Implementation of dual iterative substructuring methods on a parallel computer

Advisor : Professor Lee, Chang-Ock

by

Cho, Heyrim
Department of Mathematical Sciences
Korea Advanced Institute of Science and Technology

A thesis submitted to the faculty of the Korea Advanced Institute of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Department of Mathematical Sciences

Daejeon, Korea
2009. 5. 26.
Approved by

Professor Lee, Chang-Ock
Advisor

MMAS 20073560 Cho, Heyrim. Implementation of dual iterative substructuring methods on a parallel computer.
Department of Mathematical Sciences 2009. 27p.
Advisor Prof. Lee, Chang-Ock.

Abstract

This thesis discusses parallel implementation of dual iterative substructuring methods, particularly FETI-DP method and enhanced penalty method. We optimize the algorithm for the enhanced penalty method and compare its performance with the FETI-DP method, which is the most renowned substructuring method. The substructuring method divides the domain into local subdomains and allocates each of them into parallel processors to solve the local problems. Enhanced penalty method adds a strong continuity constraint to the FETI-DP method by measuring the difference on the edge, which we call the penalty term. This additional term accelerates the convergence, but produces more data communication between processors. It is not easy to compare the performance between the two methods. However, it is obvious that optimizing the communication routine will be crucial in the performance of the enhanced penalty method.

Here, we present the process of optimizing communication routines in the enhanced penalty method. Our analysis shows that calculating the global inner product is the most expensive communication step in this algorithm and a parallel computer with effective network and memory system that can gather the values from all processor simultaneously is needed for efficient implementation. Overall, we conclude that FETI-DP method is recommended when the subdomain problem size is small and enhanced penalty method becomes more effective when the subdomain problem size is above certain level. Both method turn out to be efficient solvers for the elliptic partial differential equation when parallel computer is available. But the choice among FETI-DP and enhanced penalty method must be made carefully considering the number of processors, computing power, and network performance of the parallel computer. Particularly, for our test problem in two dimensional domain, enhanced penalty method becomes more efficient when the local problem size becomes larger than 128×128 in 2 dimensional problem in the Gaia system.

Contents

List of Tables

List of Figures

1. Introduction

For recent several decades, parallel computers have become popular and as a result, parallel algorithms have been studied as effective solvers for large problems. Domain decomposition method is a numerical solver for partial differential equation, that can be used in parallel computers. The basic idea is to divide the domain into several parts and solve the governing equation on each of the local subdomains. Computation on the local subdomains can be distributed into the parallel processors. After solving the local problems, the solutions are assembled to obtain the global solution. The method usually includes interconnecting procedure or some constraints that ensure the local solutions become identical on the interfaces of the local subdomains or overlapping regions.

Particularly, dual iterative substructuring method is adapted to solve the self adjoint second order elliptic partial differential equation with boundary condition. The domain is divided into non-overlapping subdomains, and then a finite element solution is obtained with iterative methods.

As a dual iterative substructuring method, the Finite Element Tearing and Interconnecting (FETI) method [1] was suggested by Farhat and Roux. The continuity constraints given on the boundary of the subdomains are imposed by Lagrange multipliers. The system is reduced by eliminating the subdomain variables except for the Lagrange multipliers, then the remaining system is solved with the preconditioned conjugate gradient (PCG) method. The method solves the partial differential equation on each subdomain and the Lagrange multipliers match the boundary condition. In spite of the additional variables and computational cost on the interface, the algorithm is simple and robust compared to the other substructuring methods.

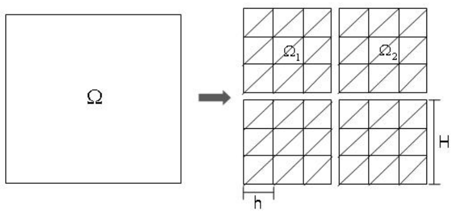


Figure 1.1: Decomposed domains with subdomain size H and mesh size h.

The convergence can be estimated by the condition number of the system matrix, which we solve with the PCG method. Let e^k be the error of the k-th iteration of the PCG method and κ be the condition number of the preconditioned system matrix. Then e^k can be estimated as $\|e^k\| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e^0\|$, where $\|\cdot\|$ is the norm defined on the solution space [2]. It is shown in [3] that the condition number of the FETI method is bounded by $O\left(\left(1+\log\frac{H}{h}\right)^3\right)$, which means that it is numerically scalable algorithm with respect to H/h, where H is the subdomain size and h is the mesh size. However, the FETI method includes a procedure for solving the singular problems that occur when all corners of the subdomain are not attached to the boundary of the whole domain.

Dual Primal FETI (FETI-DP) method [4], also suggested by Farhat *et al.*, is one of the most advanced dual iterative substructuring methods. It improved the FETI method by removing the corner singularity. The degree of freedoms at the corners are reduced by defining them as a common node over the surrounding subdomains. The continuity constraint on the interface is stronger than the FETI method, since the corner's continuity is directly imposed. Lagrange multipliers are defined as usual on the edge nodes except the corner nodes. In two dimensional problems, the condition number bound estimation turned out to be $O\left(\left(1+\log\frac{H}{h}\right)^2\right)$ in [5] and the convergence was accelerated.

Enhanced penalty method [6] is a variant of FETI-DP method, on which this paper is primarily based. A penalty function is added to the Lagrangian functional and it measures the difference between the interface nodes of the adjacent subdomains. This term contains positive parameter η across the subdomains so that it enforces a stronger continuity constraint than FETI-DP method. As a result, the condition number bound is estimated as $3\left(\frac{C}{\eta}+1\right)$, a constant only containing the imposed parameter η. In other words, the convergence is independent of the mesh size h.

The implementation of these methods on parallel computers includes several issues and it is hard to compare the performance between them at a glance. Enhanced penalty method seems to be superior to FETI-DP method in terms of the condition number estimate, but the algorithm includes additional computation on the interface nodes coming from the added penalty function. Also, it contains far more communication between processors than FETI-DP method, which means that the overall effectiveness can be largely affected by the network system. Moreover, while FETI-DP method is computed separately on each subdomain scale, enhanced penalty method is solved as a whole system in a global scale.

In this thesis, we will briefly review the dual iterative substructuring methods including FETI-DP method and enhanced penalty method and computationally verify the theoretical results. Then we will compare the performance on parallel computers between them.

2. Domain Decomposition Method

2.1 Dual substructuring method

In this section, we will show the basic idea of dual substructuring method by considering the simple Poisson problem with homogeneous Dirichlet boundary condition on two separated domains. First, we consider the Poisson problem on a bounded polygonal domain $\Omega \subset \mathbf{R}^2$ as

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

where f is an L^2 function on Ω. It is well known that the solution of (2.1) is equivalent to the minimizer of the variational problem,

$$\min_{v \in H_0^1(\Omega)} \mathcal{J}(v) = \min_{v \in H_0^1(\Omega)} \left(\frac{1}{2} a(v,v)_\Omega - (f,v)_\Omega \right),$$

where

$$a(u,v)_\Omega = \int_\Omega \nabla u \cdot \nabla v dx,$$

$$(f,v)_\Omega = \int_\Omega f v dx.$$

As usual, $H^1(\Omega)$ and $H_0^1(\Omega)$ are the Sobolev spaces defined as

$$H^1(\Omega) = \{ v \in L^2(\Omega) | \ \partial^\alpha v \in L^2(\Omega), |\alpha| \le 1 \},$$

$$H_0^1(\Omega) = \{ v \in H^1(\Omega) | \ v|_{\partial\Omega} = 0 \}.$$

Respectively, to investigate the idea of dual iterative substructuring method, let's divide the domain into two non-overlapping subdomains $\{\Omega_i\}_{i=1}^2$ such that $\overline{\Omega} = \bigcup_{i=1}^2 \overline{\Omega}_i$. Then (2.1) becomes equivalent to

$$\min_{\substack{v^i \in H^1(\Omega_i) \\ v^i = 0 \text{ on } \partial\Omega \cap \partial\Omega_i \\ v^1 = v^2 \text{ on } \partial\Omega_1 \cap \partial\Omega_2}} \sum_{i=1}^2 \mathcal{J}_i(v^i) = \min_{\substack{v^i \in H^1(\Omega_i) \\ v^i = 0 \text{ on } \partial\Omega \cap \partial\Omega_i \\ v^1 = v^2 \text{ on } \partial\Omega_1 \cap \partial\Omega_2}} \sum_{i=1}^2 \left(\frac{1}{2} a(v^i, v^i)_{\Omega_i} - (f, v^i)_{\Omega_i} \right).$$

The two minimizing solutions on each subdomains are attached at the interface by the continuity constraint to guarantee the continuity of the whole solution. This condition ensures the local minimizing solutions u_1 and u_2 be equal to the solution of the original problem.

To implement the constraint, various approaches have been made such as Lagrange multiplier, penalty function, and augmented Lagrange methods. In this section, we will introduce the Lagrange multiplier based method. Suppose that μ is a Lagrange multiplier defined on the interface $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$, which we call the dual variable. Then the variational problem becomes equivalent to finding the solution (u^1, u^2, λ) of the saddle point problem,

$$\min_{v^i \in H^1(\Omega_i)} \max_{\mu \in L^2(\Gamma)} \left(\sum_{i=1}^{2} \mathcal{J}_i(v^i) + (v^1 - v^2, \mu)_\Gamma \right). \tag{2.2}$$

The Lagrange multiplier forces the saddle point solution u^1 and u^2 to attain the same value on Γ. Now we will formulate a numerical method using finite elements with the parallel structure of the subdomains generated by domain decomposition.

2.2 Finite Element Tearing and Interconnecting methods

Before we introduce the numerical method, let's define the finite element space and notations related to the finite element approximation. The $\mathcal{P}_1$ conforming finite element space on Ω is defined as

$$S_h = \{v_h \in H_0^1(\Omega) \cap \mathcal{C}^0(\overline{\Omega}) | \quad {}^\forall\tau \in \mathcal{T}_h, v_h|_\tau \in \mathcal{P}_1(\tau)\},$$

where $C^0(\overline{\Omega})$ is the space of continuous function on $\overline{\Omega}$, $\mathcal{T}_h$ is the family of regular triangulations on Ω, $\mathcal{P}_1(\tau)$ is the standard $\mathcal{P}_1$ conforming space and h is the maximal mesh size of $\mathcal{T}_h$. Then the finite element approximation of the Poisson problem (2.1) becomes finding $u_h \in S_h$ to satisfy

$$a(u_h, v_h) = (f, v_h) \qquad {}^\forall v_h \in S_h, \tag{2.3}$$

which is equivalent to the variational problem

$$\min_{v \in S_h} \left(\frac{1}{2} a(v, v) - (f, v) \right).$$

Now, let's formulate the finite element space of the decomposed domains. Suppose that we divide the domain Ω into N non-overlapping subdomains $\{\Omega_s\}_{s=1}^N$, where Ω_s is a polygonally shaped open subset of Ω. Also, we define the common interface of adjacent subdomains Ω_s and Ω_t as $\Gamma_{st} = \partial\Omega_s \cap \partial\Omega_t$ and the union of common interfaces as $\Gamma = \bigcup_{s<t} \Gamma_{st}$. Then, the finite element space on the subdomain Ω_s can be similarly defined as

$$X_h^s = \{v_h^s \in H^1(\Omega_s) \cap \mathcal{C}^0(\overline{\Omega}_s) |^\forall \tau \in \mathcal{T}_h^s, v_h^s|_\tau \in \mathcal{P}_1(\tau), v_h^s|_{\partial\Omega \cap \partial\Omega_s} = 0\},$$

where $\mathcal{T}_h^s$ is the regular triangulation of subdomain Ω_s and h is the maximal mesh size among all subdomains. X_h^s's can be assembled into X_h as

$$X_h = \left\{ v \mid v = \{v_h^s\}_{s=1}^N \in \prod_{s=1}^N X_h^s \right\}$$

equipped with the norm $\|v\|_{X_h} = \left(\sum_{s=1}^N \|v_h^s\|_{H^1(\Omega_s)}^2 \right)^{\frac{1}{2}}$, where $\|\cdot\|_{H^1(\Omega_s)}$ is the usual Sobolev norm.

The variational formulation can be converted into finding the minimizer of energy functional $\mathcal{J} : X_h \to \mathbf{R}$ defined as

$$\min_{v \in X_h} \mathcal{J}(v) = \min_{v \in X_h} \left(\frac{1}{2} a_h(v, v) - (f, v) \right) \qquad \text{subject to } Bv = 0, \tag{2.4}$$

where $a_h(u, v) = \sum_{s=1}^N \int_{\Omega_s} \nabla u \cdot \nabla v dx$ is a bilinear functional on $X_h \times X_h$, B is a signed Boolean matrix enforcing the pointwise continuity condition on the interface, and $(f, v) = \sum_{s=1}^N \int_{\Omega_s} f v dx$.

As we mentioned, the FETI method imposes the continuity constraint along the interface using Lagrange multipliers. Suppose that the Lagrange multiplier vector μ is in the Euclidean space $\mathbf{R}^E$ with the usual Euclidean inner product $\langle \cdot , \cdot \rangle$, where E is the number of the matching points on the interface. Then the saddle point solution (u, λ) of Lagrangian functional $\mathcal{L} : X_h \times \mathbf{R}^E \to \mathbf{R}$,

$$\min_{v \in X_h} \max_{\mu \in \mathbf{R}^E} \mathcal{L}(v, \mu) = \min_{v \in X_h} \max_{\mu \in \mathbf{R}^E} (\mathcal{J}(v) + \langle Bv, \mu \rangle)$$

equals to the solution of (2.4). Moreover, the above saddle point problem can be converted into a saddle point system as finding $(u, \lambda) \in X_h \times \mathbf{R}^E$ satisfying

$$\begin{aligned} a_h(u, v) - \langle Bv, \lambda \rangle = (f, v) \qquad & {}^\forall v \in X_h \\ \langle Bu, \mu \rangle = 0 \qquad & {}^\forall \mu \in \mathbf{R}^E. \end{aligned} \tag{2.5}$$

This formula can be written into an algebraic system, and we obtain the finite element solution of (2.3) by solving the algebraic system with an appropriate solver.

The FETI method is a highly efficient method regarding the computation and convergence property. However, the problem includes solving the pseudo inverse because it involves singular problems when all corners of the subdomain are not attached to the boundary of the whole domain. FETI-DP method resolves this problem by imposing a single degree of freedom at the corner node. The corner point of the surrounding subdomains is defined as a common node. Let's define the finite element space including this property by assembling

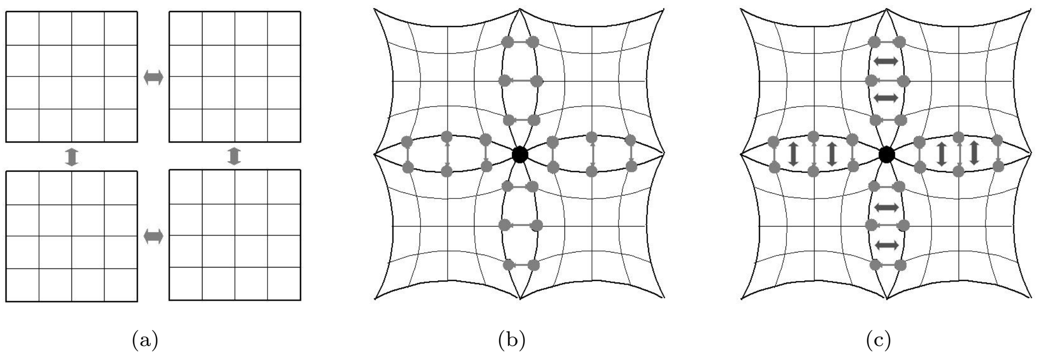


Figure 2.1: Characterization of Finite Element Tearing and Interconnecting methods : (a) FETI, (b) FETI-DP, and (c) enhanced penalty method

the X_h^s's as

$$X_h^C = \{ \ v \mid v = \{v_h^s\}_{s=1}^N \in \prod_{s=1}^N X_h^s, v \text{ is continuous at each corner } \},$$

equipped with the same norm defined as in the space X_h.

FETI-DP method finds the solution on X_h^C instead of X_h. The only difference from the FETI method is that the continuity constraint at the corner is directly imposed in the solution space. The variational formula and the saddle point formula are defined on X_h^C. The saddle point problem of FETI-DP method becomes solving the Lagrangian functional $\mathcal{L}(v,\mu)$ defined on $X_h^C \times \mathbf{R}^E$ as

$$\min_{v\in X_h^C} \max_{\mu\in\mathbf{R}^E} \mathcal{L}(v,\mu) = \min_{v\in X_h^C} \max_{\mu\in\mathbf{R}^E} \left(\mathcal{J}(v) + \langle Bv, \mu\rangle\right), \tag{2.6}$$

where $\mathcal{J}(v)$ is the energy functional on X_h^C. We can easily observe that the saddle point solution $(u,\lambda) \in X_h^C \times \mathbf{R}^E$ equals to the solution of the original problem (2.3).

Enhanced penalty method slightly changes FETI-DP method by imposing a stronger continuity constraint on the interface than FETI-DP method. We define a bilinear function on $X_h^C \times X_h^C$ as

$$J_\eta(u,v) = \sum_{s<t} \frac{\eta}{h} \int_{\Gamma_{st}} (u^s - u^t)(v^s - v^t)ds, \qquad \eta > 0.$$

The method adds this penalty function to the Lagrangian functional, which evaluates the difference of a function between the interface nodes across the subdomains. The difference is measured with the L^2 norm multiplied by a positive parameter η and divided by the maximal mesh size h. Now our saddle point formulation becomes

$$\min_{v \in X_h^C} \max_{\mu \in \mathbf{R}^E} \mathcal{L}_\eta(v,\mu) = \min_{v \in X_h^C} \max_{\mu \in \mathbf{R}^E} \left(\mathcal{L}(v,\mu) + \frac{1}{2} J_\eta(v,v) \right). \tag{2.7}$$

The symmetry and X_h^C-ellipticity of the $a_h(v,v)$ are maintained in $a_\eta(u,v) = a_h(u,v) + J_\eta(u,v)$ and $\mathcal{J}_\eta(v) = \frac{1}{2} a_\eta(v,v) - (f,v)$ satisfies

$$\mathcal{J}_\eta(u_h) = \min_{v \in X_h^C, Bv=0} \mathcal{J}_\eta(v) = \min_{v \in X_h} \mathcal{J}(v).$$

Therefore, the saddle point problem (2.7) is uniquely solvable and it obtains the same solution as the original problem (2.3). We can also check the convergence of the finite element solution u_h to the Poisson problem (2.1).

We give a remark that the Lagrangian of enhanced penalty method (2.7) contains two terms $\langle Bv, \mu \rangle$ and $J_\eta(v,v)$ to impose the continuity constraint. If we choose $\eta = 0$, the enhanced penalty method is identical to FETI-DP method. Alternatively, if we remove the Lagrangian multiplier term, the penalty function solely imposes the continuity constraint to the minimizer. It seems that the combination of these two terms is redundant, but the enhanced penalty method benefits from the advantage of both approaches. Generally, the pointwise matching term $\langle Bv, \mu \rangle$ has slow convergence but ensures the convergence to the solution. On the other hand, the penalty term $J_\eta(v,v)$ accelerates the convergence of the method but needs sufficiently large η for numerical stability. By merging the both terms, the penalty term accelerates the convergence of $\langle Bv, \mu \rangle$ and the pointwise matching term ensures the numerical stability of $J_\eta(v,v)$ without increasing the parameter η. This combination of both terms accomplishes a powerful result which makes the enhanced penalty method as a strongly scalable algorithm.

3. Implementation of Dual Iterative Substructuring Methods

3.1 Algebraic formulation and computational issues

In this section, we will derive the algebraic formulation of the dual iterative substructuring methods and discuss the computational issues of the algorithm.

We will start from the saddle point system of the FETI method (2.5). Suppose that K represent the stiffness matrix generated from $a_h(v,v)$. Then the algebraic system can be written as

$$\begin{bmatrix} K & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix},$$

where $K = \operatorname{diag}(K^1, \cdots, K^N)$ is a block diagonal matrix, K^s is the subdomain stiffness matrix on Ω_s, and $B = \begin{bmatrix} B^1 & \cdots & B^N \end{bmatrix}$ is a signed Boolean matrix.

The algebraic system above can be rewritten in terms of the dual variable λ. Note that while solving the system in terms of u^s, we must calculate the pseudo inverse in the subdomains where K^s is singular.

To introduce the algorithm of FETI-DP method [4], we regard the corner nodes as having multiple values but in fact equal. First, let's split the degree of freedom vector on each subdomain as

$$u^s = \begin{bmatrix} u_i^s \\ u_e^s \\ u_c^s \end{bmatrix} = \begin{bmatrix} u_r^s \\ u_c^s \end{bmatrix},$$

where u_i^s denotes the degree of freedom in the interior of the subdomains, u_c^s the degree of freedom at the corner nodes, and u_e^s the remaining degree of freedom on the interface edge. Again, let u_r^s be the assembled degree of freedom on the interior and edge except for the corner nodes. We decompose the subdomain stiffness matrix K^s using these notations as

$$K^s = \begin{bmatrix} K_{ii}^s & K_{ie}^s & K_{ic}^s \\ K_{ie}^{s\,T} & K_{ee}^s & K_{ec}^s \\ K_{ic}^{s\,T} & K_{ec}^{s\,T} & K_{cc}^s \end{bmatrix} = \begin{bmatrix} K_{rr}^s & K_{rc}^s \\ K_{rc}^{s\,T} & K_{cc}^s \end{bmatrix}.$$

Here, each block matrix represents the corresponding stiffness matrix. For example K_{ii}^s is the interior stiffness matrix on the subdomain Ω_s. Then, the system matrix of FETI-DP method can be formulated as

$$\begin{bmatrix} K_{rr}^1 & \cdots & 0 & K_{rc}^1 L_c^1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & K_{rr}^N & K_{rc}^N L_c^N \\ {L_c^1}^T {K_{rc}^1}^T & \cdots & {L_c^N}^T {K_{rc}^N}^T & \sum_{s=1}^N {L_c^s}^T K_{cc}^s L_c^s \end{bmatrix} \begin{bmatrix} u_r^1 \\ \vdots \\ u_r^N \\ u_c \end{bmatrix} = \begin{bmatrix} f_r^1 - {B^1}^T \lambda \\ \vdots \\ f_r^N - {B^N}^T \lambda \\ \sum_{s=1}^N {L_c^s}^T f_c^s \end{bmatrix},$$

$$\sum_{s=1}^N B^s u_r^s = 0$$

where u_c is the global vector of the corner degree of freedoms and L_c^s is the matrix connecting the global corner vector u_c to each subdomain corner nodes, i.e., $u_c^s = L_c^s u_c$. We rewrite the system as

$$\begin{bmatrix} K_{rr} & K_{rc} & B^T \\ {K_{rc}}^T & K_{cc} & 0 \\ B & 0 & 0 \end{bmatrix} \begin{bmatrix} u_r \\ u_c \\ \lambda \end{bmatrix} = \begin{bmatrix} f_r \\ f_c \\ 0 \end{bmatrix}, \tag{3.1}$$

where $K_{cc} = \sum_{s=1}^N {L_c^s}^T K_{cc}^s L_c^s$ is the global corner stiffness matrix, K_{rr} is a block diagonal subdomain stiffness matrix restricted to the remaining nodes, and K_{rc} is the coupling stiffness matrix between them. Note here that the K_{rr}^s matrix is symmetric positive definite, which means that it is invertible. This is one of the major advantage of the FETI-DP method, so that we can invert each block system in terms of u_r^s without calculating the pseudo inverse. Then we can convert the system into

$$\begin{bmatrix} F_{cc} & {F_{rc}}^T \\ F_{rc} & F_{rr} \end{bmatrix} \begin{bmatrix} u_c \\ \lambda \end{bmatrix} = \begin{bmatrix} -d_c \\ d_r \end{bmatrix},$$

where

$$F_{rr} = \sum_{s=1}^N B^s (K_{rr}^s)^{-1} {B^s}^T,$$

$$F_{rc} = \sum_{s=1}^N B^s (K_{rr}^s)^{-1} K_{rc} L_c^s,$$

$$F_{cc} = K_{cc} - \sum_{s=1}^N \left[(K_{rc}^s L_c^s)^T (K_{rr}^s)^{-1} (K_{rc}^s L_c^s) \right],$$

$$d_r = \sum_{s=1}^N B^s (K_{rr}^s)^{-1} f_r^s,$$

$$d_c = \sum_{s=1}^N \left({L_c^s}^T f_c^s - {L_c^s}^T {K_{rc}^s}^T (K_{rr}^s)^{-1} f_r^s \right).$$

The system can be reduced in terms of the dual variable λ as

$$F\lambda := \left(F_{rr} + F_{rc}(F_{cc})^{-1}{F_{rc}}^T\right)\lambda = d_r - F_{rc}(F_{cc})^{-1}d_c \tag{3.2}$$

and this dual system matrix can be solved iteratively by preconditioned conjugate gradient (PCG) method since F is symmetric positive definite. Inside the CG algorithm, calculation with the matrix F is split into calculation with F_{rr} and with $F_{rc}(F_{cc})^{-1}{F_{rc}}^T$. The second step includes solving F_{cc} which we call the course problem, and this can also be solved with the CG method.

Every matrix notated in (3.2) consists of the summation over the subdomains. This feature makes FETI-DP method easily parallelizable, i.e., it can be calculated separately in subdomain level and then summed up. Another thing to notice is that all matrices contain solving the stiffness matrix ${K^s_{rr}}^{-1}$. This can also be solved directly or iteratively with CG method. In fact, we will use direct Cholesky decomposition to solve this matrix. For preconditioning, simple Dirichlet preconditioner $F_D = BS_{rr}B^T$ will be used, where $S_{rr} = \mathrm{diag}\left(K^s_{ee} - {K^s_{ie}}^T(K^s_{ii})^{-1}K^s_{ie}\right)_{s=1}^N$ [4].

Enhanced penalty method [6] can be similarly computed as above. The remaining nodes and the corner nodes are eliminated step by step, and the system of the Lagrangian variable is obtained. However, the major difference is that the stiffness matrix of the remaining nodes can not be solved in a local subdomain scale, but in a global scale.

The system matrix of the enhanced penalty method differs from (3.1) only in K_{rr}. Let's notate it as K^η_{rr}, then

$$K^\eta_{rr} = \begin{bmatrix} K_{ii} & K_{ie} \\ K_{ie}^T & K_{ee} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \eta J \end{bmatrix}$$

and J is expressed as $J = B^T\mathrm{diag}(J_B)B$, where J_B is a subdomain size matrix induced from

$$\frac{1}{h}\int_{\Gamma_{ij}} \phi\psi ds, \qquad {}^\forall\phi, \psi \in X_h^C|_{\Gamma_{ij}}.$$

Note that K^η_{rr} is not a block diagonal matrix with respect to the subdomains, because the added penalty term includes the relationship between the adjacent subdomains in J. The final system matrix of the dual variable becomes

$$F_\eta\lambda := \left(F^\eta_{rr} + F^\eta_{rc}(F^\eta_{cc})^{-1}{F^\eta_{rc}}^T\right)\lambda = d^\eta_r - F^\eta_{rc}(F^\eta_{cc})^{-1}d^\eta_c \tag{3.3}$$

where

$$F^\eta_{rr} = B_r(K^\eta_{rr})^{-1}B_r^T, \quad F^\eta_{rc} = B_r(K^\eta_{rr})^{-1}K_{rc}, \quad F^\eta_{cc} = K_{cc} - K_{rc}^T(K^\eta_{rr})^{-1}K_{rc},$$

$$d_r = B_r^T(K^\eta_{rr})^{-1}f_r, \quad d_c = f_c - K^T_{rc}(K^\eta_{rr})^{-1}f_r.$$

As we mentioned above, K_{rr}^{η} is not block diagonal matrix with respect to subdomains and must be solved globally with the parallel CG method [7]. Perhaps, we can still implement each subdomain problems separately in a parallel machine and the data is transferred only when we compute the J matrix. Similar to FETI-DP method, the dual system matrix F_{η} and the course problem F_{cc}^{η} can be solved iteratively with CG algorithms.

The convergence rate of FETI-DP method and enhanced penalty method can be compared with the condition number bound of the system matrix solved in the CG algorithm. For FETI-DP method (3.2), it can be estimated as $\kappa(F) \leq C\left(1 + \log\frac{H}{h}\right)^2$ [5]. The bound includes H/h and the algorithm is called numerically scalable with respect to H/h. It means that the convergence is not deteriorated though we make the mesh size smaller if a certain ratio between H and h is maintained. This is a desirable property for parallel algorithms, because the mesh size h can be reduced without losing the convergence rate if the number of processor allows to reduce the subdomain size H.

The remarkable property of the enhanced penalty method is that the condition number bound of F_{η} in (3.3) is estimated as $\kappa(F_{\eta}) \leq 3\left(\frac{C}{\eta} + 1\right)$, a constant bound independent of h and H. Moreover, $\kappa(F_{\eta}) \leq 3$ for large η [6]. This is called as a strongly scalable algorithm, since the convergence is not affected regardless of dividing the mesh and subdomain size.

Yet, the enhanced penalty method has an inherent difficulty, which is the inner problem $(K_{rr}^{\eta})^{-1}$. The condition number bound of K_{rr}^{η} is estimated as

$$\kappa(K_{rr}^{\eta}) \leq C\left(\left(\frac{H}{h}\right)^2\left(1 + \log\frac{H}{h}\right)(1+\eta)\right),$$

where η is included in the bound as a factor. Therefore, a preconditioner is designed for the enhanced penalty method to remove the factor of η. The preconditioner can be defined as

$$M = \begin{bmatrix} K_{ii} & 0 \\ 0 & K_{ee} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \eta J \end{bmatrix},$$

then the condition number estimate becomes

$$\kappa(M^{-1}K_{rr}^{\eta}) \leq C\left(\left(\frac{H}{h}\right)\left(1 + \log\frac{H}{h}\right)\right)$$

. See [6].

3.2 Parallel Implementation

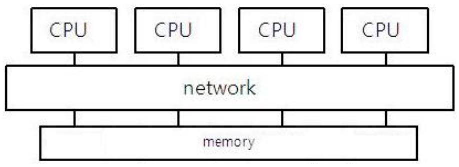
3.2.1 Hardware and software issues

Parallel computing refers to processing multiple instructions simultaneously to reduce the wall clock computing time. Recently, the computing resource for parallel programs became popular, from multi core computers to workstation clusters, so that parallel algorithms is more valuable and widely used. In this section, we will briefly overview parallel computing in perspective of hardware and software [8].

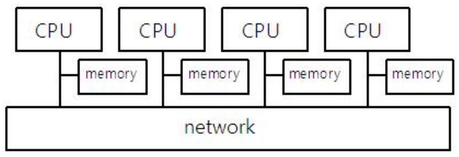
Parallel computers can be divided into two groups, shared memory system and distributed memory system, where the difference comes from the main memory access. The shared memory system locates the main memory in a single address space to share with all processers, while the distributed memory system has local address space for each processors, either distributed logically or physically; see Figure 3.1. The distributed memory system has no limit in number of processors and can be extended larger than the shared memory system, but has a drawback of non-uniformity in memory access. Nowadays, the combination, called hybrid distributed-shared memory system, became popular, which is a cluster of shared memory systems.

Another issue that we must inspect is the network of the memory system. Bus-based architectures and switch-based architectures are some of the examples. Moreover, there are variety of switch-based architecture respect to the structure of the switch, such as ring type, cross bar type, etc. The data exchanging time can be totally different, so that the communication scheme must carefully consider the memory structure of the system [9].

The parallel programming software can be classified according to the memory system. For shared memory systems, OpenMP and POSIX Threads are the most well known application program interface, which produce multiple threads. The program is initiated at a single node, and then multi-threads are made when parallelization is available. The number



(a)



(b)

Figure 3.1: Parallel computers with (a) Shared memory system, and (b) Distributed memory system.

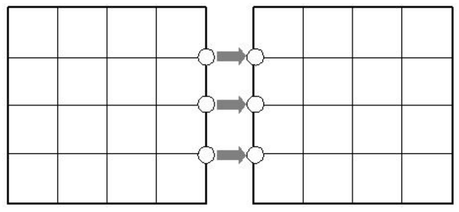
of threads need to be known and the compiler automatically allocates the tasks. These programs benefit from fast access to the outputs of other threads, since they are located together in the shared memory space. However, more general approach for parallel programs should be implemented in distributed memory system using the message passing between the processors. The most well known library for distributed memory system is Message Passing Interface (MPI). Communications occur through the network to transfer the data needed by other processors. The number of processor is not limited and implementation is available in both shared and distributed memory system. However, the tasks and communication must be explicitly allocated by the programmer, which makes the implementation more complicated. The library subroutines start with "MPI_" and they can be divided into one-to-one, one-to-all, and all-to-all communications. One-to-one communication is the basic process of sending and receiving the data. It is implemented with "MPI_ Sen" and "MPI_ Recv" functions. The argument of the command includes the name of the sending node and receiving node, data type, data size, and address, etc. One-to-all and all-to-all communications are combinations of one-to-one subroutines. The procedure of the communication is implicitly arranged in an optimized way.

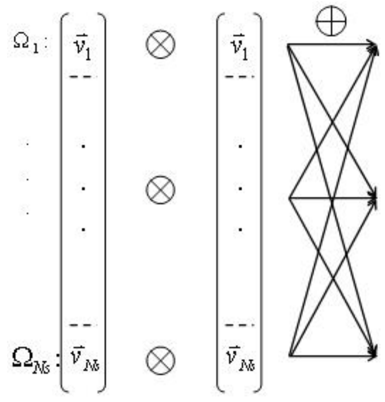
3.2.2 Communication in enhanced penalty method

The dual iterative substructuring methods are efficient solvers for self adjoint second order elliptic partial differential equations. Their apparent substructure can be easily implemented in parallel computers and numerical scalability allows them to effectively solve partial differential equations when parallel processors are available. The implementation of FETI-DP method has been tested before and the numerical scalability is well shown in [4].

In this thesis, we are primarily interested in implementing enhanced penalty method on parallel computers. As in FETI-DP method, the parallelism coming from the subdomains is similar. However, the difference between FETI-DP method and enhanced penalty method comes from the added penalty term measuring the jump across the subdomain edges. Besides the additional computation, this term produces two major data transformation process among the subdomains. Figure 3.2 illustrates them.

The first obvious communication arises in the matrix-vector product involving F_{rr}^{η} including the J matrix, where we measure the gap between the adjacent subdomains. Since we distribute the subdomains into different processors, the edge values must be sent and received across the neighboring subdomains. This process is one-to-one communication since the data is transferred from one processor to another processor. For each subdomain, $(H/h - 1)$ node values per edge must be sent and received.


(a)



(b)

Figure 3.2: Two major communication in the enhanced penalty method: (a) Send and receive the edge variables, (b) Global inner product.

The second communication comes from the parallel CG method solving F^{η}_{rr}. Unlike F_{rr} in FETI-DP method, F^{η}_{rr} is solved with its dimension in a global scale. Therefore, while working with the parallel CG algorithm, the inner product of the global vector lying on the whole domain must be computed. This computation is implemented by calculating the inner product of the local vector on their own domain, then gathering to sum all of the values, and finally spreading the result again to every nodes. This process includes all-to-all communication, where all of the processors participate. Although only single value is sent and received, the whole family of the processors must collaborate at the same time to proceed the algorithm.

4. Numerical Tests

Now we present the numerical results of FETI-DP method and enhanced penalty method, implemented on parallel computers. We verify the convergence and overall properties of the methods, make discussions on optimizing the parallel performance of the enhanced penalty method, and compare the parallel performance between two methods.

The simulation is done on IBM POWER5 Series p595, named "Gaia" maintained in Korea Institute of Science and Technology Information (KISTI) Supercomputing Center. Gaia, a hybrid shared-distributed memory system, consists of 10 nodes, where each node contains 64 Power5 processors running at 2.3GHz. All server nodes are connected by an IBM high performance federation switch network (HPS) and the Enhanced Distributed Switch network is imbedded within the nodes. Table 4.1 describes the capability of Gaia system. Gaia uses Parallel Environment (PE) 4.3 for message passing library, which is an IBM optimized library based on MPI. In addition, Engineering Scientific Subroutine Library (ESSL) 3.3 and Parallel ESSL (PESSL) 3.3 are installed as a basic linear algebra library, also IBM optimized library of the general versions such as Basic Linear Algebra Subprograms (BLAS), Linear Algebra Package (LAPACK), and Basic Linear Algebra Communication Subprograms (BLACS) [10].

Table 4.1: Computing environment of Gaia system

System	Nodes	CPU			Memory (GB)
		Total number	Processor	Rpeak (GFlops)	
p595	10	640 (64/node)	POWER 5+ (2.3GHz)	5,888 (588.8/node)	2,816

4.1 Computation and numerical scalability

The Poisson problem with homogeneous Dirichlet boundary condition (2.1) is tested on the square domain $\Omega = [0,1] \times [0,1]$ with an exact solution

$$u(x,y) = y(1-y)\sin(\pi x).$$

The domain Ω is divided into square shaped subdomains and uniform triangular element is chosen. As the usual notation, h stands for the mesh size and H is the subdomain size. Hence, the subdomain number becomes $N = 1/H \times 1/H$ and each subdomain contains $2 \times H/h \times H/h$ elements. We tested our methods for $N = 4 \times 4$, 8×8, 16×16 and various levels of H/h.

The parameter in the penalty term is chosen as $\eta = 2$ and $\eta = 10^6$. $\eta = 2$ is estimated as an optimal value considering the condition number of F_η in [6] and $\eta = 10^6$ is taken to examine the numerical stability of the enhanced penalty method. The preconditioner is used only for the $\eta = 10^6$ case.

The convergence of the CG method is investigated by the relative error with the stopping criterion $\|r_k\|/\|r_0\| < 10^{-8}$, where r_k is the residual error of the k-th iteration of the CG algorithm.

Table 4.2 shows the convergence behavior of FETI-DP method and enhanced penalty method with $\eta = 2$ and $\eta = 10^6$ for various levels of H and h. The relative error is measured with L^2 norm, i.e., $\frac{\|u-u_h\|_2}{\|u\|_2}$. We observe that the error is consistent for certain level of h in different subdomain numbers and that the enhanced penalty method converges stably for large η. Moreover, $O(h^2)$ convergence is verified.

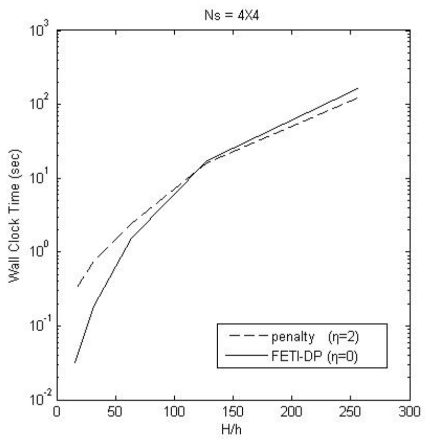
Table 4.3 compares the iteration and condition number between FETI-DP method and the enhanced penalty method. Numerical scalability is observed in both methods, since the condition number $\kappa(F)$ and $\kappa(F_\eta)$ and the iteration counts are maintained for certain level of H/h. Moreover, the strong scalability of the enhanced penalty method is shown, since the iteration number is bounded regardless of H and h, while in FETI-DP method, it increases with respect to H/h. The condition number estimation that $\kappa(F_\eta)$ is bounded by 3 is also verified. The strong scalability of the enhanced penalty method seems to be a superior property to FETI-DP method, since the outer iteration number is always kept constant. However, the enhanced penalty method has more expensive computations in the inner iteration than FETI-DP method. Also, it is clear that more communication time will be added to the overall wall clock time. Therefore, the enhanced penalty method must be implemented carefully in perspective of computation and communication.

Table 4.2: Convergence behavior of FETI-DP ($\eta = 0$) and the enhanced penalty method ($\eta = 2,\ 10^6$) for various levels of H and h.

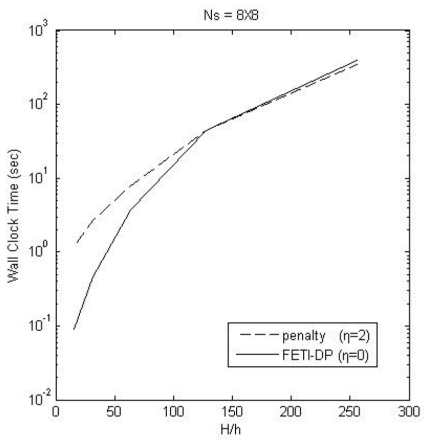
N	$\frac{H}{h}$	h	$\eta = 0$	$\eta = 2$		$\eta = 10^6$	
			error	error	ratio	error	ratio
4×4	16	1/64	2.0183e-4	2.0183e-4	-	2.0183e-4	-
	32	1/128	5.0466e-5	5.0468e-5	0.250	5.0466e-5	0.250
	64	1/256	1.2616e-5	1.2618e-5	0.250	1.2615e-5	0.250
	128	1/512	3.1529e-6	3.1544e-6	0.250	3.1531e-6	0.250
	256	1/1024	7.9021e-7	7.8856e-7	0.250	7.8702e-7	0.250
8×8	16	1/128	5.0462e-5	5.0465e-5	-	5.0463e-5	-
	32	1/256	1.2615e-5	1.2617e-5	0.250	1.2614e-5	0.250
	64	1/512	3.1607e-6	3.1535e-6	0.250	3.1485e-6	0.250
	128	1/1024	7.8727e-7	7.8959e-7	0.250	7.8825e-7	0.250
16×16	16	1/256	1.2608e-5	1.2618e-5	-	1.2611e-5	-
	32	1/512	3.1489e-6	3.1573e-6	0.250	3.1412e-6	0.249
	64	1/1024	8.1495e-7	7.9133e-7	0.250	7.7189e-7	0.246

Table 4.3: Comparison of condition number and iteration counts between FETI-DP ($\eta = 0$) and the enhanced penalty method ($\eta = 2,\ 10^6$).

N	$\frac{H}{h}$	$\eta = 0$		$\eta = 2$		$\eta = 10^6$	
		# iter.	$\kappa({F_D}^{-1}F)$	# iter.	$\kappa(F_\eta)$	# iter.	$\kappa(F_\eta)$
4×4	16	8	2.9528	5	1.1053	13	2.9243
	32	10	3.8135	5	1.1050	14	2.9771
	64	11	4.8437	5	1.1035	14	2.9733
	128	11	5.9794	5	1.1014	13	2.9707
8×8	16	12	3.2500	5	1.1002	12	2.9245
	32	14	4.2369	5	1.1024	14	2.9792
	64	16	5.3240	5	1.1016	14	2.9733
	128	17	6.6656	5	1.1004	15	2.9707



(a)



(b)

Figure 4.1: Graph of wall clock time versus H/h in (a) $N = 4 \times 4$, (b) $N = 8 \times 8$ subdomains.

4.2 Parallel Performance and Optimization

Finally, the wall clock time of the FETI-DP and enhanced penalty method in optimal $\eta = 2$, implemented on Gaia system, is presented in Table 4.4 and Figure 4.1. FETI-DP method is more efficient than the enhanced penalty method when the mesh size h is large or when the degree of freedom on each subdomain H/h is small. In spite of the less iteration number, the enhanced penalty method suffers from more computation per iteration and especially the communication time. As we can see in Table 4.5, data communication time dominates the wall clock time when the size of the subdomain problem is small. However, the enhanced penalty method becomes more efficient as the size of the subdomain problem increases. The communication time becomes relatively small and the method is rewarded from the fast convergence, i.e., small iteration number. Note that in our simulation, the enhanced penalty method becomes more favorable than FETI-DP method when the scalability constant H/h is greater than 128.

Here we note some remark on the procedure of optimizing the enhanced penalty method, which can be applied to any general parallel program.

Remark 4.2.1 Optimization of the serial code
Apparently, the parallel program must be first optimized in the sense of serial code. Optimizing the loop operation and memory access pattern is critical factors.

Particularly, matrix-vector product requires repeated and nested loops. These simple

Table 4.4: Parallel performance in wall clock time (second) of FETI-DP ($\eta = 0$) and the enhanced penalty method ($\eta = 2$).

N	$\frac{H}{h}$	h	$\eta = 0$		$\eta = 2$	
			# iter.	time(sec)	# iter.	time(sec)
4×4	16	1/64	8	0.0319	5	0.3016
	32	1/128	10	0.1856	5	0.7448
	64	1/256	11	1.5280	5	2.4389
	128	1/512	11	17.0487	5	16.1737
	256	1/1024	12	165.9410	5	123.1380
8×8	16	1/128	12	0.0918	5	1.1994
	32	1/256	14	0.4662	5	2.6973
	64	1/512	16	3.6670	5	7.9586
	128	1/1024	17	45.0788	5	44.5052
	256	1/2048	17	400.7010	5	349.5920

linear algebra calculations can be enhanced by using the linear algebra libraries, such as BLAS and BLACS. LAPACK can be used for more complicated calculations. In our test, routines that include vector addition and matrix-vector product are implemented by ESSL, IBM's linear algebra library based on BLAS supporting the PowerPC architecture under AIX and Linux.

We had prior test for the efficiency of the library. For example, the inner product was one of the most frequently used routines. The time consumed for calculating the inner product increased linearly in the usual code with loop operations. In the other hand, the time increased in log likelihood scale with a slope less than 0.65 in the optimized code using the library.

Table 4.6 shows the reduced wall clock time in percentage, after optimizing the linear algebra calculations with ESSL. The enhanced penalty method has reduced its wall clock time by $5 \sim 10\%$ for every level of H/h. However, the performance of FETI-DP method had almost no difference. We can observe that, to compensate the time required to access the library, the dimension of the problem and the frequency of library usage must exceed certain amount.

Remark 4.2.2 Load balancing and synchronization

Table 4.5: Comparing wall clock time (WCT(sec)) and communication time (sec) and its percentage of FETI-DP ($\eta = 0$) and enhanced penalty method ($\eta = 2$).

N	$\frac{H}{h}$	h	$\eta = 0$			$\eta = 2$		
			WCT	Communication		WCT	Communication	
4×4	32	1/128	0.201	0.008	3.98%	0.817	0.526	64.38%
	64	1/256	1.542	0.021	1.36%	2.574	0.931	36.17%
	128	1/512	17.518	0.096	0.55%	16.732	2.345	14.01%
	256	1/1024	165.608	1.239	0.75%	125.146	5.703	4.56%
8×8	32	1/256	0.509	0.033	6.48%	3.220	2.424	75.28%
	64	1/512	3.743	0.053	1.42%	8.379	3.904	46.59%
	128	1/1024	44.993	0.802	1.78%	47.752	9.209	19.29%
	256	1/2048	403.251	8.068	2.00%	358.331	28.618	7.99%

Synchronization means coordinating the processors to operate at the same time. For instance, the whole system must be synchronized when certain value calculated from all of the processors is needed to every processor. If the task is not equally distributed, which we call load balancing, some processors might be waiting without doing anything whenever the program needs to be synchronized. Therefore, in the parallel optimization, it is crucial to allocate the jobs fairly.

To avoid the waiting time, nonblocking communication can be used instead of blocking communication. A processor can either do calculation or communication at a single time. So, basically the blocking process stops the other calculation until the communication is properly finished. The processor waits after sending the data, until the success message is received from the counterpart. However, nonblocking communication resolves this by sending the data to the buffer, instead of directly to the receiving processor. This scheme has pros and cons. The sending processor can calculate other jobs after sending the data, but it must go back to the buffer afterwards to check if the data transfer has been completed properly. Therefore, frequently used nonblocking process can be a burden. While "MPI_ Send" and "MPI_ Recv" implements blocking communication, "MPI_ Isend" and "MPI_ Irecv" performs nonblocking communication. Furthermore, the IBM library PE 4.3 includes nonblocking communications for all-to-all subroutines, whereas the generally released MPI library does not.

In addition, even though we optimize the load balancing and communication, disparity

Table 4.6: Reduced wall clock time in percentage after optimizing the linear algebra routines in $N = 4 \times 4$ subdomains.

	H/h			
Method	16	32	64	128
FETI-DP	0.18%	0.06%	-0.12%	-0.55%
penalty($\eta = 2$)	-0.61%	-5.59%	-7.01%	-9.54%

Table 4.7: Profiling the communication time (comm.WCT) of enhanced penalty method ($\eta = 2$) : percentage of the two major communication, global inner product (MPI_Allreduce) and sending and receiving at the edge nodes (MPI_Send, MPI_Recv).

N	$\frac{H}{h}$	h	Comm.	MPI_Allreduce		MPI_Send&Recv	
			WCT	WCT	percent	WCT	percent
4×4	16	1/64	0.236	0.205	86.96%	0.031	13.04%
	32	1/128	0.488	0.423	86.72%	0.065	13.28%
	64	1/256	0.980	0.776	79.15%	0.204	20.85%
	128	1/512	2.598	1.656	63.73%	0.637	24.53%

occurs from the different performance of the physical computing resources. Therefore, the synchronizing process, especially the synchronization of the whole system, must be avoided as far as possible. One solution is to calculate the value on its own processor if it is possible. Or else, gathering the synchronization step together can reduce the frequent potential waiting time.

Alternatively, optimizing the serial code can be another solution to enhance the load balancing, since the difference between the processors will be reduced together with the computing time of each process.

Enhanced penalty method is nearly optimized in perspective of load balancing. Because the domain is divided into subdomains having the same size, the amount of calculation is almost identical. However, we should look into the two major communications mentioned in the section 3.2.2 carefully, since the physical circumstance can be always different. The profiling results are shown in Table 4.7.

The first communication to calculate the edge difference matrix J is one-to-one communication. This appears in each iteration of the CG algorithm solving F^{η}_{rr}. Instead of

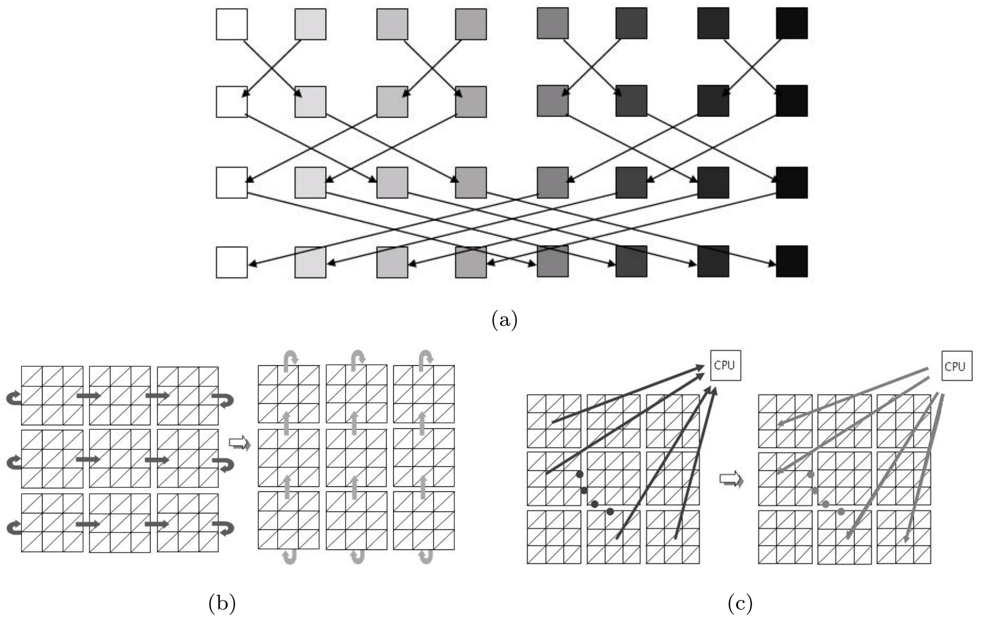


Figure 4.2: Algorithm for global inner product (a) Butterfly Algorithm, (b) Subdomain structure, and (c) Master Node.

using “MPI_ Send” and “MPI_ Recv” right before we calculate J, nonblocking subroutines “MPI_ Isend” and “MPI_ Irecv” can be commanded at the beginning of each iteration. While calculating F_{rr} and other terms, the uneven performance can be regulated by the unrestricted starting time of the communication. However, the performance was improved less than 1%. Moreover, the wall clock time rather increased when $H/h \leq 32$.

Small amount of improvement was expected since the communication cost coming from the “MPI_ Send” and “MPI_ Recv” routine accounts for little proportion out of the total wall clock time. However, the result indicates that nonblocking communication must be considered with more caution. Checking the success of nonblocking routine afterwards turns out to be a burden in some cases where load balancing is almost optimized.

The second communication calculating the global inner product occurs twice per iteration. It is implemented by “MPI_ Allreduce” routine to add the values from the whole processor group. “MPI_ Iallreduce” from PE 4.0 was applied, but the result was far worse. The necessity of checking right after the communication in each iteration has diminished the efficiency.

Remark 4.2.3 Communication routine - Butterfly algorithm

The profiling results in Table 4.7 show that all-to-all communication costs much more than one-to-one communication. The increment of cost depends on the number of processors. The communications in the MPI subroutines are optimized in its own way. However, we need to check if such communication procedure is proper to our network system [9].

To implement the global inner product, we need to add the values from all processors and their ordering can be implemented in various ways. "MPI_ Allreduce" arranges the ordering as Figure 4.2(a), what we call the butterfly algorithm. The cost is $\log(N)$ times the cost for "MPI_ send". However, if the structure of the network is not uniform, the optimal algorithm should be designed differently. Therefore, we tested several orderings implemented by other MPI routines. The first scheme is to imitate the subdomain structure and send the value to the neighbor subdomain as in the usual sending and receiving the edge routine. The second scheme is to send the value to a single master node, then broadcast the added result to the other nodes. See Figure 4.2(b) and 4.2(c). The simulation result showed that the other schemes lowered the efficiency. The wall clock time increased more than 30% to almost 100%. Therefore, we concluded that the butterfly algorithm is the optimized routine to implement all-to-all communications similar to "MPI_ Allreduce" in Gaia system.

Enhanced penalty method is optimized in perspective of the serial code, load balancing, synchronizing, and communication routine. After optimization, the enhanced penalty method performed better than FETI-DP method when the subdomain problem size $H/h \geq 128$. However, Table 4.5 shows that communicating time in the enhanced penalty method still consists of large portion among the wall clock time. This reveals that the performance of the enhanced penalty method has a potential to be improved along with the hardware resource.

5. Conclusions

This paper discussed the parallel implementation of the dual iterative subsructuring methods, particularly FETI-DP method and enhanced penalty method. We optimized the enhanced penalty method and compared its performance to FETI-DP method. The performance was analyzed in two perspectives, computation and communication, and the methods turn out to have their own strength in these two aspects, respectively.

With respect to the computation, the enhanced penalty method is superior, since the strengthened continuity constraint accelerates the convergence. Strong scalability of the enhanced penalty method results in fewer iteration number, i.e., less computation than FETI-DP method. But the enhanced penalty method compensates its fast convergence with added communication cost. Enhanced penalty algorithm contains two major communicating process and their cost is expensive. This offsets the saved wall clock time and makes FETI-DP method more effective if the subdomain problem size is small. However, as H/h becomes larger, the convergence of FETI-DP method diminishes and the enhanced penalty method reveals its advantage over FETI-DP method.

In conclusion, FETI-DP method having less communication is superior than the enhanced penalty method when H/h is small. On the other hand, the enhanced penalty method becomes more efficient when H/h is beyond a certain level, because the reduced computation compensates the additional communication cost. Considering the results, the enhanced penalty method can be useful when we are solving a large problem in a limited number of parallel processors, while FETI-DP is preferable when we have enough number of processor nodes to execute.

In case of our Poisson test problem in the Gaia system, the enhanced penalty method becomes more efficient than FETI-DP method when $H/h \geq 128$ for 4×4 and 8×8 subdomains. The decision between FETI-DP and the enhanced penalty method must be made carefully, considering the number of processors, computing power, and network performance of the parallel resource.

The performance of the enhanced penalty method can be improved with advanced hardware resources. Since the prevailing communication is calculating the sum over all processors, faster network and cache memory for these routines can make enhanced penalty method more promising.

After calculation of the global inner product is improved, the penalty term can be modified by sending and receiving less values without disturbing the convergence so large. For example, sending only the even nodes among the edge or only the average value can be considered.

Further more, FETI-DP method and enhanced penalty method in three dimension must be deliberately optimized in parallel circumstance. It is well known that FETI-DP method suffers more severely from diminished convergence in three dimensional case and the enhanced penalty method has more communication. Since the data transfers not only on the edge, but also on the surface, optimizing those routines will be an critical issue for the enhanced penalty method.

References

[1] C. Farhat and F. X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *Int. J. Numer. Methods Engrg.*, 32: 1205-1227, 1991.

[2] D. Braess. Finite Elements - Theory, Fast Solvers, and Applications in Elasticity Theory. *Cambridge University Press, Cambridge*, 2007.

[3] J. Mandel and R. Tezaur. Convergence of a substructuring method with Lagrange multipliers. *Numer. Math.*, 73: 473-487, 1996.

[4] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numer. Lin. Alg. Appl.*, 7: 687-714, 2000.

[5] J. Mandel and R. Tezaur. On the convergence of a dual-primal substructuring method. *Numer. Math.*, 88: 543-558, 2001.

[6] C.-O. Lee and E.-H. Park. A dual iterative substructuring method with a penalty term. *Numer. Math.*, 112: 89-113, 2009.

[7] A. Meyer. Basic approach to parallel finite element computations: the DD data splitting. *Parallel algorithms and cluster computing*, 52: 23-35, 2006.

[8] P. S. Pacheco. Parallel programming with MPI. *Morgan Kaufmann, San Francisco*, 1996.

[9] Y. Saad and M. H. Schultz. Data communication in parallel architectures. *Parallel Comput.*, 11: 131-150, 1989.

[10] *http://www.ksc.re.kr/new/04consumer/GAIA_Userguide _1st[1]3.pdf*