreover, the CPU usage of the scheduler is quite low.

Keywords: Nested Summation Symbol, Adaptive Scheduling, Computational Grid, Parameter Sweep

1. Introduction

Among the different scientific and technological research fields there is a huge amount of problems involving the processing of many different tasks. A usual characteristic of these problems is that the associated tasks are independent. These kinds of problems are very common in simulation experiments. This is because if experiments are run

*Corresponding author

Email addresses: javier.diaz@uclm.es (Javier Díaz), sebastian.reyes@uclm.es (Sebastián Reyes), rosa.m.badia@bsc.es (Rosa M. Badia), alfonso.nino@uclm.es (Alfonso Niño), camelia.munoz@uclm.es (Camelia Muñoz-Caro)

under multiple configurations, we can develop laws and theories that can encapsulate knowledge of the systems being studied. Consequently, we can increase the predictive abilities on different conditions not yet studied. In practical terms, this means that we need to perform multiple tests with different input parameters. These kinds of problems are examples of parameter sweep experiments (PSE) [1]. To deal with these problems in realistic cases, it is necessary a large amount of computational power. Here, computational systems like Grids of computers have, and are, being used [2]. For this reason, the parameter sweep model has emerged as a "killer application model" for composing high-throughput computing (HTC) applications on global Grids [3, 4].

The parameter sweep model finds increasing applications in different scientific areas, for example: Bioinformatics [5–7], Earth Science [8], High-Energy Physics [9–11], Molecular Science [12–15], Electrical Engineering [16], or Social Sciences [17].

In parameter sweep experiments there are several issues to tackle. First, we have to generate the set of all possible combinations of input parameters. This is a hardworking task, which should be automated. However, it is not trivial to provide a general solution, since each problem has a different number of parameters and each of them has it own variation interval. Second, the scheduling of parameter sweep applications on distributed environments, like a computational Grid, is a complex activity. For that reason, it is necessary to develop efficient scheduling strategies to appropriately allocate the workload and reduce the associated computation time. Finally, it is necessary to monitor the execution of the tasks. A task level monitoring approach, specially suited for application to the Grid, has been proposed in [18].

In this work, we focus in the generation of the parameter space input set and in the scheduling of the associated tasks for parameter sweep experiments. So, we present a general approach to generate the parameter variations based on the Nested Summation Symbol (NSS) operator. Moreover, we test an adaptive scheduling approach, with fault tolerant support, to allocate efficiently the tasks generated.

The rest of the paper is organized as follow. Section 2 presents an overview of the NSS operator, and of the scheduling strategies to be tested. Section 3 details the design of the general approach to generate input parameter sets using the NSS. In Section 4, we give the details of the methodology used to test the scheduling approaches. Section 5 presents and interprets the results found in the study. Finally, in Section 6, we present the main conclusions.

2. Background

Parameter sweep experiments (PSE) involve a large number of independent jobs, since the experiment is runned under multiple initial configurations (input parameter values). In addition, different PSE's have different number of parameters. Therefore, in the most direct approach, for each study we need a different number of loops to generate the multiple configurations for the experiment. This is a troublesome task that would be necessary to automate. One way to tackle this problem is resorting to the Nested Summation Symbol (NSS) operator [19–21]. The NSS can be viewed as an operator attached to an arbitrary number of nested sums. In other words, a NSS represents a set of summations symbols where the number of these can be variable. The NSS operator notation is:

$$\sum_{n}(\mathbf{j} = \mathbf{i}, \mathbf{f}, \mathbf{s}) \tag{1}$$

where the meaning of this convention corresponds to perform all sums involved in the generation of all the possible values of the index vector j. The limits of the elements conforming the vector j are

$$i_k \leq j_k \leq f_k, \quad \forall k = 0, n-1 \tag{2}$$

The index n is called the dimension of the NSS. Thus, the operator shown in equation (1) represents the following set of n nested summations:

$$\sum_{j_0=i_0}^{f_0} \sum_{j_1=i_1}^{f_1} \cdots \sum_{j_{n-1}=i_{n-1}}^{f_{n-1}} \tag{3}$$

The operator performs all sums involved in the generation of all possible values of the index vector j. In this formulation, the k-th index of the vector j runs on the values ranging from i_k to f_k, with a particular step length. This step length is provided by the step vector $s = (s_0, s_1, ..., s_{n-1})$.

The NSS presents several interesting properties. First, it is a linear operator and, second, the product of two NSS is another NSS, see reference [19]. These properties make easy the application of the NSS to an increasing number of parameters or to generalize simulation experiments involving different parameter spaces.

The NSS operator was originally derived for dealing with nested summations in the perturbation treatment of quantum-mechanical systems. However, it has been proposed for different mathematical applications [21]: generation of variations and combinations, explicit expression of the determinant of a square matrix, or Taylor series expansion of an n-variable function.

For processing the large number of tasks generated, PSE's need a large amount of computational power. This can be provided by distributed systems like Grids of computers. In these kinds of systems, to minimize the overall computing time it is essential a correct assignment of tasks so that computer loads and communication overheads are well balanced. This problem belongs to the active research topic of the development and analysis of scheduling algorithms [22]. Different scheduling strategies have been developed along the years (for the classical taxonomy see [23]). In particular, dynamic self-scheduling algorithms are extensively used in practical applications [24–26]. These algorithms represent adaptive schemes where tasks are allocated in run-time. Self-scheduling algorithms were initially developed to solve parallel loop scheduling problems in homogeneous shared memory systems, see for instance [24]. These kinds of algorithms divides the set of tasks into subsets (chunks), and allocates them among the processors. In this way overheads are reduced.

Although self-scheduling algorithms were derived for homogeneous systems, they can be applied to heterogeneous ones such as computational Grids [25–27]. However, the problem could be the flexibility of these algorithms to adapt efficiently to a heterogeneous environment. Thus, we previously proposed two new flexible self-scheduling algorithms called Quadratic Self-Scheduling (QSS) and Exponential Self-Scheduling (ESS) [27]. The first is based on a quadratic form for the chunks distribution function. Therefore, it has

three degrees of freedom, which provide high adaptability to distributed heterogeneous systems. The second approach, ESS, is based on the slope of the chunks distribution function. In this case, we consider that the chunks distribution function decreases in an exponential way. This algorithm provides good adaptability in distributed heterogeneous systems using two parameters. Moreover, in previous works [27] we have compared QSS and ESS with other self-scheduling algorithms in an actual Grid environment. QSS and ESS algorithms outperform the previous ones, obtaining better load balance and more reduction of the communication overhead.

However, a computational Grid is made up of a large number of independent resource providers and consumers, which run concurrently, change dynamically, and interact with each other. We can use Adaptive Grid scheduling [28, 29] to tackle the problem of the unpredictable changing conditions. In general, adaptive scheduling can consider, among others, factors such as availability, performance, workload, bandwidth and latency of the interconnection links. Thus, previously [30–32] we introduced a new fault tolerant adaptive approach to scheduling tasks in dynamic Grid environments. Using this approach, new optimal QSS and ESS parameters were obtained when the environment changes. Additionally, this scheduling approach has been implemented to create a fault tolerant simulation-based scheduler [33]. To such an end, we simulate the environment and apply a simulated annealing to obtain the QSS and ESS optimal parameters. Moreover, when the environment changes, or a job fails, we recalculate the list of chunks including, if necessary, the failed tasks. Therefore, this scheduler allows to handle correctly dynamic environments where processors appear and disappear during the execution.

3. General Model for the Generation of the Input Parameter Space

When dealing with arbitrary PSE's, the starting problem is to develop a general approach to the generation of the different input parameter values. Since the parameter values can be indexed, to generalize the treatment we must focus in the parameter indexes rather than in the parameter values. This problem can be tackled by using some concepts of set theory. Thus, we are looking for the generation of all the elements of the input space set, I, for a total of n different parameters, or in other words:

How can we generate all the different groups of n elements that can be formed from n sets of elements of different cardinality, with the restriction that we can use only one element from each set?

If the n parameter indexes sets are labeled as P_i, $0 \leq i \leq n-1$, the total number of groups of n elements is given by the product of the cardinalities:

$$\mid I \mid = \prod_{i=0}^{n-1} \mid P_i \mid \tag{4}$$

This result shows that the I set can be generated from the direct sum of the P_i sets. The most direct form to obtain this direct sum, and generate all the elements of I, is to use a series of n nested symbols, one for each P_i set. For a given loop acting on the P_i indexes, this is equivalent to a NSS operator of dimension 1 where the i vector equals 0 and the s vector equals 1, in short $\sum_1(j_i, f_i)$. Taking into account the properties of the NSS operator [19–21] we have,

$$\sum_{j_0=0}^{f_0} \cdot \sum_{j_1=0}^{f_1} \cdots \sum_{j_{n-1}=0}^{f_{n-1}} = \sum_{1}(j_0, f_0) \cdot \sum_{1}(j_1, f_1) \cdots \sum_{1}(j_{n-1}, f_{n-1}) \tag{5}$$
$$= \sum_{n}(j_0 \oplus j_1 \oplus \cdots j_{n-1}, f_0 \oplus f_1 \oplus \cdots f_{n-1}) = \sum_{n}(\mathbf{j}, \mathbf{f})$$

Therefore, we have two ways to generate the I set. First, to use a set of nested loops. This is the usual option, but it needs modification each time the number of parameters, loops, changes. Second, to implement an n-dimensional NSS operator. This option is general since the number of loops is a datum. Here, we develop and implement two different general solutions for the NSS operator for PSE problems. Both implementations make use of a function pointer, which provides us a way to generalize the treatment to any problem we need.

First, we present an iterative solution, called NSSI. This is a particularization to PSE cases of the solution proposed in [19–21]. Here, we consider starting indexes of zero and step increments of one, see Pseudocode 1. This case corresponds to equation (1).

Pseudocode 1: NSSI: NSS Iterative Case

```
Procedure NSSI
  get number of parameters, n
  get final indexes vector, f
  get function to apply indexes vector  // A function pointer can be used
  initialize current indexes vector, j, to zero
  unvisited_loops=n-1
  while unvisited_loops >= 0 do
    if j [unvisited_loops] > f [unvisited_loops] then
       j [unvisited_loops] = 0
       decrement unvisited_loops by one
    else
       apply function with the current j vector
       reset unvisited_loops to n-1
    end_if
    if unvisited_loops >= 0 then
       j [unvisited_loops] = j [unvisited_loops]+1
    end_if
  done_while
end_Procedure
```

On the other hand, we propose a recursive solution, called NSSR. Again, the initial indexes are zero and the step increment is one, see Pseudocode 2.

Pseudocode 2: NSSR: NSS Recursive Case

```
Procedure NSSR
  get number of parameters, n
  get final_indexes vector, f
  get function to apply indexes vector  // A function pointer can be used
  finalize=false
  initialize current indexes vector, j, to zero
  if current_loop_index < n then
      for value from 0 to f[current_loop_index] do
          j [current_loop_index]=value
```

```
            if_not NSSR (j, f, parameters_number, current_loop_index+1,
                function) then
                apply function with the current j vector
            end_if_not
            finalize=true
        end_for
    end_if
    return finalize
end_Procedure
```

The two previous solutions meet our requirements of generalization. First, both procedures can handle an arbitrary number of parameters and an arbitrary number of parameter values since they are input data. Second, the function that uses the parameter values is also an input datum. This allows using both procedures in any knowledge field. It is interesting to note that the "function" using the parameters can represent an arbitrary workflow formed by system calls.

4. Methodology

In this work, we generate different PSE input spaces and perform several tests to determine the performance of the fault tolerant simulation-based scheduler in a dynamic environment. We perform tests using the QSS and ESS algorithms. Moreover, we perform the same tests using GSS [36], FSS [37] and TSS [38] to compare the behaviour.

As a test case, we consider the problem of the molecular potential energy hypersurface mapping [15]. Thus, we consider the CF_2 (Carbon difluoride) molecule. This is a three-dimensional problem depending on the two *C-F* distances and the *F-C-F* angle. The parameter space of this PSE is obtained using the general model proposed in the Section 3. So, we only need to define the function which generates the inputs and specify the range of each parameter. Thus, small increments on these coordinates (parameters) are used to generate 5611 independent input molecular structures. To generate another test case we have duplicated the inputs getting 11222 independent input molecular structures. The nuclear potential energy corresponding to each molecular structure is computed using the Gamess [39] molecular structure code. The tasks have a random duration between 4 and 50 seconds.

The tests are carried out in an Internet-based Grid of computers formed by four clusters. Three of them, Tales, Hermes and Aris, are located in the Universidad de Castilla-La Mancha at Ciudad Real (Spain). Tales and Hermes consist of six and eleven Pentium IV processors, respectively (CPU frequencies between 2.4 and 3.0 GHz). Aris consists on six 64-bit Xeon processors with 2.4 GHz CPU frequency. The fourth cluster (Popo) is in the Universidad de Puebla (Mexico). This last is formed by four 64-bit Opteron biprocessors with 1.66 GHz CPU frequency. Each cluster uses a 100-Mbps Fast Ethernet network, except Aris which uses Gigabit Ethernet. Connection between clusters is achieved through Internet. The Grid uses Globus Toolkit 4.2.1 as basic middleware [2]. To allocate the tasks on the Grid we use the 5.2.1 version of the GridWay metascheduler [34] through its DRMAA-C API [35].

We have considered the previous environment as dynamic. We start the execution with six processors in Tales, six processors in Hermes, four processors in Popo and five processors in Aris. During the execution we introduce changes in the system to allow

processors to disappear and appear, see Table 1. The computations for each test case have been performed three times, to obtain average results.

Table 1: Changes introduced in the system to represent a dynamic environment. The time is taken from the start of the execution and is given in seconds.

Time	Changes introduced
1200	A processor disappears from Tales
1500	A processor disappears from Hermes
2200	Two processors appear in Hermes
2800	Two processors appear in Hermes & a processor appears in Aris
3800	The cluster Tales disappears
4400	Two processors appear in Hermes
4900	The cluster Tales appears

5. Results and Discussion

We have mapped the potential energy hypersurface of the CF_2 molecule in the dynamic environment. Figure 1 shows the behaviour of the simulation-based scheduler using the QSS and ESS algorithms. In addition, the behaviour of previous self-scheduling algorithms, GSS [36], FSS [37] and TSS [38] is also shown. We see that the proposed scheduler with QSS provides the best performance in both test cases. In particular, it obtains a reduction over the total computation time between 11% and 15% with respect to ESS. On the other hand, in the first test case (5611 tasks) the scheduler with QSS and ESS is a 26% and a 17% more efficient than the best of the other self-scheduling algorithms (FSS), respectively. Meanwhile, in the second case (11222 tasks) QSS and ESS are up to a 20% and a 6% better than FSS, respectively. This is because the simulation-based scheduler obtains a good equilibrium between load balance and overheads, even in a changing environment. This is due to the flexibility provided to the QSS and ESS by the configurable parameters, which generate a good chunk distribution that reduces the overheads keeping a good load balance. In addition, the tasks are rescheduled in a transparent way by adding them into the pool of non-allocated tasks. In the second case, the total improvement of the scheduler is lower than in the previous one, because the execution environment is stable during an important part of the execution (since the overall computation time of each algorithm is higher than before).

On the other hand, we have recorded the CPU usage of the scheduler during the scheduling process. Each sample is taken each ten seconds. Figure 2 shows the percentage of CPU used by QSS and ESS. We observe that in both cases we have a high CPU usage at the beginning, since it calculates the optimal chunk distribution, and allocating many chunks at the same time. However, during the rest of the scheduling process the percentage of CPU used is only around 5-7%, except for QSS and the test case with 11222 tasks, where the CPU usage is around 9-13%.

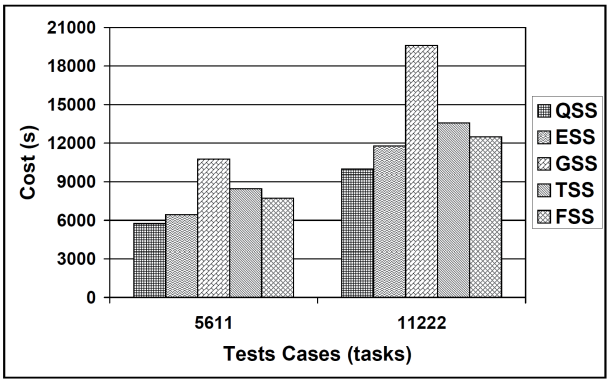

Figure 1: Computational cost to solve the test cases in a dynamic environment for the considered scheduling strategies. The cost is given in seconds.

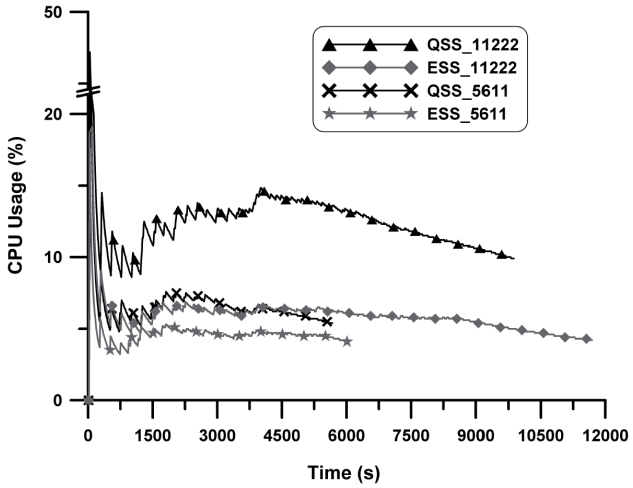

Figure 2: CPU usage of the scheduler when using QSS and ESS as basic strategies.

6. Conclusions

In this work, we presented a NSS-based general model to generate the parameters space in parameter sweep experiments. Using this general model, we can handle any number of parameters without any modification of the generator. Thus, it is possible to merge this process with the scheduling. In addition, we have tested an adaptive scheduling approach to allocate large sets of tasks in distributed heterogeneous systems, such as a Grid of Computers. This approach is focused on the efficient allocation of tasks among changing environments. We have implemented this approach into an scheduler using the DRMAA API provided by GridWay, which allows a high compatibility with different Grid infrastructures.

We have used the general NSS-based model to generate the parameter space of the experiment considered here. In this experiment, we have mapped the molecular potential

energy hypersurface of the CF_2 molecule in a dynamic environment. Thus, after generation of the parameter space, we have allocated it in a dynamic environment. To such an end, we have used the scheduler, presented here, with the QSS and ESS algorithms. Also we have allocated the tasks using other self-scheduling algorithms (GSS, TSS and FSS) to compare the behaviour. The tests have shown that the scheduler with QSS and ESS is up to a 26% and a 17% more efficient than the best of the other self-scheduling algorithms (FSS), respectively.

Acknowledgements

This work has been cofinanced by FEDER funds, the Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha (grant #PBI08-0008). The Universidad de Castilla-La Mancha is also acknowledged. The authors wish also to thank the Facultad de Ciencias Químicas and the Laboratorio de Química Teórica of the Universidad Autónoma de Puebla(Mexico), for the use of the Popocatepetl cluster. The author from BSC has been partially supported by the Ministry of Science and Technology of Spain (contract TIN2007-60625).

Bibliography

[1] Samples, M.E., Daida, J.M., Byom, M., Pizzimenti, M., "Parameter sweeps for exploring GP parameters," *Genetic And Evolutionary Computation Conf.*, Washington, USA, pp.212-219, Apr.2005.

[2] Foster, I. and Kesselman, C., *The Grid: Blueprint for a New Computing Infrastructure*, San Francisco: Morgan Kauffman, 1999.

[3] Buyya, R., Murshed, M., Abramson, D. and Venugopal, S., "Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimization Algorithm," *Software Practice and Experience*, vol. 45, no. 5, pp. 491-512, 2005.

[4] Abramson, D., Giddy, J. and Kotler, L., "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?," *Int. Parallel and Distributed Processing Symp. (IPDPS)*, Cancun, Mexico, pp. 520- 528, May 2000.

[5] Sun, C., Kim, B., Yi, G. and Park, H., "A Model of Problem Solving Environment for Integrated Bioinformatics Solution on Grid by Using Condor," Springer-Verlag LNCS 3251, pp. 935-938, 2004.

[6] Folding@Home Project Web Page. http://folding.stanford.edu. Última visita Oct. 2009.

[7] Buyya, R., Branson, K., Giddy, J. and Abramson, D., "The virtual laboratory: Enabling molecular modelling for drug design on the World Wide Grid," Technical Report CSSE-103, Monash University, 2001.

[8] Gulamali, M.Y., McGough, A.S., Newhouse, S.J., and Darlington, J., "Using ICENI to Run Parameter Sweep Applications Across Multiple Grid Resources," *GlobalGrid Forum 10*, Berlin, Germany, Mar. 2004.

[9] Basney, J., Livny, M. and Mazzanti, P., "Harnessing the Capacity of Computational Grids for High Energy Physics," *Conf. on Computing in High Energy and Nuclear Physics*, Padova, Italy, 2000.

[10] The Large Hadron Collider (LHC) Computing Grid Project for High Energy Physics Data Analysis. http://lcg.web.cern.ch/LCG. Last access Jan. 2009.

[11] EGEE Project Web Page. http://www.eu-egee.org. Last access Jan. 2009.

[12] Wozniak, J.M., Striegel, A., Salyers, D. and Izaguirre, J.A., "GIPSE: Streamlining the Management of Simulation on the Grid," *Proc. of the Thirty-Eight Ann. Simulation Symp.*, San Diego, USA, pp. 130-137, Apr. 2005.

[13] Duane, S., Kennedy, A.D., Pendleton, B.J. and Roweth, D., "Hybrid Monte Carlo," Physics Letters B, vol. 195, pp. 216-222, 1987.

[14] Izaguirre, J.A. and Hampton, S.S., "Shadow Hybrid Monte Carlo: An Efficient Propagator in Phase Space of Macromolecules," *Journal of Computational Physics*, vol. 200, no. 2, pp. 581-604, 2004.

[15] Reyes, S., Muñoz-Caro, C., Niño, A., Badia, R.M. and Cela, J.M., "Performance of Computationally Intensive Parameter Sweep Applications on Internet-based Grids of Computers: the Mapping

of Molecular Potential Energy Hypersurfaces," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 4, pp. 463-481, 2007.
[16] Spice Web Page. http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE. Last access Jan. 2009.
[17] Axelrod, R., "The Dissemination of Culture: A Model with Local Convergence and Global Polarization," *Journal of Conflict Resolution*, vol. 41, no. 2, pp. 203-226, 1997.
[18] Reyes, S., Muñoz-Caro, C., Niño, A., Sirvent, R., Badia, R., "Monitoring and Steering Grid Applications with GRID superscalar," *Future Generation Computer Systems*, 2009, accepted, DOI: 10.1016/j.future.2009.12.002.
[19] Carbó., R. and Besalú, E., "Nested Summation Symbols and Perturbation Theory," *Journal of Mathematical Chemistry*, vol. 13, pp. 331-342, 1993.
[20] Besalú, E. and Carbó., R., "Generalized Rayleigh-Schrödinger Perturbation Theory in Matrix Form," *Journal of Mathematical Chemistry*, vol. 15, pp. 397-406, 1994.
[21] Carbó., R. and Besalú, E., "Definition and Quantum Chemical Applications of Nested Summation Symbols and Logical Kronecker Deltas," *Computers and Chemistry*, vol. 18, no. 2, pp.117-126, 1994.
[22] Sirvent, R., Badia, R.M. and Labarta, J., "Graph-Based Task Replication for Workflow Applications," 11th IEEE Int. Conf. on High Performance Computing and Communications (HPCC 2009), Seoul, Korea, 2009.
[23] Casavant, T.L. and Kuhl, J.G., "A Taxonomy of Scheduling in General-Purpose Distributed Computing," *IEEE Trans. on Software Engineering*, vol. 14, no. 2, pp. 141-154, 1988.
[24] Lilja, D.J., "Exploiting the Parallelism Available in Loops," *IEEE Computer*, vol. 27, no. 2, pp. 13-26, 1994.
[25] Penmatsa, S., Chronopoulos, A.T., Karonis, N.T. and Toonen, B., "Implementation of Distributed Loop Scheduling Schemes on the TeraGrid," *Proc. of the 21st IEEE Int. Parallel and Distributed Processing Symp. (IPDPS 2007)*, Long Beach, California, USA, pp. 1-8, Mar. 2007.
[26] Sokolowski, P.J., Grosu, D. and Xu, C., *Analysis of Performance Behaviors of Grid Connected Clusters*, in: Ould-khaoua, M. and Min G. (Eds.), *Performance Evaluation of Parallel And Distributed Systems*, Nova Science Publishers, Jun. 2006.
[27] Díaz, J., Reyes, S., Niño, A., Muñoz-Caro, C., "Derivation of Self-Scheduling Algorithms for Heterogeneous Distributed Computer Systems: Application to Internet-based Grids of Computers," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 617-626, 2009.
[28] Huedo, E., Montero, R.S. and Llorente, I.M., "Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications," *Proc. 12th Euromicro Conf. on Parallel, Distributed and Network-Based Processing*, A Coruña, Spain, pp. 28-33, Apr. 2004.
[29] Li, P., Ji, Q., Zhang, Y. and Zhu, Q., "An Adaptive Chunk Self-Scheduling Scheme on Service Grid," *2008 IEEE Asia-Pacific Services Computing Conf.*, Yilan, Taiwan, pp. 39-44, Dec. 2008.
[30] Díaz, J., Reyes, S., Niño, A., Muñoz-Caro, C., "A Heuristic Approach to the Allocation of Different Workloads in Computational Grid Environments," *Int. Journal on Advances in Software*, vol. 2, no. 1, pp. 1-10, 2009.
[31] Díaz, J., Muñoz-Caro, C. and Niño, A., "An Adaptive Approach to Task Scheduling Optimization in Dynamic Grid Environments," *The 2009 Int. Conf. on Grid Computing and Applications*, Las Vegas, USA, Jul. 2009.
[32] Díaz, J., Muñoz-Caro, C. and Niño, A., "A Fault Tolerant Adaptive Method for the Scheduling of Tasks in Dynamic Grids," *The 3rd. Int. Conf. on Advanced Engineering Computing and Applications in Sciences (ADVCOMP)*, Sliema, Malta, Oct. 2009.
[33] Díaz, J., Muñoz-Caro, C. and Niño, A., "A Fault Tolerant Simulation-Based Scheduler for Dynamic Grids of Computers," *IEEE Transactions on Parallel and Distributed Systems*, 2010, submitted.
[34] Huedo, E., Montero, R.S. and Llorente, I.M., "A modular Meta-Scheduling Architecture for Interfacing with pre-WS and WS Grid Resource Management Services," *Future Generation Computer Systems*, vol. 23, pp. 252-261, 2007.
[35] Distributed Resource Management Application API Working Group - Open Grid Forum. http://www.drmaa.org. Last access Jan. 2009.
[36] Polychronopoulos, C.D. and Kuck, D., "Guided Self-Scheduling: a Practical Scheduling Scheme for Parallel Supercomputers," *IEEE Trans. on Computers*, vol. 36, pp. 1425-1439, 1987.
[37] Hummel, S.F., Schonberg, E. and Flynn, L.E., "Factoring: A Method for Scheduling Parallel Loops," *Comm. of the ACM*, vol. 35, pp. 90-101, 1992.
[38] Tzen, T.H. and Ni, L.M., "Trapezoid Self-Scheduling: A Practical Scheduling Scheme for Parallel Compilers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 87-98, 1993.
[39] Gamess Homepage. http://www.msg.chem.iastate.edu/gamess. Last access Jan. 2009.