ELSEVIER

Computer Physics Communications 83 (1994) 181–196

Computer Physics Communications

Massively parallel algorithms for computational nanoelectronics based on quantum molecular dynamics

Aiichiro Nakano, Priya Vashishta, Rajiv K. Kalia

Concurrent Computing Laboratory for Materials Simulations, Department of Computer Science, Department of Physics & Astronomy, Louisiana State University, Baton Rouge, LA 70803-4001, USA

Received 21 April 1994; revised 24 June 1994

Abstract

A quantum molecular dynamics (QMD) simulation scheme has been developed which is suitable for the study of highly nonlinear electron dynamics far from equilibrium in semiconducting devices of nanometer size. The core components of the QMD algorithm are: (i) solution of the time-dependent Kohn–Sham or Schrödinger equations; and (ii) solution of the Poisson equation for the direct electron–electron interaction. Efficient parallel algorithms for these components are the space-splitting Schrödinger solver and the dynamical-simulated-annealing Poisson solver. Both algorithms are scalable and require only nearest-neighbor communications. These algorithms are implemented on an 8,192-node MasPar computer. Timing tests are carried out to compare these algorithms with other parallel Schrödinger and Poisson solvers. The timing results on the MasPar are also compared with those on other parallel and vector architectures.

1. Introduction

Recent advances in the micro fabrication technology have led to the realization of nanometer (10^{-9} m) scale devices. In nanodevices, novel physical effects are used to attain logic functionality which conventional technology cannot achieve. For example, quantum effects provide logic functionality in resonant-tunneling diodes (RTD) [1] and quantum-interference transistors [2]. Another example is the enhanced electron–electron interaction in nanostructures. Electrons in small semiconducting devices can be viewed as being in a highly nonlinear medium. Shallow water analogy for a ballistic field-effect transistor suggests the existence of nonlinear phenomena such as soliton propagation [3]. In a RTD, quantum effects coupled with the electron–electron interaction were shown to cause intrinsic bistability [4]. Recently, multiple-branch current–voltage (I-V) curves were observed even at zero bias [5]. This phenomenon may find a novel application as a room-temperature memory device which dissipates nearly zero power. The Coulomb interaction also leads to the Coulomb blockade effect in small tunnel junctions [6]. This effect is used to control the flow of a single electron (single electronics). Electron–phonon interaction can also be controlled by artificially tailored phonon modes (phononics) [7]. In addition, switching of nanodevices occurs far from equilibrium. A proper account of dissipation is essential to describe such highly nonequilibrium dynamics [5].

SSDI 0010-4655(94)00094-8

Microscopic simulations can significantly enhance our understanding of such nonlinear, nonequilibrium dynamics. "Computational Electronics" is a rapidly growing, interdisciplinary approach (including Computer Science, Electronic Engineering, Physics, and Mathematics) to device simulations [8]. In recent years, device simulations have played a significant role in the development of very large-scale integrated-circuit (VLSI) technology. At present, the most widely used device simulations are based on either macroscopic drift-diffusion equations or semiclassical Boltzmann transport equations combined with the Monte Carlo sampling technique [8]. In nanodevices, however, conventional device simulations cannot describe the physical effects mentioned above. It is necessary to develop quantum-transport simulation methods which include novel physical effects crucial to the operation of nanodevices. This so-called "Computational Nanoelectronics" is the subject of this paper.

Recently we have applied the quantum molecular dynamics (QMD) method [9–11] to the study of highly nonlinear, far-from-equilibrium electron dynamics in nanodevices [12–16]. Various physical effects which are dominant in nanodevices but are beyond the scope of conventional semiclassical device simulations are investigated. These include: (i) intrinsic dynamic instability in a RTD [12,13]; (ii) spin-dependent Coulomb-blockade in a quantum-dot diode [14]; (iii) phonon-induced electron localization in a double quantum dot [15]; and (iv) multiple-trapping electron transport in amorphous silicon [16].

The core computational kernel of the QMD approach has two parts: Parabolic partial differential equations (the time-dependent Kohn–Sham equation or the time-dependent Schrödinger equation) for quantum particles (electrons or phonons) and second-order ordinary differential equations (Newton's second law of motion) in molecular dynamics (MD) simulations for classical N-body systems. Special algorithms are needed for the long-range Coulomb interaction in both classical and quantum simulations.

QMD simulations of realistic nanodevices require enormous computing resources. It is thus crucial to exploit the recent development of parallel computing technology. According to the classification of problems by Fox, the solution of the time-dependent Schrödinger equation is a synchronous problem and the problem maps well onto a single instruction multiple data (SIMD) architecture [17]. On the other hand, classical MD corresponds to a loosely synchronous problem and these simulations are more appropriate for multiple instruction multiple data (MIMD) architectures [17].

Implementations of classical MD on MIMD architectures are discussed in our previous publications [18–21]. We have developed efficient algorithms based on multiresolutions in both time and space [21]. In the multiresolution MD scheme, the long-range Coulomb interaction in periodic systems is computed with the fast multipole method [22] and the reduced cell multipole method [23].

The present paper deals with the quantum aspect of the QMD method, i.e., solutions of parabolic partial differential equations (PDE) such as the time-dependent Kohn–Sham equation or the time-dependent Schrödinger equation. This problem is coupled with another computationally intensive problem, i.e., solution of elliptic PDEs (the Poisson equation) for the long-range electron–electron interaction. We have designed parallel algorithms for both problems on SIMD architectures. In this paper, we consider a specific parallel architecture in which the processors are connected in a two-dimensional rectangular mesh. Physical systems we consider are also two dimensional, and are represented on a rectangular grid using a finite difference scheme. The algorithms we report take advantage of the connectivity of the parallel architecture.

Conventional solutions of the time-dependent Schrödinger equation are based on the split-operator approach using either the spectral method (SM) [24] or the alternative-direction-implicit generalization of the Crank–Nicholson (CN) method [25]. The SM uses the fast Fourier transform (FFT) algorithm [26]. Accordingly the complexity of the SM is $\mathcal{O}(M \log M)$, where M is the number of grid points used to discretize wave functions. By assigning the value of a wave function at each grid point to a processor in an array of M processors, the parallel time complexity of the SM algorithm becomes $\mathcal{O}(\log M)$. (Here, we define the parallel time complexity of an algorithm to be the time required to solve a problem with M

grid points on an array of M processors.) Also the SM involves considerable communication because of the butterfly communication in the FFT algorithm [26]. The CN method is implemented on parallel computers using, e.g., the recursive doubling algorithm [27]. The resulting parallel time complexity is $\mathcal{O}(\log M)$ and the implementation involves considerable communication.

Recently, a powerful algorithm called the space-splitting method (SSM) has been developed to solve the time-dependent Schrödinger equation [28]. The SSM is based on the decomposition of a tridiagonal matrix, related to the kinetic energy operator in the Schrödinger equation, into direct sums of 2×2 matrices. This decomposition provides an explicit scheme to propagate wave functions in time with $\mathcal{O}(M)$ operations. The parallel time complexity of the SSM algorithm is $\mathcal{O}(1)$, i.e., the SSM is scalable. (An algorithm is considered scalable if the required number of parallel operations remains constant when the number of processors scales linearly with the problem size.) In addition, the operations in the SSM are local, requiring only nearest-neighbor communications with little communication overhead.

The solution of the Poisson equation for the electron–electron interaction is the most time consuming part of a QMD simulation. Conventional Poisson solvers include the fast Poisson solver (FPS) and the multigrid method (MGM). The FPS solves the Poisson equation with $\mathcal{O}(M \log M)$ operations using the FFT [29]. The MGM reduces the complexity to $\mathcal{O}(M)$ and is asymptotically faster than the FPS [30]. This smaller complexity of the MGM algorithm is achieved through computationally less intensive coarser grids for long wave-length components of the solution. In contrast to the FPS, the MGM is applicable to general elliptic PDEs with nonconstant coefficients on irregular domains and with irregular boundary conditions. One of the difficulties with a massively parallel MGM algorithm is that most of the processors are idle when operating on coarser grids (idle processor problem). Because of this, the parallel time complexity of the parallel MGM algorithm is $\mathcal{O}(\log M)$, the same as that of the FPS, even though the sequential complexity of the MGM is superior to that of the FPS. Although we cannot reduce the parallel time complexity of the MGM algorithm, the idle processors can be used to accelerate the convergence of the MGM. In the parallel superconvergent multigrid method (PSMG), the idle processors are used to solve multiple coarse grid problems simultaneously and their results are combined to provide the optimal convergence [31]. On parallel computers, both the FPS and MGM algorithms involve considerable communications.

When the Poisson equation is coupled to a dynamical equation, as is the case in the QMD method, successive calls to a Poisson solver are correlated with each other (the solution of the Poisson equation at a call is a good initial guess for the solution at the next call). In such a case, it is possible to design a highly efficient algorithm by introducing fictitious dynamics for the electrostatic potential in the Poisson equation [32]. This dynamical-simulated-annealing (DSA) method is implemented with $\mathcal{O}(1)$ parallel time complexity on a machine with the number of processors comparable to the number of grid points, M. In contrast, conventional Poisson solvers require $\mathcal{O}(\log M)$ parallel operations. The difference stems from the fact that the DSA is an initial value problem (hyperbolic PDE) instead of the Cauchy problem in conventional Poisson solvers. Since the DSA algorithm requires only nearest-neighbor communications, it can be implemented efficiently on SIMD architectures.

In this paper, we describe the implementations of the SSM algorithm for the time-dependent Schrödinger equation and the DSA algorithm for the Poisson equation on a massively parallel computer. Numerical experiments are performed on an in-house 8,192-node SIMD computer from MasPar. The performance is compared with other parallel Schrödinger and Poisson solvers. The timing tests on the MasPar are also compared with those on other parallel and vector architectures.

The outline of this paper is as follows: In the next section, we describe the QMD method, and in Section 3, we discuss the space-splitting Schrödinger solver. The dynamical-simulated-annealing Poisson solver is described in Section 4. We compare performances of these parallel QMD algorithms in Section 5, and Section 6 contains the conclusion.

2. Quantum molecular dynamics

In the QMD method, we simulate the dynamics of coupled systems of either electrons and classical ions or electrons and phonons. In this scheme, the electron–electron interaction is included in the framework of the time-dependent density functional theory [33]. The time evolution of electron orbitals, $\Psi_i(\boldsymbol{r}, t)$, is given by the time-dependent Kohn–Sham equations [33,34]:

$$\left[i\hbar\frac{\partial}{\partial t} - \frac{1}{2m}\left(\frac{\hbar\nabla}{i} + \frac{e}{c}\left[\boldsymbol{A}(\boldsymbol{r}, t) + \boldsymbol{A}_{\mathrm{xc}}(\boldsymbol{r}, t)\right]\right)^2 \right.$$
$$- \frac{e^2}{2mc^2}\left(\boldsymbol{A}^2(\boldsymbol{r}, t) - \left[\boldsymbol{A}(\boldsymbol{r}, t) + \boldsymbol{A}_{\mathrm{xc}}(\boldsymbol{r}, t)\right]^2\right) - v_0(\boldsymbol{r}, t) - v_{eI}(\boldsymbol{r}, t)$$
$$\left. - \int \mathrm{d}\boldsymbol{r}' \, \frac{e^2 n(\boldsymbol{r}', t)}{|\boldsymbol{r} - \boldsymbol{r}'|} - v_{\mathrm{xc}}(\boldsymbol{r}, t) \right] \Psi_i(\boldsymbol{r}, t) = 0, \tag{1}$$

where $v_0(\boldsymbol{r}, t)$ is an external potential, $v_{\mathrm{xc}}(\boldsymbol{r}, t)$ is the exchange-correlation potential [33], and $\boldsymbol{A}_{\mathrm{xc}}(\boldsymbol{r}, t)$ is the exchange-correlation vector potential [34]. In Eq. (1), m and e are the mass and charge of an electron, and c is the light speed. The electron density $n(\boldsymbol{r}, t)$ is calculated from

$$n(\boldsymbol{r}, t) = \sum_{i=1}^{N_e} |\Psi_i(\boldsymbol{r}, t)|^2, \tag{2}$$

where N_e is the number of electrons.

For a coupled system of electrons and classical ions, $v_{eI}(\boldsymbol{r}, t) = V_{eI}(\boldsymbol{r}, \{\boldsymbol{R}_I(t)\})$ is the interaction potential between an electron and ions, where $\boldsymbol{R}_I(t)$ is the coordinate of the Ith ion. Assuming adiabatic dynamics for classical ions, we solve Newton's equations,

$$M_I \frac{\mathrm{d}^2 \boldsymbol{R}_I(t)}{\mathrm{d}t^2} = -\frac{\partial V_I(\{\boldsymbol{R}_I(t)\})}{\partial \boldsymbol{R}_I(t)} - \int \mathrm{d}\boldsymbol{r}\, n(\boldsymbol{r}, t)\, \frac{\partial V_{eI}(\boldsymbol{r}, \{\boldsymbol{R}_I(t)\})}{\partial \boldsymbol{R}_I(t)}, \tag{3}$$

concurrently with Eq. (1). In Eq. (3) M_I is the mass of the Ith ion, and $V_I(\{\boldsymbol{R}_I(t)\})$ is the interionic potential.

Eqs. (1) and (3) are suitable for electron transport in disordered materials at relatively high temperatures [16]. At very low temperatures, on the other hand, we need to incorporate the quantum nature of lattice vibrations. Coupled systems of electrons and phonons can be studied either by perturbation [35] or in a mean-field approximation [15,36]. In the mean-field approximation, $v_{eI}(\boldsymbol{r}, t)$ in Eq. (1) is expressed as [15],

$$v_{eI}(\boldsymbol{r}, t) = \sum_{\boldsymbol{k}} \left(Q_{\boldsymbol{k}}\, \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}} \sum_{\nu} \sqrt{\nu + 1}\, \phi_{\boldsymbol{k}}^{(\nu)*}(t)\, \phi_{\boldsymbol{k}}^{(\nu+1)}(t) + \mathrm{c.c.} \right), \tag{4}$$

where $\phi_{\boldsymbol{k}}^{(\nu)}$ is the phonon wave function for the $\boldsymbol{k}$th phonon mode in the number representation and $Q_{\boldsymbol{k}}$ is the corresponding electron–phonon interaction coefficient. Phonon wave functions evolve in time following the time-dependent Schrödinger equations [15],

$$\left(i\hbar\frac{\partial}{\partial t} - \nu\hbar\omega_{\boldsymbol{k}} \right) \phi_{\boldsymbol{k}}^{(\nu)}(t) = Q_{\boldsymbol{k}} n_{\boldsymbol{k}}^*(t)\, \sqrt{\nu + 1}\, \phi_{\boldsymbol{k}}^{(\nu+1)}(t) + Q_{\boldsymbol{k}}^* n_{\boldsymbol{k}}(t) \sqrt{\nu}\, \phi_{\boldsymbol{k}}^{(\nu-1)}(t), \tag{5}$$

where $n_{\boldsymbol{k}}(t)$ is the Fourier transform of $n(\boldsymbol{r}, t)$.

Dissipation can be included by solving the Langevin equation for the center-of-mass, $\boldsymbol{X}(t)$, of the electrons,

$$M\frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{X}(t) = -\frac{M}{\tau}\frac{\mathrm{d}}{\mathrm{d}\tau}\boldsymbol{X}(t) + \boldsymbol{R}(t), \tag{6}$$

in a relaxation-time approximation [12,37]. In Eq. (6), M is the total mass of electrons, and τ is the relaxation time. For a given external condition, Eq. (6) is solved together with Eq. (1) many times with different time sequences of the random force, $\boldsymbol{R}(t)$. The random force satisfies,

$$\langle \boldsymbol{R}(t+t_0)\cdot\boldsymbol{R}(t_0)\rangle = \frac{6Mk_{\mathrm{B}}T}{\tau}\delta(t), \tag{7}$$

where T is the temperature, k_{B} is the Boltzmann constant, and the angular bracket denotes an ensemble average.

The solution of the ordinary differential equation, Eq. (3), together with the calculation of N-body interparticle interactions, $V_I(\{\boldsymbol{R}_I(t)\})$, constitute the MD simulation for which efficient algorithms have been designed on MIMD architectures [18–21].

The remaining QMD algorithms consist of the following two computational kernels:

(i) The solution of parabolic PDEs, i.e., the time-dependent Kohn–Sham equation, Eq. (1), for electrons and the time-dependent Schrödinger equation, Eq. (5), for phonons.

(ii) Calculation of the Hartree potential,

$$v_{\mathrm{H}}(\boldsymbol{r}, t) = \int \mathrm{d}\boldsymbol{r}'\,\frac{e^2 n(\boldsymbol{r}', t)}{|\boldsymbol{r}-\boldsymbol{r}'|}, \tag{8}$$

in Eq. (1), or equivalently, solution of the elliptic PDE (the Poisson equation),

$$\nabla^2 v_{\mathrm{H}}(\boldsymbol{r}, t) = -4\pi e^2 n(\boldsymbol{r}, t). \tag{9}$$

In the next sections, we discuss parallel algorithms for (i) and (ii).

3. Space-splitting Schrödinger solver

Let us consider an electron in two dimensions under the influence of an external potential $v_0(x, y)$ and a uniform magnetic field of strength B perpendicular to the two-dimensional plane. The Hamiltonian operator H is given by

$$\begin{aligned} H &= \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\frac{\partial}{\partial x} - \frac{m\omega_{\mathrm{c}}}{2}y\right)^2 + \frac{1}{2m}\left(\frac{\hbar}{\mathrm{i}}\frac{\partial}{\partial y} + \frac{m\omega_{\mathrm{c}}}{2}x\right)^2 + v_0(x, y) \\ &= T_x + T_y + V, \end{aligned} \tag{10}$$

where $\omega_{\mathrm{c}} = eB/mc$ is the cyclotron frequency. The time evolution of the wave function, $\Psi(x, y, t)$, follows from,

$$\mathrm{i}\hbar\frac{\partial}{\partial t}\Psi(x, y, t) = H\Psi(x, y, t). \tag{11}$$

In the split-operator method [24], the wave function is propagated for a small time interval, Δt, as

$$\begin{aligned} \Psi(x, y, t+\Delta t) = {} & \exp(-\mathrm{i}V\Delta t/2\hbar)\,\exp(-\mathrm{i}T_x\Delta t/2\hbar)\,\exp(-\mathrm{i}T_y\Delta t/\hbar) \\ & \times\exp(-\mathrm{i}T_x\Delta t/2\hbar)\,\exp(-\mathrm{i}V\Delta t/2\hbar)\,\Psi(x, y, t) + \mathscr{O}\left[(\Delta t)^3\right]. \end{aligned} \tag{12}$$

The spectral method (SM) is based on the fact that T_x and T_y become local operations when the wave function is Fourier transformed in the x and y directions, respectively. The SM algorithm involves repeated applications of the fast Fourier transform (FFT) to the wave function and multiplications by $\exp(-V\Delta t/2\hbar)$ in the real space, and $\exp(-T_x\Delta t/2\hbar)$ and $\exp(-T_y\Delta t/\hbar)$ in the Fourier space [24].

The SM is suitable when periodic boundary conditions are imposed on the wave function. Otherwise, the alternative-direction-implicit generalization of the Crank–Nicholson (CN) method is appropriate [25]. In the CN approach, Eq. (12) reads

$$\Psi(x, y, t+\Delta t) = \exp(-\mathrm{i}V\Delta t/2\hbar)\left(\frac{2}{1+\mathrm{i}T_x\Delta t/4\hbar}-1\right)\left(\frac{2}{1+\mathrm{i}T_y\Delta t/2\hbar}-1\right)$$
$$\times\left(\frac{2}{1+\mathrm{i}T_x\Delta t/4\hbar}-1\right)\exp(-\mathrm{i}V\Delta/2\hbar)\ \Psi(x, y, t)+\mathscr{O}\left[(\Delta t)^3\right]. \tag{13}$$

The operators $2/[1+\mathrm{i}T_x\Delta t/4\hbar]$ and $2/[1+\mathrm{i}T_y\Delta t/2\hbar]$ become a set of tridiagonal equations for the discretized wave function, and these equations are solved by Gaussian elimination and back substitution on sequential computers [25].

The wave function in the finite-difference scheme is dicretized as $\psi_{k,j}=\psi(j\Delta x, k\Delta y)$. In the SM, the propagation of $\psi_{k,j}$ requires $\mathscr{O}(M\log M)$ operations, where M is the number of grid points. In the CN method, the complexity is reduced to $\mathscr{O}(M)$ operations. On massively parallel machines with the number of processors comparable to the number of grid points, the parallel time complexity of the SM algorithm is $\mathscr{O}(\log M)$. One of the difficulties of parallel SM algorithms is the large communication overhead due to butterfly communication for the FFT [26], see Fig. 1b. Another difficulty is that the SM is applicable only when a wave function is discretized on a uniform rectangular grid. The CN method can be implemented on parallel computers using, e.g., the recursive doubling algorithm [27]. The resulting algorithm requires $\mathscr{O}(\log M)$ parallel operations. The parallel CN algorithm also requires large communication overhead associated with the communication pattern shown in Fig. 1c.

The space-splitting method (SSM) is free of these difficulties [28]. In the SSM approach, the operation of T_x on $\Psi_{k,j}$ is expressed as

$$T_x\Psi_{k,j} = b_k\Psi_{k,j-1}+2a_k\Psi_{k,j}+b_k^*\Psi_{k,j+1}, \tag{14}$$

where

$$a_k=\frac{\hbar}{2m\Delta x^2}+\frac{m\omega_c^2}{16}(k\Delta y)^2, \qquad b_k=-\frac{\hbar}{2m\Delta x^2}+\frac{\hbar\omega_c}{4\mathrm{i}\Delta x}k\Delta y. \tag{15}$$

For each index k, the operation of T_x on $\Psi_{k,j}$ amounts to a multiplication of a tridiagonal matrix. In the SSM, this tridiagonal matrix is expressed as a direct sum of 2×2 submatrices,

$$T_x=\begin{pmatrix} 2a & b^* & & & & & & \\ b & 2a & b^* & & & & & \\ & b & 2a & b^* & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & b & 2a & b^* & & \\ & & & & b & 2a & b^* & \\ & & & & & b & 2a & \end{pmatrix} = \frac{1}{2}\begin{pmatrix} a & b^* & & & & & & \\ b & a & & & & & & \\ & & a & b^* & & & & \\ & & b & a & & & & \\ & & & & \ddots & & & \\ & & & & & & a & b^* \\ & & & & & & b & a \end{pmatrix}$$

$$+\begin{pmatrix} a & & & & & & \\ & a & b^* & & & & \\ & b & a & & & & \\ & & & a & b^* & & \\ & & & b & a & & \\ & & & & & \ddots & \\ & & & & & & a \end{pmatrix} + \tfrac{1}{2}\begin{pmatrix} a & b^* & & & & & \\ b & a & & & & & \\ & & a & b^* & & & \\ & & b & a & & & \\ & & & & \ddots & & \\ & & & & & a & b^* \\ & & & & & b & a \end{pmatrix}, \tag{16}$$

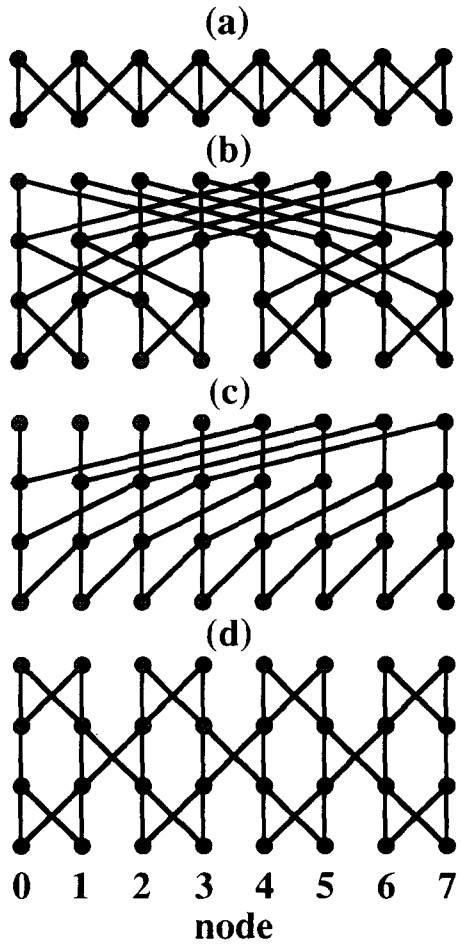

Fig. 1. (a) Communication pattern involved in a tridiagonal Hamiltonian operator. (b) Communication pattern involved in calculating the exponential of a tridiagonal Hamiltonian in the spectral method. (c) The same in the Crank–Nicholson method. (d) The same in the space-splitting method.

where we have omitted the index k. The exponential of a 2×2 matrix on the right-hand side of Eq. (16) is calculated analytically:

$$\exp\left(-\frac{\mathrm{i}\Delta t}{2\hbar}T_x\right)$$

$$= \begin{pmatrix} \epsilon_4^0 & \epsilon_4^+ & & & & & & \\ \epsilon_4^- & \epsilon_4^0 & & & & & & \\ & & \epsilon_4^0 & \epsilon_4^+ & & & & \\ & & \epsilon_4^- & \epsilon_4^0 & & & & \\ & & & & \ddots & & & \\ & & & & & \epsilon_4^0 & \epsilon_4^+ & \\ & & & & & \epsilon_4^- & \epsilon_4^0 & \end{pmatrix} \times \begin{pmatrix} \epsilon_2^0 & & & & & & & \\ & \epsilon_2^0 & \epsilon_2^+ & & & & & \\ & \epsilon_2^- & \epsilon_2^0 & & & & & \\ & & & \epsilon_2^0 & \epsilon_2^+ & & & \\ & & & \epsilon_2^- & \epsilon_2^0 & & & \\ & & & & & \ddots & & \\ & & & & & & & \epsilon_2^0 \end{pmatrix}$$

$$\times \begin{pmatrix} \epsilon_4^0 & \epsilon_4^+ & & & & & & \\ \epsilon_4^- & \epsilon_4^0 & & & & & & \\ & & \epsilon_4^0 & \epsilon_4^+ & & & & \\ & & \epsilon_4^- & \epsilon_4^0 & & & & \\ & & & & \ddots & & & \\ & & & & & \epsilon_4^0 & \epsilon_4^+ & \\ & & & & & \epsilon_4^- & \epsilon_4^0 & \end{pmatrix} + \mathcal{O}\left[(\Delta t)^3\right], \tag{17}$$

where

$$\epsilon_n^0 = \tfrac{1}{2}\left[\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a+|b|)\right)+\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a-|b|)\right)\right],$$

$$\epsilon_n^+ = \frac{b^*}{2|b|}\left[\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a+|b|)\right)-\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a-|b|)\right)\right],$$

$$\epsilon_n^- = \frac{b}{2|b|}\left[\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a+|b|)\right)-\exp\left(-\frac{\mathrm{i}\Delta t}{n\hbar}(a-|b|)\right)\right]. \tag{18}$$

The operation of $\exp(-\mathrm{i}T_y\Delta t/\hbar)$ is executed in a similar manner.

On the MasPar, processors are arranged in a two-dimensional mesh. If we assign each grid point to a processor in the mesh, all the operations in Eq. (17) are local, i.e., only nearest-neighbor communications are required. Consequently, the communication overhead of the SSM is much smaller than those of the SM and the CN. The communication pattern involved in Eq. (17) is shown in Fig. 1d. The complexity of the SSM on sequential machines is $\mathcal{O}(M)$, while on massively parallel architectures with the number of processors comparable to M, the parallel time complexity is $\mathcal{O}(1)$, i.e., the SSM is scalable.

4. Dynamical-simulated-annealing Poisson solver

The fast Poisson solver (FPS) is based on the fact that the Fourier transform of the Laplacian ∇^2 in Eq. (9) becomes local. This enables the solution of Eq. (9) with $\mathcal{O}(M)$ operations [29]. The FPS algorithm

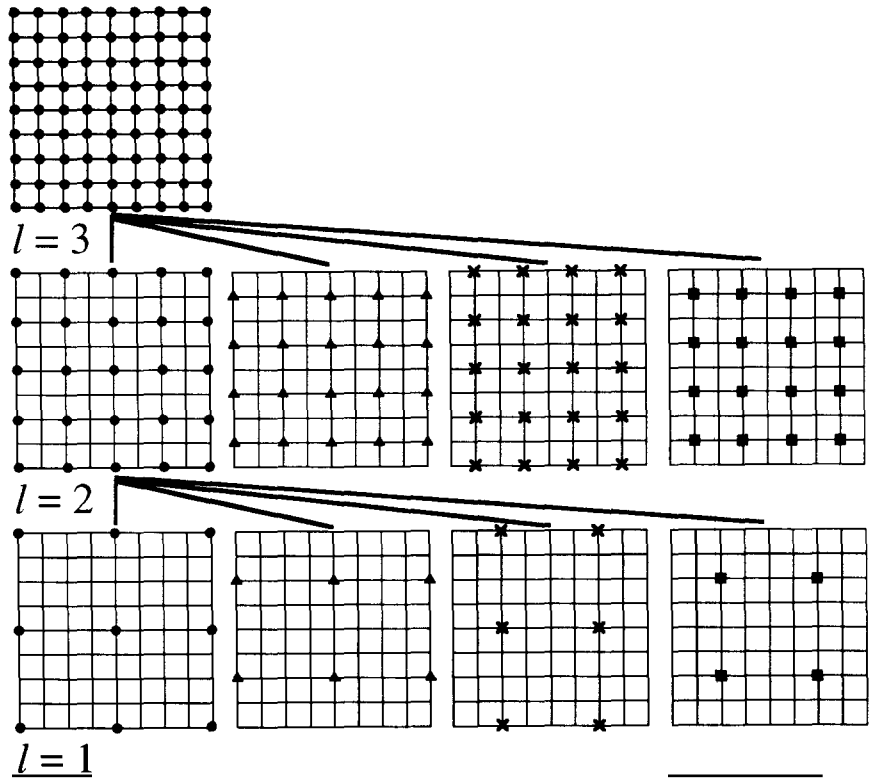

Fig. 2. Hierarchical multilevel grids in the multigrid method. The first column represents the conventional multigrid method where a grid at the $(l-1)$th level is generated by eliminating the even rows and columns from the finer grid at the lth level. In the parallel superconvergent multigrid method, a grid at the lth level produces four coarser grids at the $(l-1)$th level.

first calculates the Fourier transform n_k of $n(\boldsymbol{r})$ with $\mathcal{O}(M \log M)$ operations. Then Eq. (9) is solved in the Fourier space as $v_k^{(\mathrm{H})} = -4\pi e^2 n_k / k^2$, with $\mathcal{O}(M)$ operations, where $v_k^{(\mathrm{H})}$ is the Fourier transform of $v_{\mathrm{H}}(\boldsymbol{r})$. $v_{\mathrm{H}}(\boldsymbol{r})$ is obtained by taking the inverse Fourier transform of $v_k^{(\mathrm{H})}$. The complexity of the FPS is thus $\mathcal{O}(M \log M)$. A major drawback of the FPS is that it applies only to the finite-difference Poisson equation expressed on rectangular grid points with simple boundary conditions.

The multigrid method (MGM) revolutionized the solution of the Poisson equation, since it solves the Poisson equation with $\mathcal{O}(M)$ operations [30]. In Jacobi or Gauss–Seidel relaxation methods, short wave-length components of $v_{\mathrm{H}}(\boldsymbol{r})$ converge quickly to the correct value within a small number of iteration steps. However, the convergence of the long wave-length components is slow, and this limits the performance of relaxation methods. The MGM is based on the simple idea that slowly varying long-wavelength components can be accurately represented on a coarser mesh. By employing hierarchical meshes with coarser grid spacings, the MGM solves the Poisson equation with $\mathcal{O}(M)$ operations.

In the MGM, there is a set of multilevel meshes. The mesh at the lth level has a grid spacing $h_l = L/2^l (l = 1, 2, \ldots, l_{max})$, where L is the system's length. A mesh at the $(l-1)$th level is obtained by eliminating the even columns and rows of the finer mesh at the lth level, see Fig. 2. The implementation of the MGM on a processor array with the number of processors comparable to the number of grid points requires $\mathcal{O}(\log M)$ parallel operations. The algorithm is inefficient when it operates on coarser grids because most of the processors are inactive.

In the parallel superconvergent multigrid (PSMG) method [31], idle processors are used to accelerate the convergence of the conventional MGM method. In fact the mapping from a grid at the lth level to a coarser grid at the $(l-1)$th level is not unique. In two dimension, four coarse grids can be generated from a fine grid, see Fig. 2. At coarser levels, the finest grid is regarded as the union of a set of coarse grids. The solutions of the multiple coarse grid equations are combined to optimize the convergence rate.

The PSMG algorithm starts from an initial guess, v_{H}, for the solution of the Poisson equation on the mesh at the lth level. The exact solution, v_{H}^{*}, at the lth level satisfies the Poisson equation,

$$A^{(l)}v_{\mathrm{H}}^{*} = -4\pi e^{2}n, \tag{19}$$

in matrix notation. Here $A^{(l)}$ is the Laplacian at the lth level,

$$\left(A^{(l)}v^{(\mathrm{H})}\right)_{k,j} = \frac{1}{h_l^2}\left(v^{(\mathrm{H})}_{k,j+\delta_l} + v^{(\mathrm{H})}_{k,j-\delta_l} + v^{(\mathrm{H})}_{k+\delta_l,j} + v^{(\mathrm{H})}_{k-\delta_l,j} - 4v^{(\mathrm{H})}_{k,j}\right), \tag{20}$$

where $\delta_l = 2^{l_{\max}-l}$ is the stride at the lth level.

Eq. (19) is solved following steps (i)–(v):

(i) *Presmoothing*. First, a relaxation method is applied to converge the short wave-length components of v_{H}. We apply the smoothing operations iteratively to v_{H},

$$v_{\mathrm{H}} \leftarrow [1 + Z^{(l)}A^{(l)}]v_{\mathrm{H}} + 4\pi e^{2}Z^{(l)}n. \tag{21}$$

Different choices for the iteration matrix $Z^{(l)}$ define different relaxation methods. The number of iterations applied for the presmoothing is denoted by γ_1.

(ii) *The residual equation*. After presmoothing, the residue, r, is calculated from

$$r = A^{(l)}v_{\mathrm{H}} + 4\pi e^{2}n. \tag{22}$$

The error which is defined as $e = v_{\mathrm{H}}^{*} - v_{\mathrm{H}}$ satisfies the residual equation,

$$A^{(l)}e = -r. \tag{23}$$

Since the presmoothing has already removed the short wave-length components of the error, the error is accurately evaluated on the coarser grid, i.e., by solving Eq. (23) with l replaced by $l-1$.

(iii) *Interpolation*. Suppose that Eq. (23) at the $(l-1)$th level has been solved by some method. Then the error, e, should be interpolated to give the correction to the approximate solution, v_{H}, at the lth level. We denote the interpolation operator at the $(l-1)$th level as $Q^{(l-1)}$ so that the interpolation of e is expressed as

$$e \leftarrow Q^{(l-1)}e. \tag{24}$$

(iv) *Correction*. The solution v_{H} at the lth level is corrected by adding the interpolated error as

$$v_{\mathrm{H}} \leftarrow v_{\mathrm{H}} + e. \tag{25}$$

(v) *Post-smoothing*. Now that the long wave-length components have been corrected by Eq. (25), the short wave-length components should be smoothed again by applying Eq. (21) repeatedly. The number of the post-smoothing iterations is denoted by γ_2.

The residual equation, Eq. (23), at the $(l-1)$th level is solved by applying the above procedure (i)–(v) with v_{H} and $4\pi e^{2}n$ replaced by e and r, at the coarser level, i.e., with l replaced by $l-1$. The solution at the $(l-1)$th level then requires the solution of Eq. (23) at the $(l-2)$th level. The procedure is continued recursively until we reach the coarsest residual equation. At the coarsest level $l=1$, Eq. (23)

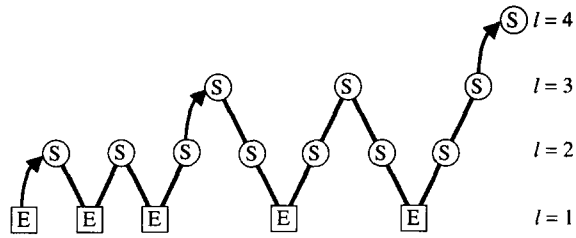

Fig. 3. Schematic representation of the full multigrid method for $l_{max} = 4$. E represents the exact solution at the coarsest level. S denotes the smoothing operation.

can be solved exactly, and the recursive procedure is terminated. The recursion may be repeated to get a more accurate result. The number of such iterations is denoted by ν.

In the full multigrid method, instead of starting from an arbitrary initial guess for v_H, we first start from the exact solution to the Poisson equation at the coarsest level. The solution is then used as an initial guess for the Poisson equation at the second level. The recursive procedure, (i)–(v), is applied ν times to obtain a converged solution at this level. The solution at the second level is used as an input for the Poisson equation at the third level. This procedure is continued until the solution at the finest level $l = l_{max}$ is obtained, see Fig. 3. The interpolation matrix $Q^{(l)}$ and the iteration matrix $Z^{(l)}$ are determined to achieve the best convergence rate in Ref. [31].

The PSMG method is implemented on the MasPar. Each grid point is assigned to a processor in the two-dimensional processor array. Interprocessor communications are implemented with the X-Net communication constructions with stride δ_l in the MPL (MasPar Parallel Application Language) which is a parallel extension of the C language [38].

As shown above, an elliptic PDE such as the Poisson equation is solved with a parallel time complexity $\mathcal{O}(\log M)$. In fact the Poisson equation, Eq. (9), is an approximation for a hyperbolic PDE,

$$\left(\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - \nabla^2\right) v_H(\boldsymbol{r}, t) = 4\pi e^2 n(\boldsymbol{r}, t), \tag{26}$$

which may be derived from Maxwell's equations by employing the Lorentz gauge. Contrary to Eq. (9) which is a Cauchy problem, Eq. (26) is an initial value problem. The complexity of an initial value problem is smaller than that of Cauchy problems. For example, as explained below an algorithm with $\mathcal{O}(1)$ parallel operations can be used to solve Eq. (26) instead of the $\mathcal{O}(\log M)$ parallel operations of Eq. (9). The reason Eq. (26) is not used to determine $v_H(\boldsymbol{r}, t)$ is that the characteristic time scale, $\tau_H = \hbar^2/me^2c = 2 \times 10^{-19}$ s, is much shorter than the characteristic time for electronic motion, $\tau_e = \hbar^3/me^4 = 2 \times 10^{-17}$ s. The two orders-of-magnitude difference in time scales makes it difficult to solve Eqs. (1) and (26) concurrently.

This situation is reminiscent of coupled equations for electrons and ions. For ionic motion, the time step in MD simulations is $\tau_I \sim 10^{-15}$ s, which is two orders-of-magnitude larger than τ_e. Dynamic simulations of coupled electron-ion systems are therefore prohibitively time consuming. In 1985, Car and Parrinello introduced the dynamical-simulated-annealing (DSA) method to overcome this problem [39]. Proposing fictitious dynamics for electrons, Car and Parrinello's approach solves the problem much faster than the original electronic structure problem. Correct statistical information is retained in the new coupled dynamical system.

The DSA method can be applied to the coupled equations, Eqs. (1) and (9). Instead of solving Eq. (9), which assumes an adiabatic approximation for the electromagnetic field, we can introduce its dynamics through Eq. (26). As long as the Hartree potential follows the electron motion adiabatically, we can

replace the speed of light c in Eq. (26) by a much smaller fictitious velocity $\tilde{c}$. In this way the time step to integrate Eq. (26) can be made longer which speeds up the program. The basic idea of the DSA is to replace the Cauchy problem by an initial value problem with less complexity.

A similar DSA scheme, or the auxiliary field method, was proposed by Car and Parrinello in 1987 in the context of the first-principles MD [32]. They replace the factor c^{-2} in Eq. (26) by a negative fictitious mass, $-\mu_{\mathrm{H}}$, associated with the Hartree potential. Their derivation of the scheme is based on a variational principle where an energy functional,

$$E[\{\Psi_i(\boldsymbol{r})\}, v_{\mathrm{H}}(\boldsymbol{r})] = \sum_{i=1}^{N_e} \int \mathrm{d}\boldsymbol{r}\ \Psi_i^*(\boldsymbol{r}) \left(-\frac{\hbar^2}{2m}\nabla^2\right)\Psi_i(\boldsymbol{r}) + \int \mathrm{d}\boldsymbol{r}\, n(\boldsymbol{r})\ v_0(\boldsymbol{r}) + E_{\mathrm{xc}}[n(\boldsymbol{r})]$$
$$+ \frac{1}{8\pi e^2}\int \mathrm{d}\boldsymbol{r}\, v_{\mathrm{H}}(\boldsymbol{r})\nabla^2 v_{\mathrm{H}}(\boldsymbol{r}) + \int \mathrm{d}\boldsymbol{r}\, n(\boldsymbol{r})\ v_{\mathrm{H}}(\boldsymbol{r}), \tag{27}$$

is minimized with respect to both the electron wave functions, $\{\Psi_i(\boldsymbol{r})\}$, and the Hartree potential, $v_{\mathrm{H}}(\boldsymbol{r})$, where $E_{\mathrm{xc}}[n(\boldsymbol{r})]$ is the exchange-correlation energy functional [32]. We have found that their functional, in fact, has a maximum with respect to the Hartree potential and a minimum with respect to electron wave functions. By expanding Eq. (27) around the correct Hartree potential which satisfies the Poisson equation, Eq. (9), we obtain

$$E[\{\Psi_i(\boldsymbol{r})\}, v_{\mathrm{H}}(\boldsymbol{r}) + \delta v_{\mathrm{H}}(\boldsymbol{r})] - E[\{\Psi_i(\boldsymbol{r})\}, v_{\mathrm{H}}(\boldsymbol{r})] = -\frac{1}{8\pi e^2}\int \mathrm{d}\boldsymbol{r}\, |\nabla \delta v_{\mathrm{H}}(\boldsymbol{r})|^2 < 0. \tag{28}$$

Therefore, their dynamical equation for $v_{\mathrm{H}}(\boldsymbol{r})$, which minimizes the energy functional by quenching, must be replaced by one which maximizes it. This is readily achieved by replacing the negative fictitious mass by a positive one. In that case, the resulting equation is identical to Eq. (26) with c^{-2} replaced by a fictitious speed factor $c^{-2} = \mu_{\mathrm{H}}$.

Let us consider a two dimensional system in the finite difference scheme. The Hartree potential and electron density are discretized as $v_{k,j}^{(\mathrm{H})} = v_{\mathrm{H}}(j\Delta, k\Delta)$ and $n_{k,j} = n(j\Delta, k\Delta)$ on a mesh with grid spacing Δ. The standard finite-difference representation of the second order spatial derivative operator in Eq. (26) leads to the equation

$$\left(\frac{\Delta}{\tilde{c}}\right)^2 \frac{\partial^2}{\partial t^2} v_{k,j}^{(\mathrm{H})}(t) = v_{k,j-1}^{(\mathrm{H})}(t) + v_{k,j+1}^{(\mathrm{H})}(t) + v_{k-1,j}^{(\mathrm{H})}(t) + v_{k+1,j}^{(\mathrm{H})}(t) - 4v_{k,j}^{(\mathrm{H})}(t) + 4\pi e^2 \Delta^2 n_{k,j}(t), \tag{29}$$

where the fictitious velocity $\tilde{c}$ has been introduced. Eq. (29) is similar to molecular dynamics with interactions only among the nearest-neighbor grid points. The time integration of the ordinary differential equation in Eq. (29) can be performed by one of the many available integration schemes [40]. In this paper, we employ the fourth-order predictor-corrector method of Gear [40,41].

The equivalence of Eq. (29) to MD may lead to another application of the QMD method. By introducing a heat bath coupled to the Hartree potential in Eq. (29) and using the isothermal MD method [42], the effect of environment on electron transport can be studied.

We have implemented the DSA Poisson solver on an 8192-node MasPar SIMD machine. In the MasPar, 8192 processors are arranged in a two-dimensional rectangular array of size 128×64. We use 128×64 grid points and assign $v_{k,j}^{(\mathrm{H})}(t)$ and $n_{k,j}(t)$ at each grid point to a processor in the array. The right hand side of Eq. (29) includes the values of the Hartree potential at the nearest-neighbor grid points. These values are fetched by the fast X-Net communication constructions in MPL [38]. After evaluating the right-hand side of Eq. (29), $v^{(\mathrm{H})}(t)$ is integrated locally within a processor by the predictor-corrector method.

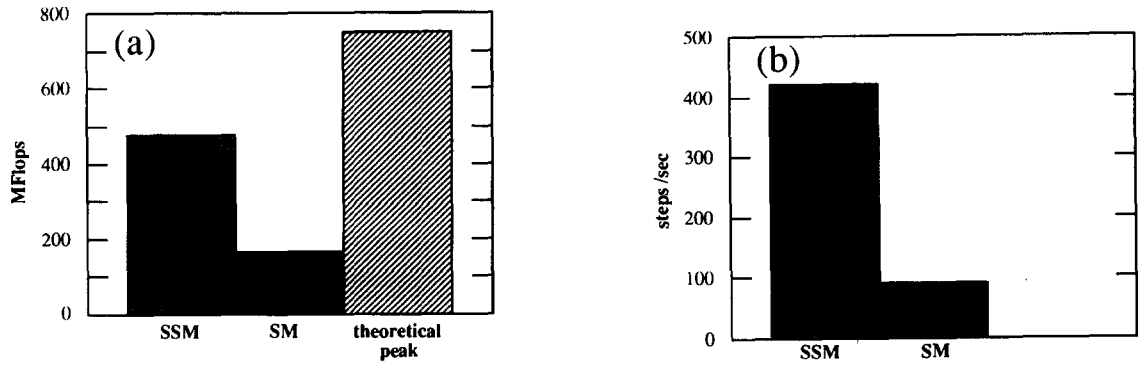

Fig. 4. (a) Performance in million floating point operations per second (MFlops) of the space-splitting method (SSM) and the spectral method (SM) for the solution of the time-dependent Schrödinger equation for an electron in two dimensions under a magnetic field. The calculation employs 128×64 grid points on an 8192-node MasPar. Measured performances (solid) in Mflops are compared with the theoretical peak speed (hatched) of the MasPar. (b) Performance of the SSM and SM for the same problem in time steps per second.

The DSA for the Hartree potential as well as the SSM for the Schrödinger equation are scalable algorithms with $\mathcal{O}(1)$ parallel time complexity. This will have a significant impact on a broad range of applications such as conventional device simulations, computational fluid dynamics, and computational nanoelectronics.

5. Performance of parallel QMD algorithms on MasPar

In this Section, we discuss the performances of the parallel QMD algorithms described in Sections 3 and 4, on an 8, 192-node MasPar 1208B.

5.1. Parallel Schrödinger solvers

Fig. 4a shows the performance in million floating point operations per second (MFlops) of the SSM and SM for the solution of the time-dependent Schrödinger equation of an electron in two dimensions under a magnetic field. The number of grid points is taken to be 128×64. The SSM achieves 479 MFlops (64% of the theoretical peak speed, 750 MFlops), while the SM achieves 22.3% of the theoretical peak. Because the SSM involves only nearest-neighbor communications it executes 2.9 times faster than the SM.

In our example, the SSM requires $144M$ floating point operations where M is the number of grid points. On the other hand, the SM requires $(16 \log_2 M + 18)M$ operations. Asymptotically, the SSM requires less floating point operations. At $M = 8{,}192$, the number of operations in the SSM is already smaller than that for the SM. Combined with the smaller communication overhead, it makes the SSM much faster than the SM. Fig. 4b shows the performance of the SSM and SM for the same problem in time steps per second. The SSM is 4.7 times faster than the SM and the performance of the SSM becomes better for larger problems.

We have tested the SSM Schrödinger solver on various other computers including (1) an IBM 3090/600J vector facility (14.5 ns, 6 processors), (2) an IBM RISC System/6000-560 (50 MHz), and (3) a Silicon Graphics (SGI) 4D/380 (33 MHz, 8 processors) shared-memory MIMD machine. For all the

Table 1
Performance of the space-splitting Schrödinger solver and the dynamical-simulated-annealing Poisson solver

Computer	Compiler	SS Schrödinger solver (MFlops)	DSA Poisson solver (MFlops)	Theoretical Peak (MFlops)
MasPar 1208B (8,192 proc)	MPL 3.0.28	475.4	459.8	750
IBM 3090/600J (14.5 ns, 1 proc out of 6)	VS Fortran 2.4 (opt(3), vector)	58.7	37.5	138
SGI 4D/380 (33 MHz, 8 proc)	f77-pfa-O3	58.4	21.7	106
IBM RISC Sys/6000-560 (50 MHz)	f77-O	54.6	19.6	100

computers except the MasPar, the same program written in Fortran is run using compiler options listed in Table 1. On the MasPar, the program is written in the MPL language and no machine-specific optimization is applied except that register variables are explicitly declared. On the IBM 3090, only one processor is used, and all the inner do loops are vectorized. On the SGI, all the major loops are executed in parallel on the eight processors.

The results of numerical experiments are summarized in Table 1. The SSM algorithm runs efficiently on many architectures, particularly on SIMD machines. On the shared-memory MIMD architecture, SGI 4D/380 with 8 processors, we achieve a speedup of 6.6.

5.2. *Parallel poisson solvers*

Fig. 5a compares the performance in MFlops of various Poisson solvers on the MasPar. We consider (1) the dynamical-simulated-annealing (DSA) method, (2) a fast Poisson solver (FPS), and (3) a parallel superconvergent multigrid (PSMG) method. For the DSA and FPS, the results are obtained with 128×64 grid points. The fourth-order Gear predictor-corrector method is used for the DSA algorithm. For the PSMG, the result on 63×63 nodes are extrapolated to 8,192 nodes. The PSMG parameters are

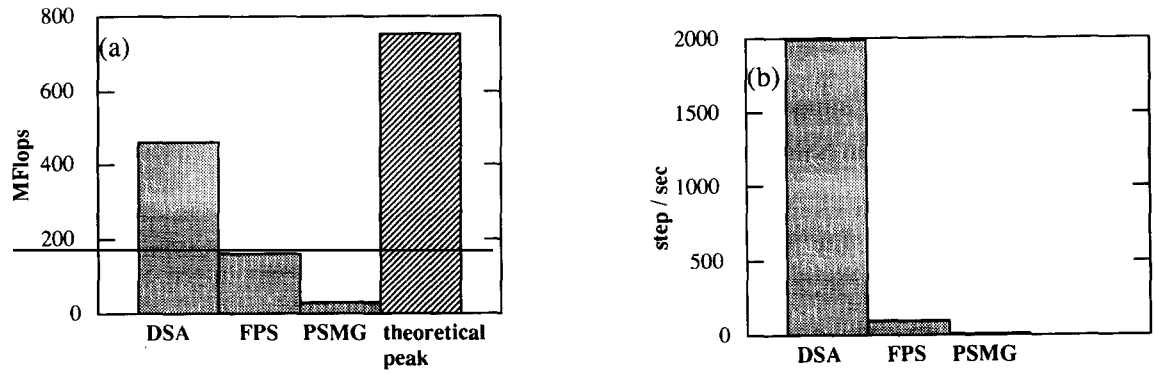

Fig. 5. (a) Performances in MFlops of the (1) dynamical-simulated-annealing (DSA) method, (2) fast Poisson solver (FPS), and (3) parallel superconvergent multigrid (PSMG) method on an 8,192-node MasPar compared with the theoretical peak speed of the MasPar. For the DSA and FPS, the results are for the Poisson equation on 128×64 grid points. For the PSMG, the result is calculated on 63×63 processors and are extrapolated to 8192 nodes. (b) Speed of the (1) DSA, (2) FPS, and (3) PSMG Poisson solvers for the same problem in steps executed in a second on the MasPar.

chosen to be $\gamma_1 = \gamma_2 = \nu = 1$. The DSA runs at 460 MFlops (61.3% of the theoretical peak, 750 MFlops), while the FPS and the PSMG run at 160 MFlops (21.4% of the theoretical peak) and 63 MFlops (8.4%), respectively. The high speed of the DSA is due to small communication overhead. When the grid geometry and appropriate boundary conditions are used, the FPS is faster than the PSMG. The latter, however, is applicable to a broader class of problems. The DSA is applicable regardless of the grid geometry and boundary conditions but only for dynamical problems.

Fig. 5b shows the speed of various Poisson solvers in steps executed per second. Here 22.2 DSA steps are executed in the same time as one FPS step. Suppose that in a QMD program with the DSA Poisson solver, we execute $n_H = 22$ time steps of the temporal evolution of the Hartree potential between two successive steps for the time-dependent Kohn–Sham equations. Then we get the same speed as that of the QMD program with the FPS Poisson solver. By choosing the fictitious speed $\tilde{c}$ in Eq. (29) properly, the time step for the Hartree potential which is $1/22$ of the time step for electrons provides sufficient accuracy to describe the dynamics of many-electron systems. For larger systems, the DSA algorithm, needing only nearest-neighbor communications, becomes much faster than the FPS algorithm because the maximum communication distance in the FPS grows as $\mathcal{O}(M^{1/2})$. Furthermore, the execution time of the FPS algorithm grows as $\mathcal{O}(\log M)$, while the execution time for the DSA remains the same. Therefore, for large-scale simulations, the DSA Poisson solver is superior to the FPS. For systems with nonuniform grid points, where the FPS is not applicable, the DSA is already faster than the PSMG method at $M = 8192$. (With the choice of $n_H = 22$, the DSA is 8 times faster than the PSMG.)

Table 1 summarizes the performance of the DSA Poisson solver on various computers. The SIMD architecture exhibits the highest percentage of the theoretical peak speed. The DSA Poisson solver runs efficiently on many architectures. On the 8 processor SGI 4D/380, the speedup is 8.6. The superlinear speedup is possible partly because we do not optimize the parallel and sequential programs separately; the sequential program is obtained simply by setting the environmental parameter for the number of processors to be one. Another reason for the superlinear speedup is that with more processors the program can utilize more resources such as fast cache memories attached to processors.

6. Conclusion

We have implemented the quantum molecular dynamics method to study nanodevices on an 8,192-node SIMD architecture from MasPar. The space splitting Schrödinger solver and the dynamical-simulated-annealing Poisson solver run efficiently on the architecture. Both programs are scalable with the parallel time complexity $\mathcal{O}(1)$, and local, i.e., they require only nearest-neighbor communications. Both programs achieve more than 60% of the theoretical peak speed of the MasPar. The parallel quantum molecular dynamics codes have been used on the MasPar to study electron transport in various nanodevices [12,15].

Acknowledgement

This work was supported by the U.S. Department of Energy, Office of Energy Research, Basic Energy Science, Materials Science Division, Grant No. DE-FG05-92ER45477. The computations were performed using the computing facilities in the Concurrent Computing Laboratory for Materials Simulations (CCLMS) at Louisiana State University. The facilities in the CCLMS were acquired with the Equipment Enhancement Grants awarded by the Louisiana Board of Regents through Louisiana Education Quality Support Fund (LEQSF).

References

[1] F. Capasso, K. Mohammed and A.Y. Cho, IEEE J. Quant. Electron. 22 (1986) 1853.
[2] S. Datta, Superlat. Microstruct. 6 (1989) 83.
[3] M. Dyakonov and M. Shur, Phys. Rev. Lett. 71 (1993) 2465.
[4] V.J. Goldman, D.C. Tsui and J.E. Cunningham, Phys. Rev. Lett. 58 (1987) 1256.
[5] K.K. Gullapalli, A.J. Tsao and D.P. Neikirk, Appl. Phys. Lett. 62 (1993) 2856;
K.K. Gullapalli, D.R. Miller and D.P. Neikirk, Phys. Rev. B 49 (1994) 2622.
[6] K.K. Likharev, IBM J. Res. Develop. 32 (1988) 144.
[7] M.A. Stroscio, K.W. Kim G.J. Iafrate, M. Dutta, and H.L. Grubin, Philos. Mag. Lett. 65 (1992) 173;
A. Madhukar, P.D. Lao, W.C. Tang, M. Aidan and F. Voillot, Phys. Rev. Lett. 59 (1987) 1313.
[8] K. Hess, J.P. Leburton and U. Ravaioli, Computational Electronics (Kluwer, Boston, 1991).
[9] A. Selloni, P. Carnevali, R. Car and M. Parrinello, Phys. Rev. Lett. 59 (1987) 823.
[10] R.N. Barnett, U. Landman and A. Nitzan, J. Chem. Phys. 89 (1988) 2242.
[11] R.K. Kalia, P. Vashishta, L.H. Yang, F.W. Dech and J. Rowlan, Int. J. Supercomput. Appl. 4 (1990) 22.
[12] A. Nakano, R.K. Kalia and P. Vashishta, Appl. Phys. Lett. 64 (1994) 2569.
[13] A. Nakano, P. Vashishta and R.K. Kalia, Phys. Rev. B 43 (1991) 9066.
[14] A. Nakano, R.K. Kalia and P. Vashishta, Phys. Rev. B 44 (1991) 8121.
[15] A. Nakano, R.K. Kalia and P. Vashishta, Appl. Phys. Lett. 62 (1993) 3470.
[16] A. Nakano, P. Vashishta, R.K. Kalia and L.H. Yang, Phys. Rev. B 45 (1992) 8363.
[17] G.C. Fox, in: The Third Conference on Hypercube Concurrent Computers and Applications, Vol. 2, G.C. Fox, ed. (ACM Press, New York, 1988) p. 897.
[18] R.K. Kalia, S.W. de Leeuw, A. Nakano, D.L. Greenweil and P. Vashishta, Supercomputer 54(X-2) (1993) 11.
[19] R.K. Kalia, S.W. de Leeuw, A. Nakano and P. Vashishta, Comput. Phys. Commun. 74 (1993) 316.
[20] A. Nakano, P. Vashishta and R.K. Kalia, Comput. Phys. Commun. 77 (1993) 303.
[21] A. Nakano, R.K. Kalia and P. Vashishta, Comput. Phys. Commun., in press.
[22] L. Greengard and V. Rokhlin, J. Comput. Phys. 73 (1987) 325.
[23] H.-Q. Ding, N. Karasawa and W.A. Goddard, Chem. Phys. Lett. 196 (1992) 6.
[24] M.D. Feit, J.A. Fleck and A. Steiger, J. Comput. Phys. 47 (1982) 412.
[25] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes in Fortran, 2nd ed. (Cambridge University Press, New York, 1992).
[26] J.W. Cooley and J.W. Tukey, Math. Comput. 19 (1965) 297.
[27] R.W. Hockney and C.R. Jesshope, Parallel Computers 2 (Adam Hilgar, Philadelphia, 1988).
[28] J.L. Richardson, Comput. Phys. Commun. 63 (1991) 84.
H. de Raedt, Comput. Phys. Rep. 7 (1987) 1.
[29] R.W. Hockney and J.W. Eastwood, Computer Simulation Using Particles (Adam Hilger, New York, 1988).
[30] A. Brandt, Math. Comput. 31 (1977) 333.
[31] P.O. Frederickson and O.A. McBryan, in: Multigrid Methods, S.F. McCormick, ed. (Marcel Dekker, New York, 1988) p. 195.
[32] R. Car and M. Parrinello, Solid State Commun. 62 (1987) 403.
[33] E.K.U. Gross and W. Kohn, in: Advances in Quantum Chemistry, Vol. 21, S.B. Trickey, ed. (Academic Press, San Diego, 1990) p. 255.
[34] G. Vignale and M. Rasolt, Phys. Rev. Lett. 59 (1987) 2360.
[35] L.F. Register and K. Hess, Phys. Rev. B 49 (1994) 1900.
[36] B. Jackson, Comput. Phys. Commun. 63 (1991) 154.
[37] M. Grilli and E. Tosatti, Phys. Rev. Lett. 62 (1989) 2889.
[38] MPL Reference Manual, Technical Report 9302-0001, MasPar Computer Corporation (1992).
[39] R. Car and M. Parrinello, Phys. Rev. Lett. 55 (1985) 2471.
[40] M.P. Allen and D.J. Tildesley, Computer Simulation of Liquids (Clarendon, Oxford, 1987).
[41] C.W. Gear, Numerical Initial Value Problem in Ordinary Differential Equations (Prentice-Hall, Englewood Cliffs, NJ, 1971).
[42] S. Nosé, Mol. Phys. 52 (1984) 255.