


Distributed replica dynamics

Cite as: J. Chem. Phys. **143**, 174112 (2015); <https://doi.org/10.1063/1.4934987>

Submitted: 03 August 2015 . Accepted: 19 October 2015 . Published Online: 06 November 2015

Liang Zhang, Samuel T. Chill, and Graeme Henkelman 



View Online



Export Citation



CrossMark

ARTICLES YOU MAY BE INTERESTED IN

Atomic layer deposition of environmentally benign SnTiO_x as a potential ferroelectric material

Journal of Vacuum Science & Technology A **34**, 01A119 (2016); <https://doi.org/10.1116/1.4935650>

Chemistry, growth kinetics, and epitaxial stabilization of Sn^{2+} in Sn-doped SrTiO_3 using $(\text{CH}_3)_6\text{Sn}_2$ tin precursor

APL Materials **4**, 126111 (2016); <https://doi.org/10.1063/1.4972995>

Polymer piezoelectric energy harvesters for low wind speed

Applied Physics Letters **104**, 012902 (2014); <https://doi.org/10.1063/1.4861187>

Lock-in Amplifiers
up to 600 MHz



Watch



Distributed replica dynamics

Liang Zhang, Samuel T. Chill, and Graeme Henkelman^{a)}

*Department of Chemistry and the Institute for Computational Engineering and Sciences,
The University of Texas at Austin, Austin, Texas 78712-0165, USA*

(Received 3 August 2015; accepted 19 October 2015; published online 6 November 2015)

A distributed replica dynamics (DRD) method is proposed to calculate rare-event molecular dynamics using distributed computational resources. Similar to Voter's parallel replica dynamics (PRD) method, the dynamics of independent replicas of the system are calculated on different computational clients. In DRD, each replica runs molecular dynamics from an initial state for a fixed simulation time and then reports information about the trajectory back to the server. A simulation clock on the server accumulates the simulation time of each replica until one reports a transition to a new state. Subsequent calculations are initiated from within this new state and the process is repeated to follow the state-to-state evolution of the system. DRD is designed to work with asynchronous and distributed computing resources in which the clients may not be able to communicate with each other. Additionally, clients can be added or removed from the simulation at any point in the calculation. Even with heterogeneous computing clients, we prove that the DRD method reproduces the correct probability distribution of escape times. We also show this correspondence numerically; molecular dynamics simulations of Al(100) adatom diffusion using PRD and DRD give consistent exponential distributions of escape times. Finally, we discuss guidelines for choosing the optimal number of replicas and replica trajectory length for the DRD method. © 2015 AIP Publishing LLC. [<http://dx.doi.org/10.1063/1.4934987>]

I. INTRODUCTION

One of the most significant challenges for molecular dynamics (MD) simulations is to overcome the time scale gap between vibrational motion and the rare events of interest for chemical and material properties. One of the pioneers of this field is Voter who has developed accelerated molecular dynamics methods including parallel replica dynamics (PRD), hyperdynamics (HD), and temperature accelerated dynamics (TAD).^{1–5} In the most relevant method to the work here, PRD, a set of replica trajectories are started within the same initial state. These trajectories are thermalized using independent streams of random numbers so that they become decorrelated and statistically independent of each other. The trajectories are then run until one detects a transition to an adjacent state. At this point, the transition is reported and all trajectories are stopped. The total simulation time is advanced by the sum of the simulation times accumulated on the replicas. The replicas are then started in the product state of the replica for which a transition was detected.

The PRD method is well adapted to parallel architectures, for example, using a message passing interface. It is nontrivial, however, to use PRD within a distributed computing environment (DCE) due to the difficulty of ensuring that the replicas simultaneously stop and then restart their trajectories in a new state when a transition is detected. In a typical client-server DCE, communication is initiated from the clients to the server but not from the server to the clients or between clients. This makes it difficult to rapidly propagate the information of a transition from a single client to the entire network. Additionally,

one cannot rely on clients reporting back to the server in a DCE so that algorithms must be fault-tolerant with respect to clients dropping out of the network. In this paper, we propose a modification to the PRD algorithm, which we call distributed replica dynamics (DRD) that is designed to work in a DCE. Similar to the PRD method, the configuration of a system is replicated on multiple clients and then decorrelated with independent MD trajectories. As with PRD, the clients run MD until a transition is detected. The difference with DRD is that the detection of a transition from a single client is not able to stop the trajectories on other clients. Instead, each client runs for a fixed trajectory length and reports back to the server when the assigned simulation task is done. As we will show, this allows the server to accumulate simulation time as clients report their results without introducing errors in the calculation. Our scheme has no requirement for synchronization or direct communication between replicas and is thus well suited for DCEs.

II. METHOD

Following the work of Voter,¹ we consider a classic, canonical system consisting of N atoms evolving on a $3N$ -dimensional potential energy surface. The escape from a reactant state, when the escape is a rare event, is a first-order process. In other words, the probability of a successful crossing from the initial state per unit time is constant and defined as the rate constant k ,

$$p(t) = k \exp(-kt). \quad (1)$$

The procedure of the DRD method is as follows.

^{a)}henkelman@utexas.edu

Step 1: The current configuration of the system is replicated and sent to M independent clients in the DCE.

Step 2: On each client, independent initial momenta are randomly generated according to the Maxwell-Boltzmann distribution at the desired simulation temperature. Then, a short dephasing trajectory is run for Δt_{dph} to decorrelate the replicated trajectories from each other. During this Δt_{dph} all transition attempts are rejected by reflecting the trajectory back into the initial state. Lelièvre and coworkers showed that this dephasing step is crucial for preparation of the quasi-stationary distribution, which gives rise to the exponential distribution of escape times.⁷ Formally, it would be more accurate to restart the trajectory in the reactant state and attempt to run the dephasing time again entirely within the initial state. We choose, however, to reflect rather than restart the trajectory because a restart loop will not necessarily finish in a fixed amount of computational time, especially in problematic cases where the escape time is faster than the dephasing time. Our algorithm requires that a fixed amount of work be done by each client before reporting the results to the server.

Step 3: An MD trajectory of fixed length, t_{rep} , is integrated on each replica. A check to see if the trajectory has made a transition to a new state is performed every Δt_{blk} . In our implementation, this is done by minimizing the configuration of the trajectory to see if the resulting minimum has changed from that of the initial state. If a transition is detected on a client, an additional MD correlation time, Δt_{cor} , is performed to capture any correlated events that might result from the transition. Otherwise, the MD trajectory on this client runs until the total time, t_{rep} , is reached. Information about whether a transition is found, the transition time, and the new state configuration is transmitted by the client back to the server.

Step 4: On the server side, data returned from the clients are registered and processed in chronological order. As illustrated in Fig. 1, the simulation clock is accumulated by t_{rep} for each replica in which no transition is found. When the first transition is reported, the simulation time is incremented by the transition time of the reporting replica for which a transition was detected, t_{tns} . The product configuration from this transition is then taken as the new global state in which new trajectories are initiated.

When a transition is detected, *steps 1-4* are repeated from within the new state.

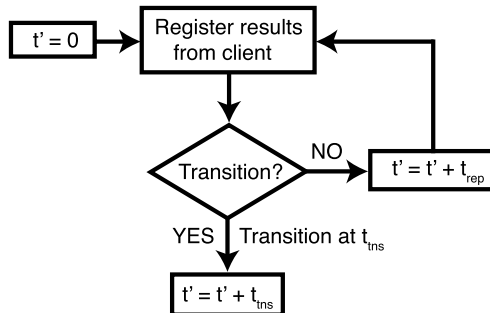


FIG. 1. Flow chart showing how the simulation clock is accumulated on the server in the DRD method.

A. Proof that DRD is correct

To prove the validity of the DRD method, we show that the moment generating function (MGF) of the DRD escape time t' is the same as t in Eq. (1) for a rare event with rate constant k . The MGF of a random valuable X is defined as

$$M_X(m) = E[e^{mX}], \quad m \in \mathbb{R}, \quad (2)$$

where $E[\dots]$ is the expectation value. The MGF of the exponentially distributed waiting time t is

$$M_t(m) = \int_0^\infty e^{mt} k e^{-kt} dt = \frac{k}{k-m}. \quad (3)$$

In DRD, the waiting time t' is generated by accumulating N_f reports of trajectories of fixed length t_{rep} , followed by one in which a transition was detected at time t_{tns} ,

$$t' = N_f t_{\text{rep}} + t_{\text{tns}}. \quad (4)$$

The MGF of t' is

$$M_{t'}(m) = \int_0^\infty e^{mt'} p(t') dt', \quad (5)$$

where $p(t')$ is the probability density of waiting times t' . It is non-trivial to write $p(t')$ explicitly, but it can be written as a mixed joint probability density of the discrete random variable, N_f , and the continuous random variable t_{tns} ,

$$p(t') = P_f(t_{\text{rep}})^{N_f} p(t_{\text{tns}}), \quad (6)$$

where $P_f(t_{\text{rep}})$ is the probability that a trajectory does not find a transition in the time t_{rep} ,

$$P_f(t_{\text{rep}}) = 1 - \int_0^{t_{\text{rep}}} k e^{-kt} dt = e^{-kt_{\text{rep}}}, \quad (7)$$

and $p(t_{\text{tns}})$ is the probability density of finding a transition at t_{tns} (from Eq. (1)) on the interval $[0, t_{\text{rep}}]$,

$$p(t_{\text{tns}}) = k e^{-kt_{\text{tns}}} \text{ for } t_{\text{tns}} \in [0, t_{\text{rep}}]. \quad (8)$$

Since N_f and t_{rep} are independent variables, the moments of t' are

$$\begin{aligned} M_{t'}(m) &= \int_0^\infty e^{mt'} p(t') dt' \\ &= \sum_{N_f=0}^\infty \int_0^{t_{\text{rep}}} e^{m(N_f t_{\text{rep}} + t_{\text{tns}})} P_f(t_{\text{rep}})^{N_f} k e^{-kt_{\text{tns}}} dt_{\text{tns}} \\ &= \sum_{N_f=0}^\infty e^{m(N_f t_{\text{rep}})} P_f(t_{\text{rep}})^{N_f} \int_0^{t_{\text{rep}}} k e^{(m-k)t_{\text{tns}}} dt_{\text{tns}} \\ &= \frac{k}{m-k} [e^{(m-k)t_{\text{rep}}} - 1] \sum_{N_f=0}^\infty [e^{mt_{\text{rep}}} P_f(t_{\text{rep}})]^{N_f} \\ &= \frac{k}{m-k} [e^{(m-k)t_{\text{rep}}} - 1] \frac{1}{1 - e^{mt_{\text{rep}}} e^{-kt_{\text{rep}}}} \\ &= \frac{k}{k-m}. \end{aligned} \quad (9)$$

Given that the MGF of the DRD transition time probability distribution is the same as the exponential distribution, the uniqueness theorem states that the two probability distributions are also the same.⁸

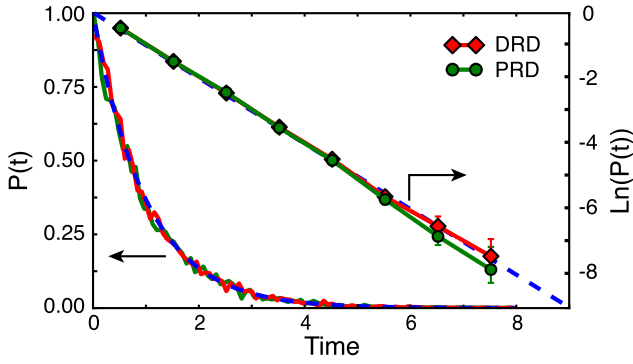


FIG. 2. Probability distribution function of escape times calculated by DRD (red) and PRD (green). The blue line is the exponential distribution from which random numbers were generated, with a rate constant $k = 1.0$.

III. RESULTS

A. Numerical simulation of escape times

Figure 2 shows results from numerical simulations of the probability distribution of the escape time using DRD and PRD. For each replica, instead of running MD of a real physics system, we generated random numbers t_i from an exponential distribution with rate constant $k = 1/\langle T_{\text{esp}} \rangle$ chosen to be 1.0. A distribution of transition times collected according to the DRD and PRD method is compared with the exponential distribution. In the DRD simulation, random numbers $\{t_i\}$ were generated sequentially until the first transition was detected within a specified simulation time $t_{\text{rep}} = 0.1$ (the first $t_i \leq t_{\text{rep}}$). For each replica, the simulation clock accumulated $\min\{t_i, t_{\text{rep}}\}$. In PRD, $N_{\text{rep}} = 20$ random numbers were generated; the minima of these N_{rep} transition times, multiplied by N_{rep} , was recorded as the escape time. As shown in Fig. 2, the DRD and PRD calculations (red and green solid lines, respectively) are consistent with the exact exponential distribution (blue dashed line).

B. Adatom hopping on Al(100)

Using the DRD method, we simulated the diffusion of an adatom on the Al(100) surface at $T = 225$ K. The Al interatomic interactions were described by an embedded-atom potential from Voter and Chen.⁹ The system was modeled as six atomic layers containing a 10×10 lattice, with the bottom two layers frozen. The Langevin-Verlet algorithm was used to integrate the dynamics of the system with a time step of 1.0 fs and a Langevin friction of 0.01 fs^{-1} . Our simulation was distributed to 150 clients using the Eon software¹⁰ and the BOINC⁶ DCE. Each DRD trajectory ran for 100 ps and then reported back to server. Δt_{dph} and Δt_{cor} were set to 1 ps. Each trajectory was thermalized to 225 K using an MD trajectory of duration Δt_{dph} , during which any transitions were rejected. Every Δt_{blk} of 2.0 ps, a state check was performed to see whether the system entered in a new state. The transition detection was done by minimizing to see if the resulting point was the initial state minimum. When a transition was detected, the configuration at time t_{ms} was passed back to the server (when the client task was completed).

To establish the validity of the DRD method, we ran a PRD simulation on the same system with 50 replicas using

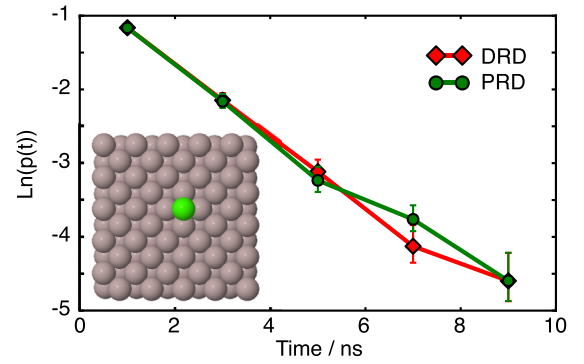


FIG. 3. Probability distribution of transition times for adatom hopping on Al(100). The red line shows the distribution from DRD while the green line is from the PRD method. The inset shows a top view of our model for Al(100) surface, with the adatom highlighted in green.

the same MD settings as the DRD simulation. A total of 500 events were collected using both DRD and PRD. The probability distributions of escape times are shown in Fig. 3. DRD produces the same exponential distribution of escape time and average escape time as PRD ($\langle T_{\text{esp}}^{\text{DRD}} \rangle = 2.07 \pm 0.09 \text{ ns}$, $\langle T_{\text{esp}}^{\text{PRD}} \rangle = 2.19 \pm 0.09 \text{ ns}$), to within the statistical uncertainty.

C. Efficiency of DRD

The efficiency of DRD is somewhat dependent on the way in which client jobs are assigned by the server. In the strategy that we used, the server distributes jobs to the clients in bundles of N_{rep} . If no transition is found in this set of jobs, another set of N_{rep} jobs is then distributed until the first transition event is reported. This simple strategy is not necessarily optimal, but it allows us to determine a qualitative relationship between the computational efficiency and the choice of N_{rep} . In order to simplify the discussion, we scale the time unit by the average transition time, $\langle T_{\text{esp}} \rangle$. In other words, we set $k = 1$ so that $\langle T_{\text{esp}} \rangle = 1$.

The computational overhead of DRD is defined as the average number of force calls required to see a single transition $\langle N_{\text{fcs}} \rangle$ as compared to the PRD algorithm. The efficiency of DRD is then expressed as

$$E_f = \frac{\langle T_{\text{esp}} \rangle}{\langle N_{\text{fcs}} \rangle} = \frac{1}{\langle N_{\text{fcs}} \rangle}. \quad (10)$$

Assuming that N_{bnl} blocks of N_{rep} replicas have been sent out before the first transition has been detected,

$$\langle N_{\text{fcs}} \rangle = (t_{\text{rep}} + \Delta t_{\text{dph}}) \langle N_{\text{bnl}} \rangle N_{\text{rep}} \quad (11)$$

and

$$N_{\text{bnl}} = \begin{cases} 1 & 0 < t < t_{\text{rep}} \\ \dots & \\ i & (i-1)N_{\text{rep}}t_{\text{rep}} < t < iN_{\text{rep}}t_{\text{rep}} \end{cases}. \quad (12)$$

The expectation value of N_{bnl} is

$$\langle N_{\text{bnl}} \rangle = \sum_{i=1}^{\infty} iP_i, \quad (13)$$

where P_i is the probability of a transition in the i th bundle of replicas, meaning that $(i-1)N_{\text{rep}}t_{\text{rep}} < t < iN_{\text{rep}}t_{\text{rep}}$. Setting

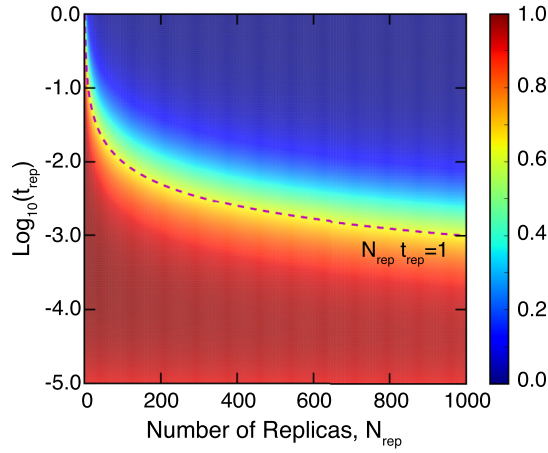


FIG. 4. Efficiency contour as a function of N_{rep} and t_{rep} described by Eq. (17) with $\Delta t_{\text{dph}} \cong 10^{-6}$ s. The dashed line is for $N_{\text{rep}} t_{\text{rep}} = 1$.

$u = N_{\text{rep}} t_{\text{rep}}$ gives

$$P_i = [e^{-u}]^{i-1} [1 - e^{-u}] \quad (14)$$

and

$$\langle N_{\text{bnl}} \rangle = \sum_{i=1}^{\infty} i (e^{-u})^{i-1} (1 - e^{-u}) \quad (15)$$

$$= (e^u - 1) \sum_{i=1}^{\infty} i e^{-ui} = \frac{1}{1 - e^{-u}}. \quad (16)$$

Substituting $\langle N_{\text{bnl}} \rangle$ into Eq. (10), and setting $v = N_{\text{rep}} \Delta t_{\text{dph}}$, gives the efficiency

$$E_f = \frac{1 - e^{-u}}{u + v}. \quad (17)$$

We expect DRD to be effective for transitions on a time scale of μs or longer, and for systems which dephase in a time of Δt_{dph} of $\sim\text{ps}$. Given these parameters, i.e., $\Delta t_{\text{dph}} \cong 10^{-6} \langle T_{\text{esp}} \rangle$,

Fig. 4 shows the efficiency of DRD as a function of N_{rep} and t_{rep} . For a fixed t_{rep} , the efficiency drops with increasing N_{rep} . However, as long as $N_{\text{rep}} t_{\text{rep}} < 1$, an efficiency over 75% can be achieved. For instance, to simulate a μs event, with $\Delta t_{\text{dph}} = 1$ ps, $\Delta t_{\text{rep}} = 100$ ps and $N_{\text{rep}} = 1000$ yield a computational efficiency near 80%.

IV. CONCLUSION

In conclusion, we show that DRD is a robust method for accelerating MD simulations in a DCE in a way that is consistent with Voter's PRD. We also show that a high efficiency can be achieved with an appropriate choice of the number of replicas and the MD time that each replica simulates.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (No. CHE-1152342) and the Texas Advanced Computing Center. We are grateful to Liangfei Qiu for his suggestion of using the uniqueness theorem.

¹A. F. Voter, *Phys. Rev. B* **57**, R13985 (1998).

²A. F. Voter, *J. Chem. Phys.* **106**, 4665 (1997).

³A. F. Voter, *Phys. Rev. Lett.* **78**, 3908 (1997).

⁴A. F. Voter and M. R. Sørensen, *Mater. Res. Soc. Symp. Proc.* **538**, 427 (1998).

⁵M. R. Sørensen and A. F. Voter, *J. Chem. Phys.* **112**, 9599 (2000).

⁶D. P. Anderson, "BOINC: A system for public-resource computing and storage," in *Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA, 8 November 2004* (IEEE, 2004), pp. 4–10.

⁷C. L. Bris, T. Lelièvre, M. Luskin, and D. Perez, *Monte Carlo Methods Appl.* **18**, 119 (2012).

⁸J. F. Kenney and E. S. Keeping, *Mathematics of Statistics*, 2nd ed. (Van Nostrand, Princeton, NJ, 1951).

⁹A. F. Voter and S. P. Chen, *Mater. Res. Soc. Symp. Proc.* **82**, 175 (1986).

¹⁰S. T. Chill, M. Welborn, R. Terrell, L. Zhang, J.-C. Berthet, A. Pedersen, H. Jónsson, and G. Henkelman, *Modell. Simul. Mater. Sci. Eng.* **22**, 055002 (2014).