npj computational materials

www.nature.com/npjcompumats

ARTICLE OPEN

Check for updates

Accurate and efficient molecular dynamics based on machine learning and non von Neumann architecture

Pinghui Mo[1], Chang Li[1], Dan Zhao[1], Yujia Zhang[1], Mengchao Shi[1], Junhua Li[1] and Jie Liu[1,2]✉

Force field-based classical molecular dynamics (CMD) is efficient but its potential energy surface (PES) prediction error can be very large. Density functional theory (DFT)-based ab-initio molecular dynamics (AIMD) is accurate but computational cost limits its applications to small systems. Here, we propose a molecular dynamics (MD) methodology which can simultaneously achieve both AIMD-level high accuracy and CMD-level high efficiency. The high accuracy is achieved by exploiting deep neural network (DNN)'s arbitrarily-high precision to fit PES. The high efficiency is achieved by deploying multiplication-less DNN on a carefully-optimized special-purpose non von Neumann (NvN) computer to mitigate the performance-limiting data shuttling (i.e., 'memory wall bottleneck'). By testing on different molecules and bulk systems, we show that the proposed MD methodology is generally-applicable to various MD tasks. The proposed MD methodology has been deployed on an in-house computing server based on reconfigurable field programmable gate array (FPGA), which is freely available at http://nvnmd.picp.vip.

npj Computational Materials (2022)8:107 ; https://doi.org/10.1038/s41524-022-00773-z

INTRODUCTION

As a cornerstone of atomistic-scale analysis, molecular dynamics (MD) is widely used in many fields, such as physics[1,2], chemistry[3,4], biology[5], materials[6,7], nanotechnology[8,9], drug design[10,11], earth science[12,13], semiconductor integrated circuit[14,15], and so on. Despite its importance, it is well-known that MD simulations suffer from a long-standing dilemma between accuracy and efficiency[16–21]. On one hand, ab-initio MD (AIMD), which is based on the first-principles density functional theory (DFT) evaluation of potential energy surface (PES), is accurate but not efficient enough to simulate large systems[16–18]. On the other hand, classical MD (CMD), which is based on artificially-crafted force fields (FF) approximation of PES, is efficient but not accurate enough in some applications[19–24].

In recent years, this dilemma is mitigated, to some extent, by the machine-learning (ML) MD (MLMD)[25–31]. By evaluating PES using ML models, the efficiency of MLMD is significantly superior than that of AIMD, while keeping the AIMD-level high accuracy. Unfortunately, though several orders of magnitude faster than AIMD, the state-of-the-art MLMD is still about two orders of magnitude slower than CMD[27,31,32]. Until now, it is still an outstanding problem to develop an MD simulator that can simultaneously achieve AIMD-level high accuracy and CMD-level high efficiency.

It is worth noting that, MD simulations are predominantly deployed on general-purpose von-Neumann (vN) computers, where the data processing hardware (e.g., central processing unit (CPU) and graphics processing unit (GPU)) and the data storage hardware (e.g., dynamic random-access memory (DRAM)) are separate hardware components. It is well known that the vN computers suffer from severe vN bottleneck (vNB)—the majority (e.g., over 90%) of computing time and energy must be spent to repeatedly shuttle data back-and-forth between the data processing hardware and the data storage hardware[33–35]. Consequently, only a very small fraction of calculation time and energy consumption is used to perform the useful arithmetic and logic operations, leading to the overall low efficiency of vN computers[33–35].

The severity of vNB depends on the characteristics of calculation—the more repeated data shuttling, the more performance-limiting vNB becomes. Given the typical MD duration (e.g., $t_{MD} \approx 10^{-9}$–10^{-3} s) and timestep (e.g., $\Delta t \approx 10^{-15}$ s), the atomistic data (e.g., positions, velocities, forces, and atomic neighbor data, etc.) must be shuttled repeatedly by a large number (e.g., $n_{MD} \approx t_{MD}/\Delta t = 10^{6}$–$10^{12}$) of times. Furthermore, in each MD timestep, a huge number of additional data shuttling is required to accomplish each PES evaluation[36]. Hindered by such nested-loop heavy-duty data shuttling, both the time efficiency and the energy efficiency of MD calculations are extremely low on general-purpose vN computers[23].

However, since the invention of the first general-purpose electronic computer in the 1940s, general-purpose vN architecture has been the dominating paradigm of the mainstream computers like laptops, desktops, and supercomputers for over 7 decades[37–39]. Researchers nowadays widely use vN computers to run MD, largely because they have no other choice. Though some special-purpose MD computers have been developed[22,23,40,41], they are all based on CMD and FF, whose accuracy is questionable in many important applications[42–46]. Therefore, considering the scientific and technological significance of MD[1–47], it deserves serious efforts to develop a special-purpose MD computer beyond the vN paradigm, to enable efficient and accurate MD calculations in various fields.

In order to approach this goal, in this paper, we propose a paradigm shift from the established vN architecture to a non vN (NvN) architecture. By leveraging the technologies in MLMD algorithms[26,27,30], artificial intelligence[48,49], and NvN architecture[50,51], the proposed special-purpose MD computer can simultaneously achieve both the AIMD-level high accuracy and the CMD-level high efficiency. This is achieved by deploying a deeply-revised MLMD algorithm, i.e., DeePMD[26–31], (to ensure high accuracy) on a carefully-optimized NvN hardware (to ensure high efficiency). In the Section Results, the calculation accuracy and

[1]College of Electrical and Information Engineering, Hunan University, Changsha, Hunan, PR China. [2]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, USA. ✉email: jie_liu@hnu.edu.cn

calculation efficiency are quantitatively analyzed. In the Section Discussion, a discussion is briefly made. In the Section Methods, the overall system design of the proposed special-purpose MD computer is introduced and the implementation details of the NvN architecture are presented.

RESULTS

The performance of the proposed special-purpose non von Neumann molecular dynamics (NVNMD) computer (see Methods section for more design and implementation details) is quantitatively analyzed in this section. First, the analysis procedure is introduced (Section Analysis procedure). Then, the calculation accuracy (Section Calculation accuracy), the calculation time efficiency (Section Time efficiency), and the calculation energy efficiency (Section Energy efficiency) are analyzed quantitatively.

Analysis procedure

Any user can follow two consecutive steps to run MD on the proposed NVNMD computer, which has been released online[52]: (i) to train a machine learning (ML) model that can decently reproduce the PES[25–31]; and (ii) to deploy the trained ML model on the proposed NVNMD computer, then run MD there to obtain the atomistic trajectories.

ML training (i.e., step (i)) is performed on traditional vN architecture computers (e.g., CPU/GPU) by using the training codes we open-sourced online[53], which are programmed purposefully based on TensorFlow[54] to help users train ML models that are compatible with the unique NvN computer proposed here. To accomplish step (i), the training samples should be prepared first. This can be done by using either the active learning tools[26,29,30], or the brute-force (i.e., less efficient) DFT-based AIMD sampling[55,56]. Then, these training samples are used as inputs of our training codes[53], which output the ML models. Our training procedure is comparatively speaking more complicated than that of the established MLMD[26,27]—it consists of not only the continuous neural network (CNN) training of the established MLMD, but also an additional step of quantized neural network (QNN) training (Section Quantized neural network) which uses CNN results as inputs. Typically, the CNN training uses a large number of training steps (e.g., 1×10^6) with a high learning rate (e.g., 2×10^{-2}); and the subsequent QNN training uses a small number of training steps (e.g., 1×10^4) and a low learning rate (e.g., 2×10^{-7}), as it only needs to minimize the small error induced by quantization from CNN to QNN.

ML inference (i.e., step (ii)) is performed on the proposed NvN architecture computer, after uploading the QNN ML model to our online NVNMD system[52]. In the online NVNMD system, all MD settings and parameters (e.g., timestep, microcanonical/canonical/isothermal–isobaric ensemble, thermostats, etc.) are controlled by using the same input file interface of the LAMMPS package[57], except that the force field is replaced by using the uploaded QNN ML model.

Six systems are used to run testing MD calculations, including three molecule systems (i.e., benzene, naphthalene, and aspirin) and three bulk systems (i.e., Sb, GeTe, and $Li_{10}GeP_2S_{12}$). The training data of molecule systems are from MD17 dataset[58–60]; and those of bulk systems (i.e., Sb, GeTe, and $Li_{10}GeP_2S_{12}$) come from Ref. [61], Ref. [56], and Ref. [62], respectively. The result of test accuracy, speed and energy efficiency are shown in Section Calculation accuracy, Section Time efficiency, and Section Energy efficiency, respectively.

Calculation accuracy

The high accuracy of the proposed NVNMD can be seen obviously in Table 1. The root mean square errors (RMSE) of PES fitting of benzene, naphthalene, aspirin, Sb, GeTe, and $Li_{10}GeP_2S_{12}$ systems are 0.19, 0.39, 0.32, 0.14, 0.09, and 0.14 kcal mol^{-1}, respectively. These are close to the established MLMD values in literature (Table 1), and well below the chemical accuracy threshold (1.0 kcal mol^{-1})[63,64], indicating decent accuracy of the proposed NVNMD. As a direct comparison, in Table 1, we also collected some of the energy prediction errors of the MLMD and CMD from the existing literature, after removing the obvious outliers (e.g., $|\Delta E_i| = 2.7$ kcal mol^{-1} for methylamine, and $|\Delta E_i| = 1.4$ kcal mol^{-1} for aqueous LiF pair[65–67]). It is obvious that, while CMD suffers from large PES prediction error, the proposed NVNMD has decent PES prediction accuracy, which is inherited from the highly accurate MLMD.

As shown in Table 1, the μ_e of MLMD is about 10^{-2}–10^{-1} kcal mol^{-1} (i.e., single-digit meV atom^{-1}) different from μ_e of NVNMD. It's worth noting that, we copied data of different systems from literature into Table 1, since it is distracting and time-consuming to reproduce MD results of so many systems using various MD tools by ourselves. So, Table 1 only roughly shows that the MLMD and NVNMD have the similar accuracy ($\mu_e \approx 10^{-1}$ kcal mol^{-1}) and both of them are much more accurate than CMD ($\mu_e \approx 10^{1}$ kcal mol^{-1}).

Though it is clear from Table 1 that NVNMD is much more accurate than CMD, it is difficult to tell the subtle accuracy difference between NVNMD and MLMD based on different datasets. Therefore, more rigorous and refined analysis is needed. Instead of directly fetching MLMD RMSE data from literature (as we did in Table 1), hereafter we start the entire procedure (including training, inference, and testing) all the way over by using the identical set of training/testing data on an identical set of systems (Table 2).

Since the proposed NVNMD is revised from DeePMD[26–31] (details available in the Methods section), we use DeePMD as reference and the starting point (third row of Table 2). Then, the vN-based revisions and the NvN-based revisions are made consecutively to obtain the results of CNN (second row of Table 2) and QNN (first row of Table 2), respectively. QNN results are the final results of NVNMD in Table 1. Here, the CNN includes all quantization-free (thus more appropriate for vN) revisions, e.g., the revision of symmetry-preserving feature calculation (Eq. (11)) and the revision of nonlinear activation function calculation (Section Nonlinear activation function). The QNN further includes the quantization-based (thus more appropriate for NvN) revisions, e.g., the continuous neural network is replaced by using quantized neural network (Section Quantized neural network), the multiplication operations of floating-point numbers are replaced by using shift operations of quantized numbers (Section Multiplication-less neural network), the continuous evaluation is replaced by using discretized look-up table searching (Eq. (6)) and so on.

When applied to all systems under test, the NVNMD shows a single-digit meV atom^{-1} accuracy difference, compared to the MLMD (last row of Table 2). It's worth noting that this is about 3 orders of magnitude lower than the typical interatomic bonding energy (at the order of 10^0 eV atom^{-1}), and 1 order of magnitude smaller than the chemical accuracy threshold, 1.0 kcal mol^{-1} (i.e., about 43.4 meV atom^{-1})[63,64]. Furthermore, this high accuracy is kept while calculating elementary, binary, and quaternary systems. The test systems we chose in Table 2 include complicated crystalline-amorphous phase transitions (Sb and GeTe[56,61]) and atomic diffusion through quaternary system (Li-Ge-P-S[62]), which involve repeated chemical bond rapture and re-forming with very sophisticated PES. Therefore, these test calculations can prove that NVNMD would be accurate enough to handle many complicated MD applications.

To further test the accuracy of NVNMD, we also computed atomic forces as shown in Table 3, because atomic forces are vital to reliably obtain the MD trajectories during integration of the Newton equation. Since the molecular systems in MLMD literature

Table 1. Calculation accuracy comparison between the proposed NVNMD and the established MLMD/CMD.

Method	System	Computer architecture	Computing hardware	$\lvert\Delta E_i\rvert$ (kcal mol^{-1})	μ_e (kcal mol^{-1})
NVNMD (this work)	Benzene	NvN	FPGA	0.19	0.21
	Naphthalene		FPGA	0.39	
	Aspirin		FPGA	0.32	
	Antimony (bulk)		FPGA	0.14	
	Germanium telluride (bulk)		FPGA	0.09	
	Li-Ge-P-S (bulk)		FPGA	0.14	
MLMD	Benzene[26,27,31,58]	vN	GPU	0.07/0.07	0.17
	Uracil[26,27,31,58]		GPU	0.09/0.11	
	Naphthalene[26,27,31,58]		GPU	0.12/0.12	
	Aspirin[26,27,31,58]		GPU	0.27/0.27	
	Salicylic acid[26,27,31,58]		GPU	0.12/0.12	
	Malonaldehyde[26,27,31,58]		GPU	0.16/0.16	
	Ethanol[26,27,31,58]		GPU	0.15/0.15	
	Toluene[26,27,31,58]		GPU	0.12/0.12	
	Water[26,27,31]		GPU	0.01	
	Glycine proton transfer[65,96]		CPU	0.3–0.4	
	Acetic acid[65,66]		CPU	0.2	
	Acetamide[65,66]		CPU	0.4	
	Acetone[65,66]		CPU	0.4	
	Ethanol[65,66]		CPU	0.2	
	Germanium telluride[56]		GPU	0.09	
	Li-Ge-P-S[62]		GPU	0.04	
CMD	700 different molecular structures[93]	vN	GPU/CPU	10 (est.)	≈10
	Small-molecule solvation and protein-ligand binding[67]		GPU/CPU	3.6–6.3	
	AMBER intermolecular terms w.r.t. DFT-SAPT[94]		GPU/CPU	10 (est.)	
	Nobel gas absorption in metal organic framework[95]		GPU/CPU	8.4-12.6	

ΔE_i is the RMSE with respect to ab-initio calculations. μ_e is the average value of $\lvert\Delta E_i\rvert$. The abbreviation FPGA represents field programmable gate array.

Table 2. The root mean square errors (RMSE) of system energies (meV atom^{-1}) of the three bulk systems.

	Method	Computer architecture	Sb (elementary)	GeTe (binary)	Li-Ge-P-S (quaternary)
NVNMD (this work)	QNN (after quantization)	NvN	6.2	3.7	6.1
	CNN (before quantization)	vN	4.2	3.6	3.1
MLMD	DeePMD[56,61,62]	vN	2.2	4.1	1.3
Difference between QNN (i.e., final results of NVNMD) and MLMD			4.0	−0.4	4.8

in Table 3 are all evaluated based on the same dataset[58–60], we evaluate NVNMD based on this dataset too. When compared against the ab-initio results, the atomic force mean absolute error (MAE) of NVNMD and that of MLMD show a very small difference (i.e., at the order of 10^1 meV Å^{-1}), as shown in the last row of Table 3. In all test cases, this difference is even below the default atomic force threshold (e.g., 40.0 meV Å^{-1} in SIESTA[68]; 25.7 meV Å^{-1} in Quantum Espresso[69]; and 23.1 meV Å^{-1} in CP2K[70]) to determine atomic force convergence during atomic lattice relaxation or supercell geometry optimization in the mainstream ab-initio density functional theory tools.

To visually illustrate the high accuracy, the energy and force predicted by the proposed NVNMD are plotted against those predicted by the established DFT-based AIMD, as shown in Fig. 1. The high accuracy of energy and forces laid a solid foundation for reliable calculation of physical properties. As shown in Table 4, the bond length, bond angle, and vibration frequencies of the water molecule calculated by using the proposed NVNMD are very close (<1% different), compared to the those obtained by MLMD. As shown in Fig. 2, the radial distribution function, angle distribution function, and coordination number of amorphous GeTe calculated by NVNMD are also very close to those by MLMD.

To further test accuracy in GeTe system, the canonical (NVT) ensemble MD is performed as shown in Fig. 3a. The crystalline GeTe system of 512 atoms is melted from crystalline to liquid, by increasing temperature from 300 K to 1800 K. Then it is quenched from liquid to amorphous from 1800 K to 300 K. Finally, the system is recrystallized from amorphous back into crystalline by annealing it at 600 K. The entire melt-quench-anneal phase transition processes as measured in experiments[71,72] can be successfully reproduced, as shown in Fig. 3a, indicating decent accuracy of the proposed NVNMD.

Table 3. The mean absolute error (MAE) of atomic forces (meV Å^{-1}) of three bulk systems and three molecular systems.

	Method	Computer architecture	Bulk systems			Molecular system		
			Sb	GeTe	Li-Ge-P-S	Benzene	Aspirin	Naphthalene
NVNMD (this work)	QNN (after quantization)	NvN	85.1	161.0	68.1	11.9	28.8	18.8
	CNN (before quantization)	vN	79.4	161.0	65.9	6.7	24.7	10.3
MLMD	DeePMD[30]	vN	64.2	168.0	79.6	--	19.4	13.1
	SchNet[84]		–	–	–	7.4	14.3	4.8
	DimeNet[85]		–	–	–	8.1	21.6	9.3
	sGDML[86]		–	–	–	2.6	29.5	4.7
	PaiNN[87]		–	–	–	–	16.1	3.6
	SpookyNet[88]		–	–	–	–	11.2	3.9
	GemNet[89]		–	–	–	6.3	9.5	2.4
	NewtonNet[90]		–	–	–	–	15.1	3.6
	UNiTE[91]		–	–	–	–	6.1	1.5
	NequIP[92]		–	–	–	–	8.2	1.6
	μ_F		64.2	168.0	79.6	6.1	15.1	4.9
Difference between QNN (i.e., final results of NVNMD) and μ_F of MLMD			20.9	−7.0	−11.5	5.8	13.7	13.9

All molecular system calculations are based on the MD17 dataset[58–60].

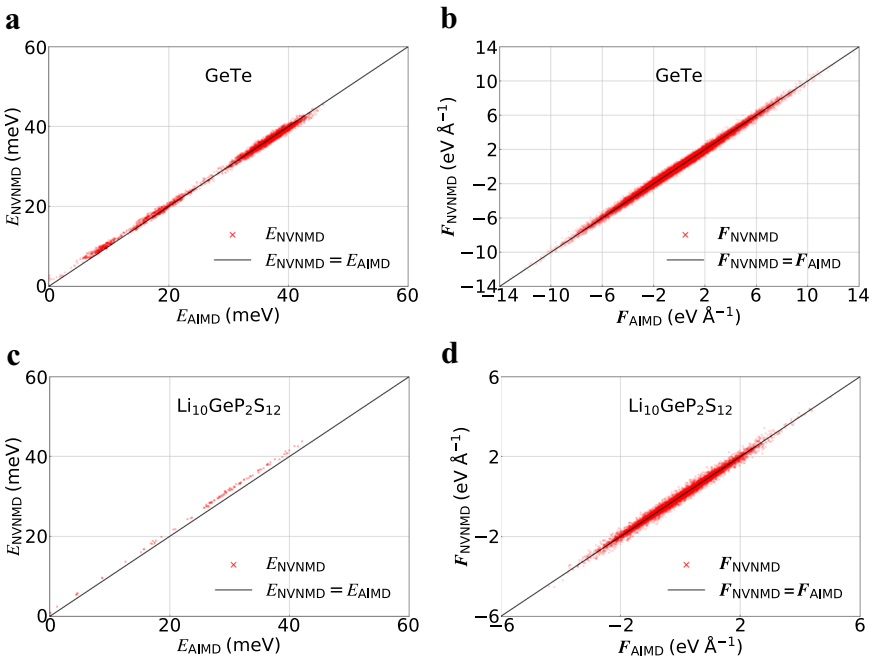

Fig. 1 The comparison of predicted energy and force. The test systems are GeTe (**a**, **b**) and $Li_{10}GeP_2S_{12}$ (**c**, **d**). $\boldsymbol{E}_{AIMD}$ and $\boldsymbol{F}_{AIMD}$ denote the energy and atomic forces of the AIMD samples. $\boldsymbol{E}_{NVNMD}$ and $\boldsymbol{F}_{NVNMD}$ denote the energy and atomic forces evaluated by NVNMD model.

Table 4. The test error $||\Delta E||_2$ and $||\Delta F||_2$, bond length, bond angle, and vibration frequencies of a single water molecule.

	$\|\|\Delta E\|\|_2$ (meV atom^{-1})	$\|\|\Delta F\|\|_2$ (meV Å^{-1})	O-H bond length (Å)	H-O-H bond angle (°)	Vibration frequency (cm^{-1})		
					Scissoring bend	Symmetric stretch	Antisymmetric stretch
NVNMD	0.5	21.4	0.97	104.77	1601.41	3643.20	3816.68
MLMD (DeePMD)	0.3	20.4	0.97	104.87	1588.06	3636.53	3796.67
Difference between NVNMD and MLMD	0.2	1.0	0.00	0.10	13.35	6.67	20.01

$||\Delta E||_2$ and $||\Delta \boldsymbol{F}||_2$ are root mean errors (RMSE) of system energies and atomic forces, respectively.

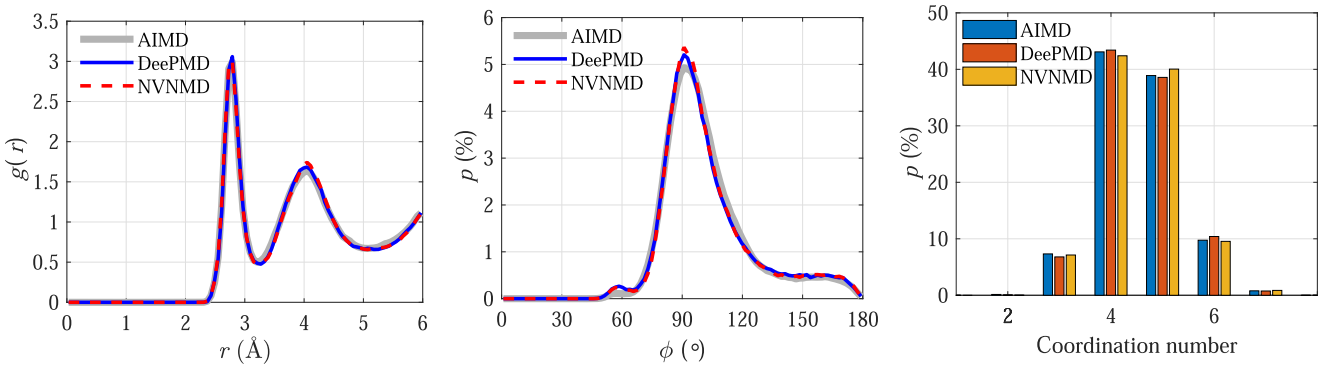

Fig. 2 The structure properties of amorphous GeTe. Radial distribution function (left), angle distribution function (middle), and coordination number (right) are computed by using AIMD, MLMD (e.g., DeePMD), and the proposed NVNMD.

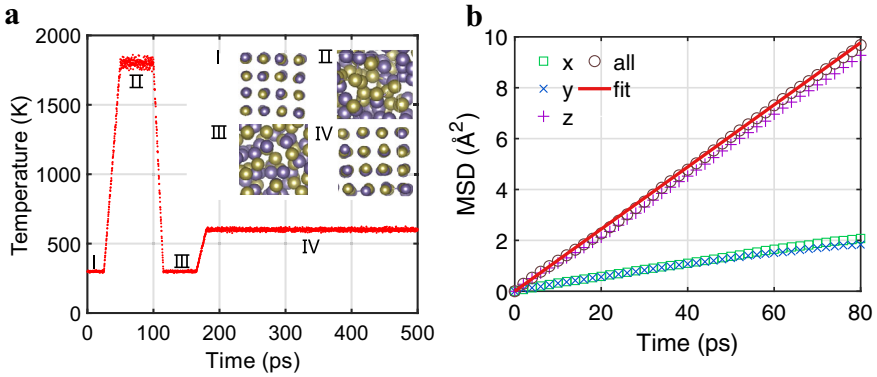

Fig. 3 The molecular dynamics simulation results of GeTe and $Li_{10}GeP_2S_{12}$ using the proposed NVNMD. In panel **a**, the phase transition processes of GeTe is reproduced, which contains initial crystalline phase (I), liquid phase (II), amorphous phase (III), and recrystallization (IV) shown in the insets. In panel **b**, the mean square displacement (MSD) of $Li_{10}GeP_2S_{12}$ is computed using NVNMD and the diffusion coefficient is extracted from its fitting line. The components of *x*, *y*, and *z* direction show the anisotropic diffusion.

For the MD test of $Li_{10}GeP_2S_{12}$, the system of 900 atoms is initialized and equilibrated at 500 K using NVT ensemble for 10 ps, then simulated at microcanonical (NVE) ensemble for 100 ps. The trajectory is used to calculate diffusion coefficients of system (shown as Fig. 3b). The mean square displacement (MSD) is calculated from trajectory using following expression

$$\mathrm{MSD} \approx \left(\frac{1}{6t}\right)\left\langle |r_i(t) - r_i(0)|^2 \right\rangle, \quad (1)$$

The diffusion coefficient is computed by extracting the slope of MSD in Fig. 3b. We get the diffusion coefficient 2.03×10^{-10} m^2 s^{-1} which is close to the value of Ref. [62] (i.e., 2.00×10^{-10} m^2 s^{-1}). Furthermore, according to the NVNMD results (Fig. 3b), the MSD in *x* and *y* directions are significantly smaller than that in *z* direction. This is in line with the anisotropic diffusion properties of $Li_{10}GeP_2S_{12}$ well known in literature[73–75].

Time efficiency

Using the six test systems, the calculation time efficiency of the proposed NVNMD is shown in Table 5. It is obvious that the time efficiency of the proposed NVNMD is around two orders of magnitude better than that of the MLMD. This means that NVNMD runs at a high speed like CMD, despite calculation complexity of MLMD is much higher than that of CMD. As schematically shown in Fig. 4, the proposed NVNMD simultaneously achieves both the CMD-level high efficiency and the MLMD/AIMD-level high accuracy. It's worth noting that the same high efficiency is kept, no matter elementary (e.g, Sb), binary (e.g., GeTe), and quaternary (e.g., Li–Ge–P–S) systems are simulated.

Energy efficiency

The energy efficiency η is calculated via the formula $\eta = T \times P$, where T represents the calculation time efficiency (Section Time efficiency); and P denotes the power consumption, which is measured by using a local power tester (PUUCAI P26A-10PN). The total power of the proposed NVNMD system is measured to be only about 108 W. Therefore, η of the proposed NVNMD system is around 10^{-5} J step^{-1} atom^{-1} (Table 6).

By using T and P of MLMD from Ref. [26,31], it can be estimated that η of the established vN-based MLMD is around 10^{-3}–10^{-2} J step^{-1} atom^{-1}. Here, P of MLMD is calculated by using the number of CPU/GPU used in Ref. [26,31]; and we use 30 W per CPU and 250 W per GPU for estimation[76–78]. For instance, based on Summit supercomputer, MLMD uses 27.3 thousand CPU cores and 27.3 thousand GPUs[31], so $P \approx 27.3 \times 10^3 \times (250 + 30) \approx 7.6$ MW (around 50–60% of Summit supercomputer's total power consumption 13 MW) is used to achieve $T \approx 2.7 \times 10^{-10}$ s step^{-1} atom^{-1}.

As shown in Table 6, the calculation energy efficiency of the proposed NVNMD is around 2–3 orders of magnitude better than that of the established vN-based MLMD, with similar calculation accuracy (Table 5 and Fig. 4). Such high energy efficiency is achieved, because in NVNMD there is no repeated data shuttling, which consumes most of the energy in its vN-based counterpart[33–35]. Consequently, the calculation energy

Table 5. Calculation time efficiency comparison between the proposed NVNMD and the established MLMD/CMD.

Method	System	Computer architecture	Hardware resource	T_i (s step^{-1} atom^{-1})	μ_t (s step^{-1} atom^{-1})
NVNMD (this work)	Benzene	NvN	FPGA	2.0×10^{-7}	2.0×10^{-7}
	Naphthalene		FPGA	2.0×10^{-7}	
	Aspirin		FPGA	2.0×10^{-7}	
	Antimony (bulk)		FPGA	2.0×10^{-7}	
	Germanium telluride (bulk)		FPGA	2.0×10^{-7}	
	Li-Ge-P-S (bulk)		FPGA	2.0×10^{-7}	
MLMD	H_2O[26,27,31]	vN	GPU	5.6×10^{-5}	2.4×10^{-5}
	SiO_2[123]		80 CPU cores	3.6×10^{-5}	
	Cu (original DP)[124]		GPU	2.8×10^{-5}	
	H_2O (original DP)[124]		GPU	9.5×10^{-6}	
	Al-Cu-Mg (original DP)[124]		GPU	8.7×10^{-5}	
	Cu (compressed DP)[124]		GPU	2.8×10^{-6}	
	H_2O (compressed DP)[124]		GPU	2.6×10^{-6}	
	Al-Cu-Mg (compressed DP)[124]		GPU	5.4×10^{-6}	
	GeTe (compressed DP)[56]		GPU	3.7×10^{-6}	
	Li-Ge-P-S (compressed DP)[62]		GPU	9.4×10^{-6}	
CMD	Lennard-Jones[32]	vN	12 CPU cores	7.0×10^{-7}	7.3×10^{-7}
	Chain[32]			3.2×10^{-7}	
	EAM[32]			1.9×10^{-6}	
	Chute[32]			1.7×10^{-7}	
	GROMACS with SIMD[125]		64 CPU cores	7.6×10^{-8}	
	GROMACS without SIMD[125]			1.3×10^{-6}	

T_i is the calculation time efficiency. μ_t is the average value of T_i. For fair comparison, we list MD calculations based on one GPU, or one FPGA, or one supercomputer node with a moderate number (e.g., <100) of CPU cores.

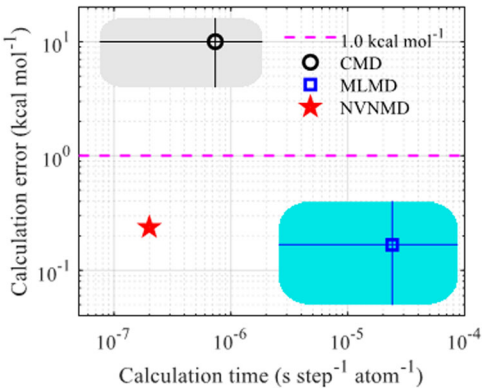

Fig. 4 Calculation time and error of CMD, MLMD, and the proposed NVNMD. The black circle, blue square and red star mark the points of (μ_t, μ_e), which are fetched from Table 1 and Table 5; the error bars and gray/blue regions are used to schematically illustrate the variations of data generated by different researchers; the horizontal dashed line is the quantum chemistry accuracy threshold, i.e., 1.0 kcal mol^{-1},[63,64]. Note: (1) we use μ_e values in Table 1, in order to cover more results from literature, which are based on different datasets; (2) for more rigorous accuracy analysis based on identical datasets, please refer to Tables 2 and 3.

efficiency of the proposed NVNMD is comparable to that of CMD, but accuracy of the proposed NVNMD is much superior than that of CMD (Table 1).

DISCUSSION

As an early-stage pilot version, we implemented the NVNMD on an FPGA (details available in the Methods section). It is well known that FPGA has merits of low-cost and field-programmability (i.e., short turnaround time for design revisions and iterations), and the disadvantages of limited hardware resources and low clock frequency. In contrast, the application-specific integrated circuit (ASIC) has merits of more abundant hardware resources and much higher clock frequency, and the disadvantages of high fabrication cost and long development cycle. So, FPGA is typically used as a debugging and testing tool (research phase), before taping out the ASIC (mass-production phase).

It's worth noting that the proposed NVNMD, which simultaneously achieved high calculation accuracy (Table 1), high calculation time efficiency (Table 5), and high calculation energy efficiency (Table 6), is based on a low-end device (Xilinx xcvu9p) in the Xilinx Virtex UltraScale+ FPGA product family (Section Hardware implementation)[79,80]. This has three significant technological implications on the future ASIC development scenarios of the proposed paradigm (NvN-based MD).

Firstly, the NVNMD we use here is based on a low clock frequency (i.e., 250 MHz), which is about one order of magnitude lower than ordinary ASIC like the commodity-level vN-based GPU/CPU whose clock frequency can reach several GHz[76]. This implies that the time efficiency of NVNMD could be enhanced by another order of magnitude (i.e., $C_1 \approx 10^1$), in a straightforward fashion by boosting the clock frequency, if we move from the research phase (FPGA) to the production phase (ASIC).

Secondly, the NVNMD we use here is implemented using a rather limited amount of hardware resources (i.e., about 10^6 logic cells in FPGA as shown in Table 7). It is well known that a single ASIC chip could integrate around 10^9–10^{10} transistor devices (e.g., 1.6×10^{10} and 2.1×10^{10} transistor devices in one Apple M1 5 nm chip and one NVIDIA Tesla V100 12 nm chip, respectively[76,81]). Even though we use about 10^1 transistor devices to realize 1 logic cell, the ASIC could be 2 to 3 orders of magnitude more resource

Table 6. Calculation energy efficiency comparison between the proposed NVNMD and the established MLMD/CMD.

		T (s step^{-1} atom^{-1})	P (W)	$\eta = T \times P$ (J step^{-1} atom^{-1})
NVNMD (this work)		2.0×10^{-7}	108	2.1×10^{-5}
MLMD	1 CPU + 1 GPU[26,31]	5.6×10^{-5}	280	1.6×10^{-2}
	Summit supercomputer[31]	2.7×10^{-10}	7.6×10^{6}	2.1×10^{-3}
CMD		$\approx 10^{-7}$	$\approx 10^{2}$	$\approx 10^{-5}$

T is the calculation time efficiency. P is the power consumption. The NVNMD (this work) uses three CPU cores (Intel i7-10700K) and one FPGA (Xilinx xcvu9p). The MLMD uses CPU (IBM POWER9) and GPU (NVIDIA Tesla V100), as introduced in Ref. [26,31]. The CMD results are taken from Table 5.

Table 7. FPGA resource consumption of the proposed system.

Resource	Utilization	Available	Utilization ratio
LUT	500111	1182240	42.30%
LUTRAM	52035	591840	8.79%
FF	674294	2364480	28.52%
DSP	4058	6840	59.33%
BRAM	188	2160	8.7%
URAM	560	960	58.33%

The abbreviations LUT, LUTRAM, FF, DSP, BRAM, and URAM represent Look-Up Table[110], Look-Up Table RAM[110], Flip-Flop[110], Digital Signal Processor[109], Block RAM[105], and UltraRAM[105], respectively.

abundant than the FPGA we are using. By leveraging the decent parallelization scaling property of MLMD (inherited by the proposed NVNMD)[82], we anticipate at least two orders of magnitude enhancement of time efficiency (i.e., $C_2 \approx 10^2$), by purely increasing the intra-ASIC parallelization.

Thirdly, the logic and arithmetic circuit can be deployed much more freely in ASIC than in FPGA, since FPGA uses more resources to ensure flexibility and programmability. Given much less constraints, the same set of functionalities can be implemented with much fewer transistor devices in ASIC than in FPGA. Therefore, by using ASIC to replace FPGA, the time efficiency can be increased by roughly ten times when the logic and arithmetic circuits are simplified (i.e., $C_3 \approx 10^1$)[83].

To summarize, the FPGA-based results we presented in this paper is a very early development stage of the proposed NVNMD paradigm. Based on the architecture design we verified using FPGA in this paper, we are working on developing ASIC-based NVNMD computer, which could be around 4 orders of magnitude (i.e., $C = C_1 \times C_2 \times C_3 \approx 10^4$) more efficient than the results we showed in this paper. In another word, by moving from FPGA to ASIC, the time efficiency of NVNMD could be enhanced from 10^{-7} s step^{-1} atom^{-1} (Table 5) to around 10^{-11} s step^{-1} atom^{-1}. This means that NVNMD based on a single ASIC chip (with cm^2-level size and 10^1-10^2 Watt-level power) could be faster than the MLMD based on the whole Summit supercomputer (around 10^{-10} s step^{-1} atom^{-1}, with one entire building size and 10^6–10^7 Watt-level power)[31,82]. Of course, during the implementation of the ASIC-based NVNMD, we need to consider factors other than speed, e.g., the generality to run all kinds of MD and the flexibility to control/dump MD simulation results, etc. All these considerations may compromise the speed somehow, which deserves future research attention. Finally, there are lots of machine learning methods other than DeePMD (e.g., SchNet[84], DimeNet[85], sGDML[86], PaiNN[87], SpookyNet[88], GemNet[89], NewtonNet[90], UNiTE[91], NequIP[92], and so on). The NvN acceleration of these methods deserves future research attention, too.

Compared to the other special-purpose MD computers already existing in literature (e.g., Anton)[22,23,40,41], the NVNMD proposed here is different in several aspects. Anton focuses on accelerating biology-related MD simulations and, thus, mainly implements biology-oriented classical force fields. While these classical force fields offer valuable insights to simulate biological MD problems (e.g., protein folding), they suffer serious accuracy problems in many applications in other fields, because it only has the CMD-level accuracy[67,93–95]. The accuracy of the proposed NVNMD, however, is at the AIMD/MLMD-level. Furthermore, Anton is implemented on the ASIC using advanced semiconductor technology nodes (e.g., 7 nm node), which offers much higher speed than the FPGA used in this pilot version of NVNMD.

METHODS

In this Section, the overall system design of the proposed special-purpose MD computer is introduced. Section Heterogeneous parallelization describes the heterogeneous parallelization between the proposed NVNMD computer's two major units—the master processing unit (MPU) and the slave processing unit (SPU). Section Pipeline and high-speed transmission interface discusses the high-speed transmission interface (HTI) between MPU and SPU. Section Master processing unit and Section Slave processing unit introduce the functionalities of MPU and SPU, respectively.

As the most important part of the proposed NVNMD computer, the SPU bears the predominant majority (e.g., over 99%) of the total computational load. To maximize calculation efficiency of SPU, we propose a paradigm shift from the established general-purpose vN architecture (e.g., CPU and GPU)[26,27,31,56,58,62,65,66,96,97] to a special-purpose NvN architecture. The proposed NvN architecture efficiently computes the energy (Section Energy calculation) and atomic force and virial (Section Force and virial calculation) by leveraging the processing-in-memory (PIM) technology (Section Processing in memory), based on the algorithms of DeepPot-SE MLMD[26,30,98] after three crucial modifications. These three modifications are indispensable to realize high calculation efficiency using very limited amount of hardware resources. Firstly, the traditional continuous neural network (CNN) widely used in MLMD is replaced by using the quantized neural network (QNN) (Section Quantized neural network). Secondly, the resource-consuming multiplication-based neural network is replaced by using a resource-economical multiplication-less neural network deliberately-designed here for the NVNMD (Section Multiplication-less neural network). Thirdly, the widely-used trigonometric function-based nonlinear activation functions are replaced by using the lightweight nonlinear activation functions specially-crafted for the NVNMD (Section Nonlinear activation function). With the help of these three significant modifications, the NvN-based SPU is implemented in a field programmable gate array (FPGA), to quantitatively test the overall performance of the proposed MD computer (Section Hardware implementation).

Heterogeneous parallelization

The MD simulation consists of a certain number of timesteps in a loop. In each timestep, there are two parts of calculations – (i) the evaluation of PES, $E = E(\{\boldsymbol{R}_i\})$, and atomic forces, $\boldsymbol{F}_i = -\nabla_i E(\{\boldsymbol{R}_i\})$; and (ii) all other calculations, including numerical integration of the Newton equation to update $\{\boldsymbol{R}_i\}$ and $\{\boldsymbol{v}_i\}$. Here, E is the system energy; $\boldsymbol{R}_i$, $\boldsymbol{v}_i$, and $\boldsymbol{F}_i$ are the Cartesian coordinate, velocity, and force of the atom i ($i = 1, 2, \ldots, N$), respectively; and N is the total number of atoms in the simulation system[99].

In large-scale AIMD/MLMD simulations (i.e., $N > 10^2$), the overwhelming majority (e.g., over 99%) of computing time is spent to evaluate PES[82]. So, we focus on accelerating the calculation of part (i) by using the SPU based

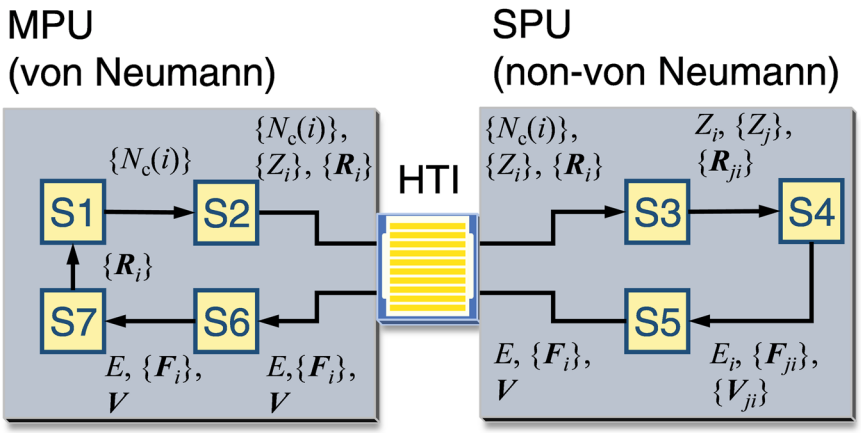

Fig. 5 Schematic figure of heterogeneous parallelization in the NVNMD system. The system consists of master processing unit (MPU), slave processing unit (SPU), and high-speed transmission interface (HTI). Calculation of each timestep of MD in the system consists of seven consecutive steps: building neighbor list (S1); encoding atomic information into compact data (S2); receiving and decoding atomic compact data (S3); evaluating potential energy surface (PES) (S4); encoding and sending data of PES (S5), decoding data of PES (S6), and other calculations, e.g., numerical integration (S7). Here, E and $\boldsymbol{V}$ represent the energy and virial of atomic system, respectively; Z_i, $\boldsymbol{R}_i$, E_i, $\boldsymbol{F}_i$, and $N_c(i)$ represent the chemical specie, coordinate, energy, force, and neighbor list of atom i, respectively; $\boldsymbol{R}_{ji}$, $\boldsymbol{F}_{ji}$, and $\boldsymbol{V}_{ji}$ represent the relative coordinate, force component, and virial component between atom i and atom j.

on the proposed special-purpose NvN architecture. In contrast, the calculation of part (ii) is much less computationally demanding, so the calculation of part (ii) is based on the traditional general-purpose vN architecture in MPU. This vN/NvN heterogeneous architecture is designed to leverage the flexibility of vN architecture. As a consequence, the proposed MD computer can efficiently run all kinds of MD simulations, e.g., canonical ensemble MD, microcanonical ensemble MD, isothermal–isobaric ensemble MD, enhanced sampling, and so on[100–104].

As illustrated in Fig. 5, the calculation of each MD timestep consists of seven consecutive steps (i.e., S1, S2, S3, S4, S5, S6, and S7). In the S1, all atoms j in the vicinity of the atom i are chosen as the neighbor atoms, whose indices are stored in a neighbor list $N_c(i) = \{j, |\boldsymbol{R}_j - \boldsymbol{R}_i| < R_c\}$ where R_c is a predefined cutoff. In the S2, the neighbor list $\{N_c(i)\}$, together with all atoms' chemical species $\{Z_i\}$ and atomic coordinates $\{\boldsymbol{R}_i\}$, is encoded into a compact data format using 16, 2, and 64 bits, respectively. Then, the compact data $\{N_c(i)\}$, $\{Z_i\}$, and $\{\boldsymbol{R}_i\}$ are transmitted from the MPU to the SPU. In the S3, after receiving these compact data from MPU, the global atomic information is transferred into the i^{th} atom's local atomic information which includes Z_i, $\{Z_j, j \in N_c(i)\}$ and $\{\boldsymbol{R}_{ji}\} = \{\boldsymbol{R}_i - \boldsymbol{R}_j, j \in N_c(i)\}$. In the S4, the i^{th} atom's energy component E_i, the atomic force components $\{\boldsymbol{F}_{ji}\} = \{\partial E_i/\partial \boldsymbol{R}_{ji}, j \in N_c(i)\}$ and virial components $\{\boldsymbol{V}_{ji}\} = \{\boldsymbol{R}_{ji}^{\mathrm{T}} \times \boldsymbol{F}_{ji}, j \in N_c(i)\}$ are evaluated by feeding the local atomic information into the PES computation module. In the S5, the system energy E, atomic forces $\{\boldsymbol{F}_i\}$, and virial $\boldsymbol{V}$ are computed by summing the contributions of each atom. The relationship can be represented as $E = \sum_{i=1}^{N} E_i$,

$$F_i = -\nabla_i E(\{R_i\}) = -\sum_{j \in N_c(i)} F_{ji} + \sum_{j \in N_c(i)} F_{ij} \tag{2}$$

and

$$V = \sum_i R_i^{\mathrm{T}} \times F_i = -\sum_{i \neq j} V_{ji} \tag{3}$$

Here, E, $\{\boldsymbol{F}_i\}$, and $\boldsymbol{V}$ are encoded with 64 bits, 32 bits, and 64 bits, respectively, and written into a random-access memory (RAM) ready to be read by MPU. In the S6, MPU reads and decodes compact data from the RAM. In the S7, numerical integration of the Newton's equation, MD thermostat, and material properties are computed.

While S1, S2, S6 and S7 are executed in the MPU, S3, S4 and S5 are executed in the SPU. During integration of these seven steps into a whole functional MD computer, the high calculation efficiency is ensured by two key design ideas. Firstly, the MPU and SPU are linked by the high-speed transmission interface (HTI), and the time spent in MPU calculation and MPU-SPU communication is minimized by the parallel pipeline computation (Section "Pipeline and high-speed transmission interface"). Secondly, the PES evaluation, which is the most time-consuming part of MD, is significantly accelerated by the processing in memory (PIM) calculations in SPU based on the proposed NvN architecture, under the coordination of

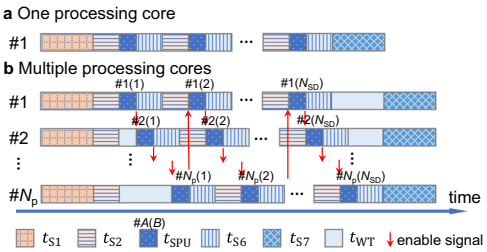

Fig. 6 Schematic figure of calculation time based on pipeline design when one processing core a **or multiple processing cores** b **are used in MPU.** The t_{S1}, t_{S2}, t_{S3}, t_{S4}, t_{S5}, t_{S6}, and t_{S7} are the calculation time of S1, S2, S3, S4, S5, S6, and S7, respectively, in Fig. 5. Here, $t_{SPU} = t_{S3} + t_{S4} + t_{S5}$; and t_{WT} is the waiting time; '#A(B)' stands for the sub-domain B processed by the core #A of MPU; N_p is the number of cores in MPU; N_{SD} is the number of sub-domains decomposed according to the SPU capacity. Enable signals are used to coordinate the calling of SPU.

MPU (Sections Master processing unit and Slave processing unit). The MPU is implemented by running a revised LAMMPS package[57] on a multicore CPU (Section Master processing unit); and the SPU is implemented using an FPGA (Section Slave processing unit).

Pipeline and high-speed transmission interface

To mitigate, as much as possible, the efficiency bottleneck caused by MPU-SPU communication, four high-speed technologies are used together here. Firstly, the MPU-SPU HTI is designed as a full-duplex channel to enable simultaneous data sending and data receiving, by using two separate memory hardware units. For instance, in the FPGA implementation (Section Hardware implementation), one on-chip block random access memory (BRAM)[105] of SPU is used to read data, and another on-chip BRAM of SPU is used to write data. Secondly, high-speed peripheral component interconnect express (PCIe) technology is used to send/receive data between MPU and SPU. For instance, here we use 16-lane PCIe 3.0, whose maximum bandwidth is 7.88 Gbit s^{-1} per lane, so the total bandwidth is as high as 15.75 GByte s^{-1} (Section Hardware implementation)[106,107]. Thirdly, the direct memory access (DMA) technology is used to transfer data between MPU and SPU. Using DMA, MPU-SPU data communication can be achieved without MPU control, so that the data transfer latency is minimized and the burden of MPU is alleviated. Fourthly, the seven

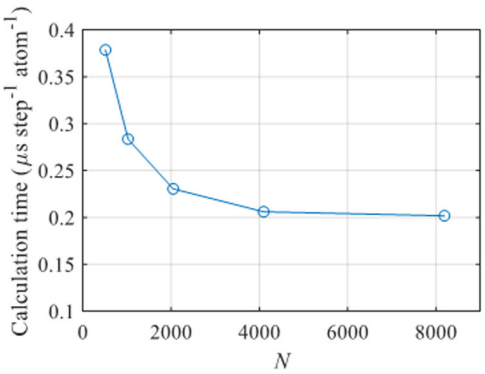

Fig. 7 The calculation time efficiency depending on the number of atoms. *N* denotes the number of atoms in the GeTe system.

consecutive steps (Fig. 5) are organized in a carefully-designed pipeline (Fig. 6).

While the first three high-speed technologies (i.e., full-duplex, PCIe, DMA) minimize the SPU's idle time (i.e., time other than t_{SPU} in Fig. 6a), the fourth one (i.e., pipeline) tries to vanish the SPU's idle time. As shown in Fig. 6b, by using a small number (typically less than ten) of CPU cores in the MPU, the NvN-based SPU is always busy performing heavy-duty calculations, which is beneficial to maximize the overall efficiency.

The proposed NVNMD is based on a pipeline, in which the MPU and the SPU work in a complementary manner (Figs. 5 and 6). Thus, to maximize the overall efficiency, it is desirable to keep the SPU always busy. In another word, t_{SPU} in Fig. 6 should be large enough (i.e., *N* should be large enough) to minimize T_i in Table 5. This trend can be seen in Fig. 7 – the calculation efficiency drops if *N* is small. Since our focus is the MD simulations of large systems (e.g., $N > 10^4$ atoms), this should not be a concern.

Master processing unit

MPU performs S1, S2, S6, and S7, as illustrated in Fig. 5. While using a MPU (e.g., CPU with N_P cores) to process the system of *N* atoms, the whole system is spatially decomposed into N_P domains with equal volume, and each core processes one domain for parallel acceleration[57]. To account for the interaction between atoms located within different domains, neighbor atoms of the domain $\boldsymbol{\Omega}$ are copied from the neighbor domains to form a shell domain (referred to as $\boldsymbol{\Omega}^n$ hereafter)[57]. The indices of atoms inside $\boldsymbol{\Omega}^n$ are stored in a list $\boldsymbol{I}_{\boldsymbol{\Omega}}{}^n = \{j \mid R_j^{\boldsymbol{\Omega}} < R_c \text{ and } j \notin \boldsymbol{I}_{\boldsymbol{\Omega}}\}$, where $R_j^{\boldsymbol{\Omega}}$ denotes the minimum distance between atom *j* and $\boldsymbol{\Omega}$; $\boldsymbol{I}_{\boldsymbol{\Omega}}$ is the list of indices of atoms inside $\boldsymbol{\Omega}$. For notational convenience, the list of indices of all atoms inside $\boldsymbol{\Omega}$ and $\boldsymbol{\Omega}^n$ is denoted as $\boldsymbol{I}_{\boldsymbol{\Omega}}{}^a = (\boldsymbol{I}_{\boldsymbol{\Omega}}, \boldsymbol{I}_{\boldsymbol{\Omega}}{}^n)$, which is a combined list of $\boldsymbol{I}_{\boldsymbol{\Omega}}$ and $\boldsymbol{I}_{\boldsymbol{\Omega}}{}^n$.

SPU can only store and process limited amount of data at one time due to hardware resource restriction, so domain $\boldsymbol{\Omega}$ is further divided into $N_{SD} = \lceil \mu \times N_{\boldsymbol{\Omega}} / N_{SPU} \rceil$ sub-domains (denoted as $\boldsymbol{\omega}$ hereafter) with equal volume, where μ is set as 2 to account for the spatial fluctuation of atom density; $N_{\boldsymbol{\Omega}}$ is the number of atoms within $\boldsymbol{\Omega}$; N_{SPU} is set as 4096 to strike a balance between communication efficiency and resource utilization; and $\lceil x \rceil$ is the ceiling function which rounds *x* to upper integer. While processing one sub-domain $\boldsymbol{\omega}$, a shell of $\boldsymbol{\omega}$ (referred to as $\boldsymbol{\omega}^n$ hereafter) is additionally created to account for the interaction between atoms located within $\boldsymbol{\Omega}^n$ and sub-domains other than $\boldsymbol{\omega}$. The indices of atoms inside $\boldsymbol{\omega}^n$ are stored in a list $\boldsymbol{I}_{\boldsymbol{\omega}}{}^n = \{j \mid R_j^{\boldsymbol{\omega}} < R_c \text{ and } j \notin \boldsymbol{I}_{\boldsymbol{\omega}}\}$, where $R_j^{\boldsymbol{\omega}}$ denotes the minimum distance between atom *j* and $\boldsymbol{\omega}$; $\boldsymbol{I}_{\boldsymbol{\omega}}$ is the list of indices of atoms inside $\boldsymbol{\omega}$.

Based on the abovementioned two-level decomposition (i.e., '*A*' and '*B*' in Fig. 6b), the enable signal is utilized to ensure that the MPU cores call the SPU in a proper order. After running S2, the p^{th} core ($p = 1, 2, \ldots, N_P$) doesn't call SPU until it receives an enable signal from its previous core (i.e., the $N_P{}^{th}$ core when $p = 1$ and the $(p-1)^{th}$ core otherwise). After obtaining the results from SPU, the p^{th} core sends an enable signal to its next core (i.e., the 1^{st} core when $p = N_P$ and the $(p+1)^{th}$ core otherwise). It's worth noting that the 1^{st} core doesn't require an enable signal to process its 1^{st} sub-domain. The steps (i.e., S1, S2, S6, and S7) are discussed in detail below. In the S1, each core builds the neighbor list $\{N_c(i), i \in \boldsymbol{I}_{\boldsymbol{\Omega}}\}$ of atoms located within its domain $\boldsymbol{\Omega}$.

In the S2, MPU processes the sub-domain $\boldsymbol{\omega}$'s data, including neighbor list $\{N_c(i), i \in \boldsymbol{I}_{\boldsymbol{\omega}}\}$, chemical species $\{Z_i, i \in \boldsymbol{I}_{\boldsymbol{\omega}}{}^a\}$, and coordinates $\{\boldsymbol{R}_i, i \in \boldsymbol{I}_{\boldsymbol{\omega}}{}^a\}$, where $\boldsymbol{I}_{\boldsymbol{\omega}}{}^a = (\boldsymbol{I}_{\boldsymbol{\omega}}, \boldsymbol{I}_{\boldsymbol{\omega}}{}^n)$ is a list obtained by combining $\boldsymbol{I}_{\boldsymbol{\omega}}$ and $\boldsymbol{I}_{\boldsymbol{\omega}}{}^n$. First, the $\{N_c(i)\}$ is recoded as local neighbor list. For example, if one element of $\{N_c(i)\}$ is 5, and the value 5 is located at the 1^{st} position of $\boldsymbol{I}_{\boldsymbol{\omega}}{}^a$, this element of $\{N_c(i)\}$ will be encoded as 1. In this step, the encoded $\{N_c(i)\}$ is compressed from 32 bits to 16 bits. Second, $\{Z_i\}$ is compressed from 32 bits to 2 bits through encoding it as the order of chemical species. Third, $\{\boldsymbol{R}_i\}$ is encoded by multiplying 2^{48} and rounding into 64-bit integer from 64-bit floating number. The encoded data $\{N_c(i)\}$, $\{Z_i\}$, and $\{\boldsymbol{R}_i\}$ are stored in the buffer until they are transmitted to SPU by HTI.

In the S6, the cores of MPU decode the data fetched from SPU. Take the sub-domain $\boldsymbol{\omega}$ inside domain $\boldsymbol{\Omega}$ as an example, the data consists of energy $E_{\boldsymbol{\omega}}$, atomic forces $\{\boldsymbol{F}_i, i \in \boldsymbol{I}_{\boldsymbol{\omega}}{}^a\}$, and virial $\boldsymbol{V}_{\boldsymbol{\omega}}$. $E_{\boldsymbol{\omega}}$ is decoded from 64-bit integer to 64-bit floating point number by multiplying a factor of 2^{-13}, and then summed up to obtain the energy $E_{\boldsymbol{\Omega}}$ of domain $\boldsymbol{\Omega}$. $\{\boldsymbol{F}_i\}$ is decoded from 32-bit integer to 64-bit floating point number by multiplying a factor 2^{-25}, and then summed up into the corresponding atomic forces of domain $\boldsymbol{\Omega}$ (i.e., $\{\boldsymbol{F}_i, i \in \boldsymbol{I}_{\boldsymbol{\Omega}}{}^a\}$) according to the index in $\boldsymbol{I}_{\boldsymbol{\omega}}{}^a$. $\boldsymbol{V}_{\boldsymbol{\omega}}$ is decoded from 64-bit integer to 64-bit floating number by multiplying a factor 2^{-25}, and then added up to the virial $\boldsymbol{V}_{\boldsymbol{\Omega}}$ of $\boldsymbol{\Omega}$.

In the S7, the cores of MPU perform numerical integration, thermostat, and so on. After the core requests SPU to evaluate PES of its domain $\boldsymbol{\Omega}$, the energy $E_{\boldsymbol{\Omega}}$, atomic forces $\{\boldsymbol{F}_i, i \in \boldsymbol{I}_{\boldsymbol{\Omega}}{}^a\}$, and virial $\boldsymbol{V}_{\boldsymbol{\Omega}}$ are obtained, but they are incomplete. Therefore, the cores exchange the forces $\{\boldsymbol{F}_i, i \in \boldsymbol{I}_{\boldsymbol{\Omega}}{}^n\}$ of atoms located within the shell $\boldsymbol{I}_{\boldsymbol{\Omega}}{}^n$ to obtain the complete atomic forces $\{\boldsymbol{F}_i, i = 1, 2, \ldots, N\}$. In addition, $E_{\boldsymbol{\Omega}}$ and $\boldsymbol{V}_{\boldsymbol{\Omega}}$ are also exchanged to obtain the complete energy *E* and virial $\boldsymbol{V}$ of the whole system. Afterward, the atomic forces are used for numerical integration and other procedures in parallel. After S7 is finished, one timestep of MD is accomplished. The abovementioned steps repeat until all timesteps in the MD trajectory are accomplished.

Slave processing unit

As shown in Fig. 5, the SPU runs S3, S4, and S5 in each MD timestep. Among the three categories of MD (i.e., CMD, AIMD, and MLMD), we choose MLMD to implement the proposed special-purpose MD computer, because the FF-based PES evaluation in CMD is too inaccurate and the DFT-based PES evaluation in AIMD is too sophisticated. We modify the Deep Potential-Smooth Edition (DeepPot-SE)[30] and deploy it in the S4 of the SPU. The local atomic information Z_i, $\{Z_j\}$, $\{\boldsymbol{R}_{ji}\}$ are used to compute the many-body descriptor $\boldsymbol{D}_i$ which preserves the translation invariance, rotation invariance, and permutation invariance. Then, $\boldsymbol{D}_i$ is used to calculate the i^{th} atom's energy E_i. Finally, atomic force components $\{\boldsymbol{F}_{ji}\}$ are obtained by computing the negative derivative of E_i. These steps are discussed in more details below.

In order to preserve the translation invariance, the global coordinates $\boldsymbol{R}_j = (x_j, y_j, z_j)$ are transformed into the relative coordinates $\boldsymbol{R}_{ji} = \boldsymbol{R}_j - \boldsymbol{R}_i = (x_{ji}, y_{ji}, z_{ji})$. To describe the smooth cutoff, a new coordinate $\boldsymbol{u}_{ji}$ is constructed through multiplying $\boldsymbol{R}_{ji}$ by a cutoff function s_{ji}, which describes the contribution decay by the increase of R_{ji} until R_c. The new coordinate is expressed as

$$\boldsymbol{u}_{ji} = \left(s_{ji}, s_{ji}\frac{x_{ji}}{R_{ji}}, s_{ji}\frac{y_{ji}}{R_{ji}}, s_{ji}\frac{z_{ji}}{R_{ji}}\right) \tag{4}$$

Here, the cutoff function is defined as

$$s_{ji} = f_c(R_{ji}) = \begin{cases} \frac{1}{R_{ji}}, & 0 \le R_{ji} < R_{cs} \\ \frac{1}{R_{ji}}\left\{\frac{1}{2}\cos\left[\pi\frac{(R_{ji}-R_{cs})}{(R_c-R_{cs})}\right]+\frac{1}{2}\right\}, & R_{cs} \le R_{ji} < R_c \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where R_{cs} is a predefined cutoff parameter[26,30]. Next, multilayer perceptron (MLP) neural network G_{Z_j}[108], which is called Feature NN (FeaNN) hereafter, is constructed. FeaNN has one input node and *M* output nodes, which is written as

$$g_{ji} = G_{Z_j}(s_{ji}) \tag{6}$$

The output of l^{th} layer of MLP is

$$x^{l+1} = \xi^l(x^l \times w^l + b^l) \tag{7}$$

where x^l, w^l, b^l, and ξ^l are input, weight, bias, and nonlinear activation function of the l^{th} layer, respectively. The weights of FeaNN depend on

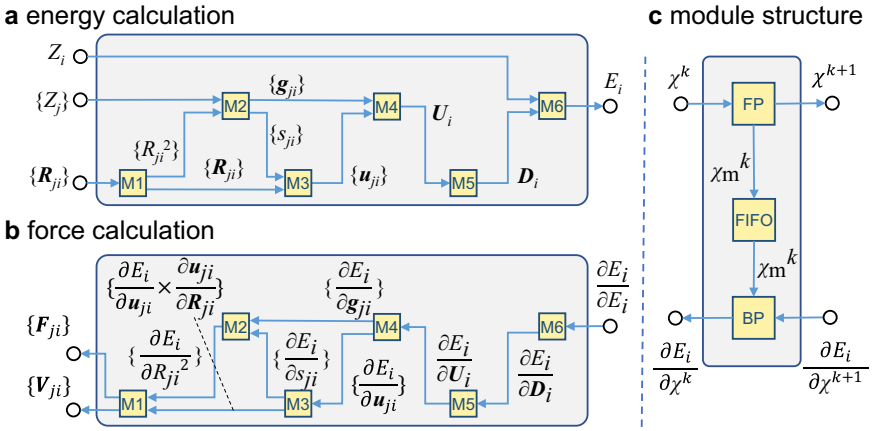

Fig. 8 The implementation of PES calculation. Six modules (i.e., M1, M2, M3, M4, M5, and M6) are used to implement the forward propagation calculation (**a**) from the chemical species Z_i and $\{Z_j\}$ and relative coordinates $\{\boldsymbol{R}_{ji}\}$ to the energy E_i as Section Energy calculation. The backward propagation (**b**) of modules is employed to calculate the component of force $\{\boldsymbol{F}_{ji}\}$ and virial $\{\boldsymbol{V}_{ji}\}$ as Sectioin Force and virial calculation. Each module can be divided into three submodules (**c**): FP, BP, and FIFO, where FP and BP are used to implement the calculation of forward propagation and backward propagation, respectively; FIFO is employed to transmit data from FP to BP.

chemical species Z_j. Therefore, the output $\boldsymbol{g}_{ji}$ can distinguish the contribution of neighbors with different chemical species.

In order to preserve the permutation invariance, matrix $\boldsymbol{U}_i$ of size $M \times 4$ is written as

$$U_i = \sum_{j \in N_c(i)} U_{ji} \tag{8}$$

where

$$U_{ji} = g_{ji}^{\mathrm{T}} \times u_{ji} \tag{9}$$

is a $M \times 4$ matrix[26,30].

In order to preserve the rotational invariance,

$$D_i' = U_i \times U_i^{\mathrm{T}} \tag{10}$$

is defined[26,30]. The subset of D_i' is extracted as a new $M \times M_2$ ($1 \le M_2 \le M$) matrix $\boldsymbol{D}_i$ for reducing unnecessary computational cost.

$$D_i[l,k] = D_i'[l,(k+l)\%M]; l = 0,1,\ldots,M-1; k = 0,1,\ldots,M_2-1 \tag{11}$$

where l and k are the matrix indexes of row and column, respectively; % is modulo operation.

The total energy is written as $E = \sum_{i=1}^{N} E_i$. The energy E_i of i^{th} atom is only determined by its chemical species Z_i and the symmetry-preserving feature $\boldsymbol{D}_i$[26,27]. MLP neural network is used to fit the relation between input $\boldsymbol{D}_i$ and output E_i (referred to as FitNN hereafter)

$$E_i = E_{Z_i}(D_i) \tag{12}$$

Then, the force and virial can be calculated by using Eq. (2) and Eq. (3), respectively.

Energy calculation

The evaluation of PES (S4 in Fig. 5) is realized by using six calculation modules (i.e., M1, M2, M3, M4, M5, and M6) in SPU, and the atomic energy E_i is predicted during forward propagation, as shown in Fig. 8a. In the M1, $\{R_{ji}^2\}$ is computed from the relative coordinate $\{\boldsymbol{R}_{ji}\}$. In the M2, $\{R_{ji}^2\}$ is used to calculate the cutoff function $\{s_{ji}\}$ and the outputs $\{\boldsymbol{g}_{ji}\}$ of FeaNN according to Eq. (5) and Eq. (6). The weights and biases of FeaNN are switched according to $\{Z_j\}$. In the M3, the new coordinate $\{\boldsymbol{u}_{ji}\}$ is obtained by using $\{s_{ji}\}$ and $\{\boldsymbol{R}_{ji}\}$ (Eq. (4)). In the M4, $\{\boldsymbol{u}_{ji}\}$ is multiplied by $\{\boldsymbol{g}_{ji}\}$ to get $\{\boldsymbol{U}_{ji}\}$, and then $\{\boldsymbol{U}_{ji}\}$ is summed together to get $\boldsymbol{U}_i$ (Eq. (9) and Eq. (8)). In the M5, the many-body descriptor $\boldsymbol{D}_i$ is extracted from the subset of the symmetric matrix D_i' which is the matrix product of $\boldsymbol{U}_i$ and $\boldsymbol{U}_i^{\mathrm{T}}$ (Eq. (10) and Eq. (11)). In the M6, the FitNN is implemented to evaluate E_i from $\boldsymbol{D}_i$, whose weights and biases are switched according to Z_i (Eq. (12)).

In order to simplify the computation complexity of M2, an interpolation method is used to map from $\{R_{ji}^2\}$ to $\{\boldsymbol{h}_{ji}\}$, where $\boldsymbol{h}_{ji}$ is a vector of R_{ji}^2 and Z_j (e.g., $\boldsymbol{s}_{ji}$ and $\boldsymbol{g}_{ji}$). At the beginning, N_{T} mapping tables with N_{M} rows are built to store the data $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ of R_{ji}^2 and Z_j in their k^{th} row ($k = 1, 2, \ldots, N_{\mathrm{M}}$), where N_{T} represents the number of different chemical species; $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ stand for the value and derivative value of $\boldsymbol{h}_{ji}$, respectively, when $R_{ji}^2 = r_k$ (here $r_k = (k-1)\cdot R_c^2 / N_{\mathrm{M}}$). Then, when R_{ji}^2 ($r_k \le R_{ji}^2 < r_{k+1}$) and Z_j is entered, one of the mapping tables is enabled according to Z_j and its k^{th} row data (i.e., $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$) is fetched. Finally, $\boldsymbol{h}_{ji}$ is computed via the formula $\boldsymbol{h}_{ji} = (R_{ji}^2 - r_k)\cdot \boldsymbol{b}_k + \boldsymbol{a}_k$. The interpolation method employs a few mapping tables and multiplication to replace the complex computation of trigonometric function (Eq. (5)) and FeaNN (Eq. (6)), and it is utilized to compute $\{s_{ji}\}$ and $\{\boldsymbol{g}_{ji}\}$ from $\{R_{ji}^2\}$ in the M2. The N_{M} is set at 1024 to strike a compromise between accuracy loss and resource usage.

The digital signal processor (DSP)[109] resources are used to implement multiplication in M1, M2, M3, M4, and M5. The on-chip memory UltraRAM (URAM)[105] is used to implement the mapping table in M2. Look-Up Table (LUT)[110] implements FitNN's matrix multiplication in the M6, and FitNN's weights and biases are stored in on-chip Look-Up Table RAM (LUTRAM)[110] to avoid frequent fetching from off-chip memory. There is no need to temporarily store the intermediate results in the off-chip memory since the output of the former module is the input of the latter.

Force and virial calculation

The atomic force is defined as the negative gradient of energy, so the force component $\{\boldsymbol{F}_{ji}\}$ is calculated in the backward propagation of the models (i.e., M1, M2, M3, M4, M5, and M6) in SPU (Fig. 8b). In order to perform both FP and BP calculation in these modules, each module consists of three submodules: FP, BP, and first-in-first-out (FIFO) (Fig. 8c). FP and BP are used to execute the calculation of forward propagation and backward propagation, respectively. FIFO is employed to transmit the required intermediate results calculated by FP to BP. As illustrated in Fig. 8c, in the FP, the input of k^{th} module is χ^k, and the output χ^{k+1} is calculated according to the expression corresponding to the module in the forward propagation; in the BP, the input is $\partial E_i/\partial \chi^{k+1}$, and the output $\partial E_i/\partial \chi^k$ is calculated by using chain rule as $\partial E_i/\partial \chi^k = \partial E_i/\partial \chi^{k+1} \times \partial \chi^{k+1}/\partial \chi^k$; in the FIFO, the intermediate results ${\chi_{\mathrm{m}}}^k$ is transmitted in order to calculated $\partial \chi^{k+1}/\partial \chi^k$.

Using the module structure shown in Fig. 8c, the gradient of each module's input is computed in the backward propagation (Fig. 8b). More specially, in the M6, FIFO transmits Z_i and the input of each layer's activation function, and BP computes $\partial E_i/\partial \boldsymbol{D}_i$. In the M5, FIFO transmits $\boldsymbol{U}_i$, and BP computes $\partial E_i/\partial \boldsymbol{U}_i$. In the M4, FIFO transmits $\{\boldsymbol{u}_{ji}\}$ and $\{\boldsymbol{g}_{ji}\}$, and BP computes $\{\partial E_i/\partial \boldsymbol{u}_{ji}\}$ and $\{\partial E_i/\partial \boldsymbol{g}_{ji}\}$. In the M3, FIFO transmits $\{s_{ji}\}$ and $\{\boldsymbol{R}_{ji}\}$, and BP computes $\{\partial E_i/\partial s_{ji}\}$ and $\{\partial E_i/\partial \boldsymbol{u}_{ji} \times \partial \boldsymbol{u}_{ji}/\partial \boldsymbol{R}_{ji}\}$. In the M2, FIFO transmits $\{R_{ji}^2\}$ and $\{Z_j\}$, and BP computes $\{\partial E_i/\partial R_{ji}^2\}$. It's worth noting that $\{\partial s_{ji}/\partial R_{ji}^2\}$ and $\{\partial \boldsymbol{g}_{ji}/\partial R_{ji}^2\}$ are obtained by using the interpolation method proposed in Section Energy calculation. In the M1, FIFO transmits $\{\boldsymbol{R}_{ji}\}$, and BP computes $\{\boldsymbol{F}_{ji}\} = \{(\partial E_i/\partial R_{ji}^2 \times \partial R_{ji}^2/\partial \boldsymbol{R}_{ji}) + (\partial E_i/\partial \boldsymbol{u}_{ji} \times \partial \boldsymbol{u}_{ji}/\partial \boldsymbol{R}_{ji})\}$ and $\{\boldsymbol{V}_{ji}\}$.

In the BP, the matrix multiplication in the FitNN is implemented by LUT, and other multiplication is realized by DSP resources; on-chip LUTRAM is used to hold the FitNN's parameters; on-chip URAM is used to implement the mapping tables in the M2. The parameters and the mapping tables only need to be initialized once at the start of NVNMD, and they don't need to be fetched from off-chip memory on a regular basis. In the FIFO,

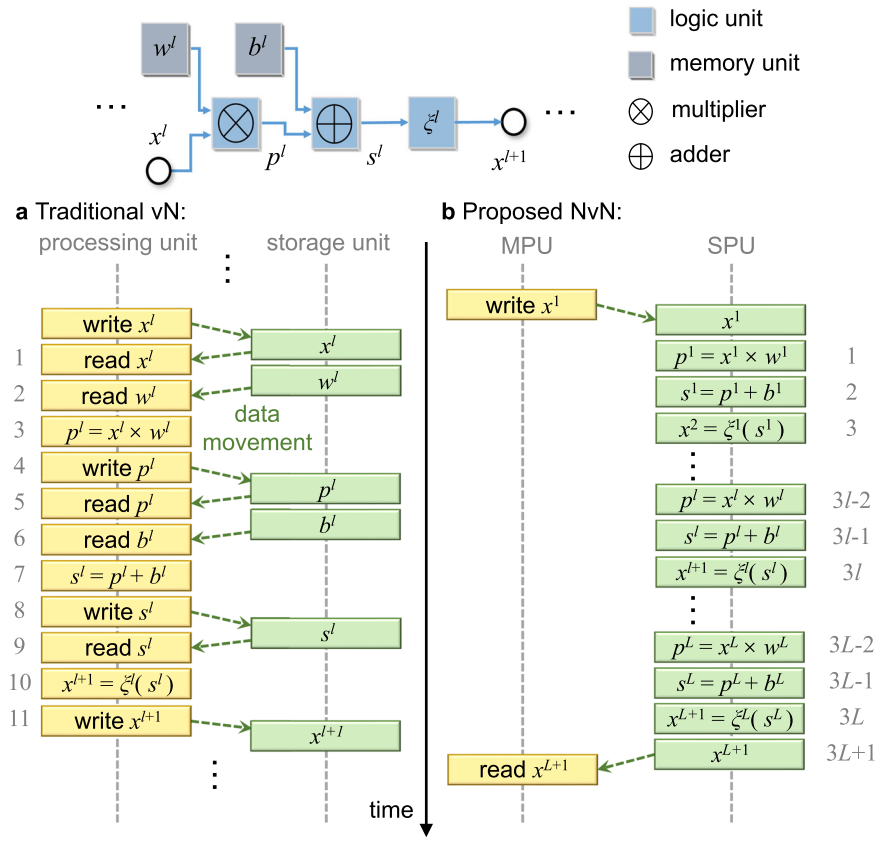

Fig. 9 Schematic comparison of calculation step between vN and NvN architecture. Equation (7) is used as an example to indicate the difference between traditional vN architecture (**a**) and proposed NvN architecture (**b**) based on the processing-in-memory (PIM) technology. Here x^l, w^l, b^l, and ξ^l are input, weight, bias, and nonlinear activation function of the l^{th} layer, respectively; $p^l = x^l \times w^l$; $s^l = b^l + p^l$.

on-chip URAM is also used to implement the function which transmits the data from FP to BP to avoid communication with off-chip memory. The entire procedure is designed to run in pipeline mode for optimal performance.

Processing in memory

If the energy, force, and virial (Sections Energy calculation and Force and virial calculation) are calculated on traditional vN computers, the efficiency is very low. For instance, to calculate the l^{th} layer of MLP (i.e., Eq. (7)), it needs 11 steps, as shown in Fig. 9. These calculations are not as efficient as they could be, because the data storage unit needs to be accessed 8 times (i.e., steps 1, 2, 4, 5, 6, 8, 9, and 11 in Fig. 9a). Due to the limited size of on-chip memory (e.g., cache) of the vN processing unit (e.g., CPU/GPU), the system has to frequently access the off-chip data storage unit (e.g., main memory), which is typically two orders of magnitude slower than the processing unit (i.e., well known as the vNB).

To overcome the vNB, the proposed NvN-based SPU leverages the processing-in-memory (PIM) technology to avoid heavy-duty data shuttling[111–115]. Specifically, the logic devices and memory cells are integrated together, vanishing the data fetching latency of its vN counterparts. In the proposed NvN-based SPU, w^l and b^l are stored in the local on-chip memory and x^{l+1} is directly used as the input of the $(l+1)^{th}$ layer of MLP without accessing off-chip memory (Fig. 9), such that the repeated data shuttling from/to the off-chip memory (i.e., vNB) can be avoided. It's worth noting that the parameters such as w^l and b^l represent the high-dimensional PES, which are material-dependent. Thus, to compute a long MD trajectory of a particular material, w^l and b^l are only loaded once from off-chip memory and then kept unchanged in on-chip memory during all timesteps of the MD trajectory. The logic and arithmetic operations (e.g., multipliers, adders, and activation functions) are implemented using reconfigurable circuit, to link on-chip memory cells (e.g., w^l and b^l). Using PIM, the calculation is pipelined without interruption of data shuttling latency, such that the calculation time is consumed purely for useful logic and arithmetic operations and, thus, the efficiency is maximized.

Quantized neural network

To implement NvN PIM (Fig. 9b), it is very hardware resource-consuming if variables (e.g., x^l, w^l, b^l, p^l, etc.) were represented using floating-point numbers[116]. So, despite continuous neural network (CNN) based on floating-point numbers is adopted in nearly all existing MLMD, we use the quantized neural network (QNN), which has been proposed to replace CNN in hardware devices with limited power supply and computational resources[117–119]. In the QNN, the weights and activations are quantized to save power consumption and computation resources. For example, we use quantization

$$\chi^q = \sigma_\gamma(\chi) = \lfloor \chi \times 2^\gamma \rfloor \tag{13}$$

for floating-point number χ, where χ^q is quantized value with the precision $2^{-\gamma}$; $\lfloor x \rfloor$ is floor function which gives the greatest integer less than or equal to x. The quantization parameter γ is determined by the trade-off between accuracy and resources. We found that, by setting $\gamma = 13$, there is negligibly small accuracy loss after replacing CNN using QNN in the proposed NVNMD.

Multiplication-less neural network

To implement NvN PIM (Fig. 9b), it is also very hardware resource-consuming if the multiplication operations were realized in the arithmetic circuit directly[48,49]. So, despite multiplication-based neural network is adopted in nearly all existing MLMD, we propose a multiplication-less neural network, which is specially designed for the proposed NVNMD, in order to reduce the hardware circuit complexity and power consumption. Specifically, to evaluate the $(3l\text{-}1)^{th}$ step in Fig. 9b, the multiplication operation '×' is replaced by using the bitwise shift operation '>>', to evaluate

$$p^q = \lfloor (w^q \times x^q)/2^\gamma \rfloor = \left(\sum_{k=1}^{K} s_k \cdot x^q \cdot 2^{n_k} \right) >> \gamma = \left(\sum_{k=1}^{K} (s_k \cdot x^q) << n_k \right) >> \gamma \tag{14}$$

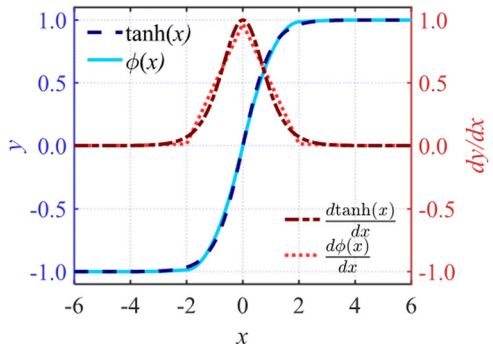

Fig. 10 The comparison between two activation functions. The normalized $\phi(x)$ is close to tanh(x) at the numerical value and first derivative.

where x^q is quantized input of the layer; γ is the quantization parameter (Eq. (13));

$$w^q = \zeta_K(\sigma_\gamma(w)) = \sum_{k=1}^{K} s_k \cdot 2^{n_k} \tag{15}$$

is the quantized weights of QNN; $s_k = -1$, 0, or 1 is the sign; n_k is a natural number;

$$\zeta_K(x) = \begin{cases} \zeta_{K-1}(x - \zeta(x)) + \zeta(x), & K > 1 \\ \zeta(x), & K = 1 \end{cases} \tag{16}$$

is the quantization function;

$$\zeta(x) = s \cdot 2^n = \begin{cases} 1 \cdot 2^{\lfloor \log_2(x/1.5) \rfloor}, & x > 0 \\ 0 \cdot 2^0, & x = 0 \\ -1 \cdot 2^{\lfloor \log_2(-x/1.5) \rfloor}, & x < 0 \end{cases} \tag{17}$$

is used to quantize value to exponent of 2; and $\lceil x \rceil$ is ceiling function which rounds x to upper integer.

Obviously, in the above multiplication-less design, the multiplication operation is replaced by bitwise shift and summation operations, which are much more resource-economical and energy-saving in digital circuit. Our test shows that if K is too small (say, $K = 1$ or 2), there is serious accuracy loss; if $K \geq 3$, the accuracy is decent to fit high-dimensional PES. So, we use $K = 3$ hereafter.

Nonlinear activation function

To implement NvN PIM (Fig. 9b), it is also very hardware resource-consuming if the trigonometric function-based nonlinear activation functions (e.g., tanh(x)) are implemented directly[120]. So, despite that these trigonometric function-based nonlinear activation functions are widely-used in existing MLMD, we design a nonlinear activation function (Fig. 10)

$$\begin{aligned} \phi_{\alpha,\beta,\gamma}(x) &= \frac{x_\gamma}{\alpha} - \frac{x_\gamma \cdot |x_\gamma|}{\beta}, x_\gamma = \begin{cases} \gamma, & x \geq \gamma \\ x, & -\gamma < x < \gamma \\ -\gamma, & x \leq -\gamma \end{cases} \\ \phi(x) &= \phi_{1,4,2}(x) + \phi_{32,256,4}(x) \end{aligned} \tag{18}$$

without trigonometric functions. In order to implement in NvN-based SPU, we redesign an activation function with continuous value and first derivative, and make it easier to use in training and prediction with fewer calculations. Because its parameters are exponents of 2, the shift operation can be used instead of the relevant multiplication and division. The most complex operation is just multiplication, not exponentiation and division in this activation function. It is easy to implement $\phi(x)$ in training and testing processes on vN-based and NvN-based computer. The curve of tanh(x) and $\phi(x)$ are compared in Fig. 10, where $\phi(x)$ is normalized to the range $[-1, 1]$ by dividing 1.0625 (max value of $\phi(x)$). Obviously, at the numerical value and first derivative, the tanh and $\phi(x)$ are similar.

Hardware implementation

To implement the heterogeneous vN/NvN (Fig. 6), we use vN-based CPU (Intel i7-10700K, 3.80 GHz, 8 cores) and NvN-based FPGA (Xilinx xcvu9p) together. The MPU in Fig. 6 is implemented by using CPU; and the SPU is implemented by using FPGA. For the neural network model deployed in SPU (Section Slave processing unit), the maximum number of neighbor atoms is set to 128; The number of FeaNN output nodes is M = 20; The $\boldsymbol{D}_i$ dimension is set as 20×10; FitNN contains three hidden layers, each having 20 nodes. The time division multiplexing (TDM) technology is adopted to reduce the number of resources[121,122]. By optimizing the design, the number of resources is reduced, the timing is improved, and the clock frequency of 250 MHz is achieved. The number of resources consumed by the whole design is shown in Table 7.

DATA AVAILABILITY

To reproduce the results in this paper, training and inference calculations are needed. The training codes and data are open-sourced at https://github.com/LiuGroupHNU/nvnmd, for generating the NVNMD-oriented inter-atomic potential models. The inference functionalities (i.e., NVNMD calculations) can be freely accessed at http://nvnmd.picp.vip.

Received: 9 December 2021; Accepted: 29 March 2022;
Published online: 09 May 2022

REFERENCES

1. Bapst, V. et al. Unveiling the predictive power of static structure in glassy systems. *Nat. Phys.* **16**, 448–454 (2020).
2. Schott, S. et al. Polaron spin dynamics in high-mobility polymeric semiconductors. *Nat. Phys.* **15**, 814–822 (2019).
3. Galib, M. & Limmer, D. T. Reactive uptake of N_2O_5 by atmospheric aerosol is dominated by interfacial processes. *Science* **371**, 921–925 (2021).
4. Widmer, D. R. & Schwartz, B. J. Solvents can control solute molecular identity. *Nat. Chem.* **10**, 910–916 (2018).
5. Karplus, M. & Petsko, G. A. Molecular dynamics simulations in biology. *Nature* **347**, 631–639 (1990).
6. Chen, S. et al. Simultaneously enhancing the ultimate strength and ductility of high-entropy alloys via short-range ordering. *Nat. Commun.* **12**, 4953 (2021).
7. Ding, W. et al. Prediction of intrinsic two-dimensional ferroelectrics in In_2Se_3 and other III2-VI3 van der Waals materials. *Nat. Commun.* **8**, 14956 (2017).
8. Wang, Y. et al. Dynamic deformability of individual PbSe nanocrystals during superlattice phase transitions. *Sci. Adv.* **5**, eaaw5623 (2019).
9. Lehtinen, O., Kurasch, S., Krasheninnikov, A. V. & Kaiser, U. Atomic scale study of the life cycle of a dislocation in graphene from birth to annihilation. *Nat. Commun.* **4**, 2098 (2013).
10. Lu, S. et al. Activation pathway of a G protein-coupled receptor uncovers conformational intermediates as targets for allosteric drug design. *Nat. Commun.* **12**, 4721 (2021).
11. Zhao, Y. et al. Augmenting drug–carrier compatibility improves tumour nanotherapy efficacy. *Nat. Commun.* **7**, 11221 (2016).
12. Laio, A., Bernard, S., Chiarotti, G. L., Scandolo, S. & Tosatti, E. Physics of iron at Earth's core conditions. *Science* **287**, 1027–1030 (2000).
13. Steinle-Neumann, G., Stixrude, L., Cohen, R. E. & Gülseren, O. Elasticity of iron at the temperature of the Earth's inner core. *Nature* **413**, 57–60 (2001).
14. Hughes, M. A. et al. n-type chalcogenides by ion implantation. *Nat. Commun.* **5**, 5346 (2014).
15. Wang, X.-P. et al. Time-dependent density-functional theory molecular-dynamics study on amorphization of Sc-Sb-Te alloy under optical excitation. *npj Comput. Mater.* **6**, 31 (2020).
16. Kohn, W. & Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **140**, A1133–A1138 (1965).
17. Car, R. & Parrinello, M. Unified approach for molecular dynamics and density-functional theory. *Phys. Rev. Lett.* **55**, 2471–2474 (1985).
18. Alavi, S. Ab initio molecular dynamics basic theory and advanced methods. By Dominik Marx and Jürg Hutter. *Angew. Chem. Int. Ed.* **48**, 9404–9405 (2009).
19. Jorgensen, W. L., Maxwell, D. S. & Tirado-Rives, J. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* **118**, 11225–11236 (1996).
20. Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A. & Case, D. A. Development and testing of a general Amber force field. *J. Comput. Chem.* **25**, 1157–1174 (2004).
21. Vanommeslaeghe, K. et al. CHARMM general force field: a force field for drug-like molecules compatible with the CHARMM all-atom additive biological force fields. *J. Comput. Chem.* **31**, 671–690 (2010).
22. Shaw, D. E. et al. Anton, a special-purpose machine for molecular dynamics simulation. *Commun. ACM* **51**, 91–97 (2008).
23. Shaw, D. E. et al. Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. in *SC14: International*

Conference for High Performance Computing, Networking, Storage and Analysis **2015-January**, 41–53 (IEEE, 2014).
24. Shaw, D. E. et al. Anton 3: twenty microseconds of molecular dynamics simulation before lunch. in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* 1–11 (ACM, 2021). https://doi.org/10.1145/3458817.3487397.
25. Behler, J. & Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **98**, 146401 (2007).
26. Wang, H., Zhang, L., Han, J. & E, W. DeePMD-kit: a deep learning package for many-body potential energy representation and molecular dynamics. *Comput. Phys. Commun.* **228**, 178–184 (2018).
27. Zhang, L., Han, J., Wang, H., Car, R. & E, W. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.* **120**, 143001 (2018).
28. Zhang, L., Lin, D.-Y., Wang, H., Car, R. & E, W. Active learning of uniformly accurate interatomic potentials for materials simulation. *Phys. Rev. Mater.* **3**, 023804 (2019).
29. Zhang, Y. et al. DP-GEN: a concurrent learning platform for the generation of reliable deep learning based potential energy models. *Comput. Phys. Commun.* **253**, 107206 (2020).
30. Zhang, L. et al. End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems. *Adv. Neural Inf. Process. Syst.* **2018-December**, 4436–4446 (2018).
31. Jia, W. et al. Pushing the Limit of Molecular Dynamics with Ab Initio Accuracy to 100 Million Atoms with Machine Learning. in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* 1–14 (IEEE, 2020). https://doi.org/10.1109/SC41405.2020.00009.
32. LAMMPS Benchmarks. Available at: https://www.lammps.org/bench.html.
33. Wulf, W. A. & McKee, S. A. Hitting the memory wall. *ACM SIGARCH Comput. Archit. N.* **23**, 20–24 (1995).
34. Horowitz, M. 1.1 Computing's energy problem (and what we can do about it). in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* **57**, 10–14 (IEEE, 2014).
35. Ielmini, D. & Wong, H. S. P. In-memory computing with resistive switching devices. *Nat. Electron.* **1**, 333–343 (2018).
36. Stegailov, V., Smirnov, G. & Vecher, V. VASP hits the memory wall: processors efficiency comparison. *Concurr. Comput. Pract. Exp.* **31**, e5136 (2019).
37. John von Neumann. *First Draft of a Report on the EDVAC*. (1945).
38. Electronic Numerical Integrator and Computer (ENIAC). Available at: https://en.wikipedia.org/wiki/ENIAC.
39. Beyond von Neumann. *Nat. Nanotechnol.* **15**, 507–507 (2020).
40. Taiji, M. et al. Protein Explorer: A Petaflops Special-Purpose Computer System for Molecular Dynamics Simulations. in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing - SC '03* 15 (ACM Press, 2003). https://doi.org/10.1145/1048935.1050166.
41. Harvey, M. J., Giupponi, G. & De Fabritiis, G. ACEMD: Accelerating biomolecular dynamics in the microsecond time scale. *J. Chem. Theory Comput.* **5**, 1632–1639 (2009).
42. Deringer, V. L. & Csányi, G. Machine learning based interatomic potential for amorphous carbon. *Phys. Rev. B* **95**, 094203 (2017).
43. Rowe, P., Csányi, G., Alfè, D. & Michaelides, A. Development of a machine learning potential for graphene. *Phys. Rev. B* **97**, 054303 (2018).
44. Zeng, J., Cao, L., Xu, M., Zhu, T. & Zhang, J. Z. H. Complex reaction processes in combustion unraveled by neural network-based molecular dynamics simulation. *Nat. Commun.* **11**, 5713 (2020).
45. Li, R., Lee, E. & Luo, T. A unified deep neural network potential capable of predicting thermal conductivity of silicon in different phases. *Mater. Today Phys.* **12**, 100181 (2020).
46. Rowe, P., Deringer, V. L., Gasparotto, P., Csányi, G. & Michaelides, A. An accurate and transferable machine learning potential for carbon. *J. Chem. Phys.* **153**, 034702 (2020).
47. Bettini, J. et al. Experimental realization of suspended atomic chains composed of different atomic species. *Nat. Nanotechnol.* **1**, 182–185 (2006).
48. Wu, B. et al. Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions. in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* 9127–9135 (IEEE, 2018). https://doi.org/10.1109/CVPR.2018.00951
49. Chen, H. et al. AdderNet: Do We Really Need Multiplications in Deep Learning? in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 1465–1474 (IEEE, 2020). https://doi.org/10.1109/CVPR42600.2020.00154
50. Ahn, J., Yoo, S., Mutlu, O. & Choi, K. PIM-enabled instructions. in *Proceedings of the 42nd Annual International Symposium on Computer Architecture* **43**, 336–348 (ACM, 2015).
51. Mutlu, O., Ghose, S., Gómez-Luna, J. & Ausavarungnirun, R. Processing data where it makes sense: Enabling in-memory computation. *Microprocess. Microsyst.* **67**, 28–41 (2019).
52. Liu, J. & Mo, P. The server website of NVNMD. (2021). Available at: http://nvnmd.picp.vip/.
53. Liu, J. & Mo, P. The training and testing code for NVNMD. (2021). Available at: https://github.com/LiuGroupHNU/nvnmd.
54. Abadi, M. et al. TensorFlow: A system for large-scale machine learning. *Proc. 12th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2016* 265–283 (2016). https://doi.org/10.5555/3026877.3026899
55. Sosso, G. C., Miceli, G., Caravati, S., Behler, J. & Bernasconi, M. Neural network interatomic potential for the phase change material GeTe. *Phys. Rev. B* **85**, 174103 (2012).
56. Shi, M., Mo, P. & Liu, J. Deep Neural Network for Accurate and Efficient Atomistic Modeling of Phase Change Memory. *IEEE Electron Device Lett.* **41**, 365–368 (2020).
57. Plimpton, S. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics* **117**, (1993).
58. Chmiela, S. et al. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **3**, e1603015 (2017).
59. Chmiela, S., Sauceda, H. E., Müller, K.-R. & Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **9**, 3887 (2018).
60. Christensen, A. S. & von Lilienfeld, O. A. On the role of gradients for machine learning of molecular energies and forces. *Mach. Learn. Sci. Technol.* **1**, 045018 (2020).
61. Shi, M., Li, J., Tao, M., Zhang, X. & Liu, J. Artificial intelligence model for efficient simulation of monatomic phase change material antimony. *Mater. Sci. Semicond. Process.* **136**, 106146 (2021).
62. Huang, J. et al. Deep potential generation scheme and simulation protocol for the $Li_{10}GeP_2S_{12}$-type superionic conductors. *J. Chem. Phys.* **154**, 094703 (2021).
63. Bogojeski, M., Vogt-Maranto, L., Tuckerman, M. E., Müller, K.-R. & Burke, K. Quantum chemical accuracy from density functional approximations via machine learning. *Nat. Commun.* **11**, 5223 (2020).
64. Narayanan, B., Redfern, P. C., Assary, R. S. & Curtiss, L. A. Accurate quantum chemical energies for 133000 organic molecules. *Chem. Sci.* **10**, 7449–7455 (2019).
65. Morawietz, T. & Artrith, N. Machine learning-accelerated quantum mechanics-based atomistic simulations for industrial applications. *J. Comput. -Aided Mol. Des.* **35**, 557–586 (2021).
66. Zhang, P., Shen, L. & Yang, W. Solvation Free Energy Calculations with Quantum Mechanics/Molecular Mechanics and Machine Learning Models. *J. Phys. Chem. B* **123**, 901–908 (2019).
67. Lu, C. et al. OPLS4: Improving force field accuracy on challenging regimes of chemical space. *J. Chem. Theory Comput.* **17**, 4291–4300 (2021).
68. Soler, J. M. et al. The SIESTA method for ab initio order-N materials simulation. *J. Phys. Condens. Matter* **14**, 2745–2779 (2002).
69. Giannozzi, P. et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys. Condens. Matter* **21**, 395502 (2009).
70. VandeVondele, J. et al. Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. *Comput. Phys. Commun.* **167**, 103–128 (2005).
71. Ahn, S. *Phase Change Memory*. (Springer International Publishing, 2018). https://doi.org/10.1007/978-3-319-69053-7.
72. Kolobov, A. V., Krbal, M., Fons, P., Tominaga, J. & Uruga, T. Distortion-triggered loss of long-range order in solids with bonding energy hierarchy. *Nat. Chem.* **3**, 311–316 (2011).
73. Mo, Y., Ong, S. P. & Ceder, G. First principles study of the Li 10GeP 2S 12 lithium super ionic conductor material. *Chem. Mater.* **24**, 15–17 (2012).
74. Marcolongo, A., Binninger, T., Zipoli, F. & Laino, T. Simulating Diffusion Properties of Solid-State Electrolytes via a Neural Network Potential: Performance and Training Scheme. *ChemSystemsChem* **2**, e1900031 (2020).
75. Kamaya, N. et al. A lithium superionic conductor. *Nat. Mater.* **10**, 682–686 (2011).
76. NVIDIA Corporation. Nvidia Tesla V100 GPU Volta Architecture. *White Paper* 53 (2017). Available at: https://images.nvidia.cn/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.
77. Summit. Available at: https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.
78. NVIDIA. NVIDIA V100. Available at: https://www.nvidia.com/en-us/data-center/v100/.
79. Xilinx. UltraScale Architecture and Product Data Sheet: Overview. *Xilinx.com* 1–46 (2020). Available at: https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.
80. Xilinx. UltraScale+ FPGAs Product Tables and Product Selection Guide. *Xilinx.com* 1–11 (2021). Available at: https://www.xilinx.com/support/documentation/selection-guides/ultrascale-plus-fpga-product-selection-guide.pdf.
81. Ic, S. P., Dube, B., Elisabeth, S. & Scansen, D. Apple M1 System-on-Chip. *systemplus.fr* 1–36 (2020). Available at: https://www.systemplus.fr/wp-content/uploads/2020/12/SP20608-Apple-M1-System-on-Chip-Sample.pdf.

82. Lu, D. et al. 86 PFLOPS Deep Potential Molecular Dynamics simulation of 100 million atoms with ab initio accuracy. *Comput. Phys. Commun.* **259**, 107624 (2021).
83. Samir, N. et al. ASIC and FPGA Comparative Study for IoT lightweight hardware security algorithms. *J. Circuits, Syst. Comput*. **28**, (2019).
84. Schütt, K. T. et al. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *Adv. Neural Inf. Process. Syst*. **2017-Decem**, 992–1002 (2017).
85. Klicpera, J., Groß, J. & Günnemann, S. Directional Message Passing for Molecular Graphs. Preprint at http://arxiv.org/abs/2003.03123 (2020).
86. Chmiela, S., Sauceda, H. E., Poltavsky, I., Müller, K. R. & Tkatchenko, A. sGDML: Constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* **240**, 38–45 (2019).
87. Schütt, K., Unke, O. & Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. in *Proceedings of the 38th International Conference on Machine Learning* (Vol. 139 eds. Meila, M. & Zhang, T.) 9377–9388 (PMLR, 2021).
88. Unke, O. T. et al. SpookyNet: Learning force fields with electronic degrees of freedom and nonlocal effects. *Nat. Commun.* **12**, 7273 (2021).
89. Klicpera, J., Becker, F. & Günnemann, S. GemNet: Universal Directional Graph Neural Networks for Molecules. Preprint at http://arxiv.org/abs/2106.08903 (2021).
90. Haghighatlari, M. et al. NewtonNet: A Newtonian message passing network for deep learning of interatomic potentials and forces. Preprint at http://arxiv.org/abs/2108.02913 (2021).
91. Qiao, Z. et al. UNiTE: Unitary N-body Tensor Equivariant Network with Applications to Quantum Chemistry. Preprint at http://arxiv.org/abs/2105.14655 (2021).
92. Batzner, S. et al. E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. Preprint at http://arxiv.org/abs/2101.03164 (2021).
93. Kanal, I. Y., Keith, J. A. & Hutchison, G. R. A sobering assessment of small-molecule force field methods for low energy conformer predictions. *Int. J. Quantum Chem.* **118**, e25512 (2018).
94. Zgarbová, M., Otyepka, M., Šponer, J., Hobza, P. & Jurečka, P. Large-scale compensation of errors in pairwise-additive empirical force fields: Comparison of AMBER intermolecular terms with rigorous DFT-SAPT calculations. *Phys. Chem. Chem. Phys.* **12**, 10476–10493 (2010).
95. Demir, H. et al. DFT-based force field development for noble gas adsorption in metal organic frameworks. *J. Mater. Chem. A* **3**, 23539–23548 (2015).
96. Shen, L. & Yang, W. Molecular Dynamics Simulations with Quantum Mechanics/Molecular Mechanics and Adaptive Neural Networks. *J. Chem. Theory Comput.* **14**, 1442–1455 (2018).
97. Jinnouchi, R., Karsai, F. & Kresse, G. Making free-energy calculations routine: combining first principles with machine learning. *Phys. Rev. B* **101**, 060201 (2020).
98. Han, J., Zhang, L., Car, R. & E, W. Deep potential: a general representation of a many-body potential energy surface. *Commun. Comput. Phys.* **23**, 629–639 (2018).
99. Allen, M. P. & Tildesley, D. J. *Computer Simulation of Liquids*. **1**, (Oxford University Press, 2017).
100. Parrinello, M. & Rahman, A. Polymorphic transitions in single crystals: a new molecular dynamics method. *J. Appl. Phys.* **52**, 7182–7190 (1981).
101. Martyna, G. J., Tobias, D. J. & Klein, M. L. Constant pressure molecular dynamics algorithms. *J. Chem. Phys.* **101**, 4177–4189 (1994).
102. Dullweber, A., Leimkuhler, B. & McLachlan, R. Symplectic splitting methods for rigid body molecular dynamics. *J. Chem. Phys.* **107**, 5840–5851 (1997).
103. Shinoda, W., Shiga, M. & Mikami, M. Rapid estimation of elastic constants by molecular dynamics simulation under constant stress. *Phys. Rev. B* **69**, 134103 (2004).
104. Tuckerman, M. E., Alejandre, J., López-Rendón, R., Jochim, A. L. & Martyna, G. J. A Liouville-operator derived measure-preserving integrator for molecular dynamics simulations in the isothermal-isobaric ensemble. *J. Phys. A. Math. Gen.* **39**, 5629–5651 (2006).
105. Xilinx. UltraScale Architecture: Memory Resources User Guide (UG573). **573**, 1–136 (2018).
106. Goldhammer, A. & Ayer, J. Jr. Understanding performance of PCI express systems. *Xilinx WP350* **350**, 1–18 (2008).
107. Xilinx, P. C. I. Express for ultrascale architecture-based devices integrated block for PCIe in the ultrascale. *Architecture* **464**, 1–15 (2015).
108. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**, 359–366 (1989).
109. Xilinx. UltraScale Architecture: DSP Slice User Guide (UG579). *Xilinx.com* (2020). Available at: https://www.xilinx.com/support/documentation/user_guides/ug579-ultrascale-dsp.pdf.
110. Xilinx. UltraScale Architecture Configurable Logic Block User Guide (UG574). *Xilinx.com* (2017). Available at: https://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf.
111. Chi, P. et al. PRIME: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. *Proceedings of the 2016 43rd Int. Symp. Comput. Archit. ISCA 2016* 27–39 (2016). https://doi.org/10.1109/ISCA.2016.13
112. Ghose, S., Boroumand, A., Kim, J. S., Gomez-Luna, J. & Mutlu, O. Processing-in-memory: a workload-driven perspective. *IBM J. Res. Dev.* **63**, 3 (2019).
113. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529–544 (2020).
114. Lu, Z., Arafin, M. T. & Qu, G. RIME: A Scalable and Energy-Efficient Processing-In-Memory Architecture for Floating-Point Operations. *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC* 120–125 (2021). https://doi.org/10.1145/3394885.3431524
115. Bavikadi, S., Sutradhar, P. R., Khasawneh, K. N., Ganguly, A. & Dinakarrao, S. M. P. A review of in-memory computing architectures for machine learning applications. *Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI* 89–94 (2020). https://doi.org/10.1145/3386263.3407649
116. Are, W., Point, F. & Layout, S. IEEE Standard 754 Floating Point Numbers. 1–7 (2011).
117. Gupta, S., Agrawal, A., Gopalakrishnan, K. & Narayanan, P. Deep learning with limited numerical precision. *32nd Int. Conf. Mach. Learn. ICML 2015* **3**, 1737–1746 (2015).
118. Han, S., Mao, H. & Dally, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *Int. Conf. Learn. Represent*. 1–14 (2016).
119. Alemdar, H., Leroy, V., Prost-Boucle, A. & Petrot, F. Ternary neural networks for resource-efficient AI applications. *Proc. Int. Jt. Conf. Neural Networks* **2017-May**, 2547–2554 (2017).
120. Marra, S., Iachino, M. A. & Morabito, F. C. High speed, programmable implementation of a tanh-like activation function and its derivative for digital neural networks. *IEEE Int. Conf. Neural Networks - Conf. Proc.* 506–511 (2007). https://doi.org/10.1109/IJCNN.2007.4371008
121. Zheng, D., Zhang, X., Pui, C. W. & Young, E. F. Y. Multi-FPGA Co-optimization: Hybrid Routing and Competitive-based Time Division Multiplexing Assignment. *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC* 176–182 (2021). https://doi.org/10.1145/3394885.3431565
122. Zou, P. et al. Time-Division Multiplexing Based System-Level FPGA Routing for Logic Verification. in *2020 57th ACM/IEEE Design Automation Conference (DAC)* **2020-July**, 1–6 (IEEE, 2020).
123. Lee, K., Yoo, D., Jeong, W. & Han, S. SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials. *Comput. Phys. Commun.* **242**, 95–103 (2019).
124. Lu, D. et al. DP Train, then DP Compress: Model Compression in Deep Potential Molecular Dynamics. Preprint at http://arxiv.org/abs/2107.02103 (2021).
125. Sedova, A., Eblen, J. D., Budiardja, R., Tharrington, A. & Smith, J. C. High-performance molecular dynamics simulation for biological and materials sciences: Challenges of performance portability. *Proc. P3HPC 2018 Int. Work. Performance, Portability Product. HPC, Held conjunction with SC 2018 Int. Conf. High Perform. Comput. Networking, Storage Anal*. 1–13 (2019). https://doi.org/10.1109/P3HPC.2018.00004

ACKNOWLEDGEMENTS

We thank Han Wang, Linfeng Zhang, Denghui Lu, Wanrun Jiang, Jun Cheng, Yongbin Zhuang, and Jianxing Huang for their precious time to try and test NVNMD, and for their helpful suggestions to improve NVNMD. We thank experts from the DeePMD community for their helpful discussions and technical support. This work is supported by the National Natural Science Foundation of China (#61804049); the Fundamental Research Funds for the Central Universities of P.R. China; Huxiang High Level Talent Gathering Project (#2019RS1023); the Key Research and Development Project of Hunan Province, P.R. China (#2019GK2071); the Technology Innovation and Entrepreneurship Funds of Hunan Province, P.R. China (#2019GK5029); the Fund for Distinguished Young Scholars of Changsha (#kq1905012).

AUTHOR CONTRIBUTIONS

Pinghui Mo, Chang Li, Dan Zhao, Yujia Zhang, and Jie Liu implemented and tested the NVNMD system; Mengchao Shi and Junhua Li generated the DFT data for training and testing the NVNMD system; Jie Liu proposed the idea and led the research; Pinghui Mo and Jie Liu composed the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Correspondence and requests for materials should be addressed to Jie Liu.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.