Parallel Transport Time-Dependent Density Functional Theory Calculations with Hybrid Functional on Summit

Weile Jia
jiaweile@berkeley.edu
University of California, Berkeley
Berkeley, California

Lin-Wang Wang
lwwang@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, California

Lin Lin*
linlin@math.berkeley.edu
University of California, Berkeley
Lawrence Berkeley National Laboratory
Berkeley, California

ABSTRACT

Real-time time-dependent density functional theory (rt-TDDFT) with hybrid exchange-correlation functional has wide-ranging applications in chemistry and material science simulations. However, it can be thousands of times more expensive than a conventional ground state DFT simulation, and hence is limited to small systems. In this paper, we accelerate hybrid functional rt-TDDFT calculations using the parallel transport gauge formalism, and the GPU implementation on Summit. Our implementation can efficiently scale to 786 GPUs for a large system with 1536 silicon atoms, and the wall clock time is only 1.5 hours per femtosecond. This unprecedented speed enables the simulation of large systems with more than 1000 atoms using rt-TDDFT and hybrid functional.

KEYWORDS

Time-dependent density functional theory, real-time, GPU, Non-equilibrium system, Hybrid exchange-correlation functional, Fock exchange operator

ACM Reference Format:
Weile Jia, Lin-Wang Wang, and Lin Lin. 2019. Parallel Transport Time-Dependent Density Functional Theory Calculations with Hybrid Functional on Summit. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3295500.3356144

1 INTRODUCTION

Real-time time-dependent density functional theory(rt-TDDFT) is one of the newest trends in electronic structure calculations [4, 31, 35, 41, 46]. Its popularity rises together with the recent experimental emphasis in electronic ultrafast phenomena in material science. It can be used to study ion collision, light absorption spectrum, laser-induced demagnetization and phase change, charge transfer, excited carrier dynamics, and chemical reactions. For example, it has been shown by many recent experiments [7, 28, 29, 37, 39, 40, 47] that laser excitation can induce structure phase changes and charge density wave excitations. It has also been shown many of the previously thought adiabatic catalytic interactions are actually non-adiabatic, which requires electron excitation simulation to study them. All these present a new trend in the material science simulations, and require beyond ground state DFT simulations. Another common usage of rt-TDDFT is to simulate the optical spectrum and nonlinear optical properties of materials. Unfortunately, the semi-local exchange correlation functionals such as the local density approximation (LDA) and generalized gradient approximation (GGA) do not describe the excited states and band gaps correctly, resulting in a large error in the calculated optical spectrum. This problem can be solved using hybrid functionals [5, 32] within the framework of density functional theory (DFT). The hybrid functional mixes a fraction of the explicit Fock exchange integral with the semi-local exchange correlation functionals. It can be used to accurately describe the band gaps for a wide range of materials, especially with the recently developed range-separated hybrid functional [16, 17]. It has been shown recently, the optimally tuned range separated hybrid functional can reproduce accurate optical absorption spectrum when compared with more expensive methods like the Bethe-Salpeter equation based on GW calculation [1]. As a result, the rt-TDDFT + hybrid functional approach can be extremely powerful to describe, for example, the exciton excitation and charge transfer processes. Unfortunately, both rt-TDDFT and hybrid functional are computationally very expensive compared to conventional ground state DFT calculations with semi-local functionals. In practice, rt-TDDFT can be hundreds of times slower than the conventional ground state molecular dynamics simulations due to the need for using small time step, hybrid functional can also be tens of times slower than the semi-local exchange correlation functional due to the evaluation of the Fock exchange term. As a result, planewave-based rt-TDDFT + hybrid functional simulations are rarely found in the literature, even for small systems with a handful of atoms [36], not to mention large ones with a thousand atoms. However, for many problems, e.g., for excited state charge transfer, laser induced structure phase transition, nano systems like quantum dots and wires, large system simulation is essential. Besides, planewave basis is also important due to its flexibility, especially for excited states. This poses a question for how practical it is to carry out laser rt-TDDFT hybrid calculations, and whether this can be a realistic approach for material science simulation.

*Corresponding author

SC '19, November 17–22, 2019, Denver, CO, USA

ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356144

Fortunately, the situation has been improved due to both new algorithm developments and the emergence of new computer platforms like Summit (located in Oak Ridge National Laboratory, listed as No.1 supercomputer in the Top500 list in November 2018). On the algorithmic side, typical rt-TDDFT calculations are simultaneously limited by the accuracy and stability requirements of the ordinary differential equation (ODE) integrators. This implies that a small time step size (often in the sub-attosecond regime) needs to be used in order to yield accurate dynamics. Hence explicit time integrators, such as the Runge-Kutta 4th order (RK4) method, are often preferred to implicit time integrators, such as the Crank-Nicolson (CN) scheme, due to efficiency as well as simplicity of implementation [12]. By choosing the optimal gauge of propagation, the recently introduced parallel transport gauge formalism (PT) [2, 21] reformulated the rt-TDDFT equations, and enabled implicit integrators to be efficiently used with a large step size. Recently it is demonstrated that the parallel transport Crank-Nicolson (PT-CN) scheme can significantly increase the time step to around 50 attoseconds, and the accuracy can be fully comparable to that of the RK4 method both for semi-local exchange-correlation functionals and hybrid functionals [21, 24]. On the hardware side, heterogeneous architecture powered by GPU accelerators has become the most widely used architecture among supercomputers. The latest GPU based Summit supercomputer has significantly increased the available computing power. Given this situation, it will be of paramount interest to test the limit of the rt-TDDFT+hybrid functional method on the Summit supercomputer. Given the heterogeneous architecture of the Summit (GPU+CPU), its latest hardware for data communication, and the large amount of total memory, the computer algorithms need to be adapted accordingly.

Through our study, we would like to address the following questions: (1) How scalable is the implementation, both in terms of strong scaling and weak scaling? (2) What is the bottleneck in the most scalable case: computation or communication? (3) Whether the memory is a bottleneck? If not, what new algorithms one can use to take advantage of the large amount of memory? (4) How large is the speedup comparing GPU with CPU, in the sense of the absolute fastest time to solution, and in the sense of the same power consumption?

The contribution of this paper is as follows. (1) We find that with careful treatment of the data distribution, usage of CUDA custom kernels, batched fast Fourier transform (FFT) operations, and careful overlapping computation and communication, the resulting multi-GPU implementation of rt-TDDFT can be highly scalable. (2) While the CPU implementation of hybrid functional calculations is always dominated by the computational time (at least for large systems using up to a few thousands of cores), our GPU implementation significantly reduces the computational time. Therefore the cost of hybrid functional calculations is now comparable to that with semi-local functionals. For a large system, the MPI communication becomes the bottleneck only when more than 500 GPUs are used. (3) The large capacity of the Summit machine allows us to implement some unique algorithms in rt-TDDFT calculations, such as using Anderson mixing to accelerate the solution of nonlinear equations, where up to 20 copies of wavefunctions are used. (4) Thanks to the increase of the time step size and hence the reduction of the number of Fock exchange operator applications using the PT-CN algorithm, we can now carry out an rt-TDDFT+hybrid functional simulation with the planewave basis set for an unprecedentedly large system with 1536 silicon atoms, with a practical time to solution of 1.5 hours per femtosecond on 768 GPUs. Compared to our best CPU implementation using 3072 cores (also using the parallel transport formulation), our GPU implementation can be up to 30 times faster assuming maximum parallelization with 768 GPUs, and is still 7 times faster assuming equal power consumption using 72 GPUs. We provide a detailed performance analysis, which gives insights for how to implement similar electronic structure codes in such heterogeneous platforms, and what aspects of such platforms can be improved in the future to serve similar applications.

The rest of the manuscript is organized as follows. We review the algorithm for performing hybrid functional rt-TDDFT calculations with the parallel transport gauge formulation in section 2. The GPU implementation is shown in section 3. The setup of the test systems and the machine configuration are presented in section 4 and section 5, respectively. Then we show the numerical results in section 6, followed by the analysis in section 7 and conclusion in section 8.

2 PARALLEL TRANSPORT GAUGE FORMULATION OF RT-TDDFT

Real-time time-dependent density functional theory solves the following set of time-dependent equations

$$\mathrm{i}\partial_t \Psi(t) = H(t, P(t))\Psi(t). \tag{1}$$

Here $\Psi(t) = [\psi_1(t), \ldots, \psi_{N_e}(t)]$ is the collection of wavefunctions (also called electron orbitals), and N_e is the number of electrons (spin degeneracy omitted). The density matrix is defined as $P(t) = \Psi(t)\Psi^*(t)$, where Ψ^* is the Hermitian conjugate of Ψ. The time-dependent Hamiltonian takes the form

$$H(t, P(t)) = -\frac{1}{2}\Delta_{\mathbf{r}} + V_{\mathrm{ext}}(t) + V_{\mathrm{Hxc}}[P(t)] + \alpha V_{\mathrm{X}}[P(t)]. \tag{2}$$

Here $V_{\mathrm{ext}}(t)$ encodes the time-dependent external potential such as the nuclei-electron interaction and the laser field, and V_{Hxc} consists of the Hartree potential and the local part of the exchange-correlation potential. In the absence of the term V_{X}, this is the semi-local functional rt-TDDFT. This paper focuses on hybrid functional rt-TDDFT calculations, where V_{X}, called the Fock exchange operator, is an integral operator with kernel $V_{\mathrm{X}}[P](\mathbf{r}, \mathbf{r}') = -P(\mathbf{r}, \mathbf{r}')K(\mathbf{r} - \mathbf{r}')$. Here $K(\mathbf{r} - \mathbf{r}')$ is the kernel for the (possibly screened) electron-electron interaction [16, 17], and α is a mixing fraction (usually $\alpha = 0.25$).

In an rt-TDDFT simulation, the matrix-vector multiplication of the type $H[P]\Psi$ needs to be repeatedly performed. This is particularly the case for hybrid functional calculations, where each set of multiplications $V_X[P]\Psi$ requires the following operations:

$$\left(V_X[P]\psi_j\right)(\mathbf{r}) = -\sum_{i=1}^{N_e} \psi_i(\mathbf{r}, t) \int K(\mathbf{r} - \mathbf{r}')\psi_i^*(\mathbf{r}', t)\psi_j(\mathbf{r}')\,\mathrm{d}\mathbf{r}'. \tag{3}$$

Due to the convolutional structure of K, when the planewave basis set is used, each term $\int K(\mathbf{r} - \mathbf{r}')\psi_i^*(\mathbf{r}', t)\psi_j(\mathbf{r}')\,\mathrm{d}\mathbf{r}'$ can be evaluated by solving a Poisson-like equation using the fast Fourier transform (FFT). Hence this amounts to solving N_e^2 Poisson-like equations. Due to the Fock exchange term, hybrid functional calculations

can be tens of times more expensive than semi-local calculations. We remark that in the context of ground state hybrid functional DFT calculations, several approaches have been proposed to reduce the cost. The application of the Fock exchange operator can be accelerated with a massive number of CPUs [10, 42], and adaptive compression techniques [26]. When approximation of the Fock exchange operator can be tolerated, the cost can also be reduced through other algorithmic approaches such as localization [6, 8, 9, 45] and density fitting techniques [19]. In the current GPU work, these techniques are not used in the rt-TDDFT simulation.

The rt-TDDFT equation (1) can be equivalently expressed using a set of unitarily transformed orbitals. Physical observables such as the density matrix are by definition invariant to such unitary rotations (called gauge-invariant). This allows us to seek for the optimal gauge for numerical simulation of rt-TDDFT. Recently, such optimal gauge has been identified [2, 21], which is defined implicitly through the following equation

$$\mathrm{i}\partial_t \Psi = H\Psi - \Psi(\Psi^* H\Psi), \quad P(t) = \Psi(t)\Psi^*(t). \tag{4}$$

Here Ψ^* stands for the Hermitian conjugate of the matrix Ψ. Compared to Eq. (1), the only difference is the extra term $\Psi(\Psi^* H\Psi)$, which mixes the information from all orbitals together. The right-hand side of Eq. (4) is a residual type term. Its magnitude can be much smaller than $H\Psi$ on the right-hand side of Eq. (1), and the dynamics become smoother. In fact, $\Psi(t)$ solved from Eq. (1) can be viewed as the parallel transport (PT) of the initial wavefunctions to time t within the range of $P(t)$, and the corresponding implicitly defined gauge is called the parallel transport gauge. Coupled with implicit integrators such as the Crank-Nicolson (CN) scheme, the resulting PT-CN scheme solves the following nonlinear equation at each time step

$$\begin{aligned} &\Psi_{n+1} + \mathrm{i}\frac{\Delta t}{2}\left\{H_{n+1}\Psi_{n+1} - \Psi_{n+1}\left(\Psi_{n+1}^* H_{n+1}\Psi_{n+1}\right)\right\} \\ =&\Psi_n - \mathrm{i}\frac{\Delta t}{2}\left\{H_n\Psi_n - \Psi_n\left(\Psi_n^* H_n\Psi_n\right)\right\}. \end{aligned} \tag{5}$$

Compared to explicit time integrators such as the explicit 4th order Runge-Kutta scheme (RK4) which often requires a sub-attosecond time step, the time step allowed by PT-CN can be significantly improved to around 10–50 attoseconds. This is particularly important for reducing the number of Fock exchange operator applications in hybrid functional calculations.

Alg. 1 summarizes the procedure for one step of time propagation using the PT-CN scheme. During each time step, we first evaluate the initial residual R_n. The right-hand side of (5) can be viewed as propagating the wavefunctions by half a step. It is thus denoted by $\Psi_{n+\frac{1}{2}}$ and is fixed during the self-consistent field iteration. The new set of wavefunction Ψ_{n+1} needs to satisfy a fixed point problem and is denoted by Ψ_f during the iteration, and the residual for the fixed point problem is denoted by R_f. The fixed point problem is solved by the Anderson mixing method [3]. When the residual is sufficiently small, the SCF iteration can be terminated. In practice, we find that the SCF convergence can also be monitored by the convergence of the charge density.

Algorithm 1 One time propagation step with the PT-CN method.

INPUT: Ψ_n
OUTPUT: Ψ_{n+1}

1: Evaluate the initial residual $R_n = H_n\Psi_n - \Psi_n(\Psi_n^* H_n\Psi_n)$.
2: Evaluate $\Psi_{n+\frac{1}{2}} = \Psi_n - \frac{\mathrm{i}\Delta t}{2}R_n$, and let $\Psi_f = \Psi_{n+\frac{1}{2}}$.
3: Evaluate the electron density ρ_f corresponding to Ψ_f.
4: **for** $j = 1, 2, \ldots$ **do**
5: Update the potential and the Hamiltonian H_f.
6: Evaluate the fixed point residual $R_f = \Psi_f + \frac{\mathrm{i}\Delta t}{2}(H_f\Psi_f - \Psi_f(\Psi_f H_f\Psi_f)) - \Psi_{n+\frac{1}{2}}$.
7: Perform Anderson mixing to update wavefunctions Ψ_f.
8: Evaluate the electron density ρ_f corresponding to Ψ_f.
9: If the change of the electron density is sufficiently small, exit the loop.
10: **end for**
11: Orthogonalize Ψ_f to obtain Ψ_{n+1}.

3 MULTI-GPU IMPLEMENTATION

In hybrid functional calculations with a planewave basis set, the application of the Fock exchange operator in the $H\Psi$ step often takes around 95% of the total computation time with a CPU implementation. However, according to Amdahl's law, in order to achieve a desirable speedup factor, almost all steps of the calculation needs to be accelerated using GPUs. Our GPU code is implemented within PWDFT, which uses the planewave discretization and is an independent module of the massively parallel software package DGDFT (Discontinuous Galerkin Density Functional Theory) [18, 27]. In our implementation, all computationally intensive parts are performed using GPUs with either GPU-accelerated libraries or CUDA custom kernels. We also carefully overlap the MPI communication and GPU computation to take advantage of the heterogeneous architecture. We remark that for ground state electronic structure calculations, the use GPU of acceleration has been reported in various software packages such as ABINIT [13], BigDFT [33], NWChem [42], Octopus [4], PWmat [22, 23], Quantum ESPRESSO [34], VASP [14, 20], to name a few.

3.1 Hybrid parallelization scheme

There are two main parallel distribution schemes for the wavefunctions. The first one is the column based distribution scheme (also called the band index parallelization), i.e., each column of Ψ are distributed to different MPI tasks based on its band index (i.e. the column index). This data distribution scheme is highly efficient for the calculation of $H\Psi$. This is particularly the case for hybrid functional calculations since different MPI tasks are able to perform FFTs independently. The second one is the row based distribution scheme (also called the G-space parallelization, where G is a standard notation for the index in the Fourier space). In this scheme, the data is distributed according to the partition of the Fourier coefficients. This distribution scheme facilitates the evaluation of matrix-matrix multiplications, such as the evaluation of the overlap matrix $S = \Psi^*(H\Psi)$ once $H\Psi$ is obtained.

In PWDFT, the wavefunctions Ψ are mostly distributed in the band index parallelization to evaluate the $H\Psi$ efficiently. The conversion from band index parallelization to G-space parallelization is performed via MPI_Alltoallv for matrix-matrix multiplication operations such as the evaluation of the overlap matrix, as shown in Fig. 1. Note that after the evaluation in the G-space, the wavefunctions are converted back to the band index parallelization format via MPI_Alltoallv. In this fashion, both $H\Psi$ and matrix-matrix multiplication operations can be evaluated efficiently. In the GPU implementation, the hybrid parallelization scheme plays an even more important role. This is because the band index parallelization allows us to use the CUFFT library for the $H\Psi$ calculation. However, the band index parallelization cannot scale to a large number of processors for matrix-matrix multiplications either on CPUs or GPUs, and we therefore need to convert to the G-space parallelization scheme. The hybrid parallelization scheme for the GPU implementation has been used in [43] for ground state electronic structure calculations. We remark that there is another level of parallelization over k-points, which is another index of the wavefunction Ψ. For large physical systems containing a few hundred to a thousand atoms such as the systems calculated in this paper, only one k-point (i.e. the Γ-point) is needed. Hence k-point parallelization is not discussed in this paper.

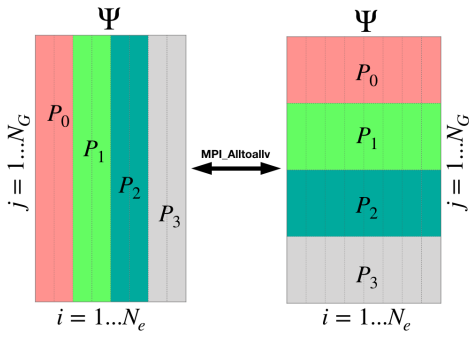


Figure 1: Illustration of the hybrid parallelization scheme with $N_e = 8$ wavefunctions on 4 MPI tasks. The number of planewaves N_G is usually on the order of $10^3 \sim 10^6$. Thus the maximum number of processes is limited by the N_e.

3.2 Evaluation of $H\Psi$

The cost of evaluating $H\Psi$ mainly consists of two parts: applying the Fock exchange operator and applying the pseudopotential.

The algorithm for applying the Fock exchange operator using a planewave discretization is shown in Alg. 2. The wavefunctions are distributed according to the band index. For a system with N_e wavefunctions and calculated on N_p processors, each processor holds $N_{e'} = N_e/N_p$ wavefunctions (assuming N_e is divisible by N_p). According to Eq. (3), each wavefunction Ψ_i needs to be multiplied to all other wavefunctions $\{\Psi\}_{j=1}^{N_e}$. This is performed using an MPI_Bcast operation in line 3 of Alg. 2. Then each processor will solve the $N_{e'}$ Poisson-like equations with FFTs. Since each MPI task will eventually receive all N_e wavefunctions, the total communication volume is $N_p \times N_G \times N_e$ multiplied by the storage cost of a complex number, where N_G is the number of planewaves to store a wavefunction.

Algorithm 2 Applying the Fock exchange operator in $H\Psi$

INPUT: Wavefunctions Ψ distributed according to the band index.
OUTPUT: $V_X\Psi$ distributed according to the band index.

1: Let $V_X\Psi$ be distributed according to the band index and initialized to zero.
2: **for** i = 1, N_e **do**
3: **if** the current processor holds Ψ_i **then**
4: Broadcast Ψ_i to all processors
5: **end if**
6: **for** j = 1, N_e **do**
7: **if** the current processor holds Ψ_j **then**
8: Solve Poisson-like equation using FFT with respect to the charge-like quantity $\Psi_i^*(\mathbf{r})\Psi_j(\mathbf{r})$, and add the solution to $(V_X\Psi)_j$.
9: **end if**
10: **end for**
11: **end for**

In order to efficiently carry out Alg. 2 on GPUs, we perform a number of optimization steps.

1. *CUFFT and CUDA custom kernels (band-by-band).* The first step of porting PWDFT onto GPU is to use the CUFFT library and CUDA custom kernels to accelerate the computation of Alg. 2. In this step, all relevant computation(from line 6 to line 10 in Alg. 2) are moved onto GPU in a band-by-band manner. In our implementation, the CUDA custom kernels are written to fill the gaps between the CUFFT calls, and there is no CPU-GPU synchronization during the computation. At the current stage, the CPUs are only used for performing MPI communication, and the data copy between CPU and GPU is necessary after the MPI_Bcast operation.

2. *Batched implementation.* Each V100 GPU on the Summit supercomputer has a peak performance of 7.8 TFLOPS and a peak bandwidth of 900 GB/s. The band-by-band implementation above cannot saturate the bandwidth of the GPUs. One way of improving GPU performance is to send more data to the GPU. In the GPU version of PWDFT, instead of sending the data $\Psi_i^*(\mathbf{r})\Psi_j(\mathbf{r})$ one by one, we batch them together and call a batched CUFFT. The corresponding CUDA custom kernels are also changed to a batched fashion. The batched version of code has two benefits: first it increases the computational intensity of the CUDA kernels to take advantage of the computing power of GPU; second, it reduces the latency between CPU and GPU by reducing the number of CUDA kernel launches. The solution of the Poisson-like equations for wavefunctions Ψ_j are batched at line 6 of Alg. 2, and the maximum batch size is set to 8 in our GPU implementation. We remark that the performance of the batched implementation can vary with the GPU architecture, and batched FFTs work well when the GPU memory bandwidth is yet not saturated, which is mostly the case for the V100 GPUs on Summit when the testing system is less than a few hundred of atoms.

3. *GPUDirect and CUDA-aware MPI.* On the Summit supercomputer, the IBM Spectrum CUDA-aware MPI is supported by hardware, which means inter-node GPUs can communicate with each other via MPI without explicitly coping data to the CPU. In PWDFT, we take advantage of this feature, and MPI communication is performed directly on the GPU data. In this step, the wavefunctions are always kept on the GPU, and MPI_Bcast is performed using the CUDA-aware MPI in a band-by-band manner. This fine-grained (band-by-band MPI_Bcast) communication creates more opportunity to overlap the GPU computation and the MPI communication, since conceptually the CUDA-aware MPI and GPU computation can be performed simultaneously.

4. *Single precision MPI.* As will be seen later in the performance analysis, the MPI communication is mainly limited by the bandwidth of network adapters (NIC). To reduce the communication time of the Fock exchange operator applications, we use the single precision format for sending and receiving the wavefunctions, which reduces the communication volume by half. We also remark that the single precision format is only used in the MPI communication, which means wavefunctions will be converted back to the double precision format for computation. We find that this leads to negligible changes in the accuracy of the rt-TDDFT dynamics, for example, the error of the total energy is as little as 10^{-9} eV for a silicon system with 768 atoms after a single step of time propagation, compared to the double precision implementation. This agrees with observations of ground state DFT calculations [11]. We remark that at the current stage, the MPI_Bcast is performed with CUDA-aware MPI with single precision wavefunctions.

5. *Overlap computation and communication.* In principle, CUDA-aware MPI communication can naturally overlap with the GPU computation. However, we find that the MPI_Bcast and computation of the Fock exchange operator cannot be fully overlapped when the CUDA-aware MPI_Bcast is involved on Summit. Profiling of the Fock exchange part shows that there are two synchronized CPU-GPU memory copy operations in the communication step, as shown in Fig. 2. This is caused by the fact that the NIC is connected to the IBM POWER 9 socket as shown in Fig. 5. Thus MPI_Bcast will first copy the data from GPU to CPU, then the inter-node communication is performed over the NIC. This memory copy operation will introduce the CPU-GPU synchronization, thus the overlapping of computation and communication is disrupted. We suspect that the synchronization between CPU and GPU is mainly caused by the specific implementation of the IBM Spectrum MPI. Therefore we propose another implementation without using the CUDA-aware MPI. This implementation uses an asynchronous CPU-GPU memory copy operation explicitly, followed by the MPI_Bcast using CPU. We remark that the CUDA-aware MPI is still used in other parts of the communication such as MPI_Alltoallv, since the CPU-GPU synchronization does not play such an important role. Fig. 3 shows that the MPI communication and GPU computation can overlap perfectly, as the MPI communication time is entirely hidden behind the computation time. We stress that the overlapping is achieved based on the fact that CPU and GPUs can work independently, not on the unblocked MPI_Isend/MPI_Irecv communication. We have also implemented the round-robin communication strategy [33] via MPI_Send/MPI_Recv. We notice that the performance using the round-robin strategy and using MPI_Bcast is approximately the same on Summit. We also find that the round-robin strategy needs to be carefully implemented to be load-balanced. On the other hand, MPI_Bcast is a simpler strategy and takes advantage of the fat-tree interconnect topology of Summit.

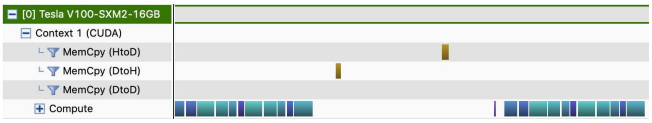


Figure 2: CUDA profiling reveals the implicit CPU-GPU synchronization introduced by the CUDA-aware Spectrum MPI.

Fig. 3 shows the reduction of the computational time associated with different stages of optimization. The testing system is a 1536 silicon atoms system, which will be discussed in section 4. The CPU version of PWDFT uses 3072 CPU cores (about 73 nodes on Summit and 96 Haswell nodes on Cori supercomputer). Each Cori node is equipped with two Intel Xeon E5-2698 v3 16-core sockets and in our tests we use one MPI per CPU core. We find that the performance of the CPU performance is almost the same for the Fock exchange operator calculation on both Summit and Cori. The GPU version uses 72 GPUs (12 Summit nodes), and is around 7 times faster than the CPU version on Summit in terms of applying the Fock exchange operator. We find that step 1 (CUFFT and CUDA custom kernels) leads to most of the performance improvement from CPU to GPU, and this step is responsible for most of the implementation efforts as well.

We remark that besides moving the computationally intensive steps from CPU to GPU, our optimization steps mainly aim at reducing the communication cost on the GPU. On the other hand, the cost of the CPU implementation is strongly dominated by the computation, and hence would benefit less from the same optimization steps. More specifically, the CPU has considerably less amount of memory bandwidth than the V100 GPU, and hence cannot use the batched implementation of FFTs efficiently. The CPU also lacks the heterogeneous architecture for overlapping communication and computation as in the GPU method. Therefore the performance comparison here and below between the GPU and CPU implementation should be fair.

Besides the application of the Fock exchange operator, we also apply the pseudopotentials to Ψ using the band index parallelization on GPU with CUFFT and CUDA custom kernels. In our implementation, we choose the real space representation for the nonlocal projectors, which can be stored as sparse vectors. This can often be more than 5 times faster compared to the reciprocal space implementation when the system size is more than a few hundred atoms [44]. In our current implementation, the entire set of local pseudopotentials and nonlocal projectors are stored on every processor. For the largest system tested in this paper with 1536 silicon atoms, the total memory cost for the nonlocal projectors is approximately 432MB. Each V100 GPU has its 16GB on-chip memory and is therefore sufficient. This simplified implementation allows us to apply the pseudopotentials without any communication cost, and thus fully takes advantage of the computational speed provided by the GPUs.

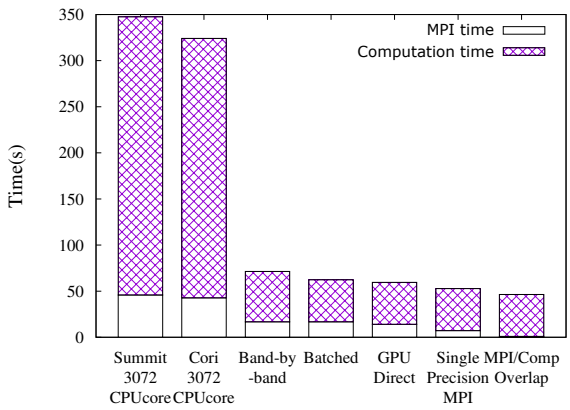


Figure 3: Wall clock time for applying the Fock exchange operator for a system with 1536 silicon atoms. The CPU version uses 3072 CPU cores, and the GPU version uses 72 GPUs.

3.3 Evaluation of the residuals

According to Alg. 1, there are two types of residuals to be evaluated in the PT-CN algorithm, denoted by R_n and R_f. For simplicity we only discuss the evaluation of R_f (Alg. 3); the evaluation of R_n is similar.

To calculate the residual R_f, the input data Ψ_f, $H_f\Psi_f$, and $\Psi_{n+\frac{1}{2}}$ are first converted from band index parallelization to G-space parallelization via an MPI_Alltoallv operation. Then the local overlap matrix S can be evaluated on GPU by calling the CUBLAS matrix-matrix multiplication routine. Next, the local information from all processors is combined into the global overlap matrix using an MPI_Allreduce operation, followed by the rotation operation on GPU through a CUBLAS GEMM call. The residual is then calculated by BLAS-1 operations. Finally, the residual R_f will be transposed back to band index parallelization using an MPI_Alltoallv operation. In order to reduce the communication cost, the single precision format for R_f is used during the communication step using MPI_Alltoallv, and is then converted back to the double precision format during the computation step.

Algorithm 3 Algorithm of residual calculation.

INPUT: Wavefunctions Ψ_f, $H_f\Psi_f$, $\Psi_{n+\frac{1}{2}}$ distributed according to the band index.

OUTPUT: Residual P_i distributed according to the band index.

1: Use MPI_Alltoallv to convert Ψ_f, $H_f\Psi_f$, $\Psi_{n+\frac{1}{2}}$ to the G-space parallelization format.
2: Evaluate the local overlap matrix $S_{\text{temp}} = \Psi_f^* H_f \Psi_f$
3: Use MPI_Allreduce operation on S_{temp} to obtain the total overlap matrix S.
4: Rotate the wavefunctions locally $\Psi_{\text{temp}} = \Psi_f S$
5: Evaluate the residual $R_f = \Psi_f + \frac{i\Delta t}{2}(H_f\Psi_f - \Psi_{\text{temp}}) - \Psi_{n+\frac{1}{2}}$
6: Use MPI_Alltoallv to convert R_f to the band index parallelization format.

3.4 Density Evaluation, Anderson mixing, wavefunction orthogonalization, and others

Because the wavefunctions are stored in the band index parallelization format, it is straightforward to evaluate the electron density $\rho(\mathbf{r}) = \sum_{i=1}^{N_e} |\psi_i(\mathbf{r})|^2$. This step requires representing the wavefunctions on a real space grid, which can be performed using FFTs, followed by an MPI_Allreduce operation across all MPI tasks. All above calculations are evaluated on the GPU, and the MPI_Allreduce operation is performed via CUDA-aware MPI.

The Anderson mixing method for solving the nonlinear equations requires the solution of a least squares problem for each wavefunction [3]. In our calculations, the maximum mixing dimension is set to 20. Thus after evaluating an overlap matrix with respect to the history of wavefunctions Ψ_f and the associated residuals R_f, the size of the least squares problem becomes very small, up to 20×20. The main cost of the Anderson mixing is then due to the evaluation of overlap matrices that can be performed efficiently using the G-space parallelization.

Note that our implementation implies that up to 20 copies of the wavefunctions are needed for performing the Anderson mixing, which could be expensive if the wavefunctions are all stored on GPUs. However, we may store these wavefunctions on the CPU main memory, which has a large capacity of 512 GB on each computing node of Summit. During the computation, we copy all the wavefunctions corresponding to a single band i(up to $20 \times N_G$) from CPU to GPU, and the overlap matrices needed for the Anderson mixing can be performed via CUBLAS matrix-matrix multiplications.

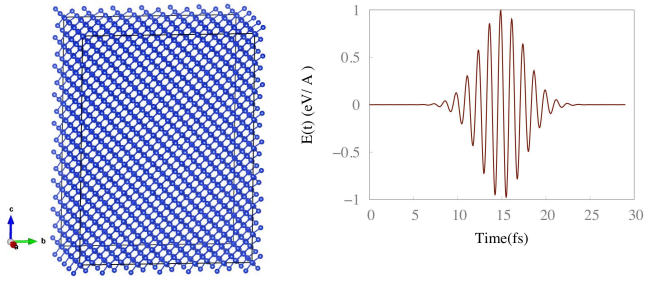
At the end of each rt-TDDFT time step, the wavefunctions Ψ will be re-orthogonalized. To improve the parallel efficiency, we again first evaluate an overlap matrix of the type $\Psi^*\Psi$ using the G-space parallelization. Then we can perform a Cholesky decomposition on the overlap matrix of size N_e, and rotate Ψ_f efficiently due to the G-space parallelization. The Cholesky decomposition is calculated on a single GPU with cuSOLVER library, and the subsequent rotation is performed via the GPU Trsm subroutine.

Besides the computationally intensive parts discussed above, all other operations such as the evaluation of the Hartree potential, the gradient of the electron density, the local part of the exchange-correlation potential, etc contributes to less than 2% of the computational time on CPUs. In the GPU version of PWDFT, these parts are all parallelized at the CPU level. For example, we parallelize the FFTs associated with the calculation of the gradient of electron density by using distributed FFTW in the Z direction. Such parallelization is important for the overall performance since all other computational intensive parts can be accelerated by up to 40 times on GPUs. We also keep the variables related to the charge density (such as the Hartree potential and the gradient of the electron density) on each MPI task. Hence MPI_AllGatherv and MPI_Bcast operation are performed after the computation.

4 SETUP OF THE TEST PHYSICAL SYSTEM

We report the efficiency of the GPU version of PWDFT using silicon systems ranging from 48 to 1536 atoms. The supercells are constructed from $1 \times 1 \times 3$ to $4 \times 6 \times 8$ unit cells, respectively, and

each simple cubic unit cell consists of 8 silicon atoms with lattice constant being 5.43 Å . In all the tests, the external potential is given by a laser pulse shown in Fig. 4(a), and its wavelength is 380nm. The total simulation time is 30 fs, and the time step using the PT-CN method is set to 50 as. Thus the total number of rt-TDDFT steps is 600. The stopping criteria is set to 1.0×10^{-6} for the electron density error. The average number of SCFs is 22 and the maximum Anderson mixing dimension is set to 20. We use the SG15 Optimized Norm-Conserving Vanderbilt (ONCV) pseudopotentials [15, 38] and HSE06 functionals [17] in all the following tests. The kinetic energy cutoff is set to 10 Hartree. For the system with 1536 atoms, the number of grid points for a wavefunction is $N_G = 60 \times 90 \times 120 = 648,000$. This corresponds to a charge density grid $120 \times 180 \times 240$. The Fock exchange operator is evaluated on the wavefunction grid. The number of occupied wavefunctions is 3072.



(a) 1536 silicon (b) 380nm laser

Figure 4: Atomic configuration and external laser field.

5 MACHINE CONFIGURATION

All numerical tests are performed on the Summit supercomputer. Fig. 5 shows the architecture of one of the 4608 Summit computing nodes. Each computing node consists of two identical groups, and each group has one IBM POWER 9 socket and 3 NVIDIA Volta V100 GPUs connected via NVLink with a bandwidth of 50GB/s. Each POWER socket has 22 physical CPU cores and share 256GB DDR4 CPU main memory, and each V100 GPU has its own 16GB high bandwidth memory. The CPU bandwidth is 135GB/s and GPU bandwidth is 900GB/s. Each GPU has a theoretical peak performance of 7.8 TFLOPS double precision operations. The two groups of hardware are connected via X-Bus with a 64GB/s bandwidth. The computing nodes are interconnected with a non-blocking fat-tree using a dual-rail Mellanox EDR InfiniBand interconnect with a total bandwidth of 25GB/s.

In this paper, we use the MPI+CUDA programming model. In all GPU tests, we use 6 MPI tasks per computing node (3 MPI tasks per socket to fully take advantage of both CPU-GPU affinity and network adapter), and each MPI task is bound to an individual GPU. For the comparison of the numerical performance, the CPU version of PWDFT only uses the CPU part of the machine. In the CPU tests, we use the maximum number of cores allowed by PWDFT. Due to the hybrid parallelization scheme, the maximum number of CPU cores is the number of wavefunctions. In the case of 1536 atom silicon system with 3072 wavefunctions, we find that the CPU version of the PWDFT efficiently scales up to 3072 CPU cores.

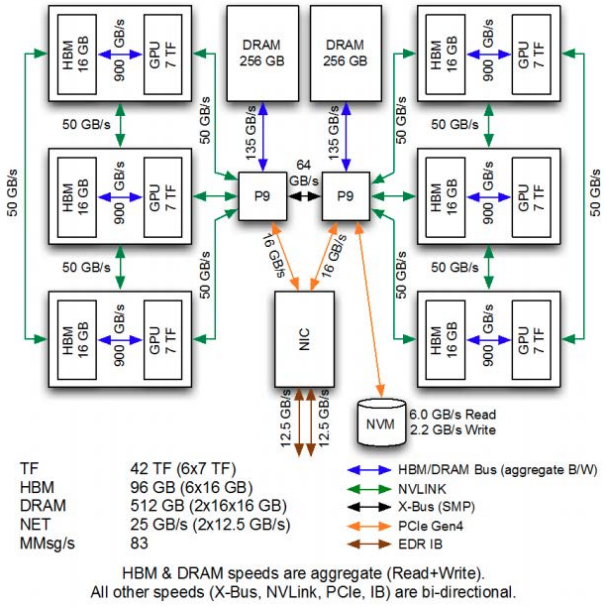


Figure 5: The architecture of a computational node on Summit.

6 NUMERICAL RESULTS

First, we demonstrate the efficiency of the PT-CN algorithm by comparing it to the explicit Runge-Kutta 4th order (RK4) integrator for the system with 1536 silicon atoms. Both algorithms are implemented on GPU, and the time step for PT-CN is set to 50 as and the time step for RK4 is 0.5 as. This is close to the largest time step allowed by RK4 due to the stability constraint. In Fig. 6, we compare the wall clock time for a 50 as simulation using PT-CN and RK4. The PT-CN method can be about 20 times faster compared to the explicit time integrator RK4 method using 36 GPUs, and becomes 30 times faster when using 768 GPUs. The increase of the speedup factor with respect to the number of GPUs is mainly due to that PT-CN can use a larger step size, and is less impacted by the cost of "others" component in section 3.4 (such as the evaluation of the Hartree potential). A detailed discussion of the scaling of different components will be presented in section 7.

Second, we compare the performance of the GPU version of PWDFT with the CPU one. The comparison is based on the power consumption, which is one of most important criteria in high performance computing, especially in the upcoming exascale era. The power consumption of a single POWER 9 socket is 190 watt and that of a single NIVIDIA V100 GPU is 300 watt. Hence the power cost for each CPU node consisting of 2 POWER 9 CPU sockets is 380 watt and each GPU computing node with 6 V100 GPUs and 2 POWER 9 CPU sockets is 2180 watt. Using 3072 CPU cores (in

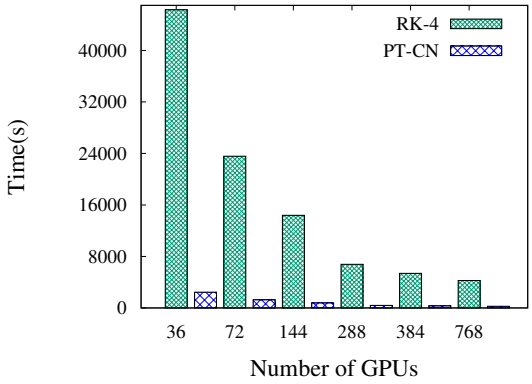


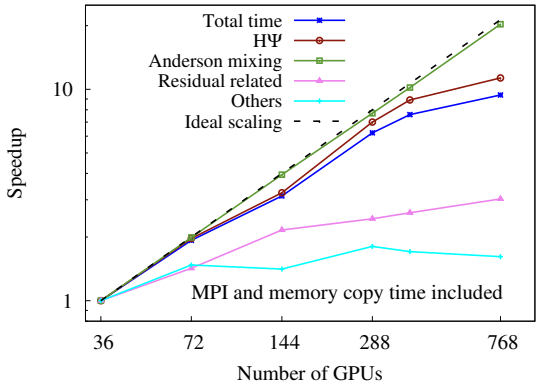
Figure 6: Wall clock time for simulating the 1536 silicon atom system for 50 attoseconds using RK4 and PT-CN methods.

practice using 73 computing nodes), the total power consumption is 27740 watt. The power consumption of 12 GPU nodes is 26160 watt, which consumes slightly less energy than 73 CPU nodes. According to Table 1, in this setup, the GPU version of PWDFT is still 7 times faster compared to the CPU version.

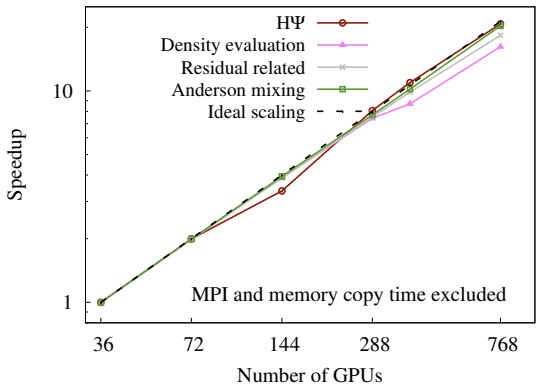
While the CPU version of PWDFT has reached its scaling limit, the GPU version can still scale beyond 72 GPUs. Fig. 7(a) demonstrates the strong scaling of the wall clock time with respect to the number of GPUs. We find that our GPU implementation can scale to 768 GPUs, and near ideal scaling is achieved when the GPU number is less than 384. After 768 GPUs, the MPI communication dominates the computational cost, which prevents the code to scale to a larger number of GPUs. According to Table 1, the GPU version is 34 times faster than the CPU version using 3072 CPU cores. We remark that the strong scaling determines the time to solution, and thus is crucially important in practical calculations. Using the GPU version of PWDFT, we can achieve 260 seconds per TDDFT step (50 attoseconds), which amounts to about 1.5 hours per femtosecond for the 1536 silicon atoms system.

The weak scaling using PT-CN method for simulating systems consisting of 48 to 1536 atoms for 50 as is shown in Fig. 8. The number of GPUs is set to $\frac{1}{2}N_{\text{atom}}$. The GPU version of PWDFT exhibits excellent weak scaling property. Since the computational complexity of hybrid functional rt-TDDFT simulation scales as $O(N_{\text{atom}}^3 \log N_{\text{atom}})$, the ideal scaling should be $O(N_{\text{atom}}^2)$ as in Fig. 8, neglecting the logarithmic factor. We find that for small systems, thanks to GPU acceleration, the Fock exchange operator applications has not yet dominated the computational cost, and hence our implementation scales even better than that indicated by the ideal scaling. Even with the system size increases to 1536 atoms, the weak scaling is still very close to the ideal scaling.

Here for a smaller system with 192 atoms, the simulation of 50 as with 96 GPUs is only 16 seconds. This means that each femtosecond simulation takes around 5 minutes. Even a picosecond rt-TDDFT simulation with hybrid functionals is now within reach and would take approximately 4 days.



(a)



(b)

Figure 7: Strong scaling: (a) Scaling of the total computation time and different components (MPI and CPU-GPU memory copy time included). (b) Scaling of different components (MPI and CPU-GPU memory time excluded). The wall clock time with 36 GPUs is set as the baseline, which is already 3.7 times faster than the CPU version with 3072 CPU cores.

7 PERFORMANCE ANALYSIS

In this section, we present a more detailed analysis of the performance of our GPU implementation.

First, we discuss the memory usage of the PT-CN method. In the GPU version of PWDFT, the most memory demanding part is the Anderson mixing, which requires up to 20 copies of wavefunction Ψ. For the system with 1536 atoms, each wavefunction takes 10MB ($N_G = 648,000$ multiplied by the cost of a complex number in double precision format). Using 36 GPUs, each MPI holds less than 100 wavefunctions (1GB). Then Anderson mixing requires less than 20 GB memory per MPI. There are 6 MPIs per computing node, and the total usage of the CPU main memory per node is less than 120GB, which is smaller than the limit of a node on Summit (512GB). Hence our implementation of PT-CN effectively takes advantage of the fat node configuration. In our code, we find that the GPU memory usage is less than 8GB for the system with 1536 atoms.

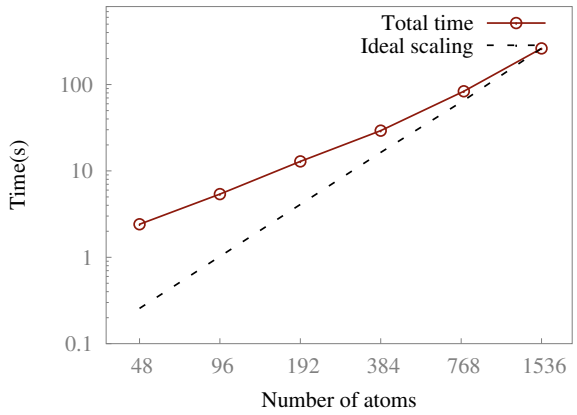


Figure 8: Weak scaling: wall clock time per 50 as for silicon systems with 48 to 1536 atoms. The number of GPUs used are always set to half of number of atoms in the calculation. The "ideal scaling" here scales as $O(N_{\mathbf{atom}}^2)$.

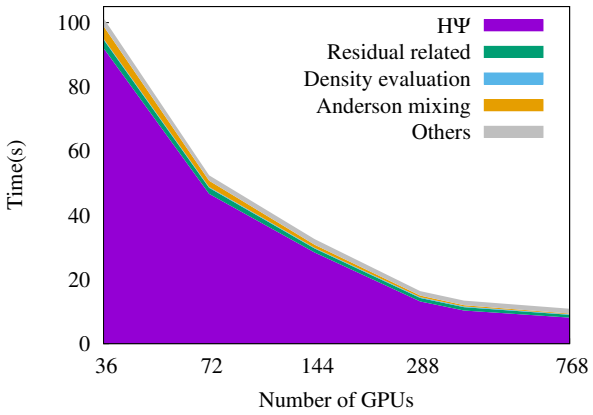


Figure 9: Strong scaling: the total time of a single SCF step and the contributions of each part using different number of GPUs.

Thus it is possible to accommodate a system with up to 3000 atoms using 72 GPUs based on the GPU memory usage.

The total number of double precision floating point operations (FLOP) for the 1536 silicon system per TDDFT step is 3.87×10^{16}. This is collected via the CUDA profiling tool NVPROF. Although NVPROF only collects the total number of FLOP on the GPUs, the number is still reasonable because in our implementation the CPUs are only responsible for the computing quantities labeled as "others" as in section 3.4, which contributes to less than 1% the total number of FLOP. The floating point operations per second (FLOPS) is then calculated as $\frac{\text{total FLOP}}{(\text{number of GPUs}) \times (\text{total time})}$, and the corresponding FLOPS efficiency is $\frac{\text{FLOPS}}{7.8\ \text{TFLOPS}}$. The efficiency of GPU version of PWDFT is 5.5% when using 36 GPUs, and goes down to 2% using 768 GPUs. The low FLOPS efficiency of GPU version of PWDFT is mainly caused by the fact that most FLOP is contributed by the FFTs in Fock exchange operator calculation. The FFT operations on GPU is mainly limited by the GPU bandwidth rather than the computational kernel. For instance, we find that CUFFT execution reaches about 11% of the peak performance of the V100 GPU in our implementation, and the result is comparable to the performance of CUFFT reported by NVIDIA [30]. The above analysis can be supported by evaluating the average required bandwidth of CUFFT and CUDA custom kernels during the Fock exchange operator calculation. We find that the GPU version of PWDFT achieves approximately 90% of the GPU maximum bandwidth in all tests. Such high GPU memory bandwidth utilization indicates that our calculation is mainly bounded by the hardware memory bandwidth, rather than the FLOPS.

The total time of a single SCF step can be divided into 5 parts: $H\Psi$, Anderson mixing, the residual related part, the electron density evaluation, and others. The contributions of each part to the total time is listed in Table 1 and shown in Fig. 9. The scaling of different computational components of the GPU version of PWDFT is shown in Fig. 7(a). Since the total computation time is dominated by the application of the Fock exchange operator, the scaling of $H\Psi$ is similar to that of the total time. In each TDDFT step, 24 Fock exchange operator evaluation is performed (22 in the SCF calculation, one in total energy evaluation, and one before SCF calculation to evaluate the residual term R_n), and this contributes to 93% of the total FLOP. The Anderson mixing step scales well with respect to the number of GPUs. The residual related part also scales with the number of GPUs. Its scaling is mainly limited by the MPI_Alltoallv and MPI_Allreduce operations. The "others" components as in section 3.4 are all parallelized on CPUs. As shown in Fig. 9, "others" does not scale with the number of GPUs. It contributes 2.6% of the total time of a single SCF when using 36 GPUs, and grows up to 15% when using 768 GPUs. Such scaling behavior is mainly caused by the fact that "others" is dominated by the MPI communication of the density related variables using MPI_Bcast.

Next, we discuss the scalability of different computational components in Fig. 7(b). Note that neither MPI communication nor CPU-GPU memory copy time is included in this figure. We find that nearly all computational time scale well with respect to the number of GPUs. The only computational part that does not scale is the Cholesky decomposition used in the orthogonalization and it is not shown in Fig. 7(b). This part only takes 0.017s on GPU for the 1536 atom system and is calculated once every TDDFT step. Thus it is negligible in the rt-TDDFT calculation. The scaling of the computational time clearly shows that the scaling bottleneck is not the computation. Therefore the main bottleneck comes from the data moment operations, which include both the CPU-GPU memory copy and the MPI communication.

The breakdown of the wall clock time in terms of the MPI communication, CPU-GPU memory copy and computation shown in Fig. 10. The detailed numbers are reported in Table 2. Since memory copies are mostly performed over wavefunctions within each node, the CPU-GPU memory copy operations scale well with respect to the number of GPUs. The MPI_Alltoallv operations is mainly used in the hybrid parallelization scheme to convert the distribution formats of wavefunctions, and is also found to be scalable.

Table 1: Wall clock time of the computationally intensive components for calculating a 1536 silicon atom system. The speedup factor is based on the best CPU implementation with 3072 CPU cores using about 73 computing nodes, and the wall clock time is 8874s.

Number of GPUs	36	72	144	288	384	768	1536	3072
Fock exchange operator MPI	0.71	0.89	1.25	1.83	1.99	3.72	6.06	8.074
Fock exchange operator computaion	90.99	45.61	27.05	11.27	8.31	4.38	2.44	1.43
Fock exchange operator total time	91.7	46.5	28.3	13.1	10.3	8.1	8.5	9.5
Local and semi-local part	0.337	0.169	0.087	0.043	0.0316	0.0158	0.00805	0.00404
$H\Psi$ total time	92.04	46.67	28.39	13.14	10.33	8.12	8.51	9.50
Wavefunction MPI_Alltoallv	0.884	0.561	0.313	0.227	0.212	0.280	0.095	0.056
$\langle\Psi\|\Psi\rangle$ MPI_Allreduce	0.354	0.593	0.552	0.676	0.667	0.523	0.522	0.5243
Computation	1.43	0.72	0.37	0.19	0.145	0.078	0.04	0.023
Residual related total time	2.67	1.87	1.24	1.09	1.02	0.88	0.66	0.60
CPU-GPU memory copy	1.64235	0.8004	0.4094	0.2018	0.1477	0.0746	0.0395	0.0202
Computation time	2.3	1.16	0.59	0.31	0.265	0.142	0.073	0.04
Anderson mixing total time	3.94	1.98	1.00	0.51	0.387	0.194	0.102	0.0553
Computation time	0.1349	0.0672	0.0341	0.0170	0.0124	0.0062	0.0032	0.0016
MPI_Allreduce	0.123	0.176	0.152	0.224	0.219	0.160	0.164	0.171
Density evaluation total time	0.258	0.243	0.186	0.241	0.232	0.167	0.167	0.172
Others	2.66	1.98	1.72	1.54	1.57	1.73	1.66	1.85
per SCF time	101.36	52.4	32.5	16.4	13.4	10.9	10.9	12.1
Total time	2453.8	1269.1	783.0	393.9	323.2	260.9	262.5	286.6
Total speedup	3.6x	7.0x	11.3x	22.5x	27.4x	34x	33.8x	30.9x
$H\Psi$ **percentage**	90%	88.3%	87%	80%	76.7%	74.6%	77.8%	79.6%

Table 2: Breakdown of the total time into the time for MPI, CPU-GPU memory copy and computation. The CPU-GPU memory copy time and MPI time are all gathered in the runtime phase, the computational time is calculated by removing all the communication time from the total time in Table 1.

Number of GPUs	36	72	144	288	384	768	1536	3072
CPU-GPU memory copy time	60.80	29.94	16.04	8.57	6.79	4.15	2.82	2.24
MPI_Alltoallv time	20.97	13.34	7.40	5.38	4.99	6.64	2.41	0.68
MPI_Allreduce time	11.50	18.39	16.70	21.27	21.15	16.19	16.44	16.62
MPI_Bcast time	18.78	20.89	31.06	44.54	48.13	92.26	146.15	193.89
MPI_AllGatherv time	0.44	1.12	1.30	1.35	1.52	1.38	0.98	1.24
MPI total time	51.69	53.74	56.45	72.54	75.79	116.47	165.97	212.43
Computational time	2341.40	1185.42	710.54	312.83	240.60	140.34	93.73	71.96

The MPI_AllGatherv operationis performed after the exchange-correlation potential is calculated via Libxc [25]. It contributes less than 0.6% of total time and is thus negligible. The MPI_Bcast operation is mainly used in the Fock exchange operator to broadcast one wavefunction to all GPUs. The MPI_Allreduce operation is performed for computing the charge density and to compute the overlap matrix. These two components are the communication bottleneck. We notice that in our testing results, there are some fluctuations in terms of the communication time in Table 2. For example, the time for MPI_Allreduce peaks at around 288 and 384 GPUs, and the time for MPI_Alltoallv has a local peak at 768 GPUs. We confirm that such fluctuation can be repeatedly observed on Summit with the same configuration, and would like to investigate the origin of such fluctuation in the future.

Let us now analyze the performance of MPI_Bcast from the receiving side. In the Fock exchange operator calculation, each node receives 3072 wavefunctions. Each wavefunction consists of $N_G = 648,000$ complex numbers, which is 5.0MB in the single precision format. Thus total communication volume is $3072 \times 5.0MB = 15.36$GB . The communication time without overlapping with computation is about 7 seconds with 768 GPUs, thus the MPI communication speed is $15.36\text{GB}/7s = 2.2$ GB/s. The Summit supercomputer has two NICs connecting to two POWER 9 sockets, respectively. The communication bandwidth for each NIC is 12.5 GB/s. Since we have three MPI tasks per socket, the network bandwidth utilization rate is about 52.7% ($3 \times 2.2/12.5$) from the receiving side. In the GPU version of PWDFT, the CPU MPI_Bcast operation is overlapped with the GPU computation, and the MPI time shown at Table 1 is part of the total communication time. For example, in the 768 GPU case, almost half of the MPI communication time is overlapped by the computation time. Besides the wavefunction MPI_Bcast, we also have the MPI_Bcast of the gradient of charge density, etc. Data

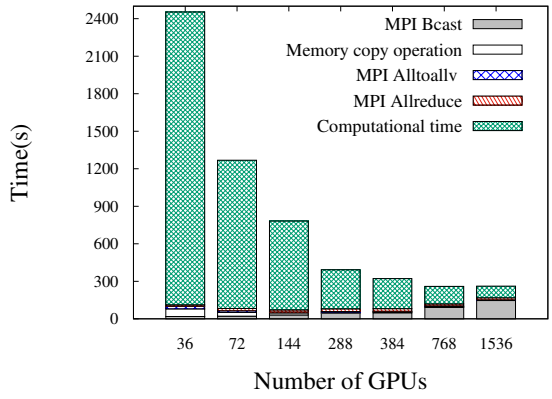


Figure 10: Strong scaling of different operations: MPI communication, CPU-GPU memory copy operation and computation.

size of the charge density is 40MB and it is also network bandwidth limited. The MPI_Allreduce operation is mainly used to evaluate the overlap matrix and charge density. Both operations are performed around 24 times in a single TDDFT step. The data size of the overlap matrix and the charge density vector are 144 MB and 40 MB, respectively. Hence the total data size for MPI_Allreduce is 4.4 GB per time step. This is less than the communication cost of MPI_Bcast but is of the same order of magnitude.

8 CONCLUSION

In this paper, we presented the GPU version of PWDFT for performing rt-TDDFT calculations with hybrid exchange-correlation functional. Our implementation is based on the planewave discretization and the parallel transport (PT) formulation. The PT formulation is used to increase the size of the time step, and therefore reduces the frequency of the Fock exchange operator applications. Our GPU version of PWDFT can effectively scale up to 768 GPUs, and calculate hybrid functional rt-TDDFT for a 1536 atom silicon system within 1.5 hours per femtosecond. The GPU code is 7 times faster than the CPU code under the same power consumption. The success of our GPU code relies on: 1) Adapt the hybrid parallelization scheme for the wavefunctions Ψ so that each Poisson-like problem can be evaluated on a single GPU; 2) Carefully optimize the Fock exchange operator calculation on GPU by combining CUFFT library and CUDA custom kernels, batch the GPU calculation to further utilize the GPU bandwidth, and overlap the CPU MPI communication and GPU computation; 3) Rewrite all other computational intensive parts, and move rt-TDDFT PT-CN algorithm almost entirely onto GPU. We remark that the performance of the hardware directly impacts the choice of optimal algorithms. For example, recently it has been shown that in CPU machines, the adaptively compressed exchange (ACE) algorithm [26] can be combined with the PT formulation to reduce the time for hybrid functional rt-TDDFT calculations [24]. In this work, we find that with the GPU acceleration of the Fock exchange calculation, the use of the PT formulation alone in fact leads to more efficient implementation on the Summit machine.

Although we only demonstrated the performance of the GPU implementation on the Summit supercomputer, our optimization strategies can also be beneficial for other heterogeneous architectures, such as the recently announced AMD GPU supercomputer Frontier targeting at exascale computing. Our optimization methods are neither limited to PWDFT, and can also be used for other rt-TDDFT as well as ground state DFT software packages. We also found that the scalability is eventually limited by the network bandwidth on the Summit supercomputer. Hence we expect the parallel performance could scale further with improved network bandwidth on future supercomputers. The unprecedented capability for large scale hybrid functional rt-TDDFT calculations can enable various practical applications, such as the study of self trapped excitons (STE). This will be our future work.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under Grant No. 1450372, No. DMS-1652330 (W. J. and L. L.), and by the Department of Energy under Grant No. DE-SC0017867 (L. L.). This work was also supported by the Director, Office of Science, the Office of Basic Energy Sciences (BES), Materials Sciences and Engineering (MSE) Division of the U.S. Department of Energy (DOE) through the theory of material (KC2301) program under Contract No. DEAC02- 05CH11231. It used resources of the Oak Ridge Leadership Computing Facility through the INCITE project. We thank Zhanghui Chen and Mauro Del Ben for helpful discussions.

REFERENCES

[1] M Alducin, R Díez Muiño, and JI Juaristi. 2017. Non-adiabatic effects in elementary reaction processes at metal surfaces. *Progress in Surface Science* 92, 4 (2017), 317–340.

[2] D. An and L. Lin. [n. d.]. Quantum dynamics with the parallel transport gauge. *arXiv:1804.02095* ([n. d.]).

[3] D. G. Anderson. 1965. Iterative procedures for nonlinear integral equations. *J. Assoc. Comput. Mach.* 12 (1965), 547–560.

[4] Xavier Andrade, Joseba Alberdi-Rodriguez, David A Strubbe, Micael J T Oliveira, Fernando Nogueira, Alberto Castro, Javier Muguerza, Agustin Arruabarrena, Steven G Louie, AlÃąn Aspuru-Guzik, Angel Rubio, and Miguel A L Marques. 2012. Time-dependent density-functional theory in massively parallel computer architectures: the octopus project. *J. Phys. Condens. Matter* 24 (2012), 233202.

[5] A. D. Becke. 1993. Density functional thermochemistry. III. The role of exact exchange. *J. Chem. Phys.* 98 (1993), 5648.

[6] I. Carnimeo, S. Baroni, and P. Giannozzi. 2018. Fast hybrid density-functional computations using plane-wave basis sets. *Electronic Structure* (2018).

[7] SK Cushing. 2017. Plasmonic hot carriers skip out in femtoseconds. *Nature Photonics* 11, 12 (2017), 748.

[8] A. Damle, L. Lin, and L. Ying. 2015. Compressed Representation of Kohn–Sham Orbitals via Selected Columns of the Density Matrix. *J. Chem. Theory Comput.* 11, 4 (2015), 1463–1469.

[9] W. Dawson and F. Gygi. 2015. Performance and Accuracy of Recursive Subspace Bisection for Hybrid DFT Calculations in Inhomogeneous Systems. *J. Chem. Theory Comput.* 11 (2015), 4655–4663.

[10] I. Duchemin and F. Gygi. 2010. A scalable and accurate algorithm for the computation of Hartree–Fock exchange. *Comput. Phys. Commun.* 181 (2010), 855–860.

[11] J. Fattebert, D. Osei-Kuffuor, E. W. Draeger, T. Ogitsu, and W. D. Krauss. 2016. Modeling Dilute Solutions Using First-Principles Molecular Dynamics: Computing more than a Million Atoms with over a Million Cores. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 12–22. https://doi.org/10.1109/SC.2016.88

[12] Adrián Gómez Pueyo, Miguel AL Marques, Angel Rubio, and Alberto Castro. 2018. Propagators for the time-dependent Kohn-Sham equations: multistep, Runge-Kutta, exponential Runge-Kutta, and commutator free Magnus methods. *J. Chem. Theory. Comput.* (2018).

[13] X. Gonze, F. Jollet, F. Abreu Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk, E. Bousquet, F. Bruneval, D. Caliste, M. CÃťtÃľ, F. Dahm, F. Da Pieve, M. Delaveau, M. Di Gennaro, B. Dorado, C. Espejo, G. Geneste, L. Genovese, A. Gerossier, M. Giantomassi, Y. Gillet, D.R. Hamann, L. He, G. Jomard, J. Laflamme Janssen, S. Le Roux, A. Levitt, A. Lherbier, F. Liu, I. LukaÄŊeviÄĞ, A. Martin, C. Martins, M.J.T. Oliveira, S. PoncÃľ, Y. Pouillon, T. Rangel, G.-M. Rignanese, A.H. Romero, B. Rousseau, O. Rubel, A.A. Shukri, M. Stankovski, M. Torrent, M.J. Van Setten, B. Van Troeye, M.J. Verstraete, D. Waroquiers, J. Wiktor, B. Xu, A. Zhou, and J.W. Zwanziger. 2016. Recent developments in the ABINIT software package. *Computer Physics Communications* 205 (2016), 106 – 131. https://doi.org/10.1016/j.cpc.2016.04.003
[14] Mohamed Hacene, Ani Anciaux-Sedrakian, Xavier Rozanska, Diego Klahr, Thomas Guignon, and Paul Fleurat-Lessard. 2012. Accelerating VASP electronic structure calculations using graphic processing units. *Journal of computational chemistry* 33, 32 (2012), 2581–2589.
[15] D. R. Hamann. 2013. Optimized norm-conserving Vanderbilt pseudopotentials. *Phys. Rev. B* 88 (2013), 085117.
[16] J. Heyd, G. E. Scuseria, and M. Ernzerhof. 2003. Hybrid functionals based on a screened Coulomb potential. *J. Chem. Phys.* 118, 18 (2003), 8207–8215.
[17] J. Heyd, G. E. Scuseria, and M. Ernzerhof. 2006. Erratum:"Hybrid functionals based on a screened Coulomb potential"[J. Chem. Phys. 118, 8207 (2003)]. *J. Chem. Phys.* 124, 21 (2006), 219906.
[18] W. Hu, L. Lin, and C. Yang. 2015. DGDFT: A massively parallel method for large scale density functional theory calculations. *J. Chem. Phys.* 143 (2015), 124110.
[19] W. Hu, L. Lin, and C. Yang. 2017. Interpolative separable density fitting decomposition for accelerating hybrid density functional calculations with applications to defects in silicon. *J. Chem. Theory Comput.* 13 (2017), 5420.
[20] Maxwell Hutchinson and Michael Widom. 2012. VASP on a GPU: Application to exact-exchange calculations of the stability of elemental boron. *Computer Physics Communications* 183, 7 (2012), 1422–1426.
[21] W. Jia, D. An, L.-W. Wang, and L. Lin. 2018. Fast real-time time-dependent density functional theory calculations with the parallel transport gauge. *J. Chem. Theory Comput.* 14 (2018), 5645.
[22] Weile Jia, Zongyan Cao, Long Wang, Jiyun Fu, Xuebin Chi, Weiguo Gao, and Lin-Wang Wang. 2013. The analysis of a plane wave pseudopotential density functional theory code on a GPU machine. *Computer Physics Communications* 184, 1 (2013), 9 – 18. https://doi.org/10.1016/j.cpc.2012.08.002
[23] Weile Jia, Jiyun Fu, Zongyan Cao, Long Wang, Xuebin Chi, Weiguo Gao, and Lin-Wang Wang. 2013. Fast plane wave density functional theory molecular dynamics calculations on multi-GPU machines. *J. Comput. Phys.* 251 (2013), 102 – 115. https://doi.org/10.1016/j.jcp.2013.05.005
[24] Weile Jia and Lin Lin. 2019. Fast real-time time-dependent hybrid functional calculations with the parallel transport gauge and the adaptively compressed exchange formulation. *Computer Physics Communications* (2019). https://doi.org/10.1016/j.cpc.2019.02.009
[25] Susi Lehtola, Conrad Steigemann, Micael J.T. Oliveira, and Miguel A.L. Marques. 2018. Recent developments in libxc âĂŤ A comprehensive library of functionals for density functional theory. *SoftwareX* 7 (2018), 1 – 5. https://doi.org/10.1016/j.softx.2017.11.002
[26] L. Lin. 2016. Adaptively Compressed Exchange Operator. *J. Chem. Theory Comput.* 12 (2016), 2242.
[27] L. Lin, J. Lu, L. Ying, and W. E. 2012. Adaptive local basis set for Kohn-Sham density functional theory in a discontinuous Galerkin framework I: Total energy calculation. *J. Comput. Phys.* 231 (2012), 2140–2154.
[28] Matteo Lucchini, Shunsuke A Sato, André Ludwig, Jens Herrmann, Mikhail Volkov, Lamia Kasmi, Yasushi Shinohara, Kazuhiro Yabana, Lukas Gallmann, and Ursula Keller. 2016. Attosecond dynamical Franz-Keldysh effect in polycrystalline diamond. *Science* 353, 6302 (2016), 916–919.
[29] Antoine Moulet, Julien B Bertrand, Till Klostermann, Alexander Guggenmos, Nicholas Karpowicz, and Eleftherios Goulielmakis. 2017. Soft x-ray excitonics. *Science* 357, 6356 (2017), 1134–1138.
[30] NVIDIA. [n. d.]. CUDA 8 Performance Overview. ([n. d.]). http://developer.download.nvidia.com/compute/cuda/compute-docs/cuda-performance-report.pdf
[31] G. Onida, L. Reining, and A. Rubio. 2002. Electronic excitations: density-functional versus many-body Green's-function approaches. *Rev. Mod. Phys.* 74 (2002), 601.
[32] J. P. Perdew, M. Ernzerhof, and K. Burke. 1996. Rationale for mixing exact exchange with density functional approximations. *J. Chem. Phys.* 105 (1996), 9982–9985.
[33] Laura E Ratcliff, A Degomme, José A Flores-Livas, Stefan Goedecker, and Luigi Genovese. 2018. Affordable and accurate large-scale hybrid-functional calculations on GPU-accelerated supercomputers. *Journal of Physics: Condensed Matter* 30, 9 (feb 2018), 095901. https://doi.org/10.1088/1361-648x/aaa8c9
[34] Joshua Romero, Everett Phillips, Gregory Ruetsch, Massimiliano Fatica, Filippo Spiga, and Paolo Giannozzi. 2018. A Performance Study of Quantum ESPRESSO's PWscf Code on Multi-core and GPU Systems. In *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, Stephen Jarvis, Steven Wright, and Simon Hammond (Eds.). Springer International Publishing, Cham, 67–87.
[35] E. Runge and E. K .U. Gross. 1984. Density-functional theory for time-dependent systems. *Phys. Rev. Lett.* 52 (1984), 997.
[36] S. A. Sato, Y. Taniguchi, Y. Shinohara, and K. Yabana. 2015. Nonlinear electronic excitations in crystalline solids using meta-generalized gradient approximation and hybrid functional in time-dependent density functional theory. *J. Chem. Phys.* 143, 22 (2015), 224116.
[37] Fabian Schlaepfer, Matteo Lucchini, Shunsuke A Sato, Mikhail Volkov, Lamia Kasmi, Nadja Hartmann, Angel Rubio, Lukas Gallmann, and Ursula Keller. 2018. Attosecond optical-field-enhanced carrier injection into the GaAs conduction band. *Nature Physics* (2018), 1.
[38] M. Schlipf and F. Gygi. 2015. Optimization algorithm for the generation of ONCV pseudopotentials. *Comput. Phys. Commun.* 196 (2015), 36–44.
[39] Martin Schultze, Krupa Ramasesha, CD Pemmaraju, SA Sato, D Whitmore, A Gandman, James S Prell, LJ Borja, D Prendergast, K Yabana, et al. 2014. Attosecond band-gap dynamics in silicon. *Science* 346, 6215 (2014), 1348–1352.
[40] Shijing Tan, Adam Argondizzo, Jindong Ren, Liming Liu, Jin Zhao, and Hrvoje Petek. 2017. Plasmonic coupling at a metal/semiconductor interface. *Nature Photonics* 11, 12 (2017), 806.
[41] C. A. Ullrich. 2011. *Time-dependent density-functional theory: concepts and applications.* Oxford Univ. Pr.
[42] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus, and W. De Jong. 2010. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.* 181 (2010), 1477–1489.
[43] Long Wang, Yue Wu, Weile Jia, Weiguo Gao, Xuebin Chi, and Lin-Wang Wang. 2011. Large Scale Plane Wave Pseudopotential Density Functional Theory Calculations on GPU Clusters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA, Article 71, 10 pages. https://doi.org/10.1145/2063384.2063479
[44] Lin-Wang Wang. 2001. Mask-function real-space implementations of nonlocal pseudopotentials. *Phys. Rev. B* 64 (Nov 2001), 201107. Issue 20. https://doi.org/10.1103/PhysRevB.64.201107
[45] X. Wu, A. Selloni, and R. Car. 2009. Order-N implementation of exact exchange in extended insulating systems. *Phys. Rev. B* 79, 8 (2009), 085102.
[46] K. Yabana and G. F. Bertsch. 1996. Time-dependent local-density approximation in real time. *Phys. Rev. B* 54 (1996), 4484–4487.
[47] Michael Zürch, Hung-Tzu Chang, Lauren J Borja, Peter M Kraus, Scott K Cushing, Andrey Gandman, Christopher J Kaplan, Myoung Hwan Oh, James S Prell, David Prendergast, et al. 2017. Direct and simultaneous observation of ultrafast electron and hole dynamics in germanium. *Nature communications* 8 (2017), 15734.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We ran the PWDFT code on Summit supercomputer with spectrum-mpi/10.2.0.11-20190201 and CUDA 9.2, FFTW3.3.8, and Lapack. We ran the GPU version of PWDFT with 6 MPI tasks per node, each MPI is connected to an individual GPU. The openMP threads number is set to 1 in both CPU and GPU runs. Compared to the CPU version of PWDFT running on 3072 CPU cores with 73 computing nodes, the GPU version is 7 times faster under the same power consumption using 72 GPUs(12 computing nodes). The GPU PWDFT can scale efficiently up to 768 GPUs, to have a speedup factor of 34 compared to the wall clock time of using 3072 CPU cores.

ARTIFACT AVAILABILITY

Software Artifact Availability: Some author-created software artifacts are NOT maintained in a public repository or are NOT available under an OSI-approved license.

Hardware Artifact Availability: There are no author-created hardware artifacts.

Data Artifact Availability: There are no author-created data artifacts.

Proprietary Artifacts: No author-created artifacts are proprietary.

List of URLs and/or DOIs where artifacts are available:

```
https://jiaweile@bitbucket.org/berkeleylab/crd-dgdft⌟
↪  .git; I think the repo can only be viewed by LBNL
↪  employee. If you want to have access to the code,
↪  please contact me at jiaweile@berkeley.edu
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Summit supercomputer, made by IBM, 6GPUs and 2 CPU sockets(44 CPU cores) per computing node

Operating systems and versions: Red Hat Enterprise Linux Server 7.5

Compilers and versions: xl/16.1.1-1, cuda/9.2.148

Applications and versions: PWDFT GPU

Libraries and versions: spectrum-mpi/10.2.0.11-20190201, CUBLAS, CUFFT, FFTW-3.3.8

Key algorithms: Parallel transport gauge and Crank Nicholson implicit time integrator

Input datasets and versions: 1536 Silicon atoms physical system

Paper Modifications: We ported the entire PWDFT code from CPU to GPU with a MPI+CUDA programming model. We write CUDA custom kernels when no CUDA libraries are available. The data are mostly kept on GPUs so that CPU-GPU memory copies are performed only when necessary. We take advantage of the CUDA-aware MPI on Summit and communicate directly using GPU data. The communications and computation are overlapped in our code. We also parallelize other parts, which takes about 1% on CPU, but can be the bottleneck once we speed up the computationally intensive parts by 30-40 times.

Output from scripts that gathers execution environment information.

```
LMOD_FAMILY_COMPILER_VERSION=16.1.1-1
MANPATH=/sw/summit/essl/6.1.0-2/essl/6.1/man:/autofs⌟
↪  /nccs-svm1_sw/summit/.swci/1-compute/opt/spack/2⌟
↪  0180914/linux-rhel7-ppc64le/xl-16.1.1-1/spectrum⌟
↪  -mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gnijoacjq⌟
↪  xrtblzk/share/man:/sw/sources/hpss/man:/sw/summi⌟
↪  t/xl/16.1.1-1/xlC/16.1.1/man/en_US:/sw/summit/xl⌟
↪  /16.1.1-1/xlf/16.1.1/man/en_US:/sw/summit/lmod/7⌟
↪  .7.10/rhel7.3_gnu4.8.5/lmod/lmod/share/man:/opt/⌟
↪  ibm/spectrumcomputing/lsf/10.1/man::
XDG_SESSION_ID=8321
XALT_ETC_DIR=/sw/summit/xalt/1.1.3/etc
LSB_EXEC_CLUSTER=summit
HOSTNAME=batch1
_ModuleTable003_=LXJ1bnRpbWUiLH0sZXNzbD17WyJmbiJdPSI⌟
↪  vc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgtcmh⌟
↪  lbDctcHBjNjRsZS9Db3JlL2Vzc2wvNi4xLjAtMiIsWyJmdWx⌟
↪  sTmFtZSJdPSJlc3NsLzYuMS4wLTIiLFsibG9hZE9yZGVyIl0⌟
↪  9MTEscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR⌟
↪  1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iZXNzbCIsfSx⌟
↪  oc2k9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVmaWxlcy9⌟
↪  zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS9oc2kvNS4⌟
↪  wLjIucDUubHVhIixbImZ1bGxOYW1lIl09ImhzaS81LjAuMi5⌟
↪  wNSIsWyJsb2FkT3JkZXIiXT0yLHByb3BUPXt9LFsic3RhY2t⌟
↪  EZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVz
LSB_EFFECTIVE_RSRCREQ=1*{select[((LN) && (type ==
↪  any))] order[r15s:pg] span[hosts=1]
↪  cu[type=rack:pref=config]}+42*{select[((CN) &&
↪  (type == any))] order[r15s:pg] span[ptile=42]
↪  cu[type=rack:pref=config]}
_ModuleTable009_=YXNlTVBBVEgiXT0iL3N3L3N1bW1pdC9sbW9⌟
↪  kLzcuNy4xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZWZpbGV⌟
↪  zL0xpbnV4Oi9zdy9zdW1taXQvbG1vZC83LjcuMTAvcmhlbDc⌟
↪  uM19nbnU0LjguNS9tb2R1bGVmaWxlcy9Db3JlOi9zdy9zdW1⌟
↪  taXQvbG1vZC83LjcuMTAvcmhlbDcuM19nbnU0LjguNS9sbW9⌟
↪  kL2xtb2QvbW9kdWxlZmlsZXMvQ29yZSIsfQ==
LSF_LOGDIR=/var/log/lsf
LSF_LIM_API_NTRIES=1
SHELL=/bin/bash
LSB_BATCH_JID=332784
HISTSIZE=1000
SSH_CLIENT=24.4.103.17 56300 22
LMOD_SYSTEM_DEFAULT_MODULES=DefApps
MODULEPATH_ROOT=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.⌟
↪  8.5/modulefiles
```

```
LIBRARY_PATH=/autofs/nccs-svm1_sw/summit/.swci/1-com
↪ pute/opt/spack/20180914/linux-rhel7-ppc64le/xl-1
↪ 6.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk
↪ 2mhdynsso6xo/lib64:/autofs/nccs-svm1_sw/summit/.
↪ swci/1-compute/opt/spack/20180914/linux-rhel7-pp
↪ c64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o
↪ 2k2myo2mssvug72icubygctlz/lib:/sw/summit/cuda/9.
↪ 2.148/lib64:/autofs/nccs-svm1_sw/summit/.swci/1-
↪ compute/opt/spack/20180914/linux-rhel7-ppc64le/x
↪ l-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd
↪ 6rixwrkcyd5gnijoacjqxrtblzk/lib:/autofs/nccs-svm
↪ 1_sw/summit/.swci/1-compute/opt/spack/20180914/l
↪ inux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1
↪ .7-uwak6exr43zvhu2vasiddrov57baf37t/lib
LSB_TRAPSIGS=trap # 15 10 12 2 1
LS_EXECCWD=/gpfs/alpine/proj-shared/nti009/USER/SC_T
↪ DDFT/si32/SI/environment
PAMI_IBV_ADAPTER_AFFINITY=1
LS_JOBPID=47341
LMOD_PKG=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmo
↪ d/lmod
LSB_MAX_NUM_PROCESSORS=43
LSB_JOBRES_CALLBACK=59349@batch1
OLDPWD=/gpfs/alpine/proj-shared/nti009/USER/SC_TDDFT
↪ /si32/SI/environment
OLCF_XL_ROOT=/sw/summit/xl/16.1.1-1
PAMI_ENABLE_STRIPING=0
LSF_SERVERDIR=/opt/ibm/spectrumcomputing/lsf/10.1/li
↪ nux3.10-glibc2.17-ppc64le-csm/etc
LSB_JOBID=332784
LSB_JOB_EXECUSER=USER
OMPI_FC=/sw/summit/xl/16.1.1-1/xlf/16.1.1/bin/xlf200
↪ 8_r
LMOD_VERSION=7.7.10
LSB_JOBRES_PID=47341
SSH_TTY=/dev/pts/162
OLCF_XLSMP_ROOT=/sw/summit/xl/16.1.1-1/xlsmp/5.1.1
OLCF_DARSHAN_RUNTIME_ROOT=/autofs/nccs-svm1_sw/summi
↪ t/.swci/1-compute/opt/spack/20180914/linux-rhel7
↪ -ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr
↪ 43zvhu2vasiddrov57baf37t
LSB_JOBNAME=si48
_ModuleTable007_=bWUiXT0ieGFsdC8xLjEuMyIsWyJsb2FkT3J
↪ kZXIiXT0zLHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTEsWyJ
↪ zdGF0dXMiXT0iYWN0aXZlIixbInVzZXJOYW1lIl09InhhbHQ
↪ iLH0seGw9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVmaWx
↪ lcy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS94bC8
↪ xNi4xLjEtMS5sdWEiLFsiZnVsbE5hbWUiXT0ieGwvMTYuMS4
↪ xLTEiLFsibG9hZE9yZGVyIl09MSxwcm9wVD17fSxbInN0YWN
↪ rRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c2V
↪ yTmFtZSJdPSJ4bCIsfSx9LG1wYXRoQT17Ii9hdXRvZnMvbmN
↪ jcy1zdm0xX3N3L3N1bW1pdC9tb2R1bGVmaWxlcy9zaXRlL2x
↪ pbnV4LXJoZWw3LXBwYzY0bGUvc3BlY3RydW0tbXBpLzEw
__LMOD_REF_COUNT_CMAKE_PREFIX_PATH=/autofs/nccs-svm1
↪ _sw/summit/.swci/1-compute/opt/spack/20180914/li
↪ nux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.
↪ 0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo:1;/autofs/ncc
↪ s-svm1_sw/summit/.swci/1-compute/opt/spack/20180
↪ 914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scala
↪ pack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz:1;/s
↪ w/summit/cuda/9.2.148:1;/autofs/nccs-svm1_sw/sum
↪ mit/.swci/1-compute/opt/spack/20180914/linux-rhe
↪ l7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-20
↪ 190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk:1;/autof
↪ s/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/
↪ 20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-r
↪ untime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t:1
__LMOD_REF_COUNT_LOADEDMODULES=xl/16.1.1-1:1;hsi/5.0
↪ .2.p5:1;xalt/1.1.3:1;lsf-tools/2.0:1;darshan-run
↪ time/3.1.7:1;DefApps:1;spectrum-mpi/10.2.0.11-20
↪ 190201:1;cuda/9.2.148:1;netlib-scalapack/2.0.2:1
↪ ;netlib-lapack/3.8.0:1;essl/6.1.0-2:1
BSUB_BLOCK_EXEC_HOST=
LSF_LIBDIR=/opt/ibm/spectrumcomputing/lsf/10.1/linux
↪ 3.10-glibc2.17-ppc64le-csm/lib
OPAL_PREFIX=/autofs/nccs-svm1_sw/summit/.swci/1-comp
↪ ute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16
↪ .1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rix
↪ wrkcyd5gnijoacjqxrtblzk
LSB_PROJECT_NAME=NTI009
USER=USER
```

```
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=0
↪  1;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;0
↪  1:mi=01;05;37;41:su=37;41:sg=30;43:ca=30;41:tw=3
↪  0;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tg
↪  z=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lh
↪  a=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.t
↪  lz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.z
↪  ip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;
↪  31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31
↪  :*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31
↪  :*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:
↪  *.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:
↪  *.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31
↪  :*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*
↪  .jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:
↪  *.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:
↪  *.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35
↪  :*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;3
↪  5:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;
↪  35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01
↪  ;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01
↪  ;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;
↪  35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01
↪  ;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;3
↪  5:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;3
↪  5:*.axv=01;35:*.anx=01;35:*.ogv=01;35:*.ogx=01;3
↪  5:*.aac=01;36:*.au=01;36:*.flac=01;36:*.mid=01;3
↪  6:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.mpc=01;
↪  36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.axa=01;3
↪  6:*.oga=01;36:*.spx=01;36:*.xspf=01;36:
LMOD_sys=Linux
LD_LIBRARY_PATH=/gpfs/alpine/stf007/world-shared/bs1
↪  /fftw/lib:/opt/ibm/spectrumcomputing/lsf/10.1/li
↪  nux3.10-glibc2.17-ppc64le-csm/lib:/sw/summit/ess
↪  l/6.1.0-2/essl/6.1/lib64:/autofs/nccs-svm1_sw/su
↪  mmit/.swci/1-compute/opt/spack/20180914/linux-rh
↪  el7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlw
↪  yyreoxj3d4aaefvk2mhdynsso6xo/lib64:/autofs/nccs-
↪  svm1_sw/summit/.swci/1-compute/opt/spack/2018091
↪  4/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapa
↪  ck-2.0.2-4pges4o2k2myo2mssvug72icubygctlz/lib:/s
↪  w/summit/cuda/9.2.148/lib64:/autofs/nccs-svm1_sw
↪  /summit/.swci/1-compute/opt/spack/20180914/linux
↪  -rhel7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11
↪  -20190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk/lib:/a
↪  utofs/nccs-svm1_sw/summit/.swci/1-compute/opt/sp
↪  ack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darsh
↪  an-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37
↪  t/lib:/sw/summit/xl/16.1.1-1/xlsmp/5.1.1/lib:/sw
↪  /summit/xl/16.1.1-1/xlmass/9.1.1/lib:/sw/summit/
↪  xl/16.1.1-1/xlC/16.1.1/lib:/sw/summit/xl/16.1.1-
↪  1/xlf/16.1.1/lib:/sw/summit/xl/16.1.1-1/lib
SBD_KRB5CCNAME_VAL=
PAMI_IBV_ENABLE_OOO_AR=1
LSB_EEXEC_REAL_UID=
LMOD_MPI_NAME=spectrum-mpi
CPATH=/autofs/nccs-svm1_sw/summit/.swci/1-compute/op
↪  t/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1
↪  /netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdyns
↪  so6xo/include:/sw/summit/cuda/9.2.148/include:/a
↪  utofs/nccs-svm1_sw/summit/.swci/1-compute/opt/sp
↪  ack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/spe
↪  ctrum-mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gni
↪  joacjqxrtblzk/include
_ModuleTable004_=ZXJOYW1lIl09ImhzaSIsfSxbImxzZi10b29
↪  scyJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmlsZXM
↪  vc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvbHNmLXR
↪  vb2xzLzIuMC5sdWEiLFsiZnVsbE5hbWUiXT0ibHNmLXRvb2x
↪  zLzIuMCIsWyJsb2FkT3JkZXIiXT00LHByb3BUPXt9LFsic3R
↪  hY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixbInV
↪  zZXJOYW1lIl09ImxzZi10b29scyIsfSxbIm5ldGxpYi1sYXB
↪  hY2siXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZpbGV
↪  zL3NpdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4xLjE
↪  tMS9uZXRsaWItbGFwYWNrLzMuOC4wLmx1YSIsWyJmdWxsTmF
↪  tZSJdPSJuZXRsaWItbGFwYWNrLzMuOC4wIixbImxvYWRP
HOSTTYPE=LINUXPPC64LE
__LMOD_REF_COUNT__LMFILES_=/sw/summit/modulefiles/si
↪  te/linux-rhel7-ppc64le/Core/xl/16.1.1-1.lua:1;/s
↪  w/summit/modulefiles/site/linux-rhel7-ppc64le/Co
↪  re/hsi/5.0.2.p5.lua:1;/sw/summit/modulefiles/sit
↪  e/linux-rhel7-ppc64le/Core/xalt/1.1.3.lua:1;/sw/
↪  summit/modulefiles/site/linux-rhel7-ppc64le/Core
↪  /lsf-tools/2.0.lua:1;/sw/summit/modulefiles/site
↪  /linux-rhel7-ppc64le/Core/darshan-runtime/3.1.7.
↪  lua:1;/sw/summit/modulefiles/site/linux-rhel7-pp
↪  c64le/Core/DefApps.lua:1;/sw/summit/modulefiles/
↪  site/linux-rhel7-ppc64le/xl/16.1.1-1/spectrum-mp
↪  i/10.2.0.11-20190201.lua:1;/sw/summit/modulefile
↪  s/site/linux-rhel7-ppc64le/xl/16.1.1-1/cuda/9.2.
↪  148.lua:1;/autofs/nccs-svm1_sw/summit/modulefile
↪  s/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.1
↪  1-20190201-6qypd6r/xl/16.1.1-1/netlib-scalapack/
↪  2.0.2.lua:1;/sw/summit/modulefiles/site/linux-rh
↪  el7-ppc64le/xl/16.1.1-1/netlib-lapack/3.8.0.lua:
↪  1;/sw/summit/modulefiles/site/linux-rhel7-ppc64l
↪  e/Core/essl/6.1.0-2:1
LSF_INVOKE_CMD=bsub
OLCF_XLMASS_ROOT=/sw/summit/xl/16.1.1-1/xlmass/9.1.1
OLCF_LMOD_ROOT=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8
↪  .5
LS_EXEC_T=START
PROJWORK=/gpfs/alpine/proj-shared
LSF_VERSION=34
LS_SUBCWD=/gpfs/alpine/proj-shared/nti009/USER/SC_TD
↪  DFT/si32/SI/environment
LSB_SUB_RES_REQ=1*{select[LN]span[hosts=1]} +
↪  42*{select[CN]span[ptile=42]}
LSB_UNIXGROUP_INT=USER
LMOD_FAMILY_MPI_VERSION=10.2.0.11-20190201
```

```
LSB_HOSTS=batch1 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13 h36n13 h36n13 h36n13 h36n13 h36n13
↪ h36n13 h36n13
NLSPATH=/sw/summit/xl/16.1.1-1/msg/en_US/%N:/sw/summ
↪ it/xl/16.1.1-1/xlC/16.1.1/msg/en_US/%N:/sw/summi
↪ t/xl/16.1.1-1/xlf/16.1.1/msg/en_US/%N
LSB_JOBFILENAME=/ccs/home/USER/.lsbatch/1554845590.3
↪ 32784
MAIL=/var/spool/mail/USER
PATH=/opt/ibm/spectrumcomputing/lsf/10.1/linux3.10-g
↪ libc2.17-ppc64le-csm/bin:/sw/sources/lsf-tools/2
↪ .0/summit/bin:/sw/summit/xalt/1.1.3/bin:/sw/summ
↪ it/cuda/9.2.148/bin:/autofs/nccs-svm1_sw/summit/
↪ .swci/1-compute/opt/spack/20180914/linux-rhel7-p
↪ pc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-201902
↪ 01-6qypd6rixwrkcyd5gnijoacjqxrtblzk/bin:/usr/bin
↪ :/usr/sbin:/autofs/nccs-svm1_sw/summit/.swci/1-c
↪ ompute/opt/spack/20180914/linux-rhel7-ppc64le/gc
↪ c-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vas
↪ iddrov57baf37t/bin:/sw/sources/hpss/bin:/sw/summ
↪ it/xl/16.1.1-1/xlC/16.1.1/bin:/sw/summit/xl/16.1
↪ .1-1/xlf/16.1.1/bin:/opt/ibm/spectrumcomputing/l
↪ sf/10.1/linux3.10-glibc2.17-ppc64le-csm/etc:/opt
↪ /ibm/csm/bin:/usr/local/bin:/usr/local/sbin:/opt
↪ /ibm/flightlog/bin:/opt/ibutils/bin:/opt/ibm/spe
↪ ctrum_mpi/jsm_pmix/bin:/opt/puppetlabs/bin:/usr/
↪ lpp/mmfs/bin
_ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNpb24
↪ iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19zaG9
↪ ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtbImN
↪ vbXBpbGVyIl09InhsIixbIm1waSJdPSJzcGVjdHJ1bS1tcGk
↪ iLH0sbVQ9e0RlZkFwcHM9e1siZm4iXT0iL3N3L3N1bW1pdC9
↪ tb2R1bGVmaWxlcy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGU
↪ vQ29yZS9EZWZBcHBzLmx1YSIsWyJmdWxsTmFtZSJdPSJEZWZ
↪ BcHBzIixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30sWyJzdGF
↪ ja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFsidXN
↪ lck5hbWUiXT0iRGVmQXBwcyIsfSxjdWRhPXtbImZuIl09Ii9
↪ zdy9zdW1taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51eC1y
__LMOD_REF_COUNT_NLSPATH=/sw/summit/xl/16.1.1-1/msg/
↪ en_US/%N:2;/sw/summit/xl/16.1.1-1/xlC/16.1.1/msg
↪ /en_US/%N:1;/sw/summit/xl/16.1.1-1/xlf/16.1.1/ms
↪ g/en_US/%N:1
LSB_JOBINDEX=0
PAMI_IBV_QP_SERVICE_LEVEL=8
OLCF_XLC_ROOT=/sw/summit/xl/16.1.1-1/xlC/16.1.1
OPAL_LIBDIR=/autofs/nccs-svm1_sw/summit/.swci/1-comp
↪ ute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16
↪ .1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rix
↪ wrkcyd5gnijoacjqxrtblzk/lib
LSB_EXIT_PRE_ABORT=99
LSB_JOBEXIT_STAT=0
OMPI_DIR=/autofs/nccs-svm1_sw/summit/.swci/1-compute
↪ /opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.
↪ 1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrk
↪ cyd5gnijoacjqxrtblzk
PWD=/gpfs/alpine/proj-shared/nti009/USER/SC_TDDFT/si
↪ 32/SI/environment
_LMFILES_=/sw/summit/modulefiles/site/linux-rhel7-pp
↪ c64le/Core/xl/16.1.1-1.lua:/sw/summit/modulefile
↪ s/site/linux-rhel7-ppc64le/Core/hsi/5.0.2.p5.lua
↪ :/sw/summit/modulefiles/site/linux-rhel7-ppc64le
↪ /Core/xalt/1.1.3.lua:/sw/summit/modulefiles/site
↪ /linux-rhel7-ppc64le/Core/lsf-tools/2.0.lua:/sw/
↪ summit/modulefiles/site/linux-rhel7-ppc64le/Core
↪ /darshan-runtime/3.1.7.lua:/sw/summit/modulefile
↪ s/site/linux-rhel7-ppc64le/Core/DefApps.lua:/sw/
↪ summit/modulefiles/site/linux-rhel7-ppc64le/xl/1
↪ 6.1.1-1/spectrum-mpi/10.2.0.11-20190201.lua:/sw/
↪ summit/modulefiles/site/linux-rhel7-ppc64le/xl/1
↪ 6.1.1-1/cuda/9.2.148.lua:/autofs/nccs-svm1_sw/su
↪ mmit/modulefiles/site/linux-rhel7-ppc64le/spectr
↪ um-mpi/10.2.0.11-20190201-6qypd6r/xl/16.1.1-1/ne
↪ tlib-scalapack/2.0.2.lua:/sw/summit/modulefiles/
↪ site/linux-rhel7-ppc64le/xl/16.1.1-1/netlib-lapa
↪ ck/3.8.0.lua:/sw/summit/modulefiles/site/linux-r
↪ hel7-ppc64le/Core/essl/6.1.0-2
OLCF_CUDA_ROOT=/sw/summit/cuda/9.2.148
OLCF_MODULEPATH_ROOT=/sw/summit/modulefiles
__LMOD_REF_COUNT_PYTHONPATH=/sw/summit/xalt/1.1.3/si
↪ te:1;/sw/summit/xalt/1.1.3/libexec:1
LSB_RES_GET_FANOUT_INFO=Y
MODULEPATH=/autofs/nccs-svm1_sw/summit/modulefiles/s
↪ ite/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.11-2
↪ 0190201-6qypd6r/xl/16.1.1-1:/sw/summit/modulefil
↪ es/site/linux-rhel7-ppc64le/xl/16.1.1-1:/sw/summ
↪ it/modulefiles/site/linux-rhel7-ppc64le/Core:/sw
↪ /summit/modulefiles/core:/sw/summit/lmod/7.7.10/
↪ rhel7.3_gnu4.8.5/modulefiles/Linux:/sw/summit/lm
↪ od/7.7.10/rhel7.3_gnu4.8.5/modulefiles/Core:/sw/
↪ summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmod/lmod/mo
↪ dulefiles/Core
LSB_CHKFILENAME=/ccs/home/USER/.lsbatch/1554845590.3
↪ 32784
_ModuleTable_Sz_=9
LOADEDMODULES=xl/16.1.1-1:hsi/5.0.2.p5:xalt/1.1.3:ls
↪ f-tools/2.0:darshan-runtime/3.1.7:DefApps:spectr
↪ um-mpi/10.2.0.11-20190201:cuda/9.2.148:netlib-sc
↪ alapack/2.0.2:netlib-lapack/3.8.0:essl/6.1.0-2
LMOD_SYSTEM_NAME=summit
LSB_DJOB_HOSTFILE=/ccs/home/USER/.lsbatch/1554845590
↪ .332784.hostfile
OLCF_ESSL_ROOT=/sw/summit/essl/6.1.0-2/essl/6.1
OMPI_LD_PRELOAD_PREPEND=/autofs/nccs-svm1_sw/summit/
↪ .swci/1-compute/opt/spack/20180914/linux-rhel7-p
↪ pc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr4
↪ 3zvhu2vasiddrov57baf37t/lib/libdarshan.so
OLCF_XLF_ROOT=/sw/summit/xl/16.1.1-1/xlf/16.1.1
```

```
LSF_JOB_TIMESTAMP_VALUE=1554845606
_ModuleTable005_=cmRlciJdPTEwLHByb3BUPXt9LFsic3RhY2t
↪ EZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVzZXJ
↪ OYW1lIl09Im5ldGxpYi1sYXBhY2siLH0sWyJuZXRsaWItc2N
↪ hbGFwYWNrIl09e1siZm4iXT0iL2F1dG9mcy9uY2NzLXN2bTF
↪ fc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgtcmh
↪ lbDctcHBjNjRsZS9zcGVjdHJ1bS1tcGkvMTAuMi4wLjExLTI
↪ wMTkwMjAxLTZxeXBkNnIveGwvMTYuMS4xLTEvbmV0bGliLXN
↪ jYWxhcGFjay8yLjAuMi5sdWEiLFsiZnVsbE5hbWUiXT0ibmV
↪ 0bGliLXNjYWxhcGFjay8yLjAuMiIsWyJsb2FkT3JkZXIiXT0
↪ 5LHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0dXM
↪ iXT0iYWN0aXZlIixbInVzZXJOYW1lIl09Im5ldGxpYi1z
LMOD_CMD=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmo
↪ d/lmod/libexec/lmod
LSB_AFFINITY_HOSTFILE=/ccs/home/USER/.lsbatch/155484
↪ 5590.332784.hostAffinityFile
LSF_BINDIR=/opt/ibm/spectrumcomputing/lsf/10.1/linux
↪ 3.10-glibc2.17-ppc64le-csm/bin
LSB_EXEC_HOSTTYPE=LINUXPPC64LE
LSB_DJOB_NUMPROC=43
HISTCONTROL=ignoredups
WORLDWORK=/gpfs/alpine/world-shared
SHLVL=4
MEMBERWORK=/gpfs/alpine/scratch/USER
HOME=/ccs/home/USER
OLCF_NETLIB_SCALAPACK_ROOT=/autofs/nccs-svm1_sw/summ
↪ it/.swci/1-compute/opt/spack/20180914/linux-rhel
↪ 7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pg
↪ es4o2k2myo2mssvug72icubygctlz
__LMOD_REF_COUNT_PATH=/sw/sources/lsf-tools/2.0/summ
↪ it/bin:2;/sw/summit/xalt/1.1.3/bin:1;/sw/summit/
↪ cuda/9.2.148/bin:1;/autofs/nccs-svm1_sw/summit/.
↪ swci/1-compute/opt/spack/20180914/linux-rhel7-pp
↪ c64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-2019020
↪ 1-6qypd6rixwrkcyd5gnijoacjqxrtblzk/bin:1;/usr/bi
↪ n:2;/usr/sbin:2;/autofs/nccs-svm1_sw/summit/.swc
↪ i/1-compute/opt/spack/20180914/linux-rhel7-ppc64
↪ le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvh
↪ u2vasiddrov57baf37t/bin:1;/sw/sources/hpss/bin:1
↪ ;/sw/summit/xl/16.1.1-1/xlC/16.1.1/bin:1;/sw/sum
↪ mit/xl/16.1.1-1/xlf/16.1.1/bin:1;/opt/ibm/spectr
↪ umcomputing/lsf/10.1/linux3.10-glibc2.17-ppc64le
↪ -csm/etc:1;/opt/ibm/spectrumcomputing/lsf/10.1/li
↪ nux3.10-glibc2.17-ppc64le-csm/bin:1;/opt/ibm/csm
↪ /bin:1;/usr/local/bin:1;/usr/local/sbin:1;/opt/i
↪ bm/flightlog/bin:1;/opt/ibutils/bin:1;/opt/ibm/s
↪ pectrum_mpi/jsm_pmix/bin:1;/opt/puppetlabs/bin:1
↪ ;/usr/lpp/mmfs/bin:1
OMPI_CC=/sw/summit/xl/16.1.1-1/xlC/16.1.1/bin/xlc_r
LSB_APPLICATION_NAME=small
__LMOD_REF_COUNT_CPATH=/autofs/nccs-svm1_sw/summit/.
↪ swci/1-compute/opt/spack/20180914/linux-rhel7-pp
↪ c64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreox
↪ j3d4aaefvk2mhdynsso6xo/include:1;/sw/summit/cuda
↪ /9.2.148/include:1;/autofs/nccs-svm1_sw/summit/.
↪ swci/1-compute/opt/spack/20180914/linux-rhel7-pp
↪ c64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-2019020
↪ 1-6qypd6rixwrkcyd5gnijoacjqxrtblzk/include:1
JOB_TERMINATE_INTERVAL=10
_ModuleTable002_=aGVsNy1wcGM2NGxlL3hsLzE2LjEuMS0xL2N
↪ 1ZGEvOS4yLjE0OC5sdWEiLFsiZnVsbE5hbWUiXT0iY3VkYS8
↪ 5LjIuMTQ4IixbImxvYWRPcmRlciJdPTgscHJvcFQ9e30sWyJ
↪ zdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFs
↪ idXNlck5hbWUiXT0iY3VkYSIsfSxbImRhcnNoYW4tcnVudGl
↪ tZSJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmlsZXM
↪ vc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvZGFyc2h
↪ hbi1ydW50aW1lLzMuMS43Lmx1YSIsWyJmdWxsTmFtZSJdPSJ
↪ kYXJzaGFuLXJ1bnRpbWUvMy4xLjciLFsibG9hZE9yZGVyIl0
↪ 9NSxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0xLFsic3RhdHV
↪ zIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJkYXJzaGFu
BINARY_TYPE_HPC=
CSM_ALLOCATION_ID=243773
LSB_ACCT_FILE=/tmp/332784.tmpdir/.1554845590.332784.
↪ acct
SHOST=login2
LSB_SUB_HOST=login2
_ModuleTable008_=LjIuMC4xMS0yMDE5MDIwMS02cXlwZDZyL3h
↪ sLzE2LjEuMS0xIiwiL3N3L3N1bW1pdC9tb2R1bGVmaWxlcy9
↪ zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUveGwvMTYuMS4xLTE
↪ iLCIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXg
↪ tcmhlbDctcHBjNjRsZS9Db3JlIiwiL3N3L3N1bW1pdC9tb2R
↪ 1bGVmaWxlcy9jb3JlIiwiL3N3L3N1bW1pdC9sbW9kLzcuNy4
↪ xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZWZpbGVzL0xpbnV
↪ 4IiwiL3N3L3N1bW1pdC9sbW9kLzcuNy4xMC9yaGVsNy4zX2d
↪ udTQuOC41L21vZHVsZWZpbGVzL0NvcmUiLCIvc3cvc3VtbWl
↪ 0L2xtb2QvNy43LjEwL3JoZWw3LjNfZ251NC44LjUvbG1vZC9
↪ sbW9kL21vZHVsZWZpbGVzL0NvcmUiLH0sWyJzeXN0ZW1C
OLCF_HSI_ROOT=/sw/sources/hpss
BASH_ENV=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmo
↪ d/lmod/init/bash
LSFUSER=USER
LSB_SUB_USER=USER
__LSF_JOB_TMPDIR__=/tmp/332784.tmpdir
LOGNAME=USER
MPI_ROOT=/autofs/nccs-svm1_sw/summit/.swci/1-compute
↪ /opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.
↪ 1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrk
↪ cyd5gnijoacjqxrtblzk
PYTHONPATH=/sw/summit/xalt/1.1.3/site:/sw/summit/xal
↪ t/1.1.3/libexec
PAMI_IBV_ENABLE_DCT=1
LSB_OUTPUTFILE=si48.332784
LSB_MCPU_HOSTS=batch1 1 h36n13 42
LSB_QUEUE=batch
LSB_OUTDIR=/gpfs/alpine/proj-shared/nti009/USER/SC_T
↪ DDFT/si32/SI/environment
```

```
OLCF_SPECTRUM_MPI_ROOT=/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk
CVS_RSH=ssh
SSH_CONNECTION=24.4.103.17 56300 128.219.134.72 22
XLSF_UIDDIR=/opt/ibm/spectrumcomputing/lsf/10.1/linux3.10-glibc2.17-ppc64le-csm/lib/uid
__LMOD_REF_COUNT_LIBRARY_PATH=/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo/lib64:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz/lib:1;/sw/summit/cuda/9.2.148/lib64:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk/lib:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t/lib:1
LSB_ECHKPNT_RSH_CMD=ssh -o 'PasswordAuthentication no' -o 'StrictHostKeyChecking no'
MODULESHOME=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmod/lmod
OMP_NUM_THREADS=7
OMPI_CXX=/sw/summit/xl/16.1.1-1/xlC/16.1.1/bin/xlc++_r
PKG_CONFIG_PATH=/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo/lib64/pkgconfig:/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz/lib/pkgconfig:/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t/lib/pkgconfig
__LMOD_REF_COUNT_LD_LIBRARY_PATH=/sw/summit/essl/6.1.0-2/essl/6.1/lib64:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo/lib64:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz/lib:1;/sw/summit/cuda/9.2.148/lib64:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk/lib:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t/lib:1;/sw/summit/xl/16.1.1-1/xlsmp/5.1.1/lib:1;/sw/summit/xl/16.1.1-1/xlmass/9.1.1/lib:1;/sw/summit/xl/16.1.1-1/xlC/16.1.1/lib:1;/sw/summit/xl/16.1.1-1/xlf/16.1.1/lib:1;/opt/ibm/spectrumcomputing/lsf/10.1/linux3.10-glibc2.17-ppc64le-csm/lib:1;/sw/summit/xl/16.1.1-1/lib:2
LMOD_SETTARG_FULL_SUPPORT=no
LESSOPEN=||/usr/bin/lesspipe.sh %s
LSF_CGROUP_TOPDIR_KEY=summit
LMOD_MPI_VERSION=10.2.0.11-20190201-6qypd6r
OMPI_MCA_io=romio321
__Init_Default_Modules=1
LMOD_FAMILY_COMPILER=xl
LSB_XFER_OP=
CMAKE_PREFIX_PATH=/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo:/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz:/sw/summit/cuda/9.2.148:/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd6rixwrkcyd5gnijoacjqxrtblzk:/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t
DISPLAY=login2:27.0
XDG_RUNTIME_DIR=/run/user/11840
__LMOD_REF_COUNT_PKG_CONFIG_PATH=/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreoxj3d4aaefvk2mhdynsso6xo/lib64/pkgconfig:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/xl-16.1.1-1/netlib-scalapack-2.0.2-4pges4o2k2myo2mssvug72icubygctlz/lib/pkgconfig:1;/autofs/nccs-svm1_sw/summit/.swci/1-compute/opt/spack/20180914/linux-rhel7-ppc64le/gcc-4.8.5/darshan-runtime-3.1.7-uwak6exr43zvhu2vasiddrov57baf37t/lib/pkgconfig:1
```

```
XALT_OLCF=1
XL_LINKER=/sw/summit/xalt/1.1.3/bin/ld
LSB_EEXEC_REAL_GID=
OLCF_NETLIB_LAPACK_ROOT=/autofs/nccs-svm1_sw/summit/
↪  .swci/1-compute/opt/spack/20180914/linux-rhel7-p
↪  pc64le/xl-16.1.1-1/netlib-lapack-3.8.0-ttlwyyreo
↪  xj3d4aaefvk2mhdynsso6xo
_ModuleTable006_=Y2FsYXBhY2siLH0sWyJzcGVjdHJ1bS1tcGk
↪  iXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3N
↪  pdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4xLjEtMS9
↪  zcGVjdHJ1bS1tcGkvMTAuMi4wLjExLTIwMTkwMjAxLmx1YSI
↪  sWyJmdWxsTmFtZSJdPSJzcGVjdHJ1bS1tcGkvMTAuMi4wLjE
↪  xLTIwMTkwMjAxIixbImxvYWRPcmRlciJdPTcscHJvcFQ9e30
↪  sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmU
↪  iLFsidXNlck5hbWUiXT0ic3BlY3RydW0tbXBpLzEwLjIuMC4
↪  xMS0yMDE5MDIwMSIsfSx4YWx0PXtbImZuIl09Ii9zdy9zdW1
↪  taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51eC1yaGVsNy1wcGM
↪  2NGxlL0NvcmUveGFsdC8xLjEuMy5sdWEiLFsiZnVsbE5h
__LMOD_REF_COUNT_MANPATH=/sw/summit/essl/6.1.0-2/ess
↪  l/6.1/man:1;/autofs/nccs-svm1_sw/summit/.swci/1-
↪  compute/opt/spack/20180914/linux-rhel7-ppc64le/x
↪  l-16.1.1-1/spectrum-mpi-10.2.0.11-20190201-6qypd
↪  6rixwrkcyd5gnijoacjqxrtblzk/share/man:1;/sw/sour
↪  ces/hpss/man:1;/sw/summit/xl/16.1.1-1/xlC/16.1.1
↪  /man/en_US:1;/sw/summit/xl/16.1.1-1/xlf/16.1.1/m
↪  an/en_US:1;/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8
↪  .5/lmod/lmod/share/man:1;/opt/ibm/spectrumcomput
↪  ing/lsf/10.1/man:1
LMOD_DIR=/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmo
↪  d/lmod/libexec
LSF_ENVDIR=/opt/ibm/spectrumcomputing/lsf/conf
__LMOD_Priority_PATH=/sw/sources/lsf-tools/2.0/summi
↪  t/bin:-9999;/sw/summit/xalt/1.1.3/bin:-9999
LMOD_FAMILY_MPI=spectrum-mpi
LSB_DJOB_RANKFILE=/ccs/home/USER/.lsbatch/1554845590
↪  .332784.hostfile
BASH_FUNC_module()=() {  eval $($LMOD_CMD bash "$@")
↪  && eval $(${LMOD_SETTARG_CMD:-:} -s sh)
}
BASH_FUNC_ml()=() {  eval $($LMOD_DIR/ml_cmd "$@")
}
_=/usr/bin/env
+ lsb_release -a
LSB Version:        :core-4.1-noarch:core-4.1-ppc64le
Distributor ID:        RedHatEnterpriseServer
Description:        Red Hat Enterprise Linux Server
↪  release 7.5 (Maipo)
Release:        7.5
Codename:        Maipo
+ uname -a
Linux batch1 4.14.0-49.18.1.el7a.ppc64le #1 SMP Thu
↪  Nov 29 03:27:24 EST 2018 ppc64le ppc64le ppc64le
↪  GNU/Linux
+ lscpu
Architecture:          ppc64le
Byte Order:            Little Endian
```

```
CPU(s):                128
On-line CPU(s) list:   0-127
Thread(s) per core:    4
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          6
Model:                 2.1 (pvr 004e 1201)
Model name:            POWER9, altivec supported
CPU max MHz:           3800.0000
CPU min MHz:           2300.0000
L1d cache:             32K
L1i cache:             32K
L2 cache:              512K
L3 cache:              10240K
NUMA node0 CPU(s):     0-63
NUMA node8 CPU(s):     64-127
NUMA node252 CPU(s):
NUMA node253 CPU(s):
NUMA node254 CPU(s):
NUMA node255 CPU(s):
+ cat /proc/meminfo
MemTotal:       601183424 kB
MemFree:        481976960 kB
MemAvailable:   508303872 kB
Buffers:                0 kB
Cached:         41386880 kB
SwapCached:             0 kB
Active:         22920448 kB
Inactive:       25564416 kB
Active(anon):   21301568 kB
Inactive(anon): 18956480 kB
Active(file):    1618880 kB
Inactive(file):  6607936 kB
Unevictable:    16840896 kB
Mlocked:        16840896 kB
SwapTotal:              0 kB
SwapFree:               0 kB
Dirty:                  0 kB
Writeback:              0 kB
AnonPages:      23940096 kB
Mapped:         21150080 kB
Shmem:          33160128 kB
Slab:           27915392 kB
SReclaimable:   20700928 kB
SUnreclaim:      7214464 kB
KernelStack:       50384 kB
PageTables:       107648 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:    300591680 kB
Committed_AS:   60759808 kB
VmallocTotal:   549755813888 kB
VmallocUsed:            0 kB
VmallocChunk:           0 kB
HardwareCorrupted:      0 kB
AnonHugePages:   3244032 kB
```

```
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:       26853376 kB
CmaFree:        23674880 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
+ inxi -F -c0
+ lsblk -a
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb       8:16   1  1.8T  0 disk
loop0     7:0    0        1 loop
sda       8:0    1  1.8T  0 disk
nvme0n1 259:0    0  1.5T  0 disk
+ lsscsi -s
+ module list
++ /sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmod/lmod
↪  /libexec/lmod bash
↪  list

Currently Loaded Modules:
  1) xl/16.1.1-1              7)
  ↪  spectrum-mpi/10.2.0.11-20190201
  2) hsi/5.0.2.p5             8) cuda/9.2.148
  3) xalt/1.1.3               9) netlib-scalapack/2.0.2
  4) lsf-tools/2.0           10) netlib-lapack/3.8.0
  5) darshan-runtime/3.1.7   11) essl/6.1.0-2
  6) DefApps
```

```
+ eval 'MODULEPATH="/autofs/nccs-svm1_sw/summit/modu
↪  lefiles/site/linux-rhel7-ppc64le/spectrum-mpi/10
↪  .2.0.11-20190201-6qypd6r/xl/16.1.1-1:/sw/summit/
↪  modulefiles/site/linux-rhel7-ppc64le/xl/16.1.1-1
↪  :/sw/summit/modulefiles/site/linux-rhel7-ppc64le
↪  /Core:/sw/summit/modulefiles/core:/sw/summit/lmo
↪  d/7.7.10/rhel7.3_gnu4.8.5/modulefiles/Linux:/sw/
↪  summit/lmod/7.7.10/rhel7.3_gnu4.8.5/modulefiles/
↪  Core:/sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmo
↪  d/lmod/modulefiles/Core";' export 'MODULEPATH;'
↪  '_ModuleTable001_="X01vZHVsZVRhYmxlXz17WyJNVHZlc
↪  nNpb24iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY
↪  19zaG9ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5P
↪  XtbImNvbXBpbGVyIl09InhsIixbIm1waSJdPSJzcGVjdHJ1b
↪  S1tcGkiLH0sbVQ9e0RlZkFwcHM9e1siZm4iXT0iL3N3L3N1b
↪  W1pdC9tb2R1bGVmaWxlcy9zaXRlL2xpbnV4LXJoZWw3LXBwY
↪  zY0bGUvQ29yZS9EZWZBcHBzLmx1YSIsWyJmdWxsTmFtZSJdP
↪  SJEZWZBcHBzIixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30sW
↪  yJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiL
↪  FsidXNlck5hbWUiXT0iRGVmQXBwcyIsfSxjdWRhPXtbImZuI
↪  l09Ii9zdy9zdW1taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51e
↪  C1y";' export
↪  '_ModuleTable001_;'
```

```
'_ModuleTable002_="aGVsNy1wcGM2NGxlL3hsLzE2LjEuMS0xL
↪  2N1ZGEvOS4yLjE0OC5sdWEiLFsiZnVsbE5hbWUiXT0iY3VkY
↪  S85LjIuMTQ4IixbImxvYWRPcmRlciJdPTgscHJvcFQ9e30sW
↪  yJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiL
↪  FsidXNlck5hbWUiXT0iY3VkYSIsfSxbImRhcnNoYW4tcnVud
↪  GltZSJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmlsZ
↪  XMvc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvZGFyc
↪  2hhbi1ydW50aW1lLzMuMS43Lmx1YSIsWyJmdWxsTmFtZSJdP
↪  SJkYXJzaGFuLXJ1bnRpbWUvMy4xLjciLFsibG9hZE9yZGVyI
↪  l09NSxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0xLFsic3Rhd
↪  HVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJkYXJzaGFu"
↪  ;' export
↪  '_ModuleTable002_;'
'_ModuleTable003_="LXJ1bnRpbWUiLH0sZXNzbD17WyJmbiJdP
↪  SIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgtc
↪  mhlbDctcHBjNjRsZS9Db3JlL2Vzc2wvNi4xLjAtMiIsWyJmd
↪  WxsTmFtZSJdPSJlc3NsLzYuMS4wLTIiLFsibG9hZE9yZGVyI
↪  l09MTEscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0Y
↪  XR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iZXNzbCIsf
↪  Sxoc2k9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVmaWxlc
↪  y9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS9oc2kvN
↪  S4wLjIucDUubHVhIixbImZ1bGxOYW1lIl09ImhzaS81LjAuM
↪  i5wNSIsWyJsb2FkT3JkZXIiXT0yLHByb3BUPXt9LFsic3RhY
↪  2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVz"
↪  ;' export
↪  '_ModuleTable003_;'
'_ModuleTable004_="ZXJOYW1lIl09ImhzaSIsfSxbImxzZi10b
↪  29scyJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmlsZ
↪  XMvc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvbHNmL
↪  XRvb2xzLzIuMC5sdWEiLFsiZnVsbE5hbWUiXT0ibHNmLXRvb
↪  2xzLzIuMCIsWyJsb2FkT3JkZXIiXT00LHByb3BUPXt9LFsic
↪  3RhY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixbI
↪  nVzZXJOYW1lIl09ImxzZi10b29scyIsfSxbIm5ldGxpYi1sY
↪  XBhY2siXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZpb
↪  GVzL3NpdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4xLL
↪  jEtMS9uZXRsaWItbGFwYWNrLzMuOC4wLmx1YSIsWyJmdWxsT
↪  mFtZSJdPSJuZXRsaWItbGFwYWNrLzMuOC4wIixbImxvYWRP"
↪  ;' export
↪  '_ModuleTable004_;'
'_ModuleTable005_="cmRlciJdPTEwLHByb3BUPXt9LFsic3RhY
↪  2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVzZ
↪  XJOYW1lIl09Im5ldGxpYi1sYXBhY2siLH0sWyJuZXRsaWItc
↪  2NhbGFwYWNrIl09e1siZm4iXT0iL2F1dG9mcy9uY2NzLXN2b
↪  TFfc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgtc
↪  mhlbDctcHBjNjRsZS9zcGVjdHJ1bS1tcGkvMTAuMi4wLjExL
↪  TIwMTkwMjAxLTZxeXBkNnIveGwvMTYuMS4xLTEvbmV0bGliL
↪  XNjYWxhcGFjay8yLjAuMi5sdWEiLFsiZnVsbE5hbWUiXT0ib
↪  mV0bGliLXNjYWxhcGFjay8yLjAuMiIsWyJsb2FkT3JkZXIiX
↪  T05LHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0d
↪  XMiXT0iYWN0aXZlIixbInVzZXJOYW1lIl09Im5ldGxpYi1z"
↪  ;' export
↪  '_ModuleTable005_;'
```

```
'_ModuleTable006_="Y2FsYXBhY2siLH0sWyJzcGVjdHJ1bS1tc
↪  GkiXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL
↪  3NpdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4xLjEtM
↪  S9zcGVjdHJ1bS1tcGkvMTAuMi4wLjExLTIwMTkwMjAxLmx1Y
↪  SIsWyJmdWxsTmFtZSJdPSJzcGVjdHJ1bS1tcGkvMTAuMi4wL
↪  jExLTIwMTkwMjAxIixbImxvYWRPcmRlciJdPTcscHJvcFQ9e
↪  30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3Rpd
↪  mUiLFsidXNlck5hbWUiXT0ic3BlY3RydW0tbXBpLzEwLjIuM
↪  C4xMS0yMDE5MDIwMSIsfSx4YWx0PXtbImZuIl09Ii9zdy9zd
↪  W1taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51eC1yaGVsNy1wc
↪  GM2NGxlL0NvcmUveGFsdC8xLjEuMy5sdWEiLFsiZnVsbE5h"
↪  ;' export
↪  '_ModuleTable006_;'
'_ModuleTable007_="bWUiXT0ieGFsdC8xLjEuMyIsWyJsb2FkT
↪  3JkZXIiXT0zLHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTEsW
↪  yJzdGF0dXMiXT0iYWN0aXZlIixbInVzZXJOYW1lIl09Inhhb
↪  HQiLH0seGw9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVma
↪  Wxlcy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS94b
↪  C8xNi4xLjEtMS5sdWEiLFsiZnVsbE5hbWUiXT0ieGwvMTYuM
↪  S4xLTEiLFsibG9hZE9yZGVyIl09MSxwcm9wVD17fSxbInN0Y
↪  WNrRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1c
↪  2VyTmFtZSJdPSJ4bCIsfSx9LG1wYXRoQT17Ii9hdXRvZnMvb
↪  mNjcy1zdm0xX3N3L3N1bW1pdC9tb2R1bGVmaWxlcy9zaXRlL
↪  2xpbnV4LXJoZWw3LXBwYzY0bGUvc3BlY3RydW0tbXBpLzEw"
↪  ;' export
↪  '_ModuleTable007_;'
'_ModuleTable008_="LjIuMC4xMS0yMDE5MDIwMS02cXlwZDZyL
↪  3hsLzE2LjEuMS0xIiwiL3N3L3N1bW1pdC9tb2R1bGVmaWxlc
↪  y9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUveGwvMTYuMS4xL
↪  TEiLCIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGlud
↪  XgtcmhlbDctcHBjNjRsZS9Db3JlIiwiL3N3L3N1bW1pdC9tb
↪  2R1bGVmaWxlcy9jb3JlIiwiL3N3L3N1bW1pdC9sbW9kLzcuN
↪  y4xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZWZpbGVzL0xpb
↪  nV4IiwiL3N3L3N1bW1pdC9sbW9kLzcuNy4xMC9yaGVsNy4zX
↪  2dudTQuOC41L21vZHVsZWZpbGVzL0NvcmUiLCIvc3cvc3Vtb
↪  Wl0L2xtb2QvNy43LjEwL3JoZWw3LjNfZ251NC44LjUvbG1vZ
↪  C9sbW9kL21vZHVsZWZpbGVzL0NvcmUiLH0sWyJzeXN0ZW1C"
↪  ;' export '_ModuleTable008_;'
↪  '_ModuleTable009_="YXNlTVBBVEgiXT0iL3N3L3N1bW1pd
↪  C9sbW9kLzcuNy4xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZ
↪  WZpbGVzL0xpbnV4Oi9zdy9zdW1taXQvbG1vZC83LjcuMTAvc
↪  mhlbDcuM19nbnU0LjguNS9tb2R1bGVmaWxlcy9Db3JlOi9zd
↪  y9zdW1taXQvbG1vZC83LjcuMTAvcmhlbDcuM19nbnU0LjguN
↪  S9sbW9kL2xtb2QvbW9kdWxlZmlsZXMvQ29yZSIsfQ==";'
↪  export '_ModuleTable009_;'
↪  '_ModuleTable_Sz_="9";' export
↪  '_ModuleTable_Sz_;'
++ MODULEPATH=/autofs/nccs-svm1_sw/summit/modulefile
↪  s/site/linux-rhel7-ppc64le/spectrum-mpi/10.2.0.1
↪  1-20190201-6qypd6r/xl/16.1.1-1:/sw/summit/module
↪  files/site/linux-rhel7-ppc64le/xl/16.1.1-1:/sw/s
↪  ummit/modulefiles/site/linux-rhel7-ppc64le/Core:
↪  /sw/summit/modulefiles/core:/sw/summit/lmod/7.7.
↪  10/rhel7.3_gnu4.8.5/modulefiles/Linux:/sw/summit
↪  /lmod/7.7.10/rhel7.3_gnu4.8.5/modulefiles/Core:/
↪  sw/summit/lmod/7.7.10/rhel7.3_gnu4.8.5/lmod/lmod
↪  /modulefiles/Core
++ export MODULEPATH
++ _ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNp
↪  b24iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19z
↪  aG9ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtb
↪  ImNvbXBpbGVyIl09InhsIixbIm1waSJdPSJzcGVjdHJ1bS1t
↪  cGkiLH0sbVQ9e0RlZkFwcHM9e1siZm4iXT0iL3N3L3N1bW1p
↪  dC9tb2R1bGVmaWxlcy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0
↪  bGUvQ29yZS9EZWZBcHBzLmx1YSIsWyJmdWxsTmFtZSJdPSJE
↪  ZWZBcHBzIixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30sWyJz
↪  dGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFsi
↪  dXNlck5hbWUiXT0iRGVmQXBwcyIsfSxjdWRhPXtbImZuIl09
↪  Ii9zdy9zdW1taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51eC1y
++ export _ModuleTable001_
++ _ModuleTable002_=aGVsNy1wcGM2NGxlL3hsLzE2LjEuMS0x
↪  L2N1ZGEvOS4yLjE0OC5sdWEiLFsiZnVsbE5hbWUiXT0iY3Vk
↪  YS85LjIuMTQ4IixbImxvYWRPcmRlciJdPTgscHJvcFQ9e30s
↪  WyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUi
↪  LFsidXNlck5hbWUiXT0iY3VkYSIsfSxbImRhcnNoYW4tcnVu
↪  dGltZSJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmls
↪  ZXMvc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvZGFy
↪  c2hhbi1ydW50aW1lLzMuMS43Lmx1YSIsWyJmdWxsTmFtZSJd
↪  PSJkYXJzaGFuLXJ1bnRpbWUvMy4xLjciLFsibG9hZE9yZGVy
↪  Il09NSxwcm9wVD17fSxbInN0YWNrRGVwdGgiXT0xLFsic3Rh
↪  dHVzIl09ImFjdGl2ZSIsWyJ1c2VyTmFtZSJdPSJkYXJzaGFu
++ export _ModuleTable002_
++ _ModuleTable003_=LXJ1bnRpbWUiLH0sZXNzbD17WyJmbiJd
↪  PSIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgt
↪  cmhlbDctcHBjNjRsZS9Db3JlL2Vzc2wvNi4xLjAtMiIsWyJm
↪  dWxsTmFtZSJdPSJlc3NsLzYuMS4wLTIiLFsibG9hZE9yZGVy
↪  Il09MTEscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0
↪  YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iZXNzbCIs
↪  fSxoc2k9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVmaWxl
↪  cy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS9oc2kv
↪  NS4wLjIucDUubHVhIixbImZ1bGxOYW1lIl09ImhzaS81LjAu
↪  Mi5wNSIsWyJsb2FkT3JkZXIiXT0yLHByb3BUPXt9LFsic3Rh
↪  Y2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVz
++ export _ModuleTable003_
++ _ModuleTable004_=ZXJOYW1lIl09ImhzaSIsfSxbImxzZi10
↪  b29scyJdPXtbImZuIl09Ii9zdy9zdW1taXQvbW9kdWxlZmls
↪  ZXMvc2l0ZS9saW51eC1yaGVsNy1wcGM2NGxlL0NvcmUvbHNm
↪  LXRvb2xzLzIuMC5sdWEiLFsiZnVsbE5hbWUiXT0ibHNmLXRv
↪  b2xzLzIuMCIsWyJsb2FkT3JkZXIiXT00LHByb3BUPXt9LFsi
↪  c3RhY2tEZXB0aCJdPTEsWyJzdGF0dXMiXT0iYWN0aXZlIixb
↪  InVzZXJOYW1lIl09ImxzZi10b29scyIsfSxbIm5ldGxpYi1s
↪  YXBhY2siXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZp
↪  bGVzL3NpdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4x
↪  LjEtMS5uZXRsaWItbGFwYWNrLzMuOC4wLmx1YSIsWyJmdWxs
↪  TmFtZSJdPSJuZXRsaWItbGFwYWNrLzMuOC4wIixbImxvYWRP
++ export _ModuleTable004_
```

```
++ _ModuleTable005_=cmRlciJdPTEwLHByb3BUPXt9LFsic3Rh
↪  Y2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN0aXZlIixbInVz
↪  ZXJOYW1lIl09Im5ldGxpYi1sYXBhY2siLH0sWyJuZXRsaWIt
↪  c2NhbGFwYWNrIl09e1siZm4iXT0iL2F1dG9mcy9uY2NzLXN2
↪  bTFfc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGludXgt
↪  cmhlbDctcHBjNjRsZS9zcGVjdHJ1bS1tcGkvMTAuMi4wLjEx
↪  LTIwMTkwMjAxLTZxeXBkNnIveGwvMTYuMS4xLTEvbmV0bGli
↪  LXNjYWxhcGFjay8yLjAuMi5sdWEiLFsiZnVsbE5hbWUiXT0i
↪  bmV0bGliLXNjYWxhcGFjay8yLjAuMiIsWyJsb2FkT3JkZXIi
↪  XT05LHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0
↪  dXMiXT0iYWN0aXZlIixbInVzZXJOYW1lIl09Im5ldGxpYi1z
++ export _ModuleTable005_
++ _ModuleTable006_=Y2FsYXBhY2siLH0sWyJzcGVjdHJ1bS1t
↪  cGkiXT17WyJmbiJdPSIvc3cvc3VtbWl0L21vZHVsZWZpbGVz
↪  L3NpdGUvbGludXgtcmhlbDctcHBjNjRsZS94bC8xNi4xLjEt
↪  MS9zcGVjdHJ1bS1tcGkvMTAuMi4wLjExLTIwMTkwMjAxLmx1
↪  YSIsWyJmdWxsTmFtZSJdPSJzcGVjdHJ1bS1tcGkvMTAuMi4w
↪  LjExLTIwMTkwMjAxIixbImxvYWRPcmRlciJdPTcscHJvcFQ9
↪  e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3Rp
↪  dmUiLFsidXNlck5hbWUiXT0ic3BlY3RydW0tbXBpLzEwLjIu
↪  MC4xMS0yMDE5MDIwMSIsfSx4YWx0PXtbImZuIl09Ii9zdy9z
↪  dW1taXQvbW9kdWxlZmlsZXMvc2l0ZS9saW51eC1yaGVsNy1w
↪  cGM2NGxlL0NvcmUveGFsdC8xLjEuMy5sdWEiLFsiZnVsbE5h
++ export _ModuleTable006_
++ _ModuleTable007_=bWUiXT0ieGFsdC8xLjEuMyIsWyJsb2Fk
↪  T3JkZXIiXT0zLHByb3BUPXt9LFsic3RhY2tEZXB0aCJdPTEs
↪  WyJzdGF0dXMiXT0iYWN0aXZlIixbInVzZXJOYW1lIl09Inhh
↪  bHQiLH0seGw9e1siZm4iXT0iL3N3L3N1bW1pdC9tb2R1bGVm
↪  aWxlcy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUvQ29yZS94
↪  bC8xNi4xLjEtMS5sdWEiLFsiZnVsbE5hbWUiXT0ieGwvMTYu
↪  MS4xLTEiLFsibG9hZE9yZGVyIl09MSxwcm9wVD17fSxbInN0
↪  YWNrRGVwdGgiXT0xLFsic3RhdHVzIl09ImFjdGl2ZSIsWyJ1
↪  c2VyTmFtZSJdPSJ4bCIsfSx9LG1wYXRoQT17Ii9hdXRvZnMv
↪  bmNjcy1zdm0xX3N3L3N1bW1pdC9tb2R1bGVmaWxlcy9zaXRl
↪  L2xpbnV4LXJoZWw3LXBwYzY0bGUvc3BlY3RydW0tbXBpLzEw
++ export _ModuleTable007_
++ _ModuleTable008_=LjIuMC4xMS0yMDE5MDIwMS02cXlwZDZy
↪  L3hsLzE2LjEuMS0xIiwiL3N3L3N1bW1pdC9tb2R1bGVmaWxl
↪  cy9zaXRlL2xpbnV4LXJoZWw3LXBwYzY0bGUveGwvMTYuMS4x
↪  LTEiLCIvc3cvc3VtbWl0L21vZHVsZWZpbGVzL3NpdGUvbGlu
↪  dXgtcmhlbDctcHBjNjRsZS9Db3JlIiwiL3N3L3N1bW1pdC9t
↪  b2R1bGVmaWxlcy9jb3JlIiwiL3N3L3N1bW1pdC9sbW9kLzcu
↪  Ny4xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZWZpbGVzL0xp
↪  bnV4IiwiL3N3L3N1bW1pdC9sbW9kLzcuNy4xMC9yaGVsNy4z
↪  X2dudTQuOC41L21vZHVsZWZpbGVzL0NvcmUiLCIvc3cvc3Vt
↪  bWl0L2xtb2QvNy43LjEwL3JoZWw3LjNfZ251NC44LjUvbG1v
↪  ZC9sbW9kL21vZHVsZWZpbGVzL0NvcmUiLH0sWyJzeXN0ZW1C
++ export _ModuleTable008_
++ _ModuleTable009_=YXNlTVBBVEgiXT0iL3N3L3N1bW1pdC9s
↪  bW9kLzcuNy4xMC9yaGVsNy4zX2dudTQuOC41L21vZHVsZWZp
↪  bGVzL0xpbnV4Oi9zdy9zdW1taXQvbG1vZC83LjcuMTAvcmhl
↪  bDcuM19nbnU0LjguNS9tb2R1bGVmaWxlcy9Db3JlOi9zdy9z
↪  dW1taXQvbG1vZC83LjcuMTAvcmhlbDcuM19nbnU0LjguNS9s
↪  bW9kL2xtb2QvbW9kdWxlZmlsZXMvQ29yZSIsfQ==
++ export _ModuleTable009_
++ _ModuleTable_Sz_=9
++ export _ModuleTable_Sz_
++ : -s sh
+ eval
+ nvidia-smi
Tue Apr  9 17:33:47 2019
+-----------------------------------------------------
↪ -------------------------+
| NVIDIA-SMI 396.64                 Driver Version:
↪  396.64                    |
|-------------------------------+----------------------
↪ ---+-----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A
↪  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage
↪  | GPU-Util  Compute M. |
|===============================+======================
↪  ===+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off
↪  |                    0 |
| N/A   37C    P0    37W / 300W |      0MiB / 16128MiB
↪  |      0%   E. Process |
+-------------------------------+----------------------
↪ ---+-----------------------+
|   1  Tesla V100-SXM2...  On   | 00000004:05:00.0 Off
↪  |                    0 |
| N/A   40C    P0    37W / 300W |      0MiB / 16128MiB
↪  |      0%   E. Process |
+-------------------------------+----------------------
↪ ---+-----------------------+
|   2  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off
↪  |                    0 |
| N/A   36C    P0    38W / 300W |      0MiB / 16128MiB
↪  |      0%   E. Process |
+-------------------------------+----------------------
↪ ---+-----------------------+
|   3  Tesla V100-SXM2...  On   | 00000035:04:00.0 Off
↪  |                    0 |
| N/A   45C    P0    38W / 300W |      0MiB / 16128MiB
↪  |      0%   E. Process |
+-------------------------------+----------------------
↪ ---+-----------------------+

+-----------------------------------------------------
↪ -------------------------+
| Processes:
↪             GPU Memory |
|  GPU       PID   Type   Process name
↪             Usage      |
|=============================================================
↪  ========================|
|  No running processes found
↪                          |
+-----------------------------------------------------
↪ -------------------------+
```

```
+ lshw -short -quiet -sanitize
+ cat
+ lspci
0000:00:00.0 PCI bridge: IBM Device 04c1
0000:01:00.0 Non-Volatile memory controller: Samsung
↪  Electronics Co Ltd NVMe SSD Controller 172Xa (rev
↪  01)
0001:00:00.0 PCI bridge: IBM Device 04c1
0001:01:00.0 USB controller: Texas Instruments
↪  TUSB73x0 SuperSpeed USB 3.0 xHCI Host Controller
↪  (rev 02)
0002:00:00.0 PCI bridge: IBM Device 04c1
0002:01:00.0 PCI bridge: ASPEED Technology, Inc.
↪  AST1150 PCI-to-PCI Bridge (rev 04)
0002:02:00.0 VGA compatible controller: ASPEED
↪  Technology, Inc. ASPEED Graphics Family (rev 41)
0003:00:00.0 PCI bridge: IBM Device 04c1
0003:01:00.0 Infiniband controller: Mellanox
↪  Technologies MT28800 Family [ConnectX-5 Ex]
0003:01:00.1 Infiniband controller: Mellanox
↪  Technologies MT28800 Family [ConnectX-5 Ex]
0004:00:00.0 PCI bridge: IBM Device 04c1
0004:01:00.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0004:01:00.1 System peripheral: PLX Technology, Inc.
↪  Device 87d0 (rev ca)
0004:01:00.2 System peripheral: PLX Technology, Inc.
↪  Device 87d0 (rev ca)
0004:01:00.3 System peripheral: PLX Technology, Inc.
↪  Device 87d0 (rev ca)
0004:01:00.4 System peripheral: PLX Technology, Inc.
↪  Device 87d0 (rev ca)
0004:02:02.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0004:02:0a.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0004:02:0b.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0004:02:0c.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0004:03:00.0 SATA controller: Marvell Technology
↪  Group Ltd. 88SE9235 PCIe 2.0 x2 4-port SATA 6 Gb/s
↪  Controller (rev 11)
0004:04:00.0 3D controller: NVIDIA Corporation
↪  GV100GL [Tesla V100 SXM2] (rev a1)
0004:05:00.0 3D controller: NVIDIA Corporation
↪  GV100GL [Tesla V100 SXM2] (rev a1)
0005:00:00.0 PCI bridge: IBM Device 04c1
0005:01:00.0 Ethernet controller: Broadcom Limited
↪  NetXtreme BCM5719 Gigabit Ethernet PCIe (rev 01)
0005:01:00.1 Ethernet controller: Broadcom Limited
↪  NetXtreme BCM5719 Gigabit Ethernet PCIe (rev 01)
0006:00:00.0 Bridge: IBM Device 04ea (rev 01)
0006:00:00.1 Bridge: IBM Device 04ea (rev 01)
0006:00:00.2 Bridge: IBM Device 04ea (rev 01)
0006:00:01.0 Bridge: IBM Device 04ea (rev 01)
0006:00:01.1 Bridge: IBM Device 04ea (rev 01)
0006:00:01.2 Bridge: IBM Device 04ea (rev 01)
0007:00:00.0 Bridge: IBM Device 04ea (rev 01)
0007:00:00.1 Bridge: IBM Device 04ea (rev 01)
0007:00:00.2 Bridge: IBM Device 04ea (rev 01)
0007:00:01.0 Bridge: IBM Device 04ea (rev 01)
0007:00:01.1 Bridge: IBM Device 04ea (rev 01)
0007:00:01.2 Bridge: IBM Device 04ea (rev 01)
0030:00:00.0 PCI bridge: IBM Device 04c1
0030:01:00.0 Ethernet controller: Broadcom Limited
↪  NetXtreme II BCM57800 1/10 Gigabit Ethernet (rev
↪  10)
0030:01:00.1 Ethernet controller: Broadcom Limited
↪  NetXtreme II BCM57800 1/10 Gigabit Ethernet (rev
↪  10)
0030:01:00.2 Ethernet controller: Broadcom Limited
↪  NetXtreme II BCM57800 1/10 Gigabit Ethernet (rev
↪  10)
0030:01:00.3 Ethernet controller: Broadcom Limited
↪  NetXtreme II BCM57800 1/10 Gigabit Ethernet (rev
↪  10)
0033:00:00.0 PCI bridge: IBM Device 04c1
0033:01:00.0 Infiniband controller: Mellanox
↪  Technologies MT28800 Family [ConnectX-5 Ex]
0033:01:00.1 Infiniband controller: Mellanox
↪  Technologies MT28800 Family [ConnectX-5 Ex]
0034:00:00.0 PCI bridge: IBM Device 04c1
0035:00:00.0 PCI bridge: IBM Device 04c1
0035:01:00.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0035:02:04.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0035:02:05.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0035:02:0d.0 PCI bridge: PLX Technology, Inc. Device
↪  8725 (rev ca)
0035:03:00.0 3D controller: NVIDIA Corporation
↪  GV100GL [Tesla V100 SXM2] (rev a1)
0035:04:00.0 3D controller: NVIDIA Corporation
↪  GV100GL [Tesla V100 SXM2] (rev a1)
```