A Robust PDE Solver for the 3D Stokes/Navier-Stokes Systems on the Grid Environment *

Huidong Yang†
Walter Zulehner‡
Ulrich Langer §
Markus Baumgartner¶

November 28, 2007

Abstract

Since grid computing provides users with more distributed computing and storage resources, it gives us an opportunity to design new efficient and robust solvers for the numerical solutions of partial differential equations (PDEs). For instance, large scale problems in computational fluid dynamics (CFD) can be efficiently simulated under this environment. However, most of the currently developed PDE solvers using the finite element method (FEM), usually tight combinations of a mesh generator and a linear system solver, are not qualified for the grid computing environment. In this paper, based on principles of grid computing, we present a grid-enabled Client/Server model for solving the Stokes/(incompressible) Navier-Stokes system in 3D.

1 Introduction

As we might see today, under the grid environment, the large scale numerical simulations in computational fluid dynamics might be resolved in a more efficient way due to its huge computational and memory storage resources, cf. [FKT01, FJ03, FK03, CJ03, PW05, Mic05, Vla05, BS06]. Unfortunately, most of the traditional FEM software packages [HPF04, MW203, FEM86, DIF], usually a combined model of meshing and solving parts, do not take into account the strength of the grid computing. In general, they offer no user friendly interface for grid versions. This restriction leads to loss of efficiency and opportunity for solving really complicated problems by using these desktop-based versions. For instance, we have to develop a robust mesh generator to discretize the computational domain smoothly. This domain, given by CAD geometry

*This work was partially supported by the Austrian Science Fund (FWF) under the grant SFB F013 "Numerical and Symbolic Scientific Computing".

†Institute of Computational Mathematics, Johannes Kepler University Linz, huidong@numa.uni-linz.ac.at

‡Institute of Computational Mathematics, Johannes Kepler University Linz, zulehner@numa.uni-linz.ac.at

§Institute of Computational Mathematics, Johannes Kepler University Linz, ulanger@numa.uni-linz.ac.at

¶Institute of Graphics and Parallel Processing, Johannes Kepler University Linz, mb@gup.jku.at

data [mes], could be so involved that a large amount of mesh data has to be controlled and shared among grid nodes in a secure and efficient way [Mic05]. On the other hand, for numerical solutions of the stabilized discretization of PDEs, robust and advanced solvers are needed as well. The solvers themselves are not able to be directly applied to different situations. In the case of solving the Stokes/Navier-Stokes system, we prefer the robust algebraic multigrid method for the numerical solutions of PDEs [Bra95, Hac85, Kic98, Wab03, Wab06, ZS03, Zul02]. So it is reasonable to separate both meshing and solving parts into distinct grid nodes. Via this separating technique, the meshing and solving parts will mainly concentrate on their own favorite jobs without knowing the details from each other. Among these nodes, one needs to define a flexible and secure interface for data transfer. Concerning the nonlinear convection part of the Navier-Stokes equations, the communication between meshing and solving nodes is necessary in order to reassemble the linearized system after each outer nonlinear iteration. For the linear iteration part (the main cost of the solver), no extra communication time is needed. In the next sections, we will first present the solver designing ideas under the grid environment, data transferring interface on the memory level using Globus_IO secure channels [FBA03], and the grid-enabled Client/Server model. Considering its application to the Stokes/Navier-Stokes system by the streamline diffusion finite element method (SDFEM) [TV96], a very short introduction to this numerical method is summarized. Afterwards, we can see in detail how this Client/Server model works under the Austrian grid environment [BGV06]. Finally, based on this separating technique, several typical test results on different Austrian grid nodes are given.

2 Client/Server Model on the Grid

Because of its own properties distinguished from usual Client/Server models, in this section, a grid-enabled Client/Server model under the Austrian grid environment is discussed in detail. The technique of separating meshing and solving parts into distinct nodes fit well into the Client/Server model.

2.1 The Secure Grid Environment

One important point concerning the grid computing is how to realize secure data transfer among client and server nodes through Internet/Intranet. The Globus Toolkit 4.0.4 includes the open source software MyProxy 3.7 for managing security credentials (certificates and private keys) [MyP]. One highlight of this package is to combine an online credential repository with an online certificate authority which allow users to obtain credentials when and where needed. Under the Austrian grid environment, the user would use *myproxy-init* command to upload a credential to the myproxy-server *hydra.gup.uni-linz.ac.at* for later retrievals by Client/Server nodes. The credential is then delegated to the myproxy-server and stored with the given MyProxy passphrase. Proxy credentials with default lifetime of 12 hours can then be retrieved via *myproxy-get-delegation* with the MyProxy passphrase. Once the Client/Server nodes obtain the proxy credentials, the authentication and authorization on the Client/Server are done. The secure mode is checked via setting Globus_IO secure mode parameters as input arguments of these functions *globus_io_attr_set_secure_authentication/channel_mode*. See Fig. 1.

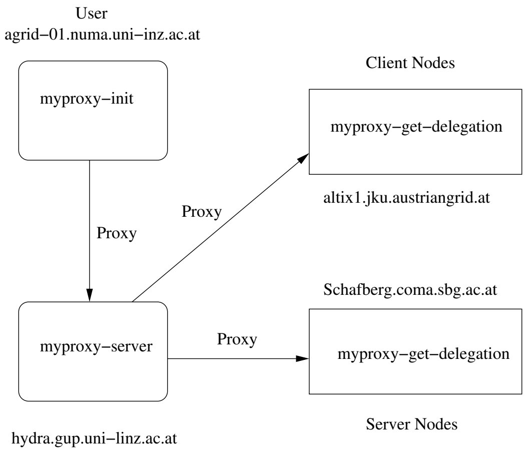

Figure 1: MyProxy process on the Austrian grid

2.2 The Separating Model

Using the previous authentication and authorization, a secure channel connecting Client and Server nodes through the Internet/Intranet is guaranteed. The meshing company (a client node) generates the system of equations arising from the mesh data it produces. Only the authorized solving company (a server node) has the right to access the database through the Globus_IO secure channel. It provides high-performance I/O with integrated security and a socket-like interface for users [FBA03]. Normal users holding no powerful machines can also succeed in doing such numerical simulations without knowing the details of meshing and solving parts on their own machines. As shown in Fig. 2, once their identities are certified by a Certification Authority and recognized by the requested resources, users can submit the job to nodes and control the data flow between nodes. For instance, *globus-url-copy* [Glo], can realize the data files transfer among nodes, and by RSL (resource specification language) [Glo], users can control the job running schedule on the nodes. Under the Austrian grid environment, using *glogin*, the identified user can realize the interactive usage of grid resources [RK04]. Via the Globus_IO secure and efficient channel mode, we can separate our mesh generator and linear system solver parts into different grid nodes. The communication between nodes is guaranteed in this channel created by calling *globus_io_tcp_connect* provided such TCP connecting is not opened and closed frequently.

2.3 The Grid-enabled Client/Server Model

As one can find in Fig. 3, an additional feature compared to standard Client/Server models on TCP/IP protocols is the authentication part on both client and server nodes by employing Globus_IO operations. The TCP connecting is mainly implemented via *globus_io_tcp_listen/connect/accept*. The server node firstly creates a TCP listener at a port and keeps listening at this port. Once it gets a notification from the client node, it will try to establish a TCP connection. If this connecting is successful, they will continue the next jobs. Otherwise, the server still listens at the opening port until it gets the next notification from the client. The server node does not start the solver until it gets the whole system given by the client node. The data transfer between nodes are

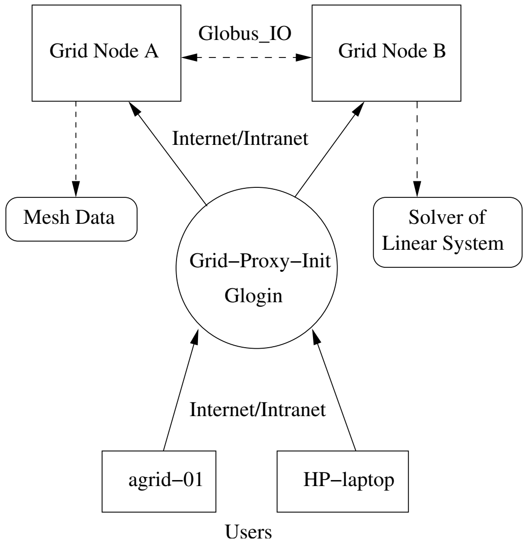

Figure 2: Separating model

hidden in the blocks of *Receive System* and *Send System*, which are implemented by offering a common data transferring interface such that both client and server nodes can apply it to their transferring processes. Using this friendly interface we defined on both nodes, the client node will send its linearized system (*Oseen equations*) [Wab03, Wab06], to the server node via calling the *globus_io_read/write* pair. Once the server node obtains the linearized system, it will start the solving procedure for the outer nonlinear iterations. Due to its nonlinear convection part $u \cdot \nabla u$ in the Navier-Stokes equations [Wab03, Wab06], after each outer nonlinear iteration, the server and client nodes have to do several communications for the process of reassembling the linearized system. When the solutions are sufficiently accurate, the server will stop calculations and send the solutions back to the client node. Finally, all I/O operations and TCP connections are closed.

2.3.1 Shared Data Transferring Interface

Because of the nonlinear part of the Navier-Stokes equations, the communication of sending and receiving the linear system between client and server nodes occurs after every outer nonlinear iteration. In this sense, a secure, stable and efficient data transferring interface has to be constructed on both client and server nodes. As we mentioned before, the Globus_IO offers such desirable operations. By calling the *globus_io_write/read* pair, one can realize the data sending and receiving through the established Globus_IO channel. Usually, hierarchical data structures of matrices and vectors are used for storing and solving the linear system. The matrix and vector themselves could also contain simple data structures. The size of these hierarchical data structures has to be measured carefully since the *globus_io_write/read* calling needs to know the exact block size of the message the nodes would like to send and receive. We designed small data structures for the entries of matrices and vectors such that the size of these elementary data structures can be evaluated by calling the *sizeof* function. There may appear different types of matrices with varying entries: *MC_Matrix1*

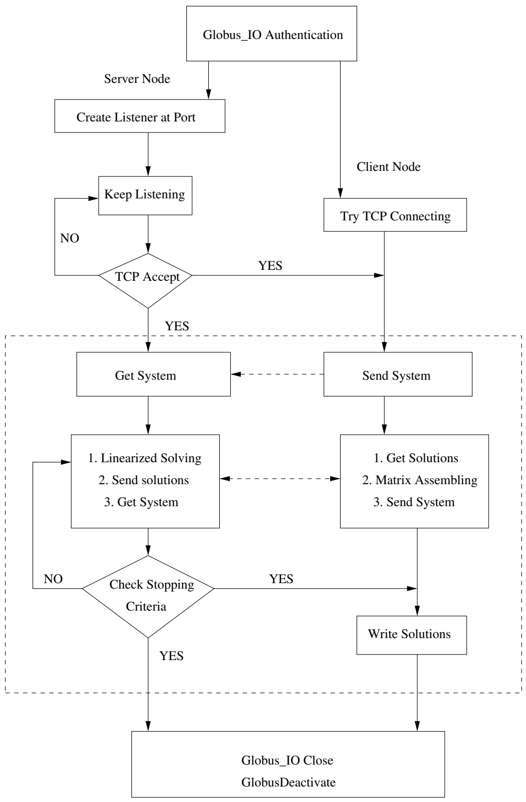

Figure 3: Grid-enabled Client/Server model

with small matrix (3×3) entries, *MC_Matrix2* with small vector (3×1) entries and *MC_Matrix3* with *double* entries. For vectors, there exist two types: *MC_Vec1* with small vector (3×1) entries and *MC_Vec2* with *double* entries respectively. Additionally, the matrix structure has to be decomposed into the vector storage. For instance, the compressed row structure (CRS) can be applied in order to fit into the vector transferring interface of the *globus_io_write/read* calling. In Fig. 4, we show a sample interface

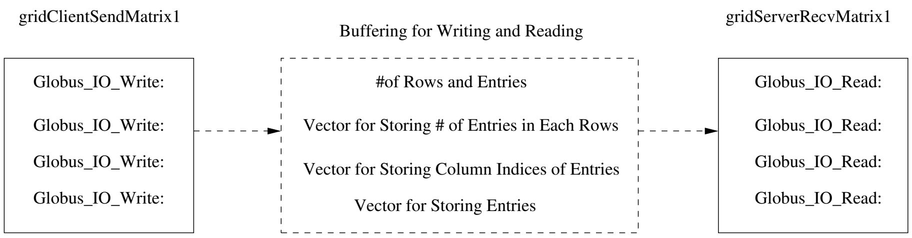

Figure 4: Sending and receiving interface on Client/Server nodes

of *gridClientSendMatrix1* and *gridServerRecvMatrix1* pair on client and server nodes. The *globus_io_write/read* callings, the matrix decomposition, and the matrix reconstructure are encapsulated inside. Users only feed in the *MC_Matrix1* type of matrices as the input parameters. For the interfaces of the other structures, we implemented it in the same way.

2.3.2 Server/Client Configuration Files

Using the high-performance and secure data transferring protocol, GridFTP [GRI], the executable file can be transferred and installed on the grid nodes. By the configuration files as in Fig. 5, one can specify the node roles: client for the mesh generator and server for the linear equations solver. We define the configuration file as an

GridClient.conf File:

```
GRIDTYPE  GClient
GRIDNODE altix1.jku.austriangrid.at
GRIDPORT 40102

MESHFILENAME cube_hex_l3.flma.gz
DATAFILENAME data.data

REYNOLDS 100
BOUNDARY_CONDITION ....
RHS_FORCE ...

SOLVER_TYPE

END
```

GridServer.conf File:

```
GRIDTYPE  GServer
GRIDPORT 40102
SOLVER_TYPE
END
```

Figure 5: Configuration files on Client/Server nodes

input parameter for starting the programming on both client and server nodes. For the configuration file on the server node, one has to specify the GRIDTYPE=GServer, GRIDPORT= $a \in [40000, 45000]$ (the Globus port range under the Austrian grid environment). One can specify the SOLVER_TYPE from different types of solvers, We will discuss them later on. In this part, no mesh data is involved. The solver is independent of mesh generation by such a separating technique. It offers a more convenient interface for users on the application level. The client node mainly handles mesh input data and generates linear system arising from it. In the case of the hybrid mesh [mes], it contains varying elements: hexahedra, prisms, pyramids and tetrahedra as in Fig. 6. This hybrid

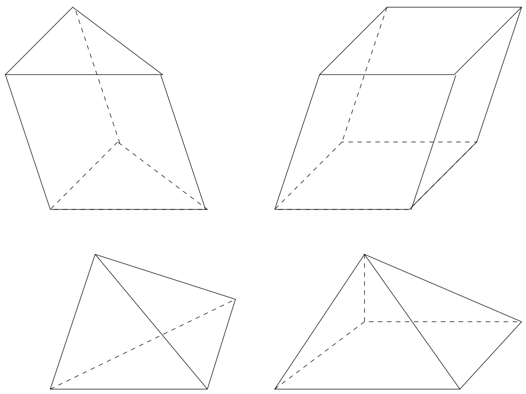

Figure 6: Elements generated by Spider

mesh requires more advanced and robust solvers to handle the linear system arising from the stabilized discretization on it. The client role is specified by setting GRIDTYPE=GCLIENT, the connecting server node GRIDNODE=*altix1.jku.austriangrid.at*, and the same port number as specified on the server node. Additionally, the client should also specify the solver type one wants to apply, but only the type name, not the detailed parameters which are jobs on the server node. This solver type specification will decide which matrix and vector will be sent to the server. The client only sends the minimal system information which fulfills the solver requirement.

2.3.3 Nonlinear Solver Interface on the Server Node

We are only interested in the stationary solutions of the Stokes/Navier-Stokes equations. The solver for these two problems is built into a unified framework. Users are able to handle these two problems in a convenient way by setting different *Reynolds* numbers ($Reynolds = 0$ will go to the Stokes problem). The solver type *NS_CoupledAMG* is the Navier-Stokes solver using the robust algebraic multigrid method (AMG) as a solver or as a preconditioner of the BiCGstab method [Wab03] (*stabilized bi-conjugate gradient method*) for solving the nonsymmetric and indefinite system. The *smoothingType* is used to distinguish solver types for the system because it is an important issue for the multigrid solvers [Wab03, ZS03, Hac85]. Once the smoothing type is selected by users, the server node will start to obtain the matrices and vectors from the client node. Afterwards, the linearized system is solved using the algebraic multigrid method. The *gridServerRecv* is a friendly data receiving interface as we discussed in the previous subsection. Once the stopping criteria (*signal*) is checked, the server will decide whether to solve the linearized system once more. See the whole process on the server node illustrated in Fig. 7. Via this *signal*, the client will determine whether it needs to reassemble the matrices and vectors for the server using the solutions it obtains. If not necessary, it only gets the solutions and write them to the local file.

2.3.4 Nonlinear Connecting Interface on the Client Node

On the client node, users only need to feed in matrices and vectors as illustrated in Fig. 8. The *gridClientSendTransfer* offers a friendly interface containing *globus_io_write* operations. This corresponds to the *gridServerSendTransfer* in server nodes. Via the *signal* it receives from the server node, the client will determine whether to reconstruct the system of matrices and vectors from the solutions *U,P* it obtains or just write the stationary solutions to the local file. As you might see here, no details of solver information are involved.

3 Numerical Approximations and Fast Solvers for the Stokes and Navier-Stokes Systems

In this section, we first present a short summary of the streamline diffusion finite element method (SDFEM) for solving the Stokes/ Navier-Stokes problems [TV96]. Then the algebraic multigrid method (AMG) for the resulting nonsymmetric and indefinite system will be discussed in a short way [Wab03, Wab06, ZS03, Zul02].

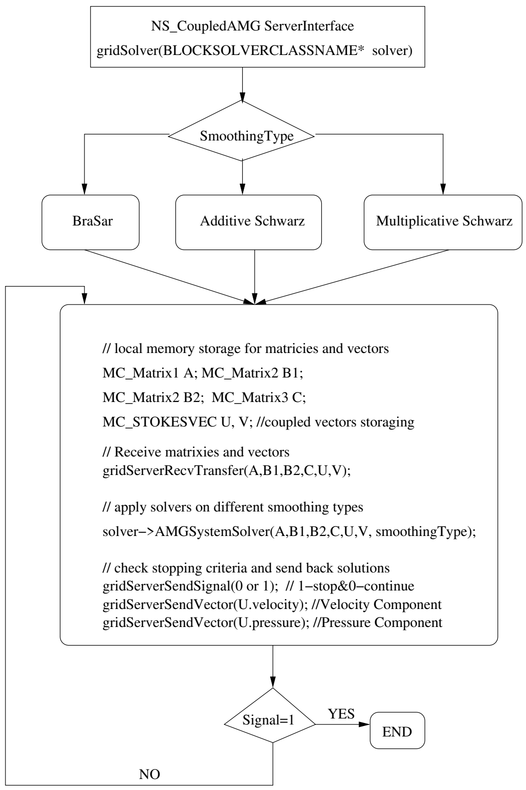

Figure 7: Solver interface on server nodes

3.1 The SDFEM for the Stokes and Navier-Stokes equations

We consider the Stokes/Navier-Stokes equations written in the velocity-pressure formulation for a bounded, connected and polyhedral domain $\Omega \subset \mathbf{R}^3$:

$$
\begin{aligned}
-\Delta u + \lambda u \cdot \nabla u + \nabla p &= f && \text{in } \Omega \\
\nabla \cdot u &= 0 && \text{in } \Omega \\
\mathcal{B}(u, p) &= g && \text{on } \Gamma
\end{aligned}
$$

where λ is the scaled Reynolds number, p the pressure, u the velocity field, $\mathcal{B}$ some types of boundary operators and Γ the boundary of domain Ω. The above system of equations describes the balance law of momentum and conservation law of mass in fluid mechanics. The discretization of this system by the SDFEM method leads to the following discrete variational problem: Find $[u_h, p_h] \in V_h \times Q_h$ such that, for all $[v_h, q_h] \in V_h \times Q_h$,

$$
\begin{aligned}
&(\nabla u_h, \nabla v_h) + \lambda(u_h \cdot \nabla u_h, v_h) - (p_h, \nabla \cdot v_h) \\
&+ \delta \sum_{T \in \mathcal{T}_h} h_T^2 (\lambda u_h \cdot \nabla u_h + \nabla p_h, \lambda u_h \cdot v_h)_T \\
&= (f, v_h) + \delta \sum_{T \in \mathcal{T}_h} h_T^2 (f, \lambda u_h \cdot \nabla v_h)_T
\end{aligned}
$$

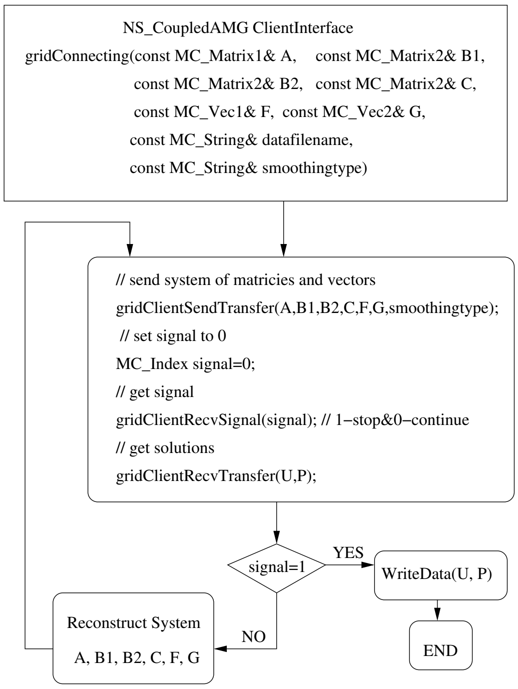

Figure 8: Connecting interface on client nodes

and

$$
\begin{aligned}
&(q_h, \nabla \cdot u_h) + \delta \sum_{T \in \mathcal{T}_h} h_T^2 (\lambda u_h \cdot \nabla u_h + \nabla p_h, \nabla q_h)_T \\
&= \delta \sum_{T \in \mathcal{T}_h} h_T^2 (f, \nabla q_h)_T,
\end{aligned}
$$

where V_h, Q_h are some finite element spaces, and δ is a small parameter.

This problem admits at least one solution $[u_h, p_h]$ under proper assumptions, see [TV96]. For sufficiently small λ (scaled *Reynolds number*), the solution of the problem is unique, see [TV96]. We choose the finite element spaces $V_h = \{v \in C_0(\bar{\Omega})^3 : v|_T \in P_1, \text{ for all } T \in \mathcal{T}_h\}$ and $Q_h = \{q \in C(\bar{\Omega}) \cap L_0^2(\Omega) : q|_T \in P_1 \text{ for all } T \in \mathcal{T}_h\}$ and appropriate bases $\{\phi_j\}$ for V_h and $\{\psi_j\}$ for Q_h. Then the approximate solutions can be represented in the following way:

$$
u_h = \sum_j u_j \phi_j, \quad p_h = \sum_k p_k \psi_k
$$

leading to the nonlinear system of the form

$$
\begin{pmatrix} A(u) & B_1^T \\ B_2 & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}
$$

for the unknown vectors $u = (u_j)$ and $p = (p_k)$. This nonlinear system was solved iteratively by replacing u in the convective part by the previous iterate. So, in each step of this inner iteration, a linear problem of the form

$$\begin{pmatrix} A(w) & B_1^T \\ B_2 & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

(a so-called *Oseen problem*) has to be solved. For the Stokes problem ($\lambda = 0$), we have $B_1 = B_2$ and A a constant matrix.

3.2 Standard Iterative Solvers for the Stokes System

For the Stokes problem ($\lambda = 0$), there exist some iterative solvers capable of solving this saddle point system [Wab03].

3.2.1 Schur(P)CG - (Preconditioned) conjugate gradient method applied to the Schur Complement

By eliminating the velocity variables, we get a reduced system $Sp = BA^{-1}f - g$ for the pressure only, where $S = BA^{-1}B^T + C$, the (negative) Schur complement, is positive semi-definite and has a one-dimensional kernel. One can apply the standard (P)CG method, see [Hac94], to it by choosing the initial guess properly. The velocity u is calculated by applying the algebraic multigrid method (AMG) to the linear equation $Au = f - B^T p$. In each (P)CG iteration, the multi-grid method has to be applied once since A^{-1} appears in the Schur complement S and has to be evaluated accurately enough. This inner iteration is the main cost of this solver.

3.2.2 ZulehnerInexactUzawa - inexact Uzawa method applied to the whole system

It is a preconditioned technique applied to the whole system

$$\hat{\mathcal{L}}^{-1}\mathcal{K}\begin{pmatrix} u \\ p \end{pmatrix} = \hat{\mathcal{L}}^{-1}\begin{pmatrix} f \\ g \end{pmatrix}, \quad \text{with} \quad \hat{\mathcal{L}} = \begin{pmatrix} \hat{A} & 0 \\ B & -\hat{S} \end{pmatrix}.$$

One big advantage of this method is, that it does not require the exact inversion of A but only of some preconditioner $\hat{A}$. It leads to an efficient improvement of the performance. For more details, see [Zul02].

3.3 Algebraic Multigrid Solvers for the Stokes/Navier-Stokes Systems

To solve the original saddle point problem (nonsymmetric and indefinite), one can apply the algebraic multigrid method (AMG) to the whole system, see [Wab03]. One crucial component of a multigrid method is the choice of the smoother. We implement two different smoothers for the coupled system: the *BraessSarazin* smoother, see [BR97], and the *Vanka* smoother, see [Van86]. The first smoother has a smoothing property with a rate of $\mathcal{O}(\frac{1}{m})$, where m is the number of smoothing steps, see [BR97], [Zul00]. The second smoother is obtained by setting up small subproblems cell by cell (patch by patch), i.e. with one degree of freedom for the pressure and a few degrees of freedom for the connected velocity unknowns. The solutions of these subproblems are combined by the multiplicative Schwarz iteration. The smoother for the additive Schwarz case has a smoothing property with a rate of $\mathcal{O}(\frac{1}{\sqrt{m}})$, see [ZS03]. One can use the multigrid method as a solver for the coupled system. If using it as a preconditioner for the system, one can apply the preconditioned stabilized bi-conjugate gradient method (Pre-BiCGstab) to the coupled system, which will lead to better convergence performance. One can see this advantage later on.

4 Client/Server Model Applied To the Stokes/Navier-Stokes System

In this part, we will see how the grid-enabled Client/Server model can be applied to the Stokes/Navier-Stokes system. We are not going to compare the efficiency of different solvers we have discussed in the previous section, but will only concentrate on how this model works under the Austrian grid environment.

4.1 Apply the Model to the Stokes System on the Austrian Grid With Standard Iterative Methods

In order to measure the communication and computing time for each linearized iteration, we first test the problem on the uniformly refined meshes for the Stokes system. The numerical solvers are mesh-independent, i.e. the number of iterations does not increase when the mesh is refined. We present four levels $L_1 - L_4$ of uniformly refined meshes as shown in TABLE 1. These mesh files are only stored on the client nodes and will be used to generate the Stokes system locally. We run the solvers on the server

Table 1: Four Uniformly Refined Levels

Levels	L_1	L_2	L_3	L_4
#Vertices	125(5^3)	729(9^3)	4,913(17^3)	35,937(33^3)
#Unknowns	500	2,916	19,652	143,748
Mesh Size(h)	$1/4$	$1/8$	$1/16$	$1/32$

node by three standard iterative methods. For each of these methods, the iteration numbers should stay on the same level. See the test results on the uniformly refined meshes in TABLE 1. Next, we are going to show some test results under the Austrian grid

Table 2: Mesh-independent Iterations

Methods on Four Levels	L_1	L_2	L_3	L_4	**#It(average)**
#It(CG)	26	41	51	49	42
#It(PCG)	13	20	24	25	21
#It(Uzawa)	18	21	21	19	20

environment. All of these examples are tested on mesh level L_3. Time is measured in seconds. The linear system arising from this level is not so huge since it is only used to show how the Client/Server model works. We specify both client and server nodes, let them do their jobs locally, and see how both of them can cooperate through shared data transferring interface which we defined before. TABLE 3 contains a list of nodes connected to the Austrian Grid [BGV06]. The node Agrid-01 is a powerful desktop located at the Institute of Computational Mathematics, JKU. The node ALTIX1 in Linz is a four 16-way SGI Altix 350 system which are configured as a cluster

Table 3: Grid Nodes Information

Node	Site	Proc Type	Arch.	RAM
Agrid-01	Linz	AMD Opteron (2)	AMD64	4 GB
SCHAFBERG	Salzburg	Intel Itanium 2 (16)	IA64	16 GB
ALTIX1	Linz	Intel Itanium 2 (64)	IA64	64 GB

and interconnected by an Infiniband fabric. The communication in this node should be really fast. The node at Salzburg is an one 16-way SGI Altix1 350 system. In our first trivial test, both client and server are on the local machine Agrid-01. Since no extra time spent on the Internet for data transfer, the system sending and receiving should run very fast. On the client node, one can see the time spent in initializing and sending the Stokes system. On the server node, we record the time for receiving the system and the iterative numbers for these three iterative solvers. The time spent on system transfer

Table 4: Client/Server Model on Agrid-01

Client Node Agrid-01		**Server Node Agrid-01**			
System	**Send**	**Recv**	**Solve**		
2.5s	U(0.8s)	U(0.8s)	**SchurCG**	**SchurPCG**	**Uzawa**
	C(0.2s)	C(0.2s)	119.3s	61.9s	8.4s
	P(0.8s)	P(0.8s)	#It(47)	#It(24)	#It(21)

for SchurCG method is $0.2s$ (C(0.2s)). For the other two, additional time is needed for sending preconditioners, $0.8s$ for both Uzawa (U(0.8s)) and SchurPCG (P(0.8s)) methods. On the server node, one can see most of the time is spent in solving the system. For each of these test cases, the cost for generating the Stokes system is more or less the same. As expected, the inexact Uzawa method has better performance than the other two by comparing the cost in TABLE 4. SchurPCG can speed up the performance by half compared to SchurCG method. In the next test in TABLE 5, we utilize ALTIX1 as a client node which deals with mesh reading, the Stokes system generation and system sending to the server node SCHAFBERG. Since they are two separated grid nodes, more time is needed for data transfer. We start from *grid-proxy-init* on the

Table 5: Client/Server Model on Altix1 and Schafberg

Client Node ALTIX1		**Server Node SCHAFBERG**			
System	**Send**	**Recv**	**Solve**		
5.8s	U(5.2s)	U(5.2s)	**SchurCG**	**SchurPCG**	**Uzawa**
	C(3.6s)	C(3.6s)	246.1s	132.4s	21.5s
	P(4.8s)	P(4.8s)	#It(47)	#It(24)	#It(21)

local machine, use *glogin* to log into altix machines on these nodes, and transfer binary files and Client/Server configuration files to them by *globus-url-copy*. Afterwards, start the server on the node SCHAFBERG, and then run the client on ALTIX1. We should mention here, that the mesh file is only transferred to the client node. For solvers on the server node, we only transfer the executable binary file and the server configuration file which are sufficient for solving the linear system. Additionally, users who are not able to afford the cost for powerful machines would benefit a lot from this model. As we see here, the mesh generation and solving the linear system are both done on the nodes. The only requirement for users is to join this grid environment, set the Server/Client configuration files, run the jobs on the nodes, and afterwards check the results from computing. This is also a big advantage compared to usual PDE solvers. For testing, the other possibility is to run the client on a HP laptop with a AMD Athlon mobile processor and 512 MB memory in Linz. The server is the grid node of SCHAFBERG. See TABLE 6. Usually, a laptop has less resources compared with grid nodes, which

Table 6: Client/Server Model on HP-Laptop and Schafberg

Client Node HP-Laptop		**Server Node SCHAFBERG**			
System	**Send**	**Recv**	**Solve**		
22.7s	U(22.4s)	U(22.4s)	**SchurCG**	**SchurPCG**	**Uzawa**
	C(16.9s)	C(16.9s)	267.1s	127.1s	21.6s
	P(22.1s)	P(22.1s)	#It(47)	#It(24)	#It(21)

leads to the dramatic increase in costs for the Stokes system setup. This tells us for really complicated mesh generation, it would be a good idea to set one powerful node as a client where we can utilize its resources. However, the solving time is comparable in these cases since they are always running on the same server node. One can try to run the solver on more grid nodes under the grid computing environment. As a test version of the grid-enabled solver for the Stokes system using the separating technique, it works well under the Austrian grid environment.

4.2 Apply the Model to the Stokes/Navier-Stokes systems on the Austrian Grid with the Multigrid Methods

In this subsection, we present the multigrid solvers for the Stokes/Navier-Stokes problems. Since the communication time is shown in previous samples, we are not going to list the communication and computing time in this subsection. One can get the approximate cost of the methods by multiplying the cost in the previous subsection with the number of nonlinear iterations. For each outer nonlinear iteration, a linear system is solved. We give the test results using the multigrid method as a solver for the linearized *Oseen equations*. See TABLE. 7. If we use the BiCGstab Krylov space method with

Table 7: Nonlinear Iterations of Multigrid Solvers

Reynolds Number(λ)	**Number of Nonlinear Iterations (Number of Linear Iterations)**		
	Braess-Sarazin	**Additive Schwarz**	**Multiplicative Schwarz**
0.0	1 (20)	1 (21)	1 (12)
1.0	4 (20)	4 (22)	4 (12)
50.0	12 (19)	12 (21)	12 (11)
100.0	21 (19)	21 (20)	21 (10)

the multigrid method as a preconditioner for the coupled system, then the number of linear iterations will decrease efficiently. See TABLE. 8. For the case of the scaled

Reynolds number $\lambda = 0.0$, we are solving the Stokes problem, and only one nonlinear iteration is needed. When increasing the value of the *Reynolds number*, we enlarge the nonlinear convection part which leads to the increase of nonlinear iterations for the coupled system. However, we are here not going into details on how to adjust these parameters for the high *Reynolds number*. The number of linear iterations for each

Table 8: Nonlinear Iterations of BiCGstab

Reynolds Number(λ)	**Number of Nonlinear Iterations (Number of Linear Iterations)**		
	Braess-Sarazin	**Additive Schwarz**	**Multiplicative Schwarz**
0.0	1 (7)	1 (8)	1 (5)
1.0	4 (7)	4 (8)	4 (5)
50.0	12 (6)	12 (10)	12 (5)
100.0	21 (7)	21 (8)	21 (4)

nonlinear iteration is illustrated by the number in the bracket.

5 conclusions

Based on techniques of grid computing and reconsiderations of PDE solvers using the finite element method for the Stokes/Navier-Stokes systems, we discussed the separating technique which is suitable for the Client/Server model under the grid environment. We developed a friendly user interface for data transfer. The robust multigrid PDE solver for the application in the computational fluid dynamics has been developed. Most of numerical experiments are implemented and tested on different grid nodes.

Acknowledgment

The authors would like to thank Ferdinand Kickinger for providing them with his mesh files, the foundation work of this solver, and very valuable discussions concerning programming techniques. Last, but not least, the authors would like to thank the project of Austrian Grid (http://www.austriangrid.at) funded by the bm:bwk (Austrian Federal Ministry for Education, Science and Culture).

References

[BGV06] M. Baumgartner, C. Glasner, and J. Volkert. An Overview of the Austrian Grid Infrastructure. In J. Volkert, T. Fahringer, D. Kranzlmüller, and W. Schreiner, editors, *Proceedings of* 1^{st} *Austrian Grid Symposium*, pages 277–286, 2006.

[BR97] D. Braess and R.Sarazin. An efficient smoother for the Stokes problem. *Numer. Math.*, 23:3–20, 1997.

[Bra95] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[BS06] G. Brajesh and L. Shilpa. *Grid Revolution: An Introduction to Enterprise Grid Computing*. McGraw-Hill/Osborne Media, 2006.

[CJ03] F. Craig and J. Joshy. *Grid Computing*. IBM Press, 2003.

[DIF] The Diffpack webside. http://www.diffpack.com/.

[FBA03] L. Ferreira, V. Berstis, and J. Armstrong. *Introduction to Grid Computing with Globus*. IBM Corp, 2003.

[FEM86] The FEMLAB webside. http://www.comsol.com/, 1986.

[FJ03] C. Fellenstein and J. Joseph. *Grid Computing*. Prentice Hall Ptr, 2003.

[FK03] I. Foster and C. Kesselman. *Grid 2 Blueprint for a New Computing*, volume 13 of *Elsevier Series in Grid Computing*. Morgan Kaufmann Pub, 2003.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *High Performance Computing Applications*, 15(3):200–222, 2001.

[Glo] The Globus Toolkit webside. http://www.globus.org/toolkit/.

[GRI] The OpenGridForum webside. http://www.ogf.org/.

[Hac85] W. Hackbusch. *Multigrid Methods and Application*. Springer Verlag, Berlin, Heidelberg, New York, 1985.

[Hac94] W. Hackbusch. *Iterative Solutions of Large Linear Systems of Equations*. Springer, New York, 1994.

[HPF04] The FEMworks website. http://www.hpfem.jku.at/, 2004.

[Kic98] F. Kickinger. Algebraic multigrid for discrete elliptic second-order problems. In W. Hackbusch, editor, *Multigrid Methods V. Proceedings of the 5th European Multigrid conference*, volume 3 of *Springer Lecture Notes in Computational Science and Engineering*, pages 157–172, 1998.

[mes] The Spider Online webside. http://www.meshing.org/.

[Mic05] D.S. Michael. *Distributed Data Management for Grid Computing*. Wiley-Interscience, 2005.

[MW203] The AMuSE website. http://www.numa.uni-linz.ac.at/, 2003.

[MyP] The MyProxy webside. http://grid.ncsa.uiuc.edu/myproxy/.

[PW05] P. Plaszczak and R. Wellner. *Grid Computing*. Morgan Kaufmann Pub, 2005.

[RK04] H. Rosmanith and D. Kranzlmüller. glogin-A Multifunctional, Interactive Tunnel into the Grid. In *Proceedings of Grid 2004, 5th IEEE/ACM Intl.Workshop on Grid Computing*, pages 266–272, Pittsburgh, PA, USA, 2004. IEEE Computer Society.

[TV96] L. Tobiska and R. Verfürth. Analysis of a streamline diffusion finite element method for the Stokes and Navier-Stokes equations. *Numerical Analysis*, 33(1):107–127, 1996.

[Van86] S. Vanka. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Comp. Mech. Apply. Eng*, 59(1):29–48, 1986.

[Vla05] S. Vladimir. *Grid Computing for Developers*. Charles River Media, 2005.

[Wab03] M. Wabro. *Algebraic Multigrid Methods for the Numerical Solution of the Incompressible Navier-Stokes Equations*. PhD thesis, Johannes Kepler University Linz, 2003.

[Wab06] M. Wabro. Amge - coarsening strategies and application to the oseen equations. *SIAM Journal for Scientific Computing*, 27:2077–2097, 2006.

[ZS03] W. Zulehner and J. Schöberl. On Schwarz-type smoothers for saddle point problems. *Numerische Mathematik*, 95(2):377–399, 2003.

[Zul00] W. Zulehner. A class of smoothers for saddle point problems. *Computing*, 65(3):227–246, 2000.

[Zul02] W. Zulehner. Analysis of iterative methods for saddle point problems: a unified approach. *Mathematics of Computation*, 71(238):479–505, 2002.