Accurate load balancing accelerates Lagrangian simulation of water ages on distributed, multi-GPU platforms

CHEN YANG[1], Reed M. Maxwell[1], and Richard Valent[2]

[1]Princeton University
[2]National Center for Atmospheric Research

November 22, 2022

Abstract

Water age is a fundamental descriptor of source, storage, and mixing of water parcels in a watershed. The Lagrangian, particle tracking, approach is a powerful tool for physically-based modeling of water age distributions, but its application has been hampered since it is computationally demanding. In this study, we present a parallel approach for particle tracking simulations. This approach uses multi-GPU with MPI parallelism based on domain decomposition. An inherent challenge of distributed parallelization of Lagrangian approaches is the disparity in computational work or load imbalance (LIB) among different processing elements (PEs). Here, load balancing (LB) schemes were proposed to dynamically balance the distribution of particles across PEs during runtime. In the followed hillslope simulations, LIB was observed in all LB-disabled runs, e.g., with a load ratio of 423.62% by using 2-GPU in LW_Shrub case. LB schemes then accurately balanced the load distribution and improved the parallel scaling. Additionally, the parallel approach showed excellent overall speedup: a 60-fold improvement using 4-GPU relative to the serial run. A regional scale application further demonstrated the LB performance. The parallel time used by 8-GPU without LB was 31.33% reduced after LB was activated. When increasing 8-GPU with LB to 16-GPU with LB, it showed parallel scalability by reducing the parallel time of ~50%. This work shows how massively parallel computing can be applied to particle tracking in water age simulations. It also demonstrates the practical importance of load balancing in this context, which enables the large-scale simulations with an increased complexity of flow paths.

Accurate load balancing accelerates Lagrangian simulation of water ages on distributed, multi-GPU platforms

Chen Yang[1*], Reed M. Maxwell[1,2*], Richard Valent[3]

[1]Department of Civil and Environmental Engineering, Princeton University, Princeton, NJ 08544, USA
[2]High Meadows Environmental Institute, Princeton University, Princeton, NJ 08544, USA
[3]Computational and Information Systems Laboratory, National Center for Atmospheric Research, Boulder, CO 80305, USA

Corresponding author:
Chen Yang cy15@princeton.edu
Reed Maxwell reedmaxwell@princeton.edu

Key points:

- Massively parallel computing of Lagrangian water-age simulations is realized
- Mechanisms of load imbalance are identified and LB schemes are proposed
- Parallel performance of the approach is demonstrated at the regional scale

Abstract

Water age is a fundamental descriptor of source, storage, and mixing of water parcels in a watershed. The Lagrangian, particle tracking, approach is a powerful tool for physically-based modeling of water age distributions, but its application has been hampered since it is computationally demanding. In this study, we present a parallel approach for particle tracking simulations. This approach uses multi-GPU with MPI parallelism based on domain decomposition. An inherent challenge of distributed parallelization of Lagrangian approaches is the disparity in computational work or load imbalance (LIB) among different processing elements (PEs). Here, load balancing (LB) schemes were proposed to dynamically balance the distribution of particles across PEs during runtime. In the followed hillslope simulations, LIB was observed in all LB-disabled runs, e.g., with a load ratio of 423.62% by using 2-GPU in LW_Shrub case. LB schemes then accurately balanced the load distribution and improved the parallel scaling. Additionally, the parallel approach showed excellent overall speedup: a 60-fold improvement using 4-GPU relative to the serial run. A regional scale application further demonstrated the LB performance. The parallel time used by 8-GPU without LB was 31.33% reduced after LB was activated. When increasing 8-GPU with LB to 16-GPU with LB, it showed parallel scalability by reducing the parallel time of ~50%. This work shows how massively parallel computing can be applied to particle tracking in water age simulations. It also demonstrates the practical importance of load balancing in this context, which enables the large-scale simulations with an increased complexity of flow paths.

Keywords

Water age, Particle tracking, Multi-GPU with MPI, Domain decomposition, Load balancing

1. Introduction

Water age is an important metric that can unravel the journey that water-parcels and pollutants take while traveling through a watershed (Botter, Bertuzzo, & Rinaldo, 2011). Methods used to quantify water ages mainly include tracer data, lumped analytical models, and distributed numerical models (Nicholas B. Engdahl & Maxwell, 2014). Tracer data are direct observations of system behavior, however they have technical challenges such as tracer selection and data interpretation (Sprenger et al., 2019). Moreover, the limited sampling cannot provide a full view of the spatiotemporal variations of water ages (Nicholas B. Engdahl, McCallum, & Massoudieh, 2016; McCallum, Engdahl, Ginn, & Cook, 2014). Analytical solutions can also provide an understanding of the real system, although the limitations of this approach due to simplifications have been well acknowledged by the community (Basu, Jindal, Schilling, Wolter, & Takle, 2012). For distributed numerical models, the application of Eulerian framework is hampered by the complications to solve the high-dimensional governing equation of water age (Gomez & Wilson, 2013). Therefore, Lagrangian approach based on integrated hydrologic modeling has become a promising tool to simulate transient age distributions (Nicholas B. Engdahl & Maxwell, 2014; Jing et al., 2019; Wilusz, Harman, Ball, Maxwell, & Buda, 2019).

The Lagrangian approach is computationally demanding which limits widespread application (Sprenger et al., 2019; Wilusz et al., 2019). Current studies of water ages using particle tracking are limited to either catchments at small scales (Wilusz et al., 2019; J. Yang, Heidbüchel, Musolff, Reinstorf, & Fleckenstein, 2018) or larger scales with a limited number of particles or for steady-state conditions (Jing et al., 2020; Maxwell et al., 2016). Recently, as global water security and climate change become increasing concerns, a growing number of studies have proposed simulating water age at larger scales over long time-periods at high resolution (Maxwell et al., 2016; McGuire et al., 2005; Starn, Kauffman, Carlson, Reddy, & Fienen, 2021). Accomplishing these goals will significantly increase the computational burden of particle tracking; massively parallel computing represents a promising solution.

Currently, there are few studies documenting parallelization of particle tracking approaches for water age (Jing et al., 2020; Wilusz et al., 2019; J. Yang et al., 2018). Maxwell, Condon, Danesh-Yazdi, and Bearup (2019) developed EcoSLIM, a particle tracking code, using OpenMP (Open Multi-Processing) on CPU (Central Processing Unit). Yang and coauthors (C. Yang et al., 2021), added CPU-based MPI (Message Passing Interface) and multi-GPU (Graphics Processing

Unit) with OpenMP into EcoSLIM. Ji, Luo, and Wang (2019) sped up MODPATH (Pollock, 2016) through multi-GPU with OpenMP/MPI based on domain decomposition (DDC). However, their parallelized codes are limited to steady state simulations, and MODPATH is unable to simulate evapotranspiration (ET) age and source water composition. N. B. Engdahl, Schmidt, and Benson (2019) proposed two schemes for speeding up particle tracking simulations with mass transfer to represent chemical reactions. One of the schemes also implemented MPI parallelism through DDC. In DDC-based MPI parallelism, a typical problem encountered is load imbalance among processing elements (MPI processes and/or GPUs), which can present a challenge for good parallel efficiency. However, load imbalance has not been quantified in water age simulations and its effects on parallel performance are unclear.

Load balancing (LB) schemes have been used to overcome these parallelization challenges discussed above. Other disciplines using particle tracking, such as the molecular dynamics (MD) and the smoothed particle hydrodynamics (SPH) (Boulmier, Raynaud, Abdennadher, & Chopard, 2019; Egorova, Dyachkov, Parshikov, & Zhakhovsky, 2019; Eibl & Rüde, 2019; Fattebert, Richards, & Glosli, 2012; Furuichi & Nishiura, 2017; Kunaseth et al., 2013) have presented LB schemes that greatly improved parallel simulation performance. However, LB has not been applied to hydrologic modeling based on particle tracking; even in the studies using MPI through DDC mentioned above. Additionally, when applying the particle tracking at larger scales in long-term simulations, spatiotemporal variations of water age drivers in the real-world applications can increase the heterogeneity of flow paths. This heterogeneity causes uneven particle distributions and velocities within the domain, presenting new challenges to efficiency. This will further complicate the distribution of particles across different subdomains and thus the processing elements (which we are using here as a generic term for compute resources such as CPU cores or a GPU). Furthermore, it becomes challenging to implement LB when considering the increasing complexity of code structure of particle tracking due to the growing capabilities such as simulating ET age and source-water composition at transient state in EcoSLIM.

In this study, we present new LB approaches implemented in the EcoSLIM code which is a particle tracking code simulating water age (ET, outflow, and groundwater) and source water mixing (initial subsurface water, rainfall, and snow). EcoSLIM is a grid-based approach which is different from the mesh-free particle tracking in other disciplines. EcoSLIM works seamlessly with ParFlow.CLM (Kollet & Maxwell, 2008a), which is an integrated hydrologic model simulating the coupled land-surface and subsurface water- and energy-processes at transient state. ParFlow.CLM provides the temporally variant hydrodynamics and spatially variable subsurface-properties for EcoSLIM as input files, such as saturation, precipitation minus ET, three-dimensional velocity fields, and IDs and porosities of the subsurface units.

Objectives of this study are twofold. Firstly, we use MPI to manage multi-GPU instead of OpenMP in our previous work (C. Yang et al., 2021). This changes the target of the decomposition from computational load to modeling domain. MPI parallelism removes the barrier of a limited number of GPUs on a single computational node by created when using OpenMP and potentially extends the particle tracking applications to massively parallel computing. Secondly, three schemes with increasing physical representation are proposed to dynamically balance the load among different MPI-processes/GPUs during runtime, which is crucial for parallel efficiency. In following sections, the EcoSLIM code, the implementation of multi-GPU with MPI, and the load balancing schemes are introduced in section 2. The setup of test cases and platforms are followed in section 3. In section 4, validation of the new code is verified and parallel performances of the code with/without the LB schemes are illustrated. Specifically, application of the code for a 40-year simulation at regional scale in the North China Plain was shown. Finally, contributions and implications of this work to hydrologic modeling using Lagrangian approach are concluded in section 5.

2. Methodology

2.1. EcoSLIM code

EcoSLIM is originally implemented in Fortran and further accelerated by GPUs using CUDA (Compute Unified Device Architecture) Fortran (C. Yang et al., 2021). Its original structure is briefly introduced here to understand the following implementations of multi-GPU with MPI and LB schemes. For more details, please refer to our previous work (Maxwell et al., 2019; C. Yang et al., 2021). In each timestep of a transient simulation, key steps are as follows:

(1) New particles are added into grid-cells where precipitation minus ET (PME) is positive.

(2) Mass balance of precipitation and ET is calculated based on PME.

(3) Advancing each active particle by a do-loop. Each particle either moves forward with an increase of age or exits the modeling domain through outflow or ET.

(4) After the particle loop, inactive particles that have left the modeling domain via outflow or ET are sorted out of the array of particles, the number of active particles is updated and space at the end of this array is made available for new particles.

(5) The time loop moves to next timestep.

In the particle loop, other attributes of each particle are also updated, such as the saturated/unsaturated travel time, the saturated/unsaturated travel length, and the travel time/length in some specified subsurface units. Statistics of outflow and ET of the whole modeling domain, such as mass, mass weighted age, and source water composition, are included. Additionally, gridding information is recorded, such as mass, cell-averaged age, and source water composition for each grid-cell. Particle loop is the main computational load in EcoSLIM which occupies more than 99% of the total simulation time when serially executing the code (C. Yang et al., 2021). Therefore, our previous work (C. Yang et al., 2021) and also this study focused on parallelizing the particle loop.

2.2. Multi-GPU with MPI

Instead of the load decomposition in our previous work (C. Yang et al., 2021), MPI parallelism is applied here based on DDC. The modeling domain is split into P and Q parts in x and y directions respectively, which generates a computing topology using $P \times Q$ MPI processes. The quantity of GPUs utilized in a simulation equals that of MPI processes. We use the method in Ruetsch and Fatica (2014) to assign a unique GPU to each MPI rank. GPUs, MPI ranks, and subdomains are all numbered from 0 to $P \times Q - 1$. Each GPU is responsible for a subdomain of the same number. Source of particles on each GPU is from the assigned subdomain while transport of these particles is in the whole modeling domain. For the particle loop, we adopt the same GPU kernel in our previous work (C. Yang et al., 2021) with a few modifications for tracking particles in specified subsurface units. After the mapping to subdomains, each MPI-process/GPU works almost independently with a limited MPI collective communications. They are the calculation of mass balance mentioned in item (2) in section 2.1 and the ET/outflow statistics of the whole modeling domain also mentioned in section 2.1, which are performed before and after the execution of the kernel respectively.

The transport of particles in the whole domain, instead of in the subdomain where they were added, avoids the MPI communications of exchanging particles between subdomains when particles move out of a subdomain. The disadvantage of such a design is the redundant copies of the global information (e.g., velocities, porosities, saturations, and PME) for all MPI processes. This is CPU-memory expensive. However, for present clusters which are commonly equipped with 2/4/8 GPUs per node, the 2/4/8 copies of necessary information on one node are not a bottleneck for regional modeling with scales of Tran, Zhang, Cohard, Condon, and Maxwell (2020) and C. Yang et al. (2020). The multi-GPU with MPI was also conducted in our previous work based on load decomposition (C. Yang et al., 2021). However, in each timestep, the overhead of distributing load from rank *0* to others by MPI communication is over the speedup by extending single GPU to multi-GPU. Though the DDC in this study avoids such a problem, the load imbalance mentioned in section 1 becomes a new issue. Therefore, three schemes with increasing physical representation are proposed to balance the load among GPUs/MPI-processes during runtime in next section.

2.3. Schemes of load balancing

2.3.1 Direct transfer (S1)

The movement of particles in current EcoSLIM are independent, so the loads on GPUs can be redistributed by directly transferring particles between MPI processes with a user specified frequency. It is implemented by communications on CPU using the MPI functions of *MPI_Send* and *MPI_Recv* after the sort of particles. At a given time, the number of active particles on each MPI process is gathered. Thus, the old numbers of the starting and ending particles (*np_lo* and *np_ro*) on each process ranked in a global queue are obtained. Then the starting and ending numbers are updated to new ones (*np_ln* and *np_rn*) based on an even division of the global queue. Then the transfer is accomplished on each MPI process by the following four steps: (1) sending particles to the upstream process if *np_lo* is smaller than *np_ln*, (2) receiving particles from the downstream process if *np_ro* is smaller than *np_rn*, (3) sending particles to the downstream process if *np_ro* is larger than *np_rn*, and (4) receiving particles from the upstream process if *np_lo* is larger than *np_ln*. The number of active particles is updated after each transfer. The overhead of this scheme is determined by the quantity of particles transferred and the bandwidth of the computing platform.

2.3.2 Cyclic mapping (S2)

In this scheme, the DDC is static which is determined after initialization of the simulation. In a simulation using *n* GPUs/MPI-processes, the mapping between subdomains and GPUs (both numbered from *0* to *n-1*) is continuously shifted with a user specified frequency. For instance, at a given time t_1, the mapping is shifted from 'the *m*th subdomain → the *m*th GPU' to 'the (*m+1*)th subdomain → the *m*th GPU' where *m* ranges from *0* to *n-1*. If *m+1* is larger than *n-1*, the subdomain numbered with the reminder of *m+1* and *n* will be mapped to the *m*th GPU. Table 1 showed the cyclic mapping between subdomains and GPUs in a simulation using four MPI processes. Using this scheme, each GPU traverses the loads of all subdomains periodically, and thus the load distribution is dynamically balanced among GPUs. The overhead of shifting the mapping is almost negligible.

Table 1. Cyclic mapping in a simulation using four GPUs/MPI-processes

t_0 (Initial)	Subdomain number	0	1	2	3
	GPU number	0	1	2	3
t_1	Subdomain number	1	2	3	0
	GPU number	0	1	2	3
t_2	Subdomain number	2	3	0	1
	GPU number	0	1	2	3

2.3.3 Dynamic DDC (S3)

In the initialization of a simulation, particles are evenly distributed in space, so we decompose the modeling domain into subdomains of an equal size. PME is spatiotemporally variable, so the number of particles added into each subdomain is different at a given timestep and such a difference varies with time. More importantly, particles added into different subdomains have different exits from the whole modeling domain. Both source and exit are responsible for heterogeneity of the flow paths. A subdomain of more source particles and longer flow paths imposes heavier load on its corresponding GPU. As a result, after the initial even-decomposition, we dynamically update the decomposition during runtime based on flow paths of particles. At a given timestep, the initial location of an active particle is identified and the corresponding grid-cell in the top layer get one score. After traversing all the active particles, we get the accumulated scores for each grid-cell in the top layer. This was implemented through atomic operations on a two-dimensional matrix in the GPU kernel mentioned in section 2.2. Then we conduct DDC based on this weight matrix. The frequency of such a dynamic DDC can be specified by users.

The orthogonal recursive bisection (ORB) method is used for DDC, which is popular in MD and SPH (Egorova et al., 2019; Fattebert et al., 2012). The domain or each subdomain is divided into two in a direction at one time. By switching the direction, the whole domain is recursively divided into the scheduled number of subdomains ($P \times Q$). DDC is only implemented in x and y directions in this study. It starts in a direction which will be divided into more pieces. For example, if Q is larger than P, DDC will start in y direction, otherwise, it starts in x direction. Two algorithms are provided in the code to determine the dividing line. One calculates the accumulated particles of columns (rows) in x (y) direction. Once the number of particles of n-1 columns (rows) is less than half the total particles in this subdomain while that of n columns (rows) is more than half the total particles in this subdomain, the dividing line is found as column (row) n. The other algorithm to find the dividing line with higher efficiency is the typical dichotomizing search.

3. Test setup

3.1. Hillslope model

Tests were conducted based on a hillslope model (Maxwell et al., 2019; C. Yang et al., 2021). The modeling domain has the length of 100-, 1-, and 9.4-m in x, y, and z directions, respectively. It was divided into 20 columns, 5 rows, and 20 layers with constant resolutions in x and y directions. In vertical direction, the layer-thickness was variable: 0.5 m for the bottom 18 layers while 0.3- and 0.1-m for the top 2 layers. Soil has the homogeneous properties: saturated hydraulic conductivity of 0.05 m/h, Manning's N of 10^{-6} $m^{1/3}h^{-1}$, porosity of 0.2, and van Genuchten parameters with α of 1.0 m^{-1} and exponent n of 2.0. Two real meteorological-forcings were used to drive ParFlow.CLM, representing a high elevation, snow dominated mountain headwaters (ER) and a semiarid, rain-dominated plains system (LW). Two homogeneous land-cover types were used which are the Shrub plant functional type (Shrub) and the Evergreen Needleleaf plant functional type (Trees). Thus, four cases were tested with a combination of the meteorological forcings and the land-cover types, which were named as: ER_Shrub, ER_Trees, LW_Shrub, and LW_Trees. For simulations of both ParFlow.CLM and EcoSLIM, no flux boundaries were adopted except the land surface which was open for precipitation, outflow and ET. For each case, ParFlow.CLM simulation of 5 years was conducted using hourly timestep. One-year forcing data were repeatedly used in the whole simulation. Dynamic equilibrium of the flow field was approached at the end of simulation. Hence, the transient flow field of the last year in ParFlow.CLM simulation was repeatedly used in EcoSLIM simulation of 20 years with hourly timestep. At the end of each simulation, EcoSLIM system achieved the dynamic equilibrium.

By injecting 2 particles into the modeling domain per precipitation event, the average particle-numbers in the last year of the simulations for four cases are 0.39-, 0.83-, 1.30-, and 1.03-million, respectively. Such quantities of particles are comparable to those in most of the previous studies (Danesh-Yazdi, Klaus, Condon, & Maxwell, 2018; Nicholas B. Engdahl & Maxwell, 2015; Jing et al., 2019; Jing et al., 2020; Kollet & Maxwell, 2008b; Maxwell et al., 2016; Weill, Lesparre, Jeannot, & Delay, 2019; Wilusz et al., 2019). It has a maximum of 6.8 million in Weill et al. (2019) to the best of our knowledge. In fact, the particle-number in most previous studies is the total injected particles while that during runtime is a fewer quantity. However, the number in this study is the active particles in the modeling domain and the particles out of the domain through ET and outflow are not included. Thus, the particle-quantities are much more than those in previous studies.

3.2. Test platform

Tests were conducted on the Casper cluster in the Computational and Information Systems Laboratory at the National Center for Atmospheric Research. The computational node used for the following simulations is equipped with 2.3-GHz Intel® Xeon® Gold 6140 processors and NVIDIA Tesla V100 32GB SXM2 GPUs with NVLink. The compiler is NVIDIA HPC SDK of version 20.11, the MPI is implemented using Open MPI of version 4.0.5, the GPU driver version is 450.51.06, and the CUDA version is 11.0.3. Tests were also repeated on a personal workstation (WS). The WS is equipped with 2.00-GHz Intel® Xeon® E5-2683 v3 processors together with four GPUs of 12 GB GeForce GTX 1080 Ti. Other necessary setups of the WS environment are NVIDIA HPC SDK 20.11, Open MPI 3.1.5, GPU driver 440.118.02, and CUDA 10.2.

4. Results and discussion

4.1. Code-to-code verification

To verify the availability of the new code, simulation results using the new code were compared to those of the original OpenMP version (Maxwell et al., 2019). Comparisons were performed for all four cases while that of the ER_Shrub case was shown in Figures 1 (outflow) and 2 (ET). Results of other test cases had performances as good as that of ER_Shrub. Tests in this study were conducted using one, two, and four GPUs successively while the results using four GPUs were illustrated in Figures 1 and 2. Subplots in Figures 1 and 2 were for results without LB and with each LB scheme. The water-age and -mass for both outflow and ET simulated by the new code well fitted those generated by the original code. The deviations between them were attributed to the generation of pseudo-random numbers (PRNs). Though the ensembles of the PRNs were statistically the same for each run, the PRN for a specific particle probably changed due to the invoking sequence of the generation-function which was dependent on the parallelism, i.e., the OpenMP or the multi-GPU with MPI. For the same parallelism, if different numbers of CPU-threads/GPUs were used, there were also such deviations during our tests. The fitness of outflow was better than that of ET because ET in EcoSLIM were directly dependent on PRNs.

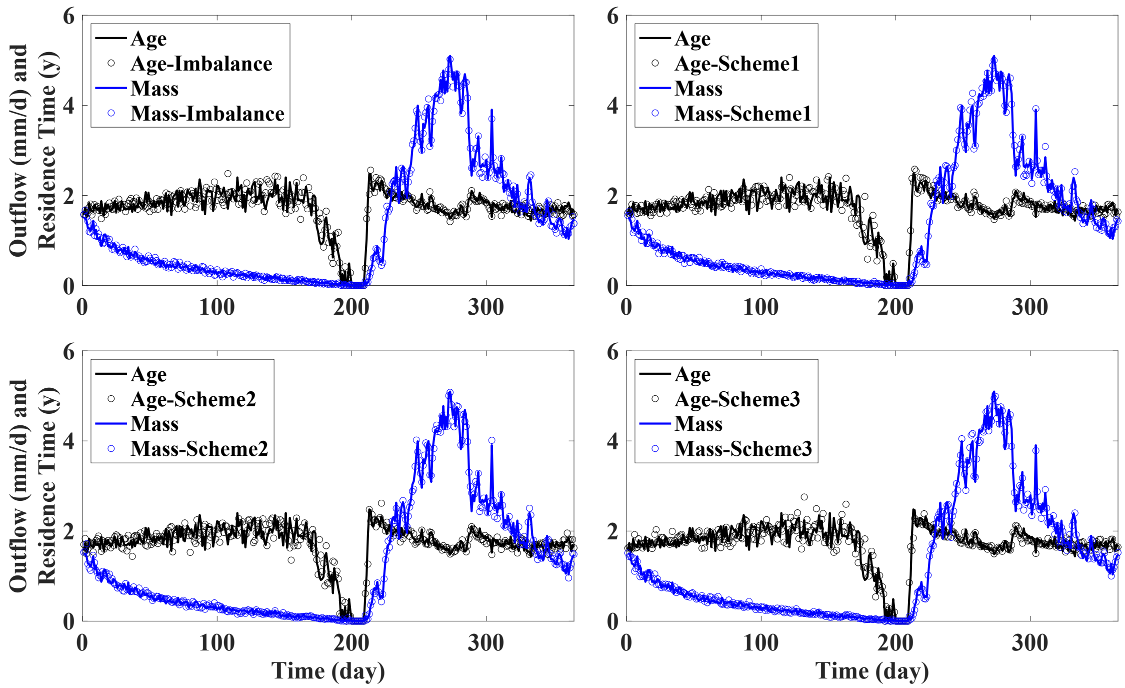

Figure 1. Comparisons for age and mass of outflow based on ER_Shrub case between the original EcoSLIM code and that parallelized in this study.

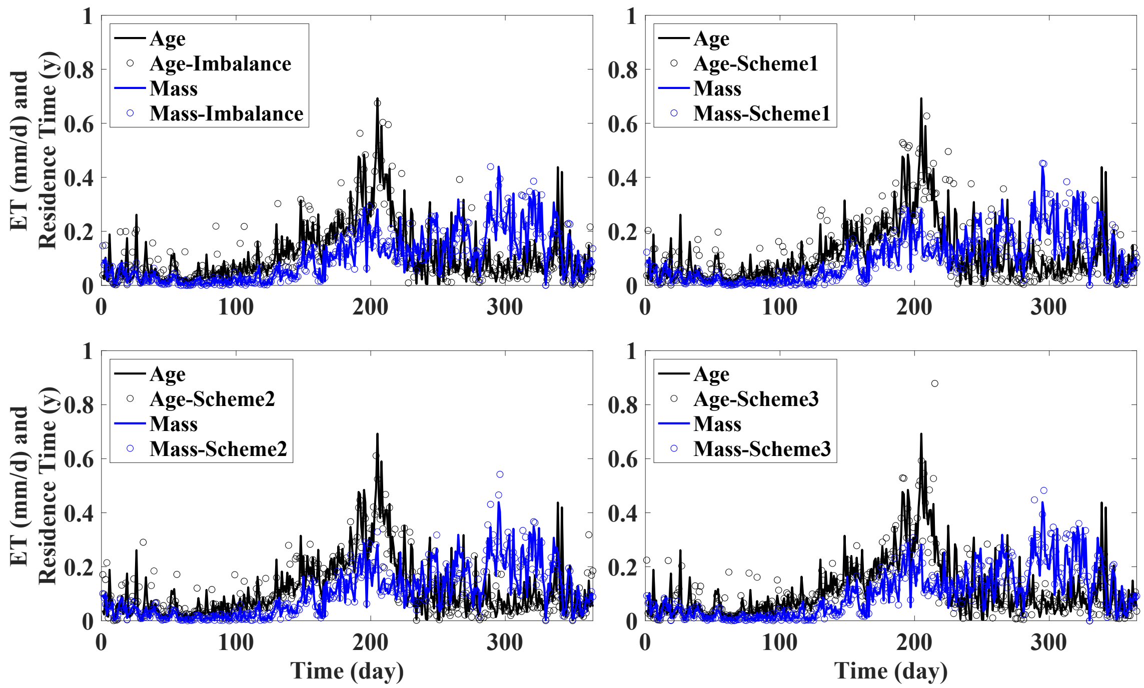

Figure 2. Comparisons for age and mass of ET based on ER_shrub case between the original EcoSLIM code and that parallelized in this study.

4.2. Parallel performance

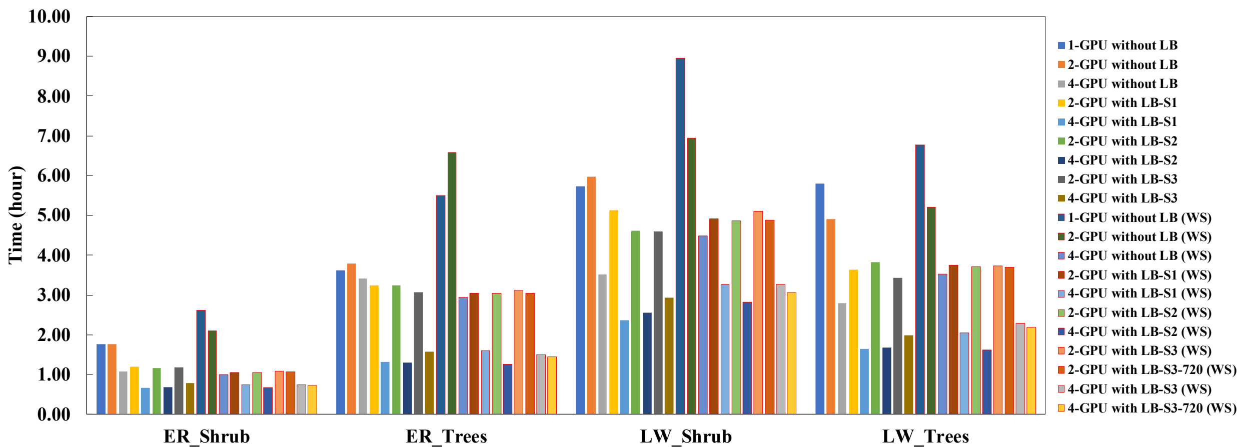

Figure 3. Wall-clock time consumption of each test. *n*-GPU represents the number of GPUs used in simulations. S*n* represents different LB schemes. WS indicates tests conducted on workstation while others on Casper. 720 indicates S3 worked every 720-hour while others worked every 8760-hour.

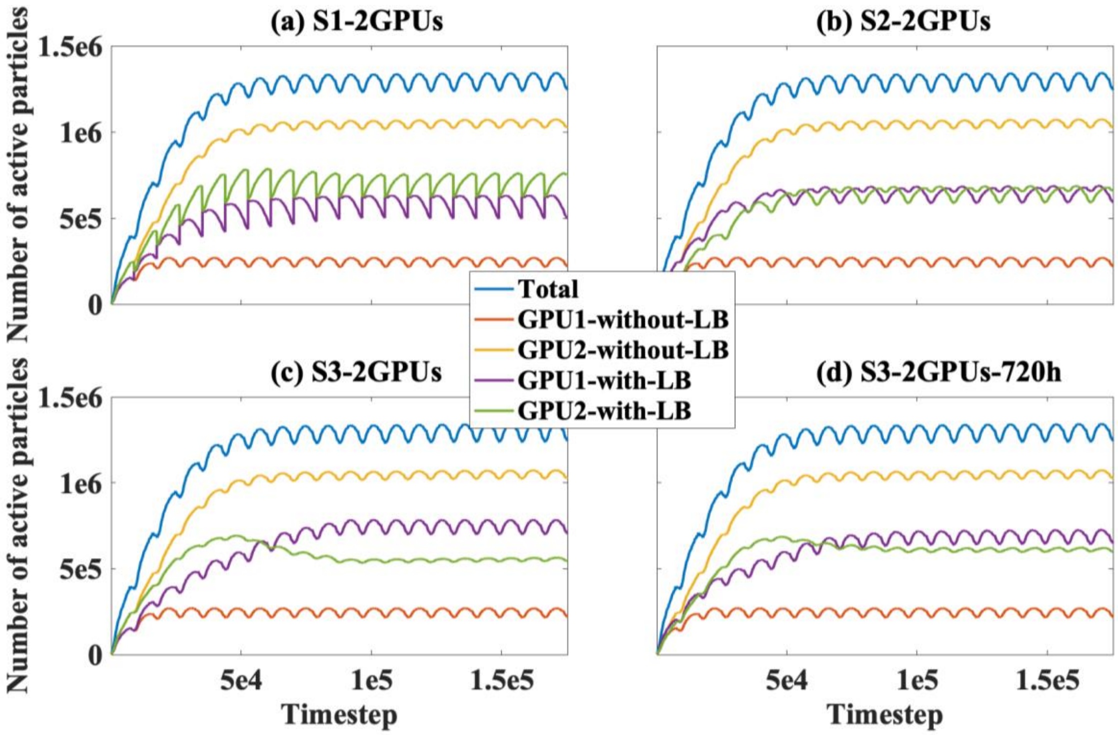

Figure 4. Load distribution for LW_Shrub by using 2-GPU. S*n* represents LB schemes. 720h represents that S3 worked every 720-hour while others were 8760-hour.

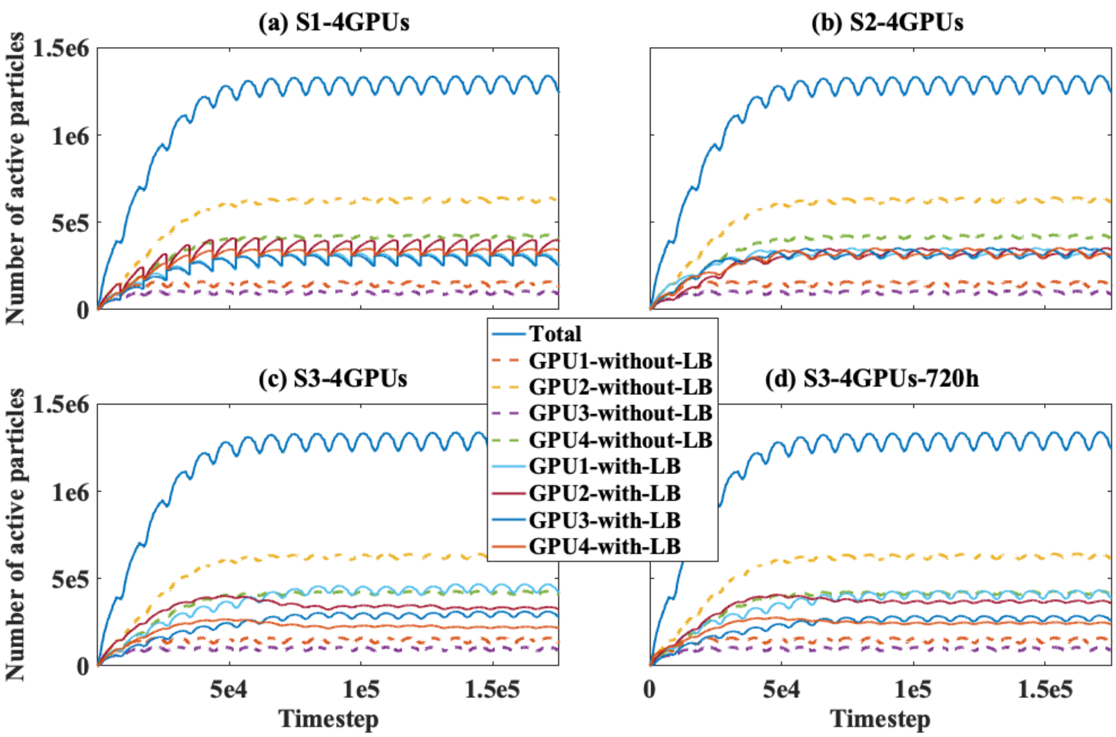

Figure 5. Load distribution for LW_Shrub by using 4-GPU. S*n* represents LB schemes. 720h represents that S3 worked every 720-hour while others were 8760-hour.

Figure 3 shows wall-clock time consumption of each test on both Casper and WS. Load distributions for tests using 2- and 4-GPU were shown in Figures 4 and 5, respectively. Tests using 2-GPU (P = 2 and Q = 1) were performed by evenly dividing the domain in x direction while those using 4-GPU (P = 2 and Q = 2) had an additional division in y direction. All LB schemes worked every 8760-hour except that in Figures 4d and 5d were 720-hour for S3. Though only LW_Shrub was taken as an example in Figures 4 and 5, other cases had similar performances. Time used for four cases by one CPU-thread based on the original code were tested on WS, which were 36.41-, 76.34-, 121.56-, and 92.40-hour for ER_Shrub, ER_Trees, LW_Shrub, and LW_Trees, respectively. Speedup of each case was then calculated and listed in Table 2.

Table 2. Speedup of each test relative to the serial run using one CPU-thread

Platform	Case name	Speedup								
		Without LB			Scheme 1		Scheme 2		Scheme 3	
		1-GPU	2-GPU	4-GPU	2-GPU	4-GPU	2-GPU	4-GPU	2-GPU	4-GPU
Casper	ER_Shrub	20.5403	20.5693	33.5134	30.5237	54.8938	31.0469	52.7465	30.6935	46.3058
	ER_Trees	21.1316	20.1687	22.3125	23.5297	57.5619	23.5857	58.7581	24.8849	48.6518
	LW_Shrub	21.1858	20.3505	34.5421	23.7138	51.5341	26.3410	47.6590	26.4183	41.5489
	LW_Trees	15.9127	18.8541	32.9764	25.3874	56.0544	24.1910	54.8824	26.9184	46.2856
WS	ER_Shrub	13.9637	17.2900	36.0809	34.4456	48.8535	34.4522	54.4261	33.4696	49.0443
	ER_Trees	13.8955	11.5971	26.0117	25.0581	47.8438	25.1381	60.4948	24.5480	51.0523
	LW_Shrub	13.5910	17.5129	27.0710	24.7589	37.1392	24.9531	43.0452	23.8150	37.1698
	LW_Trees	13.6440	17.7556	26.1808	24.6492	45.1304	24.8835	57.2562	24.7449	40.4952

In tests without LB, time used by 2-GPU was even more than that by 1-GPU for ER_Trees and LW_Shrub on Casper (Figure 3). This performance degradation has two reasons. Firstly, severe load imbalance can be observed in Figure 4 when using 2-GPU with a particle-number ratio of 423.62%. The larger one (yellow lines in Figure 4) which determined the parallel efficiency almost achieved the total load. It confirmed that load imbalance also exists in Lagrangian hydrologic modeling and decreases the parallel performance. Secondly, collective MPI communications mentioned in section 2.2 introduced overhead when increasing the GPU number from one to two. When using 4-GPU without LB, the maximum load largely decreased (Figure 5) due to the additional decomposition in y direction. The hillslope model is quasi-three-dimensional with a x slope of 0.1 and a y slope of 0. Hence, the movement of particles in y direction can be neglected which formed the parallel flow paths along x direction. Along x direction, particles added upstream had much longer flow paths than those added downstream. As a result, the division in y direction was much more effective than that in x direction to improve the parallel performance. However, load imbalance was still significant (Figure 5) and parallel scalability was not shown by increasing 2-GPU to 4-GPU (Figure 3).

When S1 was activated, time used by 2- and 4-GPU were dramatically decreased relative to those without LB. With S1, time used by 4-GPU was even less than half of that used by 2-GPU on Casper (Figure 3). The overhead of particle transfer was small for all four cases, which was less than 3 seconds in each 20-year simulation. S2 had performance as good as S1 (Figure 3). It was mentioned in section 2.3.2 that the overhead of S2 was small enough to be neglected. For S3, more time was used by 4-GPU when compared to that of S1 and S2 (Figure 3). In Figures 4c and 5c, the load distribution was not well balanced relative to that of S1 and S2. The flow paths of particles were transient, so the weight matrix at a moment cannot effectively balance the load for a long-period of simulation (i.e., 8760-hour). When S3 was activated with a higher frequency of every 720-hour, the improved load balance can be observed in Figures 4d and 5d and the further speedup was indicated in Figure 3. However, difference of the loads between GPUs 1-2 and 3-4 was still observed in Figure 5d. This is due to the model dimension which is five grid-cells in *y* direction. Hence it cannot be evenly divided by ORB introduced in section 2.3.3.

S3, a physically-based scheme, not only aims to balance the load but also to understand the load imbalance in Lagrangian hydrological modeling. To build S3, we also tried DDC based on the source of particles (i.e., PME). The score/weight of a grid-cell is determined by the accumulated particle-number added into it in a period. However, it didn't show good performance. Current implementation actually integrates the effects of both the quantity of source particles and the flow-path lengths. This trial and error indicates that the flow-path lengths instead of the quantity of added particles dominate the load distribution. This has important implications to efficiently build other physically-based LB schemes. Generally, with LB, the new code showed excellent parallel performance in the tests on both Casper and WS (Table 2). The speedup by one GPU is ~13-fold on WS with 1080 Ti while ~21-fold on Casper with Tesla V100. The speedup by 4-GPU is over 50-fold on both Casper and WS and has a maximum over 60-fold. With LB schemes, the code showed parallel scalability from 2-GPU to 4-GPU.

4.3. Application in the North China Plain

We applied the new parallel code on a North China Plain (NCP) domain to demonstrate its capacity for large-scale simulations. To the best of our knowledge, there have been no previous studies based on particle tracking at such a regional scale for water ages of ET, groundwater (GW), and outflow in a unified framework. The NCP ParFlow.CLM model was adopted from C. Yang et al. (2020) with a few modifications. The model has 509 and 921 grid-cells in x and y directions respectively while it is discretized into five layers in vertical direction. The horizontal resolution is 1 km while the layer thickness from bottom to top is 100-, 1-, 0.6-, 0.3-, and 0.1-m. Thus, the NCP model has a dimension of 509 km × 921 km × 102 m in total. We conducted an EcoSLIM simulation of 40 years on Casper with hourly timestep, in which the hourly outputs of one-year simulation from ParFlow.CLM were repeatedly used.

The EcoSLIM simulation was started using 8-GPU (P = 2 and Q = 4) without LB (abbreviated as R1 hereafter). From the 155,928th hour, S1 was activated every 240-hour (R2) while R1 was continued for the following 7.8 years. At the 247,032th hour, the load of R2 was evenly divided into 16 portions and a new run (R3) was started using 16-GPU (P = 4 and Q = 4) with S2 activated every 240-hour. The overlap between R2 and R3 is 3.4 years. We also tried 16-GPU without LB for the first five years of the simulation (R4). Figure 6 showed the active-particle-number and the wall-clock time consumption of each timestep during the latter 22 years of the simulation (the 155,929th to the 350,400th hour). R1, R2, and R3 were indicated by green, blue, and red in Figure 6 respectively. The active-particle-number is around 200 million during this simulation time-interval (Figure 6a). The discrepancy of the particle number between different runs in the overlaps are due to the generation of random numbers discussed in section 4.1.

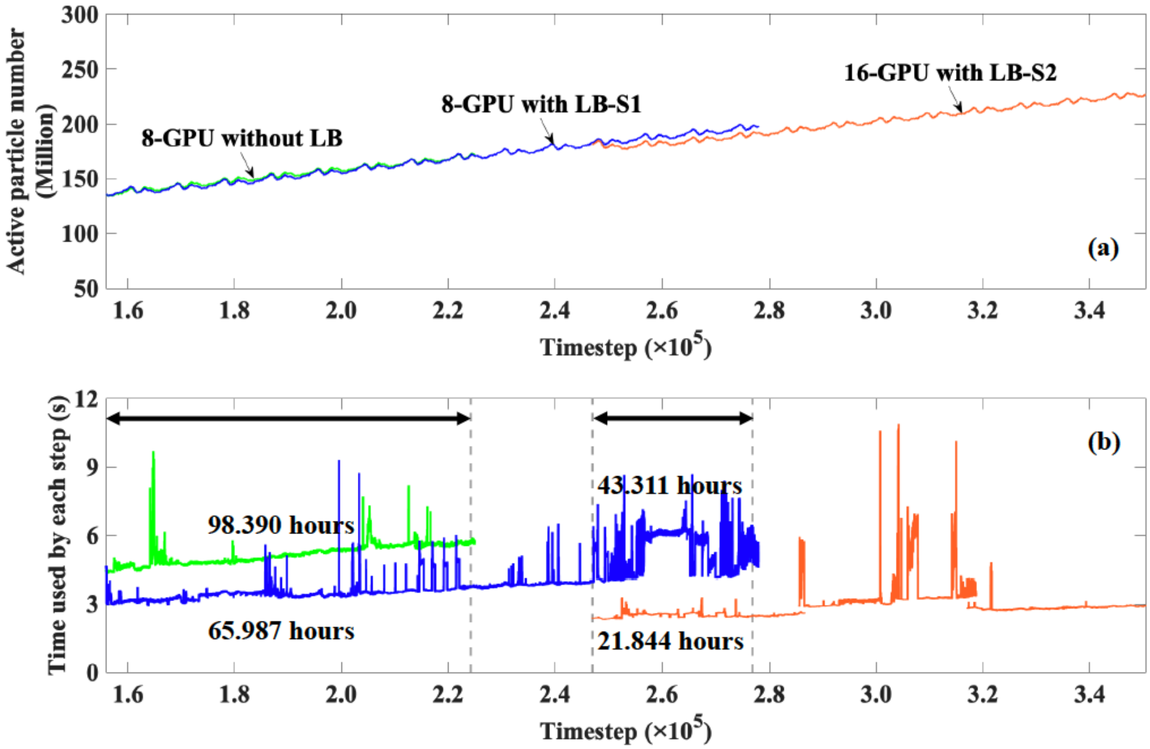

Figure 6. Computational load (a) and wall-clock time consumption (b) for the EcoSLIM simulation in the North China Plain. The time interval is from the 155929th to the 350400th hour in the 40-year simulation.

For the overlap between R1 and R2, parallel time of the particle loop was 98.390- and 65.987-hour for R1 and R2 respectively (Figure 6b). The overhead of S1 for transferring data was 1.573-hour. S1 (overhead included) decreased 31.33% of the time used by R1, which demonstrated the high efficiency of S1. Figures 7a and 7b showed the well-balanced load by S1. For the overlap between R2 and R3, the parallel time was 43.311- and 21.844-hour for R2 and R3 respectively (Figure 6b). Though it showed 50% decrease of the parallel time, the obvious jitters of the time in R2 has to be considered. Based on their baselines, the time used by R3 was a little longer than half the time used by R2. This should be due to the better load balancing effect of S1 than that of S2, which was shown in Figures 7b and 7d. However, when comparing the load distribution between R3 and R4 for a time interval of the same length (4.94-year), the load balancing effect of S2 was significant. The difference between the maximum- and minimum-load at the end of the comparing time-interval in R4 was 6.66 million (Figure 7d) while that in R3 was 3.52 million (Figure 7c), which was 47.21% decrease of the load variance. Additionally, based on the increasing trend in Figures 7c and 7d, the load variance in R3 with S2 gradually achieved a steady state while that in R4 continued increasing.

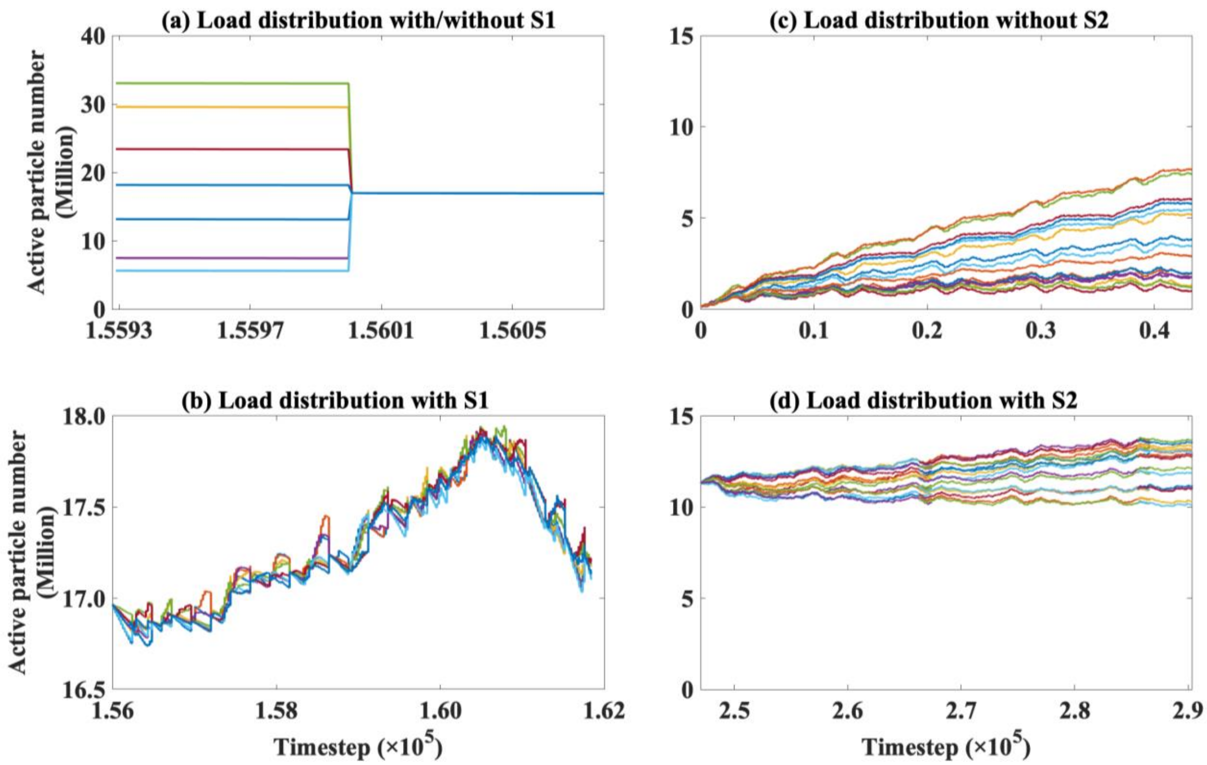

Figure 7. Load distributions in the application in the North China Plain. Load distribution on 8-GPU before and after using S1 in R2 (a), on 8-GPU with S1 in R2 (b), on 16-GPU without LB in R4 (c), and on 16-GPU with S2 in R3. (b) was magnified from (a).

5. Conclusions

Water age can reveal the source, storage, and mixing of water parcels in a watershed. Though data- and model-driven methods have significantly advanced our understanding of water ages, the quantification of water ages is still technically challenging. Lagrangian particle tracking is an invaluable tool for physically-based transient modeling of water ages, but it is computationally expensive. When considering climate change and global water security, it is essential to conduct simulations of water ages at large scale with high resolution, which makes the implementation of massively parallel computing in particle tracking for this purpose pressing. Though parallel computing is widely implemented for Eulerian hydrological modeling, applications to Lagrangian based simulations are developing. This is likely due to the inherent difficulties such as load imbalance across computational resources which will become more challenging when modeling a real hydrologic system with high spatiotemporal variability.

In this study, multi-GPU with MPI parallelism based on domain decomposition (DDC) was implemented in the Lagrangian, particle tracking code EcoSLIM, to accelerate simulations of water age and source-water mixing. Three load balancing (LB) schemes with increasing physical representation (i.e., direct transfer, cyclic mapping, and dynamic DDC) were built to dynamically balance the quantity of particles across GPUs during runtime. With LB, the code showed excellent parallel performance in the hillslope simulations on two different platforms, e.g., a maximum of 60-fold speedup on 4-GPUs and the parallel scalability from 2-GPU to 4-GPU that is almost ideal. A 40-year simulation conducted in the North China Plain further demonstrated the high parallel efficiency of LB for a large-scale application. Using 8-GPU with LB, it reduced 31.33% of the parallel time using 8-GPU without LB. When increasing 8-GPU with LB to 16-GPU with LB, ~50% reduction of the parallel time demonstrated the parallel scalability.

More importantly, results confirmed the load imbalance in Lagrangian hydrologic modeling. In LW_Shrub case using 2-GPU, the particle-number ratio achieved 423.62%, which severely degraded the parallel performance without LB. For LB schemes, physically-based dynamic DDC performed as well as other schemes in hillslope simulations. Trial and error of building this scheme identified that the distribution of flow-path lengths in the domain instead of the quantity of particles added into the domain dominates the load distribution. This illustrated both the mechanisms of load imbalance and the directions to build efficient physically-based LB schemes in this context. This study realized the massively parallel computing of particle tracking in water age simulations which is lacking in hydrologic modeling. It also demonstrated that LB have practical importance enabling its applications at large scales with increased heterogeneity of flow paths. The LB schemes can be borrowed to other hydrologic models using Lagrangian approach and the parallelized EcoSLIM is a promising tool to accelerate the scientific progress of water age studies.

Acknowledgements

This work was supported by the U.S. Department of Energy Office of Science, Offices of Advanced Scientific Computing Research and Biological and Environmental Sciences IDEAS project and Watershed Function Scientific Focus Area under Award Number DE-AC02-05CH11231. The authors acknowledge high-performance computing support from Cheyenne (doi:10.5065/D6RX99HX) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation. Fortran/CUDA-Fortran code

(https://github.com/aureliayang/EcoSLIM/tree/multi-GPU) which can reproduce the results is located on Github. Once the manuscript is accepted, this repository will be archived in Zenodo to get a DOI for citation purpose.

References

Basu, N. B., Jindal, P., Schilling, K. E., Wolter, C. F., & Takle, E. S. (2012). Evaluation of analytical and numerical approaches for the estimation of groundwater travel time distribution. *Journal of Hydrology, 475*, 65-73. doi:https://doi.org/10.1016/j.jhydrol.2012.08.052

Botter, G., Bertuzzo, E., & Rinaldo, A. (2011). Catchment residence and travel time distributions: The master equation. *Geophysical Research Letters, 38*(11). doi:10.1029/2011gl047666

Boulmier, A., Raynaud, F., Abdennadher, N., & Chopard, B. (2019, 23-26 Sept. 2019). *On the Benefits of Anticipating Load Imbalance for Performance Optimization of Parallel Applications.* Paper presented at the 2019 IEEE International Conference on Cluster Computing (CLUSTER).

Danesh-Yazdi, M., Klaus, J., Condon, L. E., & Maxwell, R. M. (2018). Bridging the gap between numerical solutions of travel time distributions and analytical storage selection functions. *Hydrological Processes, 32*(8), 1063-1076. Retrieved from <Go to ISI>://WOS:000430466700006

Egorova, M. S., Dyachkov, S. A., Parshikov, A. N., & Zhakhovsky, V. V. (2019). Parallel SPH modeling using dynamic domain decomposition and load balancing displacement of Voronoi subdomains. *Computer Physics Communications, 234*, 112-125. doi:https://doi.org/10.1016/j.cpc.2018.07.019

Eibl, S., & Rüde, U. (2019). A systematic comparison of runtime load balancing algorithms for massively parallel rigid particle dynamics. *Computer Physics Communications, 244*, 76-85. doi:https://doi.org/10.1016/j.cpc.2019.06.020

Engdahl, N. B., & Maxwell, R. M. (2014). Approximating groundwater age distributions using simple streamtube models and multiple tracers. *Advances in Water Resources, 66*, 19-31. doi:https://doi.org/10.1016/j.advwatres.2014.02.001

Engdahl, N. B., & Maxwell, R. M. (2015). Quantifying changes in age distributions and the hydrologic balance of a high-mountain watershed from climate induced variations in recharge. *Journal of Hydrology, 522*, 152-162. doi:https://doi.org/10.1016/j.jhydrol.2014.12.032

Engdahl, N. B., McCallum, J. L., & Massoudieh, A. (2016). Transient age distributions in subsurface hydrologic systems. *Journal of Hydrology, 543*, 88-100. doi:https://doi.org/10.1016/j.jhydrol.2016.04.066

Engdahl, N. B., Schmidt, M. J., & Benson, D. A. (2019). Accelerating and Parallelizing Lagrangian Simulations of Mixing-Limited Reactive Transport. *Water Resources Research, 55*(4), 3556-3566. doi:10.1029/2018wr024361

Fattebert, J. L., Richards, D. F., & Glosli, J. N. (2012). Dynamic load balancing algorithm for molecular dynamics based on Voronoi cells domain decompositions. *Computer Physics Communications, 183*(12), 2608-2615. doi:https://doi.org/10.1016/j.cpc.2012.07.013

Furuichi, M., & Nishiura, D. (2017). Iterative load-balancing method with multigrid level relaxation for particle simulation with short-range interactions. *Computer Physics Communications, 219*, 135-148. doi:https://doi.org/10.1016/j.cpc.2017.05.015

Gomez, J. D., & Wilson, J. L. (2013). Age distributions and dynamically changing hydrologic systems: Exploring topography-driven flow. *Water Resources Research, 49*(3), 1503-1522. doi:https://doi.org/10.1002/wrcr.20127

Ji, X. H., Luo, M. L., & Wang, X. S. (2019). Accelerating Streamline Tracking in Groundwater Flow Modeling on GPUs. *Groundwater*. doi:10.1111/gwat.12959

Jing, M., Heße, F., Kumar, R., Kolditz, O., Kalbacher, T., & Attinger, S. (2019). Influence of input and parameter uncertainty on the prediction of catchment-scale groundwater travel time distributions. *Hydrol. Earth Syst. Sci.*, *23*(1), 171-190. doi:10.5194/hess-23-171-2019

Jing, M., Kumar, R., Heße, F., Thober, S., Rakovec, O., Samaniego, L., & Attinger, S. (2020). Assessing the response of groundwater quantity and travel time distribution to 1.5, 2, and 3 °C global warming in a mesoscale central German basin. *Hydrol. Earth Syst. Sci., 24*(3), 1511-1526. doi:10.5194/hess-24-1511-2020

Kollet, S. J., & Maxwell, R. M. (2008a). Capturing the influence of groundwater dynamics on land surface processes using an integrated, distributed watershed model. *Water Resources Research, 44*(2). doi:Artn W0240210.1029/2007wr006004

Kollet, S. J., & Maxwell, R. M. (2008b). Demonstrating fractal scaling of baseflow residence time distributions using a fully-coupled groundwater and land surface model. *Geophysical Research Letters, 35*(7). Retrieved from <Go to ISI>://WOS:000254716500001

Kunaseth, M., Richards, D. F., Glosli, J. N., Kalia, R. K., Nakano, A., & Vashishta, P. (2013). Analysis of scalable data-privatization threading algorithms for hybrid MPI/OpenMP parallelization of molecular dynamics. *The Journal of Supercomputing, 66*(1), 406-430. doi:10.1007/s11227-013-0915-x

Maxwell, R. M., Condon, L. E., Danesh-Yazdi, M., & Bearup, L. A. (2019). Exploring source water mixing and transient residence time distributions of outflow and evapotranspiration with an integrated hydrologic model and Lagrangian particle tracking approach. *Ecohydrology, 12*(1). Retrieved from <Go to ISI>://WOS:000454601400016

Maxwell, R. M., Condon, L. E., Kollet, S. J., Maher, K., Haggerty, R., & Forrester, M. M. (2016). The imprint of climate and geology on the residence times of groundwater. *Geophysical Research Letters, 43*(2), 701-708. doi:10.1002/2015gl066916

McCallum, J. L., Engdahl, N. B., Ginn, T. R., & Cook, P. G. (2014). Nonparametric estimation of groundwater residence time distributions: What can environmental tracer data tell us about groundwater residence time? *Water Resources Research, 50*(3), 2022-2038. doi:10.1002/2013wr014974

McGuire, K. J., McDonnell, J. J., Weiler, M., Kendall, C., McGlynn, B. L., Welker, J. M., & Seibert, J. (2005). The role of topography on catchment-scale water residence time. *Water Resources Research, 41*(5). doi:10.1029/2004wr003657

Pollock, D. W. (2016). *User guide for MODPATH Version 7—A particle-tracking model for MODFLOW* (2016-1086). Retrieved from Reston, VA: http://pubs.er.usgs.gov/publication/ofr20161086

Ruetsch, G., & Fatica, M. (2014). *CUDA Fortran for scientists and engineers : best practices for efficient CUDA Fortran programming*. Amsterdam Boston: Morgan Kaufmann, an imprint of Elsevier.

Sprenger, M., Stumpp, C., Weiler, M., Aeschbach, W., Allen, S. T., Benettin, P., . . . Werner, C. (2019). The Demographics of Water: A Review of Water Ages in the Critical Zone. *Reviews of Geophysics, 57*(3), 800-834. doi:https://doi.org/10.1029/2018RG000633

Starn, J. J., Kauffman, L. J., Carlson, C. S., Reddy, J. E., & Fienen, M. N. (2021). Three-Dimensional Distribution of Groundwater Residence Time Metrics in the Glaciated United States Using Metamodels Trained on General Numerical Simulation Models. *Water Resources Research, 57*(2), e2020WR027335. doi:https://doi.org/10.1029/2020WR027335

Tran, H., Zhang, J., Cohard, J.-M., Condon, L. E., & Maxwell, R. M. (2020). Simulating Groundwater-Streamflow Connections in the Upper Colorado River Basin. *Groundwater, 58*(3), 392-405. doi:10.1111/gwat.13000

Weill, S., Lesparre, N., Jeannot, B., & Delay, F. (2019). Variability of Water Transit Time Distributions at the Strengbach Catchment (Vosges Mountains, France) Inferred Through Integrated Hydrological Modeling and Particle Tracking Algorithms. *Water, 11*(12), 2637. Retrieved from https://www.mdpi.com/2073-4441/11/12/2637

Wilusz, D. C., Harman, C. J., Ball, W. B., Maxwell, R. M., & Buda, A. R. (2019). Using particle tracking to understand flow paths, age distributions, and the paradoxical origins of the inverse storage effect in an experimental catchment. *Water Resources Research, n/a*(n/a), e24397. doi:10.1029/2019wr025140

Yang, C., Li, H.-Y., Fang, Y., Cui, C., Wang, T., Zheng, C., . . . Yang, X. (2020). Effects of Groundwater Pumping on Ground Surface Temperature: A Regional Modeling Study in the North China Plain. *Journal of Geophysical Research: Atmospheres, 125*(9), e2019JD031764. doi:10.1029/2019jd031764

Yang, C., Zhang, Y.-K., Liang, X., Olschanowsky, C., Yang, X., & Maxwell, R. (2021). Accelerating the Lagrangian particle tracking of residence time distributions and source water mixing towards large scales. *Computers & Geosciences*, 104760. doi:https://doi.org/10.1016/j.cageo.2021.104760

Yang, J., Heidbüchel, I., Musolff, A., Reinstorf, F., & Fleckenstein, J. H. (2018). Exploring the Dynamics of Transit Times and Subsurface Mixing in a Small Agricultural Catchment. *Water Resources Research, 54*(3), 2317-2335. doi:10.1002/2017wr021896