Solving Large Scale Optimization Problems via Grid and Cluster Computing[*]

Katsuki Fujisawa[†], Masakazu Kojima[‡]
Akiko Takeda[⋆] and Makoto Yamashita[♯]

Abstract. Solving large scale optimization problems requires a huge amount of computational power. The size of optimization problems that can be solved on a few CPUs has been limited due to a lack of computational power. Grid and cluster computing has received much attention as a powerful and inexpensive way of solving large scale optimization problems that an existing single-unit CPU cannot process. The aim of this paper is to show that grid and cluster computing provides tremendous power to optimization methods. The methods that this paper picks up are a successive convex relaxation method for quadratic optimization problems, a polyhedral homotopy method for polynomial systems of equations and a primal-dual interior-point method for semidefinite programs. Their parallel implementations on grids and clusters together with numerical results are reported. The paper also mentions a grid portal system for optimization problems briefly.

Key words.
Optimization Problem, Computer Network, Parallel Computing, Cluster Computing, Grid Computing, Semidefinite Program, Semidefinite Program Relaxation, Polynomial Equations, Polyhedral Homotopy Method

$*$ This paper is a revised version of [4], which is presented at SAINT 2004 (The 2004 International Symposium on Applications and the Internet), Tokyo, Japan, January 26–30, 2004.

$\dagger$ Department of Mathematical Sciences, Tokyo Denki University.
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology, Japan. *fujisawa@r.dendai.ac.jp*

$\ddagger$ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. *kojima@is.titech.ac.jp*

$\star$ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. *takeda@is.titech.ac.jp*

$\sharp$ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan.
yamashi9@is.titech.ac.jp

1 Introduction

Optimization problems have a variety of practical applications in various fields such as operations research, engineering, science, biology and economics. Many combinatorial and nonconvex optimization problems are known to be $\mathcal{NP}$-hard, which means that there exists no deterministic algorithm that finds an optimal solution in polynomial time unless $\mathcal{P} = \mathcal{NP}$. Hence, solving large scale combinatorial and nonconvex optimization problems requires a huge amount of computational time and resources, and the size of such problems that we can solve has been limited. As computing resources continue to improve, however, optimal solutions of larger scale optimization problems become more achievable. In particular, grid and cluster computing technology has recently received much attention as a powerful and inexpensive methodology for solving large scale optimization problems that an existing single-unit CPU cannot process.

To solve large scale optimization problems, we need new computing infrastructure which enables us to easily access to computational resources including hardware and software library distributed across a wide area network like the Internet. For example, Applegate et al. [2] implemented the Danzig, Fulkerson and Johnson's cutting plane method for the large scale TSP (traveling salesman problem). They obtained an optimal solution of the TSP that has 15,112 cities (nodes) in Germany. This problem is the largest scale TSPLIB [1] instance that has been solved to date. The computation was executed on a network of 110 processors located at Rice and Princeton Universities. They estimated the total computational time was 22.6 years, scaled to a Compaq EV6(21264) Alpha processor running at 500MHz. A group of researchers at the University of Iowa and Argonne National Laboratory solved the QAP (quadratic assignment problem) instance called NUG30 in QAPLIB [2] using the Condor [3] developed by the University of Wisconsin. The Condor is a system of daemons and tools to utilize commodity computing resources including idle desktop machines for high-throughput computing. NUG30 is known as a huge scale QAP that would require more than 10 years of computational time by a single CPU. They obtained an optimal solution in just seven days using the Condor system and 653 CPUs on average. See the paper [1] for more details.

In this paper, we present grid and cluster computing for some optimization problems with numerical results. In Sections 2 and 3, we focus our attention on GridRPC, which is an RPC system redesigned for grid applications. Some GridRPC systems such as Netsolve and Ninf are widely used. The Ninf (Network based Information library for global world-wide computing Infrastructure) system [11] developed by AIST (National Institute of Advanced Industrial Science and Technology, Japan) employs a client-server model, where server and client machines are connected via a local area network or the Internet. We implemented highly parallel algorithms for some optimization problems and polynomial systems of equations with the use of the Ninf system on several PC clusters connected via a high speed local area network and/or the Internet. In Section 4, we briefly explain the SDPARA (SemiDefinite Programming Algorithm paRAllel version) [16], a parallel implementation of the SDPA [15], and present numerical results on PC clusters. Section 5 presents a grid portal system which enables users with few computational resources to receive benefits of parallel computing.

[1]http://www.iwr.uni-heidelberg.de/iwr/comopt/software/TSPLIB95/
[2]http://www.opt.math.tu-graz.ac.at/qaplib/
[3]http://www.cs.wisc.edu/condor/

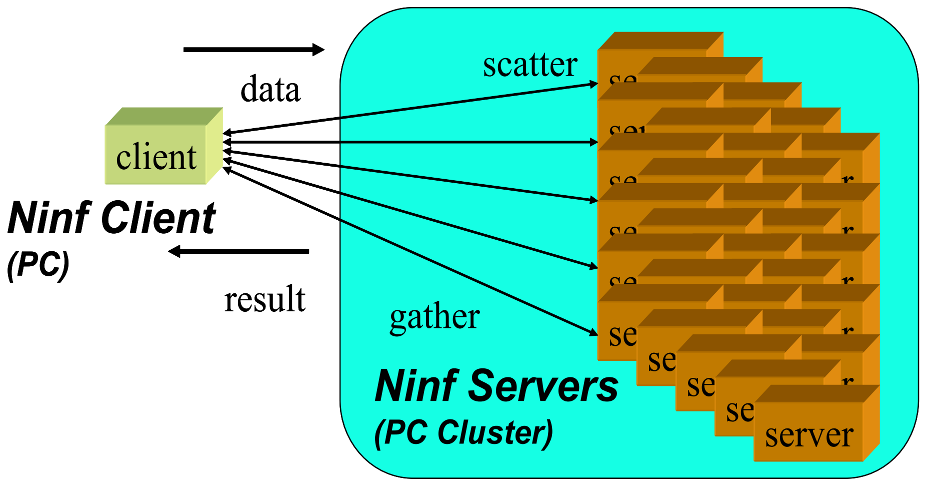

Figure 1: Ninf client-server model

2 Parallel Successive Convex Relaxation Method for Nonconvex Quadratic Optimization Problems

Let $\mathbb{R}^n$ and $\mathcal{S}^n$ denote the n-dimensional Euclidean space and the space of $n \times n$ real symmetric matrices, respectively. A general QOP is described in the following form:

$$\left.\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} & \gamma_i + 2\boldsymbol{q}_i^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} \leq 0 \quad (i = 1, \ldots m) \end{array}\right\}, \tag{1}$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\gamma_i \in \mathbb{R}$, $\boldsymbol{q}_i \in \mathbb{R}^n$ and $\boldsymbol{Q}_i \in \mathcal{S}^n$ $(i = 1, \ldots, m)$. When a given QOP has a quadratic objective function such as $\gamma_0 + 2\boldsymbol{q}_0^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_0 \boldsymbol{x}$, we can transform it into QOP (1) by replacing the quadratic objective function with a new variable t and adding $-\gamma_0 - 2\boldsymbol{q}_0^T \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{Q}_0 \boldsymbol{x} + t \leq 0$ to the set of inequality constraints. A general QOP is known as one of the most significant nonlinear programming problems. QOPs cover not only economics and engineering applications but also various important nonconvex mathematical programs, such as 0-1 integer programs, linear complementarity problems, bilevel quadratic programs, linear and quadratic fractional programs, and so on. Because of their theoretical and computational difficulties, however, solvable QOPs had been limited to convex QOPs where all $\boldsymbol{Q}_i$s are assumed to be positive semidefinite or general QOPs with small size.

The SCRM (Successive Convex Relaxation Method) proposed by Kojima-Tuncel [7] is a powerful numerical method to compute upper bounds of general QOPs by repeated applications of SDP (semidefinite programming) relaxations. The SCRM generates and solves a large number of SDP problems at each iteration. Takeda et al. [12] reported that the SCRM can deal with some larger scale QOPs through numerical experiments on a PC cluster. To get more accurate upper bounds and/or to process larger scale of QOPs, we need more and more computing resources. We implemented a highly parallel SCRM on the Ninf system. Figure 1 shows a Ninf client-server model. The Ninf client controls the SCRM applied to a QOP and generates a large number of subproblems each of which forms an SDP problem at each iteration. All generated SDP problems are sent to the Ninf server machines. Then each Ninf server solves an SDP problem using the software

SDPA [15]. Note that each Ninf server solves only one SDP problem at a time. After finishing the execution of the SDPA, the result is sent back to the Ninf client machine. Takeda et al. [12] also showed computational efficiency of the SCRM by varying the number of Ninf servers. They reported that SDP problems were allocated to each Ninf server in balance and the total computational time consumed by each Ninf server was almost the same. Therefore good performance and high scalability were attained.

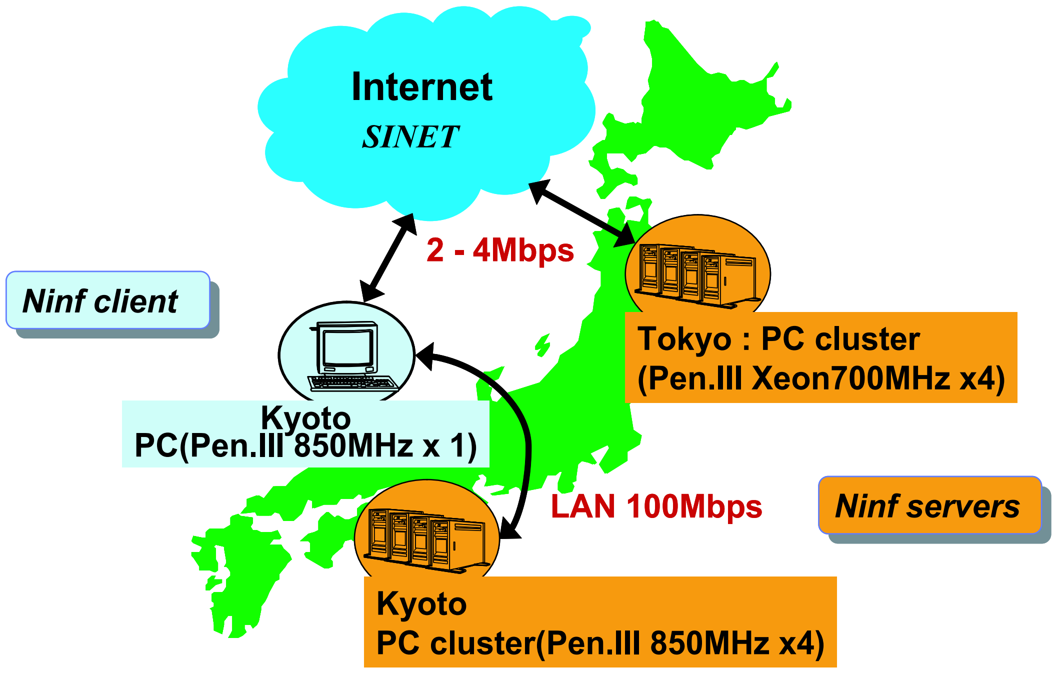

Figure 2: Grid Environment(Kyoto $\Longleftrightarrow$ Tokyo)

We now show some numerical experiments of the SCRM on a Grid environment. Figure 2 illustrates that we have two PC clusters which are located in Kyoto and Tokyo. The Kyoto cluster has 4 nodes which are connected to a Ninf client PC through LAN (100BASE-TX). The Tokyo cluster has 4 nodes which are connected to the Ninf client PC through the Internet called SINET. We estimate the average network speed of the SINET between Kyoto and Tokyo to be about 2 to 4 Mbps. Table 1 shows numerical results of the SCRM on the grid environment illustrated in Figure 2. The first row "# subproblems" denotes the total number of SDP problems which the Ninf servers in the Kyoto cluster and the Tokyo cluster solved, respectively. The second row denotes the total execution time of the SDPA in each cluster. The total execution time of the Kyoto cluster is almost the same as that of the Tokyo cluster. Note also that the total transmitting time between the Ninf client and Ninf servers is extremely small. From Table 1, we can say that the SCRM is well paralleled in the grid environment.

3 Parallel Polyhedral Homotopy Method

Polynomial systems have various applications in many fields of science and engineering. The software package "PHoM" [6] is designed to find all isolated (real and complex)

Table 1: Numerical results on grid environments

Prob. name	LC80-144		BLevel20-3	
Ninf Server	Kyoto	Tokyo	Kyoto	Tokyo
# subproblems	1795	1333	883	597
total execution time(sec.)	16486.5	16395.6	1053.6	984.2
total trans. time C $\rightarrow$ S(sec.)	0.150	0.117	0.046	0.028
total trans. time C $\leftarrow$ S(sec.)	0.275	0.069	0.130	0.182

Table 2: Numerical results of the homotopy method (noon-9 problem (3))

	StartSystem		Trace		Total	
# CPUs	time(s.)	sp-up-raito	time(s.)	sp-up-ratio	time(s.)	sp-up-ratio
1	43	1.00	44,119	1.00	44,162	1.00
2	25	1.72	22,192	1.99	22,217	1.99
4	22	1.95	11,109	3.97	11,131	3.97
8	14	3.07	5,548	7.95	5,562	7.94
16	14	3.07	2,822	15.63	2,826	15.57
32	20	2.15	1,435	30.74	1,455	30.35

solutions of a polynomial system of equations $f(x) = 0$ such as the cyclic n problem:

$$\left.\begin{array}{l} x_1 + x_2 + \cdots + x_n = 0 \\ x_1x_2 + x_2x_3 + \cdots + x_nx_1 = 0 \\ x_1x_2x_3 + \cdots + x_nx_1x_2 = 0 \\ \cdots \\ x_1x_2x_3 \cdots x_n - 1 = 0, \end{array}\right\}, \qquad (2)$$

using a polyhedral homotopy method. The package consists of three modules. The first module which we call "StartSystem" constructs a family of polyhedral-linear homotopy functions. Takeda et al. [13] implemented a highly paralleled StartSystem using the Ninf. The second module traces homotopy paths to compute all isolated solutions of a polynomial system of equations. The Ninf servers trace the homotopy paths by applying the predictor-corrector method. The third module verifies whether all isolated solutions of the polynomial system of equations have been computed correctly.

We also employ the Ninf client-server system, illustrated in Figure 1, for the parallel implementation of the first and second modules of the PHoM. Table 2 shows numerical results of the first and second modules for the noon-9 problem:

$$\left.\begin{array}{l} x_1x_2^2 + x_1x_3^2 + \cdots + x_1x_9^2 - 1.1x_1 + 1 = 0 \\ x_2x_1^2 + x_2x_3^2 + \cdots + x_2x_9^2 - 1.1x_2 + 1 = 0 \\ \cdots \\ x_9x_1^2 + x_9x_2^2 + \cdots + x_9x_8^2 - 1.1x_9 + 1 = 0 \end{array}\right\}. \qquad (3)$$

This problem has 19,665 isolated solutions. All numerical experiments were executed on the Presto I cluster which has 64 nodes (each CPU is Celeron 500MHz). The 'sp-up-ratio' stands for the computational time (# of CPUs is 1) divided by the computational time (# of CPUs is k). If 'sp-up-ratio' is sufficiently close to k, we can regard the software as well paralleled on the Ninf client-server system. Takeda et al. [13] reported that the computation of mixed cells by StartSystem (the first module) is suitable for parallel computation through some numerical experiments. We furthermore observe from Table 2 that tracing all the homotopy paths (the second module) is also suitable for parallel computation on the Ninf client-server system.

Table 3 also shows numerical results of the second module for the cyclic problems (2) with dimensions $n = 11,\ 12$ and 13. There are many homotopy paths, which can be traced independently in parallel; specifically the cyclic 13 problem has 208,012 homotopy paths, and the above noon-9 problem has 19,665 homotopy paths. We use two PC clusters which are located in Tokyo Institute of Technology. All homotopy paths of the cyclic 11 problem were computed in the Presto I cluster, while all homotopy paths of the cyclic 12 and 13 problems were computed on the Presto III cluster which has 256 nodes and 512 CPUs (each CPU is Athlon 1900+). From Table 3, we observe that the second module of the PHoM is quite suitable for parallel computation on PC clusters. Note that more than 5,000 seconds were required to compute all the homotopy paths of the cyclic 13 problem even when we used 256 CPUs simultaneously. Therefore parallel computing is indispensable to process larger scale polynomial systems of equations.

Table 3: Numerical results of tracing homotopy paths (cyclic n problem)

	cyclic-11		cyclic-12		cyclic-13	
# CPUs	time(s.)	sp-up-raito	time(s.)	sp-up-ratio	time(s.)	sp-up-ratio
2	47,345	1.00				
4	23,674	2.00				
8	11,852	3.99				
16	5,927	7.99				
32	2,967	15.96				
64	1,487	31.84	2,592	1.00		
128			1,332	1.95	10,151	1.00
256			703	3.69	5,191	1.95
# paths traced	16,796		41,696		208,012	

4 Parallel Implementation of Semidefinite Programming

In the last decade, SDP problems have been intensively studied in theoretical, numerical and practical aspects in various fields such as interior-point methods, combinatorial optimization, control and systems, robust optimization and quantum chemistry. Let $\mathcal{S}^n$ denote the vector space of $n \times n$ symmetric matrices. For a pair of matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{S}^n$, the inner product is defined as $\boldsymbol{X} \bullet \boldsymbol{Y} = \sum_{i=1}^n \sum_{j=1}^n X_{ij} Y_{ij}$. We use the notation $\boldsymbol{X} \in \mathcal{S}^n_+$ ($\mathcal{S}^n_{++}$) to indicate that $\boldsymbol{X} \in \mathcal{S}^n$ is positive semidefinite (or positive definite). Given $\boldsymbol{A}_i \in \mathcal{S}^n$ $(i = 0, 1, \dots, m)$ and $\boldsymbol{b} \in \mathbb{R}^m$, the standard form SDP problem is written as follows:

$$\left.\begin{array}{ll} \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{X} \\ \text{subject to} & \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \ (i = 1, 2, \dots, m),\ \boldsymbol{X} \in \mathcal{S}^n_+ \end{array}\right\}. \tag{4}$$

The corresponding dual problem is as follows:

$$\left.\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{m} b_i z_i \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} \boldsymbol{A}_i z_i + \boldsymbol{Y} = \boldsymbol{A}_0,\ \boldsymbol{Y} \in \mathcal{S}_+^n \end{array}\right\}. \tag{5}$$

It is known that the primal-dual interior point method (PDIPM) is capable of solving these problems in polynomial time. The SDPA (SemiDefinite Programming Algorithm) [15] is an optimization software package, written by the C++ language, of the PDIPM for solving the standard form SDP problem. The SDPA incorporates a special data structure for handling block diagonal data matrices and an efficient method proposed by Fujisawa et al. [3] for computing search directions when problems to be solved are large scale and sparse. In many applications, however, SDP problems become too large for SDP software packages including the SDPA to solve on a single processor. It is well-known that the PDIPM for SDP problems has two major time consuming parts at each iteration even if we exploit the sparsity of the date matrices. The first part is the computation of the so-called Schur complement matrix. The second part is the Cholesky factorization of the Schur complement matrix.

The SDPARA (SemiDefinite Programming Algorithm paRAllel version) [16] is a parallel version of the SDPA on multiple processors and distributed memory, which replaces these two bottleneck parts mentioned above by their parallel implementation using MPI and ScaLAPACK (Scalable Linear Algebra PACKage). The SDPARA reads input data m, n, $\boldsymbol{b}$, $\boldsymbol{A}_0$, $\boldsymbol{A}_1$, …, $\boldsymbol{A}_m$ and each processor keeps the memory space for the input data and variables $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{z}$ independent of other processors, while the Schur complement matrix is divided and stored on each processor. The SDPARA can compute each row of the Schur complement matrix independently in parallel and applies a parallel Cholesky factorization provided by ScaLAPACK to the Schur complement matrix. Figure 3 shows that the SDPARA on a PC cluster (Presto III) attains high scalability for large scale SDP problems. ELEMENTS in Figure 3 corresponds to the computation of the Schur complement matrix, and CHOLESKY does to the Cholesky factorization of the Schur complement matrix.

The largest scale problem arising from the quantum chemistry [8] which we can solve involves a $24,503 \times 24,503$ Schur complement matrix. See Yamashita et al. [16] for more details. Zhao et al. [17] proposed another SDP formulation of the quantum chemistry. The size of matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ in their formulation are very large compared with the SDP formulation proposed in [8], though the size of their Schur complement matrix is not so large (See Table 4). We solved these two problems on the PC cluster which we call ACT-JST cluster. The ACT-JST cluster in Tokyo Denki University has 40 nodes (each node has 1 or 2 GB memory) and 80 CPUs (Athlon 1.2GHz).

As we described above, each processor must maintain all input data and variables, whereas the Schur complement matrix is stored on distributed memory. Therefore, the current version of the SDPARA is suitable for a large scale SDP problem which involves a large Schur complement matrix and not so large number of variables like the SDP problem (1) in Table 4. The SDP problem (2) in Table 4 requires that each processor needs more than 1 GB for input data, variables and the distributed Schur complement matrix. When an SDP problem to be solved gets larger, we need to distribute input data and variables among the processors to save memory in each processor.

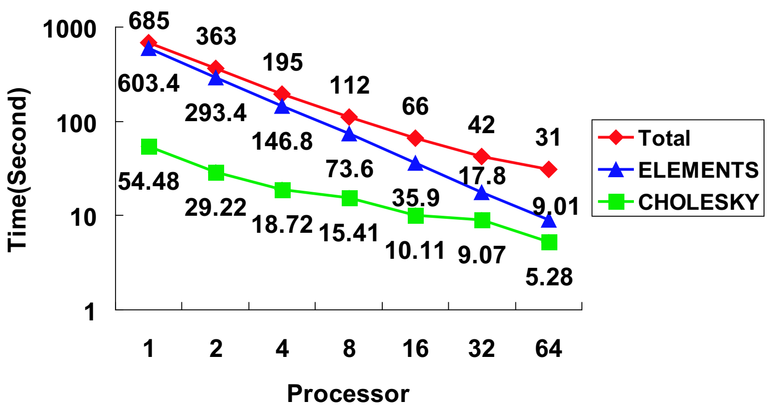

Figure 3: High scalability of the SDPARA on Presto III

Table 4: Large scale problems arising from the quantum chemistry

problem	(1):Nakata et al. [8]	(2):Zhao et al. [17]
size of the Schur complement matrix ($= m$)	24,503	7,230
size of $\boldsymbol{X}$ and $\boldsymbol{Y}$ ($= n$)	630	5,990
time(s.)	4153.5 (64CPUs)	38009.0 (32CPUs)

We also developed a new software package called the SDPA-C to exploit the sparsity structure of large scale SDP problems based on some fundamental results about positive semidefinite matrix completion [5, 9]. The SDPARA-C [10] is a parallel implementation of the SDPA-C using the parallel computing techniques including the ones used for the SDPARA. We confirmed that the SDPARA-C attains high scalability and works efficiently to solve large scale and sparse SDP problems through some numerical experiments on PC clusters.

The point we have to notice is that the SDPARA and the SDPARA-C are designed to be executed on PC clusters in which all nodes are connected by high-speed network devices, since the two software packages require much amount of network communication between the nodes for the parallel Cholesky factorization. We discuss how users can access to the SDPARA through the Internet in the next section.

5 Future Works

We are planning to apply grid and cluster computing to some other optimization problems such as the vehicle routing problem with time windows and more general mixed integer programming problems. So far we have mainly employed the Ninf GridRPC system (which we call Ninf-1). Tanaka et al. [14] have recently redesigned the Ninf and implemented a new GridRPC system called Ninf-G. The Ninf-G is a full re-implementation of the Ninf-1 using Globus Toolkit. This takes advantage of high interoperability with some other Globus-based Grid systems. We have plans to reconstruct our software packages such as the SCRM and the PHoM using the Ninf-G and Globus Toolkit.

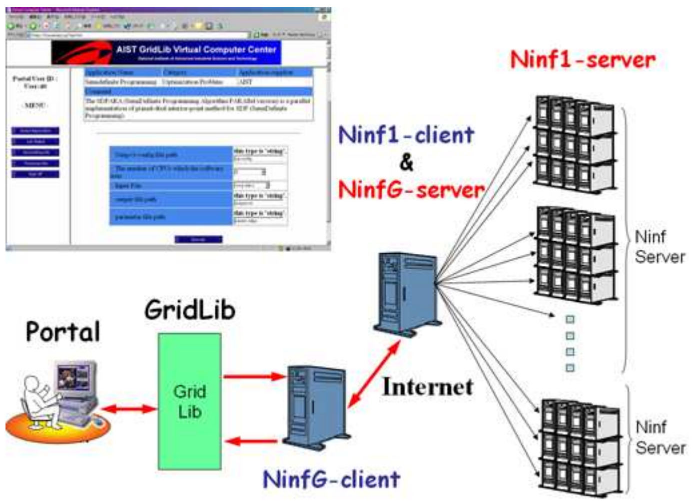

Figure 4: A Grid Portal System for the SDPARA

We are also developing a grid portal system for some optimization software packages including the SDPARA. The system will enable users to easily perform parallel computation through the Internet. GridLib in Figure 4 developed by AIST (National Institute of Advanced Industrial Science and Technology, Japan) provides a development framework to construct a virtual computer center as ASP (Application Service Provider). For a GridLib user, no particular knowledge about the Web securities and the Web programming is needed. We have already finished making a grid portal system for the SDPARA with a little deal of trouble. We consider that a few modifications to the SDPARA are necessary to portalize it. The user first accesses the Web portal site, then selects an application and a problem to be solved. GridLib starts up a Ninf-G client program associated with the application selected, and sends the problem to a Ninf-G server over the Internet. Now the Ninf-G server plays the role of a Ninf-1 client, which calls Ninf-1 servers among a PC cluster to execute the SDPARA. After finishing the execution of the SDPARA, the result is back to the Web portal site following the reverse route.

References

[1] K. Anstreicher, N. Brixius, J-P. Goux and J. Linderoth: Solving Large Quadratic Assignment Problems on Computational Grids. *Mathematical Programming*, **91** (2002) 563-588 .

[2] D. Applegate, R. Bixby, V. Chvátal and W. Cook: Implementing the Dantzig-Fulkerson-Johnson Algorithm for Large Traveling Salesman Problems. *Mathematical Programming*, **97** (2003) 91-153.

[3] K. Fujisawa, M. Kojima and K. Nakata: Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming. *Mathematical Programming*, **79** (1997) 235-253.

[4] K. Fujisawa, M. Kojima, A. Takeda and M. Yamashita: High Performance Grid and Cluster Computing for Some Optimization Problems. Research Report B-400, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo, Japan, December 2003.

[5] M. Fukuda, M. Kojima, K. Murota and K. Nakata: Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework. *SIAM Journal on Optimization*, **11** (2000) 647-674.

[6] T. Gunji, S. Kim, M. Kojima, A. Takeda, K. Fujisawa and T. Mizutani: PHoM – a Polyhedral Homotopy Continuation Method. December 2002, revised January 2003. To appear in *Computing*.

[7] M. Kojima and L. Tuncel: Cones of Matrices and Successive Convex Relaxations of Nonconvex Sets. *SIAM Journal on Optimization*, **10** (2000) 750-778.

[8] M. Nakata, H. Nakatsuji, M. Ehara, M. Fukuda, K. Nakata and K. Fujisawa: Variational Calculations of Fermion Second-Order Deduced Density Matrices by Semidefinite Programming Algorithm. *Journal of Chemical Physics*, **114** (2001) 8282-8292.

[9] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima and K. Murota: Exploit Sparsity in Semidefinite Programming via Matrix Completion II: Implementation and Numerical Results. *Mathematical Programming*, **95** (2003) 303-327.

[10] K. Nakata, M. Yamashita, K. Fujisawa and M. Kojima: A Parallel Primal-Dual Interior-Point Method for Semidefinite Programs Using Positive Definite Matrix Completion. Research Report B-398, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo, Japan, November 2003.

[11] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima and H. Takagi: Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure. *HPCN'97*, **LNCS-1225** (1997) 491-502.

[12] A. Takeda, K. Fujisawa, Y. Fukaya and M. Kojima: Parallel Implementation of Successive Convex Relaxation Methods for Quadratic Optimization Problems. *Journal of Global Optimization*, **24** (2002) 237-260.

[13] A. Takeda, M. Kojima and K. Fujisawa: Enumeration of All Solutions of a Combinatorial Linear Inequality System Arising from the Polyhedral Homotopy Continuation Method. *Journal of the Operations Research Society of Japan*, **45** (2002) 64-82.

[14] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura and S. Matsuoka: Ninf-G : A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, **1** (2003) 41-51.

[15] M. Yamashita, K. Fujisawa and M. Kojima: Implementation and Evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0). *Journal of Optimization Methods and Software*, **18** (2003) 491-505.

[16] M. Yamashita, K. Fujisawa and M. Kojima: SDPARA : SemiDefinite Programming Algorithm PARAllel Version. *Journal of Parallel Computing*, **29** (2003) 1053-1067.

[17] Z. Zhao, B. J. Braams, M. Fukuda, M. L. Overton and J. K. Percus: The Reduced Density Matrix Method for Electronic Structure Calculations and the Role of Three-Index Representability. To appear in *Journal of Chemical Physics*, 2004.