Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 54(4), 471–482(2013)

Dandan Li, Xiaohui Ji, Qun Wang

An Efficient Parallel Computing Method for the Processing of Large Sensed Data

DOI 10.7305/automatika.54-4.450
UDK 004.89:004.4'242
IFAC 2.8.4; 1.1.9

Original scientific paper

In recent years we witness the advent of the Internet of Things and the wide deployment of sensors in many applications for collecting and aggregating data. Efficient techniques are required to analyze these massive data for supporting intelligent decisions making. Partial differential problems which involve large data are the most common in the engineering and scientific research. For simulations of large-scale three-dimensional partial differential equations, the intensive computation ability and large amounts of memory requirements for modeling are the main research problems. To address the two challenges, this paper provided an effective parallel method for partial differential equations. The proposed approach combines the overlapping domain decomposition strategy and the multi-core cluster technology to achieve parallel simulations of partial differential equations, uses the finite difference method to discretize equations and adopts the hybrid MPI/OpenMP programming model to exploit two-level parallelism on a multi-core cluster. The three-dimensional groundwater flow model with the parallel finite difference overlapping domain decomposition strategy was successfully set up and carried out by the parallel MPI/OpenMP implementation on a multi-core cluster with two nodes. The experimental results show that the proposed parallel approach can efficiently simulate partial differential problems with large amounts of data.

Key words: Data processing, Domain decomposition method, Hybrid programming model, Multi-core clusters, Finite difference method

Učinkovita metoda paralelnog računanja za obradu velike količine podataka prikupljanih senzorom. Posljednjih godina svjedočimo dolasku tzv. interneta stvari i širokoj uporabi senzora u raznim primjenama prikupljanja i objedinjavanja podataka. Učinkovite metode su potrebne za analizu velike količine podataka u svrhu podrške inteligentnom odlučivanju. Parcijalno diferencijalni problemi koji uključuju veliku količinu podataka su gotovo uobičajeni u inženjerstvu i znanstvenom istraživanju. Za simulaciju masovnih trodimenzionalnih parcijalnih diferencijalnih jednadžbi potrebne su značajne računalne mogućnosti, a potreba za velikom količinom memorije za modeliranje je glavni istraživački problem. Za rješavanje oba problema ovaj rad pruža efektivnu paralelnu metodu za parcijalne diferencijalne jednadžbe. Predloženi pristup kombinira strategiju dekompozicije preklapajućih domena i tehnologiju višejezgrenih klastera za postizanje paralelnih simulacija parcijalnih diferencijalnih jednadžbi, koristi metodu konačnog diferenciranja za diskretizaciju jednadžbi te model *MPI/OpenMP* hibridnog programiranja za iskorištavanje dvorazinskog paralelizma na višejezgrenom klasteru. Formiran je trodimenzionalan model toka podzemnih voda sa strategijom dekompozicije preklapajućih domena konačnih diferencija. Za izvođenje je korištena paralelna *MPI/OpenMP* implementacija na višejezgrenom klasteru s dva čvora. Eksperimentalni rezultati su pokazali kako predloženi paralelni pristup može učinkovito simulirati parcijalno diferencijalne probleme s velikom količinom podataka.

Ključne riječi: obrada podataka, metoda dekompozicije domene, model hibridnog programiranja, višejezgreni klasteri, metoda konačnog diferenciranja

1 INTRODUCTION

The Internet of Things (IoT) has envisioned the interaction and seamless integration of smart things. In this context, smart things communicate with each other, collect and aggregate data in network for satisfying certain requirements of end-users [1]. In recent years, IoT technologies have been widely used in many applications such as environmental quality and protection, natural resource management, groundwater management, farmland and greenhouse [2, 3], etc. Among IoT-related technologies, sensor is one of the most mature and practical technologies. Sensor generates vast amounts of data that need to be stored and managed. Data processing is unanimously recognized as one of the core functionalities of the IoT [3]. Things require powerful computation ability and storage resources to handle large-scale applications. In this kind of applications, efficient parallel techniques for the processing of sensed data are critical. In this paper we aim to propose an efficient computation method which is used for improving the computation efficiency of the intensive computation problems.

Generally, many practical problems which involve massive data processing and storage in scientific research as well as engineering can be described as partial differential equations. For example, the three-dimensional groundwater flow problem can be described as a partial differential equation. Groundwater flow simulations have important significance for groundwater flow pollution control and rational use of water resources in the future. When managing large-scale groundwater flow equations with numerical discretization methods, the intensive computation ability and large amounts of memory requirements for modeling are the main bottlenecks for researchers [4]. In order to improve computation ability and reduce memory requirements, parallel computing of partial differential equations has become an important subject for studying. The domain decomposition method (DDM) has been proved to be one efficient parallel algorithm and has a wide range of applications in parallel computing platforms [5–7]. The basic principle of DDM is that the solution domain can be divided into several sub-domains. So solving the original problem can be transformed into solving the sub-problems and each sub-problem can be solved independently [8]. For solutions of partial differential equations on large-scale with complex 3D geometries, the domain decomposition techniques are natural approaches to split the problem into sub-problems and sub-problems are allocated to different processors. The DDM enables one to reduce both computation costs and the memory requirements for storing large amounts of data [9, 10].

For intensive computation tasks which involve large amounts of data, parallel computers provide powerful computation ability. Due to supercomputers' expensive price and large power consumption, it is helpful for contributing the development of multi-core clusters. Since the release of the first generation dual-core processor by IBM in 2001, Sun in 2004, and AMD in 2005, more and more cores are built into a single processor with the development of multi-core technology [11]. Multi-core architectures are today's dominant computing platforms [12, 13]. Parallel architectures and programming models are not independent. Understanding the characteristics of parallel architectures is quite important to select the most effective parallel programming model. Multi-core clusters can be thought as the hierarchical two-level parallel architectures, since they combine features of shared and distributed memory [14]. Intuitively, a parallel paradigm that uses memory access for intra-node communication and message passing for inter-node communication seems to exploit better the characteristics of a multi-core cluster. Generally, MPI is considered optimal for process-level parallelism and shared memory parallel programming paradigm OpenMP is considered optimal for loop-level parallelism [15]. Combining MPI and OpenMP parallelization to construct a hybrid program can achieve two-level parallelism and reduce the communication overhead of MPI at the expense of introducing OpenMP overhead due to threads creation [16, 17].

Domain decomposition strategy and multi-core clusters technology promise power for high performance computing. As a parallel algorithm, DDM is simple to implement parallel simulations. Because of its efficiency and flexibility, there is much research [18–23] on DDM for numerical simulations of partial differential equations. However, the majority of research is based on non-overlapping DDM rather than overlapping DDM. Additionally, these studies adopt the numerical discretization method finite element method (FEM) which involves a large amount of computation compared with finite difference method (FDM). The paper focuses on studying the overlapping DDM because the overlapping DDM has a better convergence than the non-overlapping DDM. Each sub-domain uses the FDM as the numerical discretization method. Since multi-core clusters are widely used in the high performance computing and can be thought as the hierarchical two-level parallel architectures, many researchers are dedicated to study the hybrid MPI/OpenMP programming model which can exploit the two-level parallelism for architectures of multi-core clusters. Papers [24–28] introduce MPI/OpenMP programming model in all aspects of applications, but they do not adopt the MPI/OpenMP programming model to implement the overlapping DDM for partial differential equations.

The objective of the present work is to develop a parallel approach for efficiently dealing with the processing of large sensed data. This paper combines the overlapping DDM and the multi-core cluster parallel computing technology. It uses the FDM to approximate partial differential equations and adopts the hybrid MPI/OpenMP programming model on a multi-core cluster. The principles of the proposed parallel approach can be described as follows. Firstly, the overlapping DDM is used to split the physical domain into several overlapping sub-domains with equal size. Secondly, the FDM is adopted to establish mathematical model on each sub-domain. Thirdly, the hybrid MPI/OpenMP program is proposed for parallel implementation on a multi-core cluster. Based on the approach, this paper carried out experiments taking the three-dimensional groundwater flow equation as an example on a multi-core cluster. Numerical experiment results yield the speedup of about 3.3 when the original problem is divided into four sub-problems. That means the proposed approach is effective for parallel simulations of large-scale partial differential equations.

The paper is organized as follows. Section 2 introduces the related work on IoT applications, domain decomposition method and the hybrid MPI/OpenMP programming model. Section 3 introduces methods of data processing. This section presents the division methods of overlapping DDM and the basic features of the FDM. Section 4 briefly introduces the architecture of multi-core cluster; the most two important programming models and the hybrid MPI/OpenMP programming model for multi-core clusters. Section 5 describes in detail the efficient implementation of data processing methods. This section proposes a parallel FDM overlapping domain decomposition strategy for partial differential equations and adopts the hybrid MPI/OpenMP programming model to implement the parallel algorithm on a multi-core cluster. Section 6 starts with a description of groundwater flow model, describes the parallel platform used in the study, carries out numerical experiments and discusses the experimental results. Section 7 gives the conclusions of this paper.

2 RELATED WORK

IoT is an unprecedented technological revolution which intends to get all the real world items connected to the network. IoT is likely to have a staggering impact on our daily lives and becomes an inherent part of areas such as environmental management, industrial control and water resource management [2, 3]. Sensor is one of the most mature and practical technologies among IoT-related technologies. It can monitor and transfer various data to the management system in real time. After standardizing and storing, these data can be used to compute and provide information for researchers. Since massive data generated from sensor technology need to be stored and managed, data processing is unanimously recognized as one of the core functionalities of the IoT.

Things require powerful computation ability and storage resources to handle large-scale applications. The DDM has been proved to be one effective method for providing powerful computation ability and reducing storage requirements. Because the computations in the sub-domains are ideally suited for parallel computing, the DDM has attracted more attentions in recent years. Studies have shown that it provides powerful parallel strategy for partial differential equations. Paper [18] presented a non-overlapping DDM for elasticity equations. It used the FEM for approximation and proved the domain decomposition convergence for elasticity equations. Paper [19] deal with the parallelization of finite element based Navier-Stokes codes using domain decomposition. A highly efficient sparse direct solver PARDISO was used in this study. Paper [20] presented a new non-overlapping domain decomposition method for the Helmholtz equation. Paper [21] presented a Robin-Robin non-overlapping domain decomposition method for an optimal boundary control problem associated with an elliptic boundary value problem. It showed the convergence of the sub-domain optimal solutions to the whole domain optimal solutions. Paper [22] developed an iterative scheme based on non-overlapping subdomains and Robin interface conditions for neutron diffusion problems. Paper [23] proposed a two-level scalable parallel FEM domain decomposition method for numerical simulation of partial differential equations. Great progress has been made in DDM for partial differential equations. However, these papers mainly focus on studying parallel FEM based on non-overlapping DDM and only for special class equations instead of general partial differential equations.

Multi-core clusters are playing increasingly important roles in the high-performance computing arena. But there is no single programming model for multi-core clusters. The hybrid MPI/OpenMP programming model is the emerging trend for parallel programming on multi-core clusters. There are many studies on the MPI/OpenMP programming model. Paper [15] gave a more comprehensive analysis of the hybrid MPI/OpenMP approach based on numerical accuracy and convergence in addition to presenting the performance of the code on current multi-core architectures. Paper [16] presented the hybrid MPI/OpenMP scheme for scalable parallel pseudospectral computations for fluid turbulence. Paper [25] proposed a hybrid MPI/OpenMP parallelization of an FFT-based 3D Poisson solver. Paper [26] combined the hybrid MPI/OpenMP model with adaptive integral method (AIM) in order to improve the scalability of AIM. Paper [27] proposed an efficient hybrid MPI/OpenMP algorithm for coupled Euler–Lagrange simulations. This algorithm facilitates cavitations predictions for challenging industrial applications. Paper [28] developed a parallel iterative solver for finite-element methods using an OpenMP/MPI hybrid programming model on the earth simulator. Although much research has been undertaken on the hybrid MPI/OpenMP programming model, little work has tried to use the MPI/OpenMP programming model to achieve overlapping domain decomposition strategy for parallel simulations of partial differential equations.

3 ANALYSIS OF DATA PROCESSING METHODS

IoT faces some difficulties which contain massive data storage and data processing. In order to accurately analyze large amounts of data obtained from sensors, things require powerful computation ability and storage resources to handle large-scale applications which involve intensive computation. For tasks with large sensed data, the strategy of data parallelism provides powerful computation ability and solves the problem of storage resources. In addition, data parallelism exhibits a natural form of scalability [29]. DDM is the mainstream of data parallelism. It is an efficient and flexible parallel algorithm for the large system of equations arising from the discretization of partial differential equations [18–21]. Since numerical discretization methods are not parallel algorithms, they can achieve parallel simulations of partial differential equations by means of domain decomposition strategy. This section focuses on the analysis of overlapping DDM and the numerical discretization method FDM.

3.1 Division methods of overlapping DDM

For problems involve large sensed data, DDM has been proved to be effective for improving computation ability and reducing storage requirements. In [30], DDM is defined as: domain decomposition refers to the process of subdividing the original problem of a large linear system into smaller problems. Since the sub-domains (subproblems) can be handled independently, such methods are very attractive for parallel computing platforms. Generally there are two main classes of domain decomposition strategies which depend upon the partition of the original domain into sub-domains, which is whether the subdomains are overlapping or non-overlapping. The overlapping DDM can be attractive for parallel computing since it leads to solving only local problems, and performing local communications between the neighboring sub-domains to exchange boundary conditions at the domains interfaces. The main advantage gained from using overlapping DDM is that it can lead to a better rate of convergence [31]. So this paper adopts the overlapping DDM.

There are three basic principles when using overlapping DDM to divide calculation domains. The first basic principle is that each sub-domain should be ruled as far as possible. The second principle is that the size of each sub-domain should be equal. This measure can avoid the phenomenon that the wait time between processors is too long. The third is that the area of overlapping domains should be as small as possible. The reason is that the area of overlapping domains representatives the communication overhead. For a three-dimensional solution domain, the division of the domain can be divided into three types. They are one-dimensional division, two-dimensional division and three-dimensional division, respectively. For dividing calculation domain into eight sub-domains as the example, one-dimensional division will obtain seven communication surfaces, while the two-dimensional division only four and three-dimensional division only three. Different types of division can be shown in Fig. 1. If the calculation domain is a three-dimensional cube given by Fig. 1(a-c), the three-dimensional division will get the minimum communication overhead and it should be the first choice. When the calculation domain is a cuboid shown in Fig. 1(d), the communication area of the three-dimensional division is not the smallest. In this case, one-dimensional division should be the first choice.

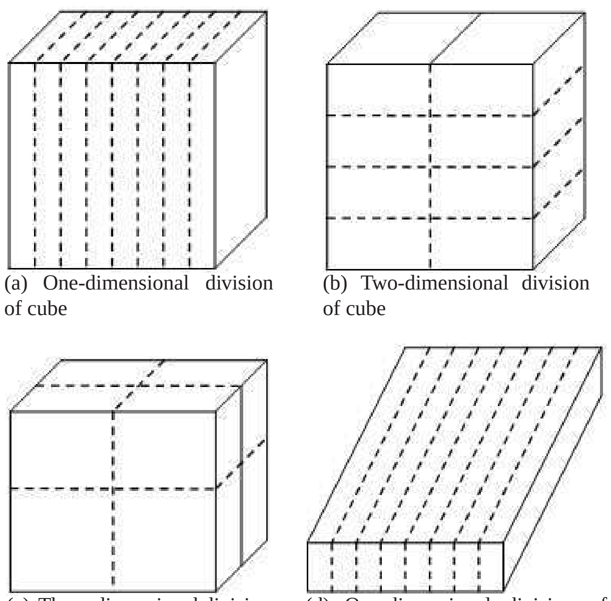
(a) One-dimensional division of cube (b) Two-dimensional division of cube (c) Three-dimensional division of cube (d) One-dimensional division of cuboid

Fig. 1. Different types of domain division

The specific overlapping DDM based on one-dimensional direction steps can be described as follows. Firstly, the solution domain Ω can be divided into several non-overlapping sub-domains D_i with equal size. Then, sub-domain Ω_i can be obtained by expending each sub-domain D_i a certain scale and removing the section outside the domain Ω. All the sub-domains Ω_i constitute an overlapping decomposition of the domain Ω. Based on above mentioned decomposition techniques, taking a two-dimensional domain for example, the case of dividing the calculation domain into two overlapping sub-domains Ω_1, Ω_2 can be shown in Fig. 2. The grey domain indicates the overlapping domain between Ω_1 and Ω_2.

3.2 Analysis of finite difference method

IoT technology can be used to monitor science as well as engineer problems which can be described as partial differential equations. The FDM is an important tool for numerical simulations of partial differential problems. FDM

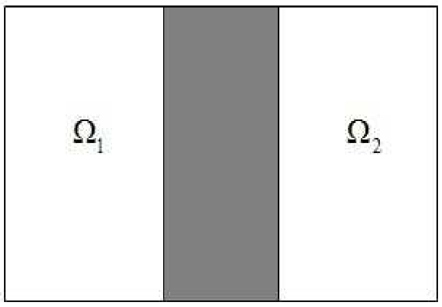

Fig. 2. An overlapping DDM based on two sub-domains

was already known by L. Euler. It has been one of the simplest forms of discretization. It proceeds by replacing the derivatives in the differential equations by finite difference approximations. This gives an algebraic system of equations to be solved in place of the differential equation. The solution domain is partitioned in space and time. The approximations of the solution are computed at the space and time points. A classical FDM approximates the differential operators constituting the field equation locally. Therefore a structured grid is required to store local field quantities. The order of the differential operator of the original problem formulation directly dictates the number of nodes to be involved.

The FDM, FEM and boundary element method (BEM) are the three most commonly used numerical discretization methods. The advent of FDM in numerical applications began in the early 1950s and its development was stimulated by the emergence of computers that offered a convenient framework for dealing with complex problems of science and technology. While other methods, such as the FEM and BEM have enjoyed recent popularity, FDM is still utilized for a wide array of computational engineering and science problems. Compared with other numerical discretization methods, FDM has obvious advantages to support its widely applications. The first advantage is that the theory of FDM is quite mature. The second advantage of FDM is that the amount of computation is less than other numerical methods. The great advantages of FDM are its simplicity, generality, flexibility, efficiency and robustness, and the wide range of systems to which FDM can be applied. The method is the arm-by-arm splitting technique which makes the method in multi-dimensional as simple as in one-dimensional. Based on the above mentioned advantages of FDM, this paper adopted the FDM to establish mathematical model on each sub-domain.

4 THE ARCHITECTURES OF COMPUTER CLUSTERS AND PROGRAMMING MODEL

One of the most important roles in IoT is computers. In order to design an efficient method for massive data processing, it needs to understand the computer architectures. Today's dominant computing platforms are multi-core clusters. This section firstly introduces the architectures of multi-core clusters. Parallel architectures and programming models are not independent. Then, this section briefly introduces the two most important programming models. Finally, this section introduces the hybrid MPI/OpenMP programming model in order to exploit better parallelism for multi-core clusters.

4.1 Architectures of multi-core clusters

The growing tendency for petascale platforms is toward a hierarchical shared-memory node structure with each node having multiple sockets. Each socket has multiple compute cores with shared or separate caches [32]. Multi-core clusters are hybrid parallel architectures that consist of a number of nodes which are connected by a fast interconnection network. Each node contains multiple sockets which have access to a shared memory, while the data on other nodes may usually be accessed only by means of explicit message passing. Examples of such systems are multi-processor clusters from SUN, SGI, IBM, a variety of multi-core PC clusters, supercomputers like the NEC SX-6. The architecture of multi-core cluster with two nodes can be shown in Fig. 3.

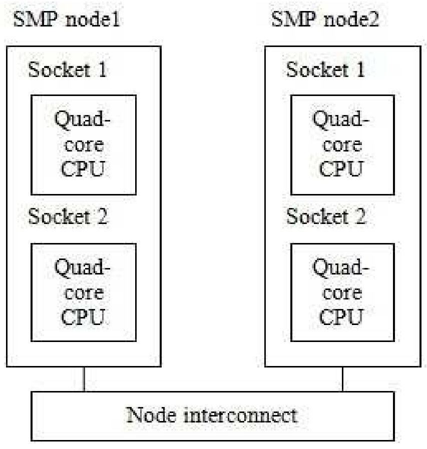

Fig. 3. The architecture of multi-core cluster with 2 nodes

Once a specific decomposition strategy is chose, programmers must choose programming models to implement. Message passing and data parallelism are the two most important parallel programming models, which are based on the programming level to distinguish. Message passing interface (MPI) is a typical representative of the message passing programming model for distributed memory parallel architectures and open specifications for multiprocessing (OpenMP) is a typical representative of the data parallellism programming model.

4.2 Programming model of distributed architectures

The IoT system is a distributed memory architecture which contains many computer nodes. MPI is a standard API for message passing which has been designed for distributed memory parallel architectures. It is the most popular parallel programming environment. MPI is an efficient, flexible, scalable and portable interface for developing parallel applications on multi-platform multi-core parallel programming architectures, including LINUX, UNIX and Windows platforms. It supports multiple languages, such as FORTRAN, C, and C++. MPI is a library, not a programming language. It must be bound to the specific programming language to be realized. At present the most mainstream programming language is the C, while in the fields of science and engineering is the FORTRAN language.

MPI provides the user with a programming model where processes communicate with other processes by calling library routines to send or receive messages. The advantage of MPI programming model is that users have completely control over data distribution, process synchronization, permitting the optimization data locality and workflow distribution. Although, MPI achieves the process-level parallelism, it dose not take full advantage of computing performance of each kernel in a processor. And it suffers from a few deficiencies. Decomposition, development and debugging of applications can be time-consuming and significant code changes are often required. Communications can create large overhead and the code granularity often has too large to increase the latency. Finally, global operations can be very expensive. These deficiencies will reduce the efficiencies of pure MPI programs carried out on multi-core clusters.

4.3 Programming model of shared architectures

Each computer node in IoT system commonly contains multiple cores which is shared memory architecture. The OpenMP is a standarddifferent types of parallel programming models shared memory API for a single application. The OpenMP supports multi-platform shared memory parallel programming in C/C++ and FORTRAN on all architectures, including UNIX platforms and Windows platforms. Jointly defined by a group of major computer hardware and software vendors, OpenMP is a portable, scalable model that gives parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from the desktop to the supercomputer. It consists of a set of compiler directives and library routines that extend FORTRAN, C, and C++ codes to express shared-memory parallelisms.

OpenMP provides the fork-and-join execution model as shown in Fig. 4. The master thread spawns a team of threads as needed. OpenMP is usually used to parallelize time-consuming loops. The OpenMP directives are placed only on the outer loops within a loop nest. The advantage of OpenMP is that an existing code can be easily parallelized by placing OpenMP directives around time-consuming loops which do not contain data dependence. And the source code is unchanged. The OpenMP implementation also shows good scalability, but has the following disadvantages when compared with the MPI implementation. In order to achieve thread-level parallelism, OpenMP requires a shared address space which limits the scalability to the number of CPUs within one multi-core socket. This disadvantage makes the pure OpenMP programming model is not suitable for multi-core clusters.

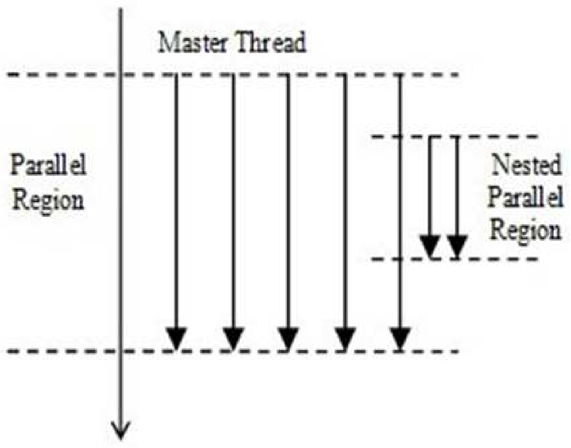

Fig. 4. Fork-join model in OpenMP

4.4 The hybrid MPI/OpenMP programmming model

Multi-core clusters can be thought as the hierarchical two-level parallel architectures, since they combine features of shared and distributed memory. Whilst message passing may be necessary to communicate between nodes, it is not immediately clear that this is the most efficient parallelization technique for multi-core clusters. Based on the above analysis of MPI and OpenMP, neither kind of programming model is suitable for multi-core architectures. How to exploit both advantages of MPI and OpenMP on multi-core clusters is a hot issue. The MPI/OpenMP hybrid parallelization paradigm is the emerging trend for parallel programming on architectures of multi-core clusters. Theoretically, a shared memory model such as OpenMP should offer a more efficient parallelization strategy within a multi-core socket. Hence a combination of MPI and OpenMP programming model provides a more efficient parallel strategy for a multi-core cluster. In this strategy, process-level parallelism is achieved through overlapping DDM by message passing among processes, and thread-level parallelism is obtained via loop-level parallelism inside each process by using OpenMP. The hybrid programming model can take full advantage of the hardware resources and greatly reduce the communication overhead because of a two-level parallellism on a multi-core cluster. The following Fig. 5 describes the hybrid MPI/OpenMP programming model on the multi-core cluster.

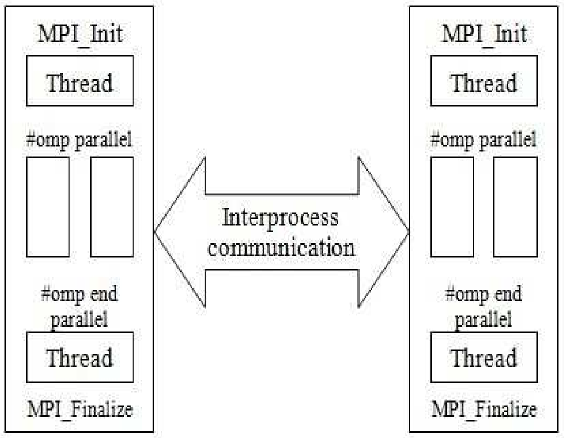

Fig. 5. The MPI/OpenMP programming model on the multi-core cluster

5 EFFICIENT IMPLEMENTATION OF DATA PROCESSING METHODS

DDM provides powerful computation ability for large-scale applications which involve large amounts of data processing. In order to implement the DDM on parallel computer architectures, there are two main issues for dealing. First, it needs to adopt a strategy for decomposing the program into parallel components. Second, it must actually write the parallel program, which requires to choose a programming model and interface for the implementation [29]. Based on the two issues, this section mainly introduces two aspects. First, this section proposes a parallel FDM overlapping domain decomposition strategy for data processing. Second, this section illustrates the hybrid MPI/OpenMP implementation for the parallel strategy.

5.1 The parallel FDM overlapping domain decomposition method

DDM is the mainstream of program decomposition strategy for multi-core clusters. Since numerical discretization methods are not parallel algorithms, they can achieve parallel simulations of partial differential equations by means of the DDM. Based on the division principles of overlapping DDM as stated in Section 3, overlappping DDM divides the domain of a problem into multiple overlapping sub-domains with equal size and assigns different processors to compute results for sub-problems. The number of sub-domains depends on the number of processors. Then we divide each sub-domain into a series of square grids. For each grid point, we adopt the FDM approximation. There are several kinds of formats to simplify a differential equation such as the forward, backward and central difference approximations. For space approximation, this paper applies the seven-point stencil FDM described in Equation (1). For time approximation, the partial differential equation is handled by the backward Euler method which is a fully implicit method described in Equation (2). The reason is that the backward approximates is unconditionally stable.

$$\frac{\partial^2 h}{\partial x^2} = \frac{h_{i+1} - 2h_i + h_{i-1}}{\Delta x^2} \tag{1}$$

$$\frac{\partial h}{\partial t} = \frac{h^{n+1} - h^n}{\Delta t} \tag{2}$$

Consequently, it can obtain a sparse linear algebraic system $Ax = b$ for each sub-problem, in which A is symmetric positive definite. For details about the deduction, readers can refer to our previous work [33]. However, it is very time-consuming by using the conventional techniques like Gauss elimination or basic stationary iterative methods to solve the sparse linear algebraic system $Ax = b$, especially when the matrix is quite large. It has been proved that the Krylov subspace methods could handle the problem efficiently and accurately. Conjugate gradient (CG) is suitable for solving symmetric algebraic systems. Meanwhile, both the robustness and convergence of the iterative methods can be improved by employing preconditioning techniques. And the preconditioned iterative methods have been proved to be one of the most efficient ways [34]. So in this paper, the preconditioned CG (PCG) is used to solve the system $Ax = b$. This paper adopts the Cholesky preconditioner. Furthermore, to take advantages of the sparsity of the coefficient matrix, the compressed sparse row (CSR) storage scheme is utilized in our code, which makes it less memory requirements.

5.2 The hybrid MPI/OpenMP implementation

Once a specific decomposition strategy is chose, it must choose the programming model to implement [29]. As stated in Subsection 4.4, the hybrid MPI/OpenMP programming model is suitable for multi-core cluster architectures. The hybrid MPI/OpenMP model is a mixed model.

MPI is used to pass message between sub-domains by calling the function `MPI_Sendrecv()` for point-to-point communications. In order to separate two adjacent massage passing, it needs to perform a synchronization call before receiving the massage. OpenMP is used to parallelize the loops under the MPI processes. The most time-consuming parts of the PCG algorithm are iterations of loops. The OpenMP directives are placed only on the outer loops within a loop nest. For details about using OpenMP parallelize loops, readers can refer to our previous work [35]. In this paper, the single program multiple data (SPMD) parallel scheme is adopted. It means that the computing tasks in sub-domains are assigned to several processors averagely. All the processors are equal in status, and no host CPU exists. Each node installs an MPI/OpenMP application. The program adopts the language C to bind MPI and OpenMP for expressing the hybrid parallelism. This scheme saves time of data allocation and is easy to implement.

6 NUMERICAL SIMULATIONS AND PERFORMANCE ANALYSIS

In order to evaluate the performance of the proposed method, this paper carried out numerical experiments based on the three-dimensional groundwater flow equation. The three-dimensional groundwater flow equation located in heterogeneous and anisotropic medium could be described by the following

$$\frac{\partial(K_{xx}\frac{\partial h}{\partial x})}{\partial x}+\frac{\partial(K_{yy}\frac{\partial h}{\partial y})}{\partial y}+\frac{\partial(K_{zz}\frac{\partial h}{\partial z})}{\partial z}-W=S_s\frac{\partial h}{\partial t} \quad (3)$$

where K_{xx}, K_{yy} and K_{zz} are values of hydraulic conductivity along the x, y and z coordinate axes, which are assumed to be parallel to the major axes of hydraulic conductivity; h is the potentiometric head; W is a volumetric flux per unit volume and represents sources or sinks of water; S_s is the specific storage of the porous material and t is the time.

6.1 The groundwater flow model based on the DDM

In recent years, groundwater flow simulation has become one of the top international issues in new generation of environmental applications. By using IoT-related technologies, it leads to enhancing the assessment of groundwater resources. For instance, in the setting of groundwater flow management, sensors can be deployed in the groundwater to monitor groundwater resource pollution, water quality and regional ecological in real time [3]. These will generate a high degree of autonomous data processing. These data are meant to tell us information of the groundwater flow pollution control and rational use of water resources in the future. After standardizing and storing, these data can be provided for mathematical models of groundwater flow, scientists and analysts. It can draw many new recognition knowledge which greatly enriches and develops to describe and predict groundwater flow theories and methods.

A simplified scenario of three-dimensional groundwater flow model was established in the paper. It is a homogeneous media ($K_{xx}=K_{yy}=K_{zz}$) with S_s, specified as a constant and Dirichelet boundary condition at the bordering portion. The distributions of real space points can be viewed as a cuboid array of $N \times N \times K$ real numbers, where *N*, *N*, *K* represents the number of grid points in row direction, column direction and layer direction. Based on the division methods of overlapping DDM shown in Subsection 3.1, this paper adopts the one-dimensional domain division along the layer K direction. The reason is that *N* is much larger than *K*. Each processor receives a calculation sub-domain of size $N \times N \times M$ grid points, where $M=(K+L)/NMPI$. The variable L indicates the number of overlapping layers and the variable *NMPI* indicates the number of MPI processes. For each grid point, FDM is used for discretizing the three-dimensional groundwater flow equation in this paper. The left-hand side of Equation (3) adopted seven-point stencil central difference method and the right-hand side uses backward difference approximations. Finally, each sub-domain can obtain a linear algebraic system as stated in Subsection 5.1. Values on overlapping domains are updated by taking averages of values on adjacent sub-domains. The stopping criterion is that values of each sub-domain satisfy this inequality $|h_{i,n+1}-h_{i,n}|/|h_{i,n+1}|<\varepsilon$ at iteration *n+1*, where i and ε denote the i-th sub-domain and the permissible error ε, respectively.

6.2 Test environment and numerical experiments

The numerical experiments were carried out on a windows cluster with two nodes. Each node includes dual-socket of Intel(R) Xeon(R) processors (1.6GHz). The parallel software environment is Windows 2003 Operating System, MPICH2-1.2-win-x86-64, OpenMP, C.

In the parallel overlapping DDM, the size of overlapping domain can affect the execution time. Theoretically, a larger overlapping domain has a better convergence. However, increasing the overlapping domain will result in increasing the amount of computation and communication. Thence the overlapping domain should not be too large to avoid the reduction in the efficiency of the algorithm. In general, the size of overlapping domain is chose arbitrarily. The optimal overlapping domain size depends on various conditions, such as problem size, initial values, architecture of hardware, performance of computing nodes, communication performance of network, the required solution accuracy, etc. Based on the above mentioned test environment and the problem in this article, the first experiment was focused on finding a better overlapping domain size which could make the execution time shorter. The first experiment was carried out with problem size $100 \times 100 \times 7$ with different overlapping domain and the numerical results can be shown in Table 1.

Table 1. Experimental results with different overlapping domain

overlapping domain size	1 layer	2 layers	3 layers
convergence speed	slow	fast	faster
execution speed	faster	fast	slow

Based on the second row of data in the Table 1, it is easy to see that the convergence of algorithm improves with the increase of overlapping domain. However, from the third row of data in the Table 1, it is found that the execution speed decreases with overlapping domain size increases. The reason is that the amount of computation for each sub-domain and the amount of communication between sub-domains significantly increase with the increase of the overlapping domain, which leads to the significant increase of computation time and communication time. Although the improvement of algorithm convergence can reduce the number of iterations, the reduction of time due to convergence improving can not make up the increase of computation time and communication time. In order to obtain a better parallel performance, the overlapping domain should not be large in this paper. The experimental results are only suitable for the problem stated in the paper, parallel approach proposed in this paper and the multi-core cluster in the paper.

Theoretically, a processor can launch multiple processes and a core can launch multiple threads. Hence, there are multiple types of hybrid MPI/OpenMP programming model. However, the excessive number of processes and threads will lead to competition for resources and the fewer number of processes and threads will lead to a waste of resources. Neither kind of the above programming model is able to get the best parallel performance. The aim of the second experiment was to find the most effective MPI/OpenMP programming model. The second experiment was tested for a fixed size of problem with $100 \times 100 \times 7$. The following different types of parallel programming model are evaluated. The execution time can be shown in Table 2.

Table 2. Execution time of different number of processes and threads

problem size	processes	threads	execution time
$100 \times 100 \times 7$	2	1	1.078
		2	0.930
		4	0.851
	4	1	0.746
		2	0.666
		4	0.635
	8	1	0.690
		2	0.692
		4	0.706

Based on Table 2, the execution time is the shortest when MPI starts four processes and each process starts four threads. The number of started processes is equal to the number of sockets within a multi-core cluster. The number of started threads by each process is equal to the number of cores in each socket. The reason is that some cores are idle when the number of threads is less than the number of cores. When the number of threads is greater than the number of cores, it will cause the problem of competing bandwidth. The same problem also exists in processes. This experimental results mean the best hybrid MPI/OpenMP programming model is that each socket starts one process and each core starts one thread. This kind of programming model ensures that all data for each process are guaranteed to be on the local memory of each socket, and so the most efficient memory access is possible.

Speedup is an important indicator for measuring the performance of parallel algorithm. The third experiment was to test the speedup of the parallel algorithm and carried out with problem size $100 \times 100 \times 50$. As stated in section of test environment, the multi-core cluster composes two nodes and four sockets. This experiment tested speedup when the number of processes is 1, 2 and 4 respectively. Based on conclusions of the second experiment, each process started four threads for solving each sub-problem. The measured speedup can be shown in Table 3.

Table 3. Speedup with different number of processes

problem size	processes	speedup
$100 \times 100 \times 50$	1	1
	2	1.88
	4	3.20

The experimental results show that the speedup increases with the number of processes grows. This phenomenon can be explained by the definition of speedup. The speedup is defined as the ratio of the solution time of original problem on one process and the solution time of sub-problem on multi-processes. And the efficiency is above eighty percent which is satisfactory.

Scalability is another important indicator for measuring the performance of parallel computing. The scalability can be shown from the relationship between speedup and problem size. The purpose of the forth experiment was to test the scalability of the proposed parallel approach. This experiment was carried out with different problem size. Each problem was divided into four sub-problems. And each process starts four threads. The measured speedup can be shown in Table 4.

Table 4. Speedup with different problem size

problem size	speedup
$100 \times 100 \times 42$	3.11
$100 \times 100 \times 46$	3.15
$100 \times 100 \times 50$	3.20
$100 \times 100 \times 54$	3.25
$100 \times 100 \times 58$	3.29
$100 \times 100 \times 62$	3.33

From the data described in Table 4, we can see that the speedup increases with problem size grows. This phenomenon can be explained as follows. With the problem size increases, the proportion of computation increases while the proportion of communication overhead reduces. That means the parallel method proposed in this paper has a better scalability.

Based on the results of the third and the last experiments, we can conclude that the hybrid MPI/OpenMP implementation for the overlapping DDM is an effective way for parallel simulations of large-scale partial differential problems with intensive computation.

7 CONCLUSION

Sensor technology is one of the most mature and practical IoT-related technologies. It generates massive data that need to be stored and processed when sensor technology is used to monitor information in real time. Partial differential problems which involve intensive computation are the most common in the engineering and scientific research. Data processing is unanimously recognized as one of the core functionalities of the IoT. Efficient techniques are required to analyze these massive data for supporting intelligent decisions making. In order to efficiently solving partial differential problems with massive data, this paper proposed a parallel approach which combines the overlapping DDM and multi-core clusters technology for parallel simulation of large-scale partial differential equations. According to the parallel characteristics of multi-core cluster architectures, this approach adopts the hybrid MPI/OpenMP programming model to implement the overlapping DDM. On a cluster with two nodes, this paper carried out experiments for taking the groundwater flow equation as an example, simulation results indicate that this parallel approach can facilitate parallel simulations of partial differential problems with large data.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China (41102158) and the Fundamental Research Funds for Universities of China.

REFERENCES

[1] E. Karmouch and A. Nayak, “A distributed constraint satisfaction problem approach to virtual device composition,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 1997–2009, 2012.

[2] N. DLODLO, “Adopting the internet of things technologies in envrionmental management in south africa,” in *2012 International Conference on Environment Science and Engieering*, pp. 14–28, 2012.

[3] L. Minbo, C. Guangyu, and ZhuZhu, “Smart data processing of agriculture iot,” in *Proceedings of 2012 International Conference on the Internet of Things*, pp. 319–325, 2012.

[4] L. Berenguer, T. Dufaud, and D. Tromeur-Dervoutz, “Aitken′s acceleration of the schwarz process using singular value decomposition for heterogeneous 3d groundwater flow problems,” *Computers & Fluids*, 2012.

[5] L. Giraud, A. Haidar, and S. Pralet, “Using multiple levels of parallelism to enhance the performance of domain decomposition solvers,” *Parallel Computing*, pp. 150–158, 2010.

[6] P. Guérin, A.-M. Baudron, and J.-J. Lautard, “Domain decomposition methods for the neutron diffusion problem,” *Parallel Computing*, pp. 2159–2167, 2010.

[7] A. M. Amini, D. Dureisseix, P. Cartraud, and N. Buannic, “A domain decomposition method for problems with structural heterogeneities on the interface: Application to a passenger ship,” *Comput. Methods Appl. Mech. Engrg*, pp. 3452–3463, 2009.

[8] D. Odiévre, P.-A. Boucard, and F. Gatuingt, “A parallel, multiscale domain decomposition method for the transient dynamic analysis of assemblies with friction,” *Comput. Methods Appl. Mech. Engrg*, pp. 1297–1306, 2010.

[9] O. Lloberas-Valls, D. Rixen, A. Simone, and L. Sluys, “Domain decomposition techniques for the efficient modeling of brittle heterogeneous materials,” *Comput. Methods Appl. Mech. Engrg*, pp. 1577–1590, 2011.

[10] I. Pultarová, “Preconditioning of the coarse problem in the method of balanced domain decomposition by constraints,” *Mathematics and Computers in Simulation*, pp. 1788–1798, 2012.

[11] G. Tang, E. F. D. Azevedo, FanZhang, J. C.Parker, D. B. Watson, and P. M. Jardine, “Application of a hybrid mpi/openmp approach for parallel groundwater model calibration using multi-core computers,” *Computer & Geosciences*, pp. 1451–1460, 2010.

[12] J. Knezović, M. Kovać, and H. Mlinarić, “Integrating streaming computations for efficient execution on novel multicore architectures,” *Automatika-Journal for Control, Measurement, Electronics, Computing and Communications*, pp. 387–396, 2010.

[13] J. Knezović, H. Mlinarić, and M. Źagar, “Lossless image compression exploiting streaming model for efficient execution on multicores,” *Automatika-Journal for Control, Measurement, Electronics, Computing and Communications*, 2012.

[14] Y. He and C. H. Q. Ding, “Mpi and openmp paradigms on cluster of smp architectures: the vacancy tracking algorithm for multi-dimensional array transposition,” in *In Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, IEEE Computer Society Press, 2002.

[15] H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, and B. Chapman, “High performance computing using mpi and openmp on multi-core parallel systemss,” *Parallel Computing*, pp. 562–575, 2011.

[16] P. D. Mininni, D. Rosenberg, R. Reddy, and A. Pouquet, “A hybrid mpi-openmp scheme for scalable parallel pseudospectral computations for fluid turbulenc,” *Parallel Computing*, pp. 316–326, 2011.

[17] X. Wu and V. Taylor, “Performance characteristics of hybrid mpi/openmp implementations of nas parallel benchmarks sp and bt on large-scale multicore clusters,” *Computer Journal*, pp. 154–167, 2012.

[18] A. Chakib, A. Ellabib, and A. Nachaoui, “A domain decomposition convergence for elasticity equations,” *Mathematics and Computers in Simulation*, pp. 154–167, 2008.

[19] M. P. Raju and S. Khaitan, “Domain decomposition based high performance parallel computing,” *International Journal of Computer Science Issuess*, pp. 27–32, 2009.

[20] Y. Boubendir, X. Antoine, and C. Geuzaine, “A quasi-optimal non-overlapping domain decompostion algorithm for the helmholtz equation,” *Journale of Computational Physics*, pp. 262–280, 2012.

[21] L. Hou and J. Lee, “A robin-robin non-overlapping domain decompostion method fro an elliptic boundary control problem,” *Internationla journal of numerical analysis and modeling*, pp. 443–465, 2011.

[22] J. Sikora and T.Grzywacz, “Domain decomposition method for diffuse optical tomography problems,” *Engineering Analysis with Boundary Elements*, pp. 1005–1013, 2012.

[23] F. Kong, X. Cai, and Y. Zhao, “Scalable parallel domain decomposition methods for numerical simulation of pde,” *Bulletin of advanced technology research*, 2011.

[24] C. Wu, *The research of three-dimensional FDTD parallel algorithm based on MPI and OpenMP*. PhD thesis, Huazhong University of Science and Technology, 2009.

[25] A. Gorobets, F. Trias, R. Borrell, O. Lehmkuhl, and A. Oliva, “Hybrid mpi+openmp parallelization of an fft-based 3d poisson solver with one periodic direction,” *Computers & Fluids*, pp. 101–109, 2011.

[26] F. Wei and A. E. Yilmaz, “A hybrid message passing/shared memory parallelization of the adaptive integral method for multi-core cluster,” *Parallel Computing*, pp. 279–301, 2011.

[27] S. Yakubov, B. Cankurt, M. Abdel-Maksoud, and T. Rung, “Hybrid mpi/openmp parallelization of an euler lagrange approach to cavitation modelling,” *Computers & Fluids*, 2012.

[28] K. Nakajima, “Parallel iterative solvers for finite-element methods using an openmp/mpi hybrid programming model on the earth simulator,” *Parallel Computing*, pp. 1048–1065, 2005.

[29] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, *Sourcebook of Parallel Computing*. Morgan Kaufmann, 2003.

[30] B. Smith, P. Bjorstad, and W. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. New York: Cambridge University Press, 1996.

[31] S. A. Nadeem, *Parallel domain decomposition preconditioning for the adaptive finite element solution of elliptic problems in three dimensions*. PhD thesis, The University of Leeds School of Computing, 2001.

[32] S. Benkner and V. Sipkova, "Exploiting distributed-memory and shared-memory parallelism on clusters of smps with data parallel programs," *International Journal of Parallel Programming*, pp. 3–19, 2003.

[33] T. Cheng, X. Ji, and Q. Wang, "An efficient parallel method for large-scale groundwater flow equation based on petsc," in *IEEE Youth Conference on In-formation, Computing and Telecommunications, IEEE YC-ICT 2009*, pp. 190–193, 2009.

[34] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 1993.

[35] D. Li, X. Ji, and QunWang, "A parallel pcg solver for large-scale groundwater flow simulation based on openmp," in *The 4th International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 306–309, 2011.

Dandan Li was born in Shanxi Province, China, in 1988. She is a currently a Ph.D. candidate at the School of Information Engineering, China University of Geosciences (Beijing). Her research interests include parallel computing and numerical simulation.

Xiaohui Ji is an associate professor at the School of Information Engineering, China University of Geosciences (Beijing), China. She received her Ph.D. from the Institute of Software, Chinese Academy of Sciences in 2005. Her research interests include numerical simulation, parallel computing, constraint processing and global optimization problems.

Qun Wang received the B.Sc. degree from China University of Geosciences (Wuhan) in 1982 and the M.Sc. and Ph.D. degree from HuaZhong University of Science and Technique in China, in 1987 and 1990, respectively. He worked as a research fellow at the University of Alberta from 1992 to 1997. He worked in soft-ware design and development in Canada from 1997 to 2003. He is currently a professor and the dean at the School of Information Engineering, China University of Geosciences (Beijing), China. He has published more than 100 scientific papers. His research interests include high performance computing and visualization techniques and geoengineering, geospatial data mining, networking engineering and security.

AUTHORS' ADDRESSES

Dandan Li, M.Sc.
Xiaohui Ji, Ph.D.
Qun Wang, Ph.D.
School of Information Engineering,
China University of Geosciences (Beijing),
University of the first two authors,
Xue Yuan Road, Haidian District, 100083, Beijing, China
email: dandanlicugb@163.com, xhji@cugb.edu.cn, qunw@cugb.edu.cn

Received: 2012-12-25
Accepted: 2013-02-28