Los Alamos NATIONAL LABORATORY

Scalable approaches to long-time atomistic dynamics

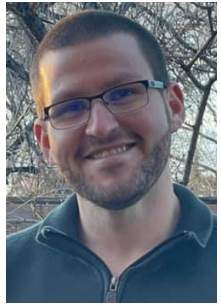
Danny Perez

Theoretical Division T-1

LA-UR-21-21287

Collaborators/Acknowledgements

- **Method development:** Arthur Voter, Andrew Garmon
- **Mathematics:** Tony Lelièvre, Claude Le Bris, Mitch Luskin, David Aristoff
- **Code:** The EXAALT ECP team
- **Funding:** DOE ECP, BES; LANL LDRD
- **Computing:** LANL IC, NERSC



Los Alamos NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

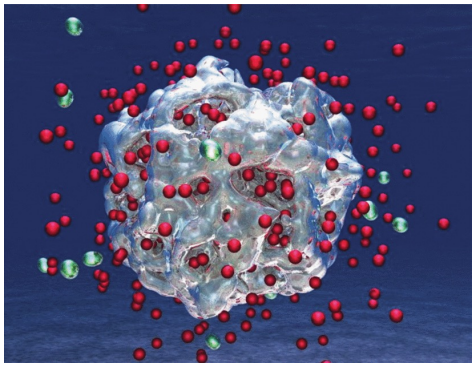
Plan

- The timescale problem of (parallel) MD
- Parallel Trajectory Splicing (ParSplice)
- Improving speculation in ParSplice
- Improving resource allocation in ParSplice
- A new ParSplice-inspired mathematical formalism for state-to-state dynamics

Why Molecular Dynamics?

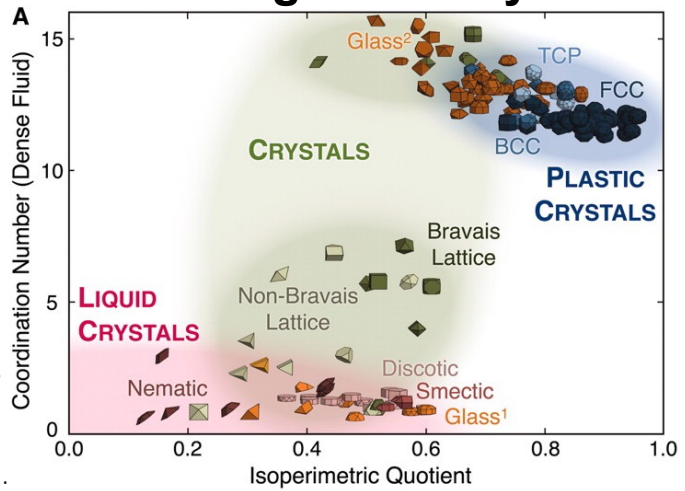
Ubiquitous: >1M hit on Google scholar

H production in Water/Al (Quantum MD)



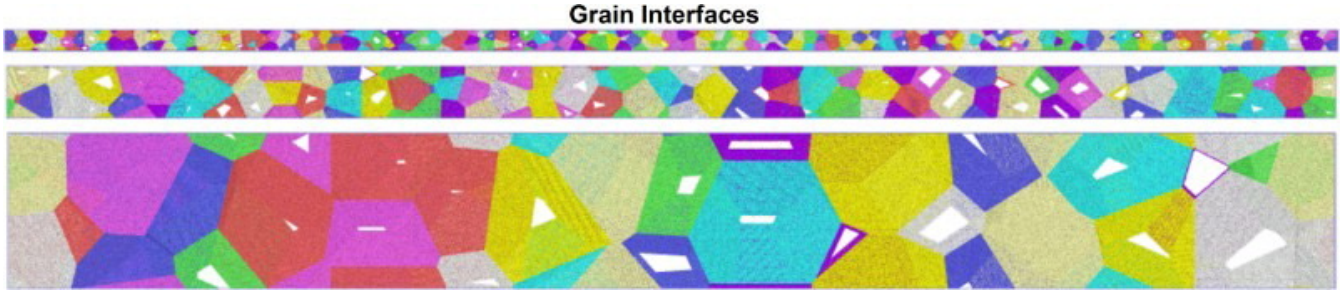
K. Shimamura et al., “Hydrogen-on-Demand Using Metallic Alloy Nanoparticles in Water,” Nano Letters, vol. 14, no. 7,2014, pp. 4090–4096

Phases of granular systems



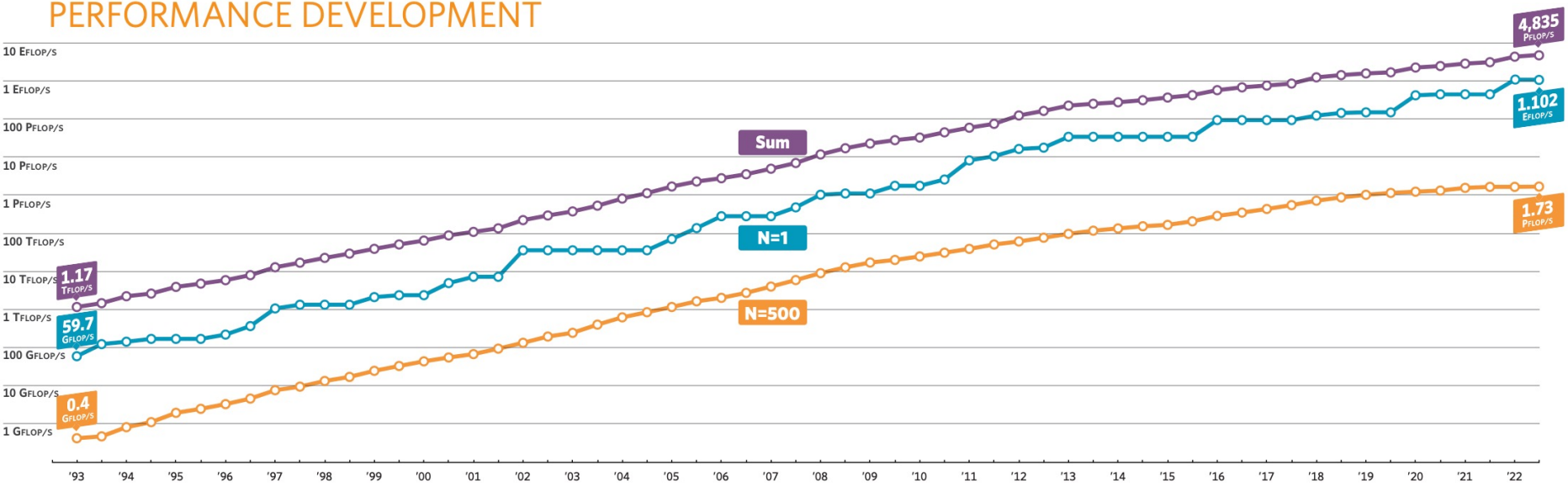
Glotzer, Sharon C., and Michael J. Solomon. "Anisotropy of building blocks and their assembly into complex structures." *Nature materials* 6.8 (2007): 557-562.

Shock Response of coarse grained explosives



Mattox, Timothy I., et al. "Highly scalable discrete-particle simulations with novel coarse-graining: accessing the microscale." *Molecular Physics* 116.15-16 (2018): 2061-2069.

PERFORMANCE DEVELOPMENT



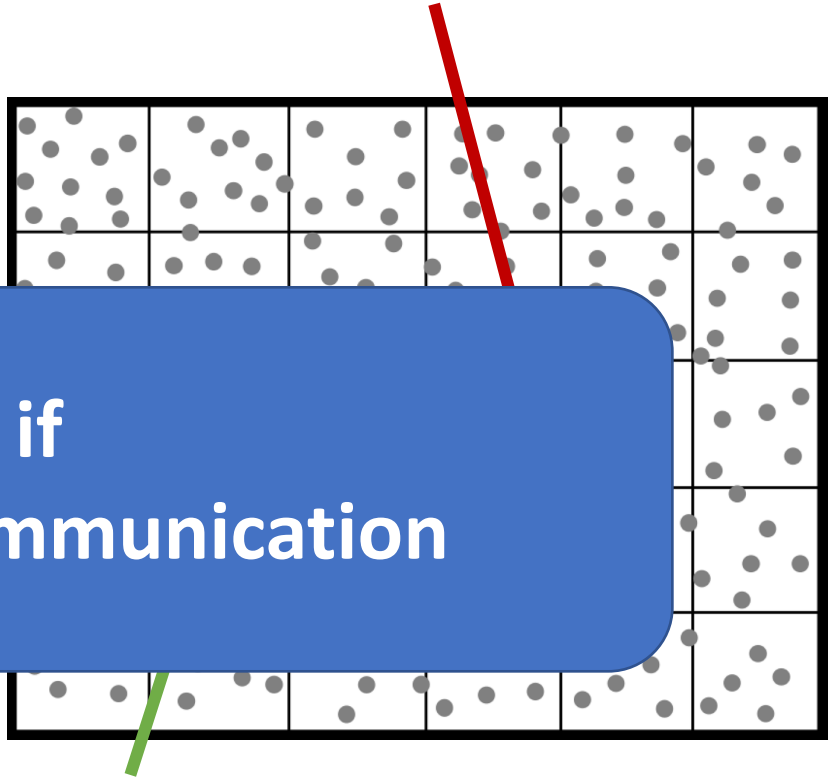
Parallel MD

Most cycles spent here

Get forces $\boldsymbol{F} = -\nabla V(\boldsymbol{r}^{(i)})$ and $\boldsymbol{a} = \boldsymbol{F}/m$

Move atoms: $\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + \boldsymbol{v}^{(i)}\Delta t + 1/2\ \boldsymbol{a}\Delta t^2 +$

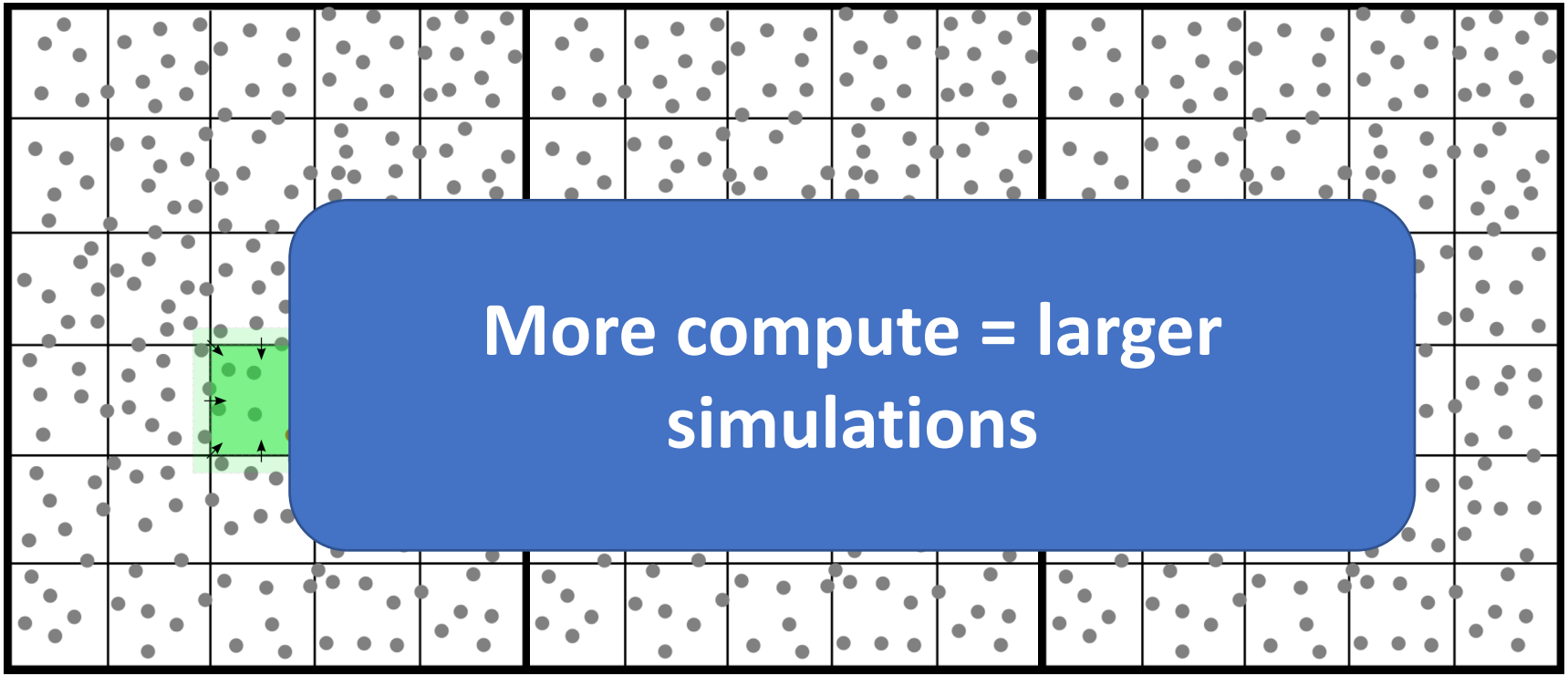
Communication required at every step



Each processor owns its domain

Scalable if
computation >> communication

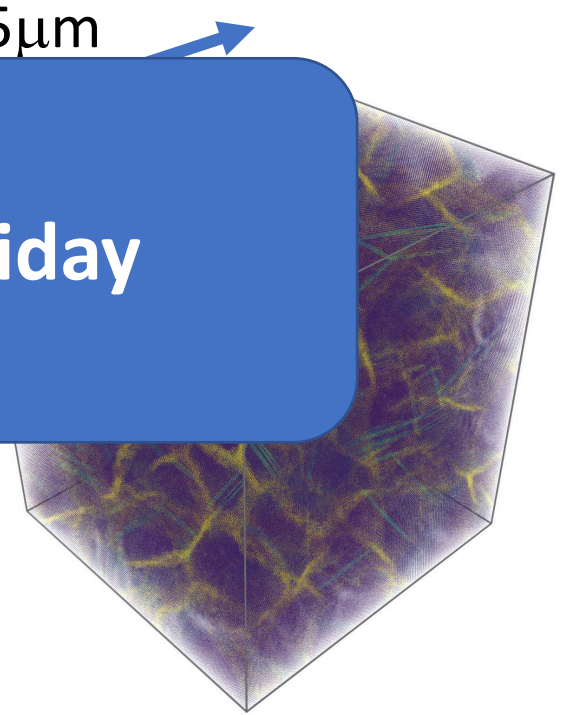
MD weak-scales



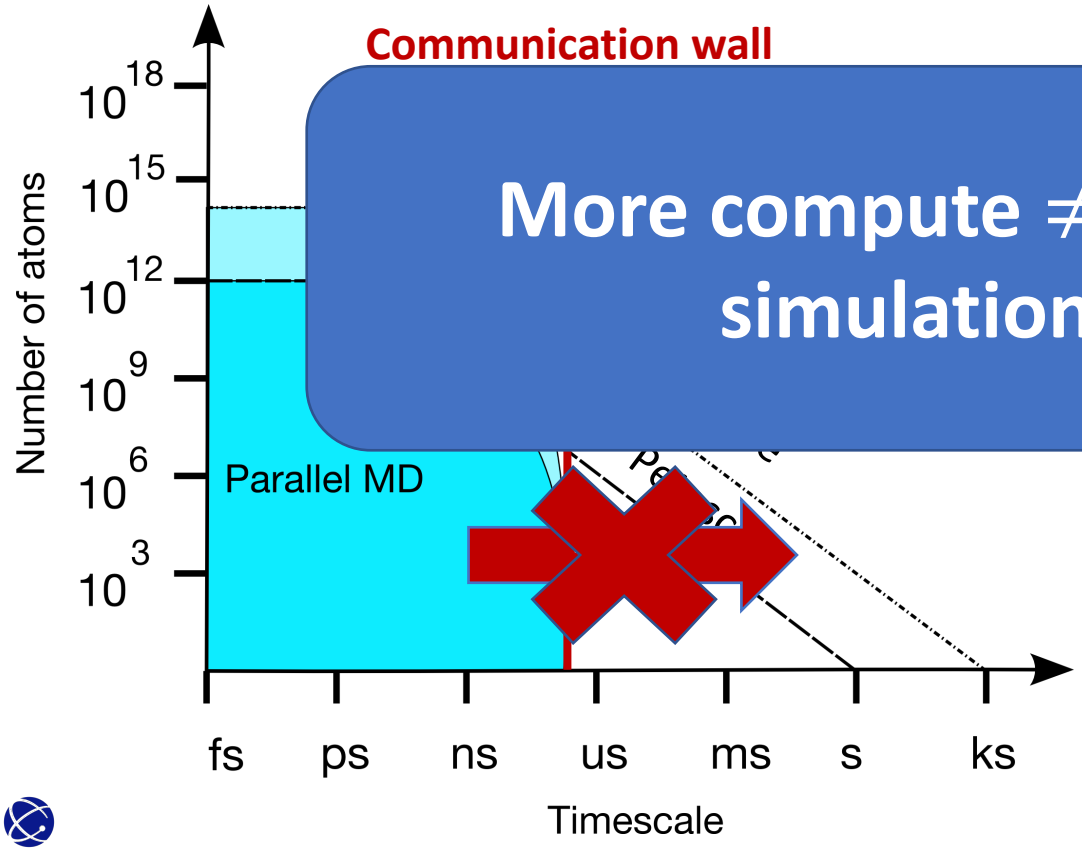
A brief history of MD

- 1959: 32 ato
- 1964: 864 a
- …
- 1996: 100 r
- 2000: 5 billi
- 2006: 320 bi
- 2008: 1 trillion atoms (Germann et al.)
- 2013: 4 trillion atoms (Eckhardt et al.)
- 2019: 20 trillion atoms (Tchipev et al.)

More on this on Friday

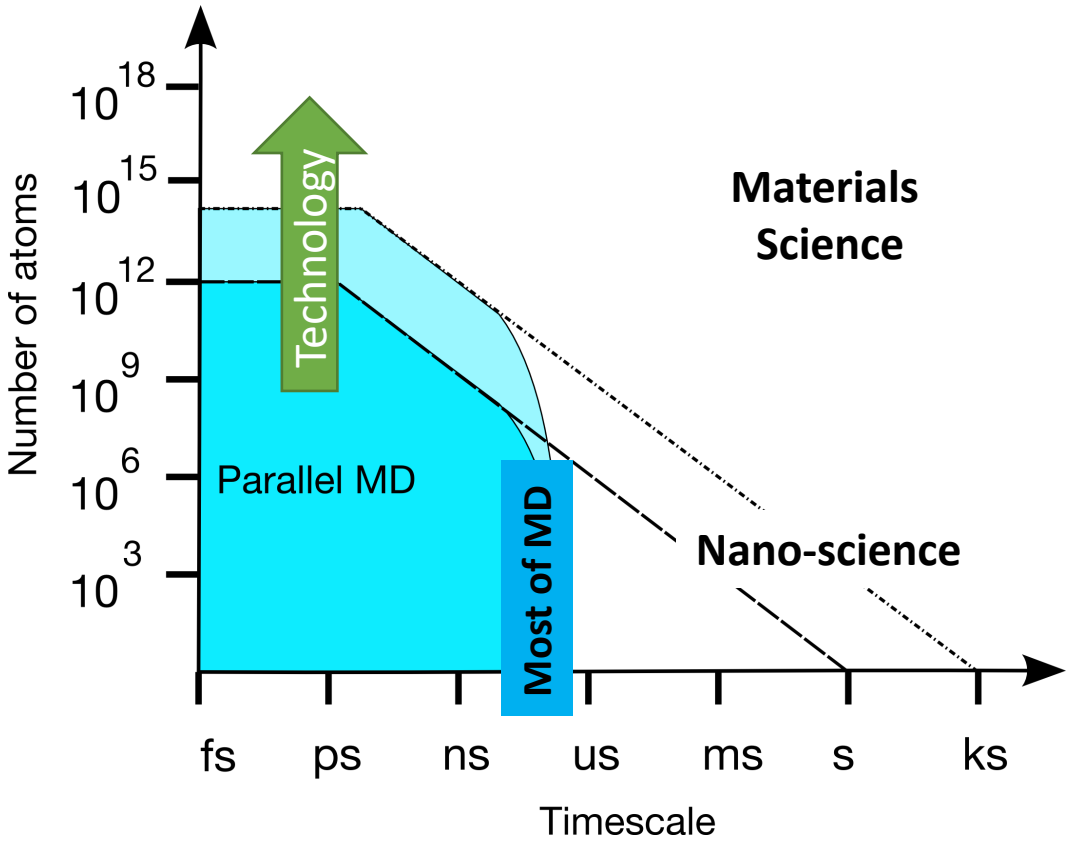


MD does not strong-scale

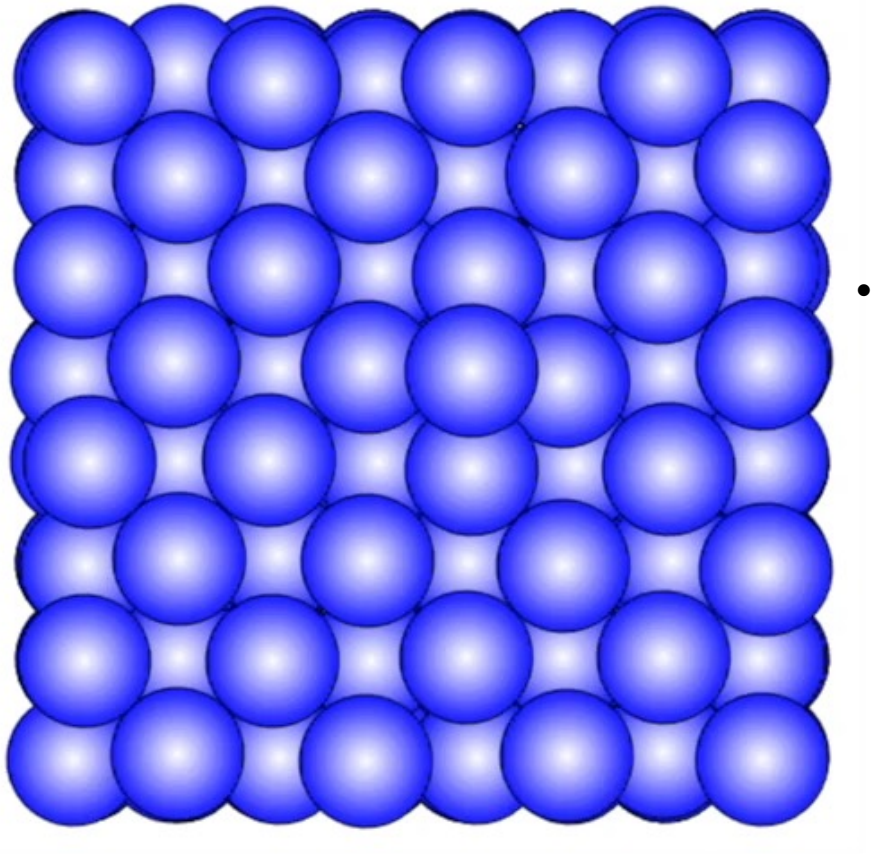


More compute ≠ longer simulations

The prospect for MD at the exascale



Metastability

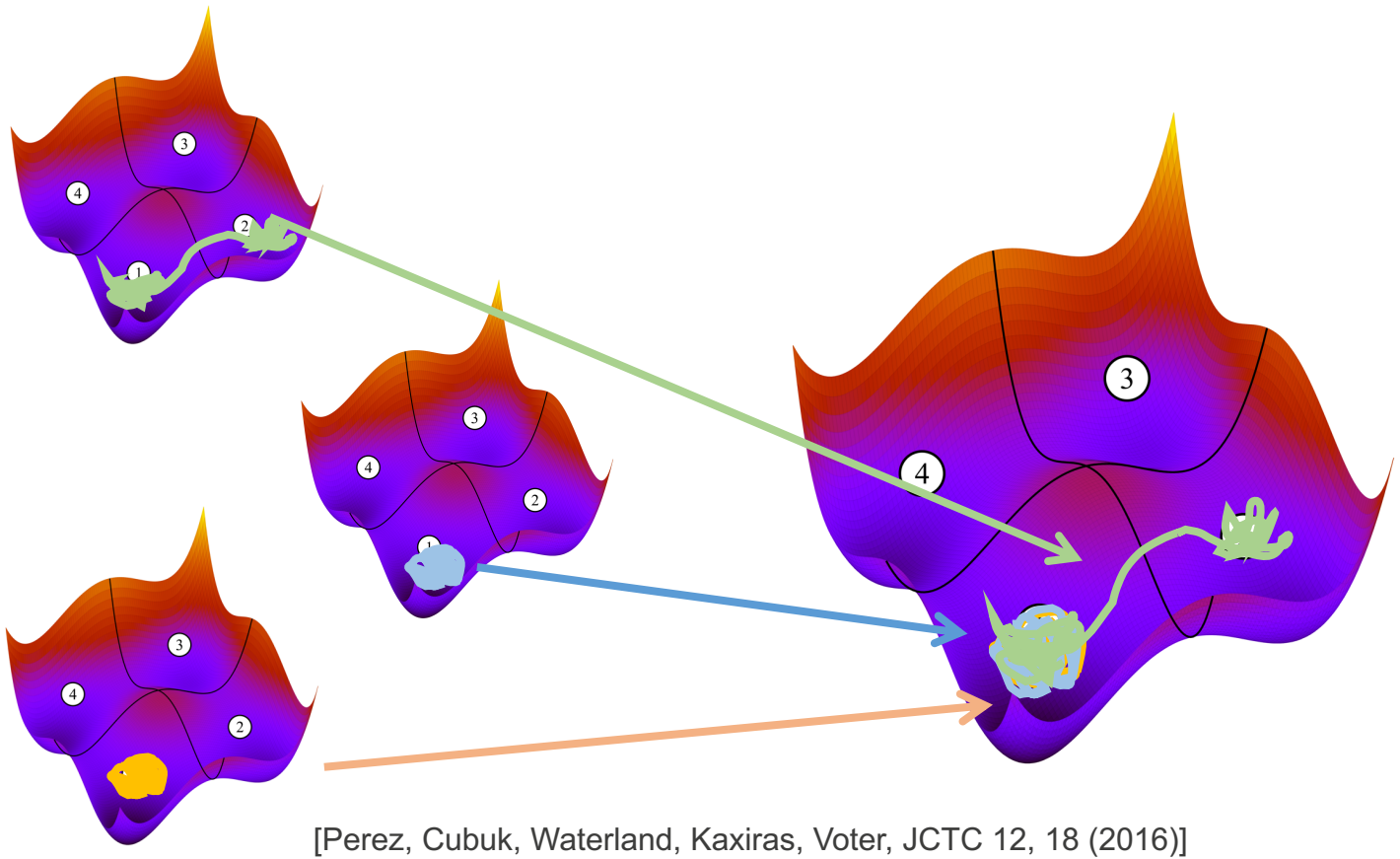


- For materials away from melting:
  - Fast vibrations/fluctuations (ps)
  - Slow conformational changes (ns-s)

- Short simulations are often **not informative** of long-time behavior

Theme of today's talk:
How can we leverage this separation of timescales to **parallelize** the dynamics **in time** instead of space

Parallel Trajectory Splicing (ParSplice)

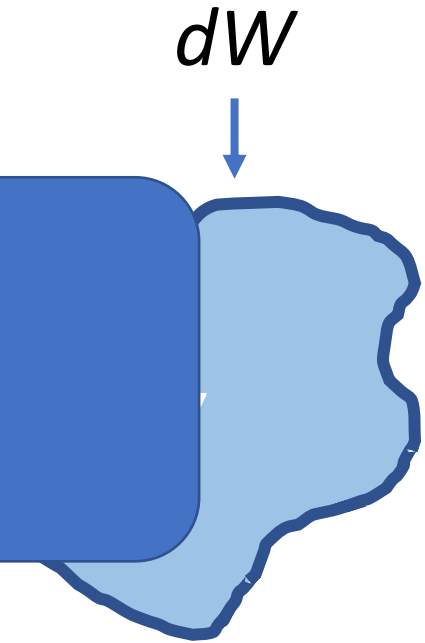


[Perez, Cubuk, Waterland, Kaxiras, Voter, JCTC 12, 18 (2016)]
[Aristoff, SIAM/ASA Journal on Uncertainty Quantification 7, no. 2 (2019): 685-719]

State-to-state dynamics

Goal is to generate a single *statistically correct* state-to-state trajectory

QSD for Langevin dynamics



In the following:

- Overdamped Langevin dynamics
- Absorbing
- Generator
- **QSD is ei**
  correspo

See Mouad's talk

Most of the following also applies to other dynamics, if:

- QSD exists
- QSD is unique
- Convergence to the QSD is fast

QSD for Langevin dynamics

$$\frac{\partial \rho}{\partial t} = L\rho \text{ on } W$$
$$\rho = 0 \text{ on } \partial W$$

With $L = -\nabla V \cdot \nabla + \beta^{-1}\Delta$

Then:

$$\rho(X,t) = \sum_k e^{-\lambda_k t}\, c_k^0 u_k(X)$$

For $t > (\lambda_2 - \lambda_1)^{-1}$ and conditional on not having escaped,

$$\hat{\rho}(X,t) \cong u_1(X) + O(e^{-(\lambda_2-\lambda_1)t})$$

Properties of the QSD

- The QSD of *W* is **unique**
- Convergence to the QSD is **exponential** with rate $(\lambda_2 - \lambda_1)$

From the QSD:

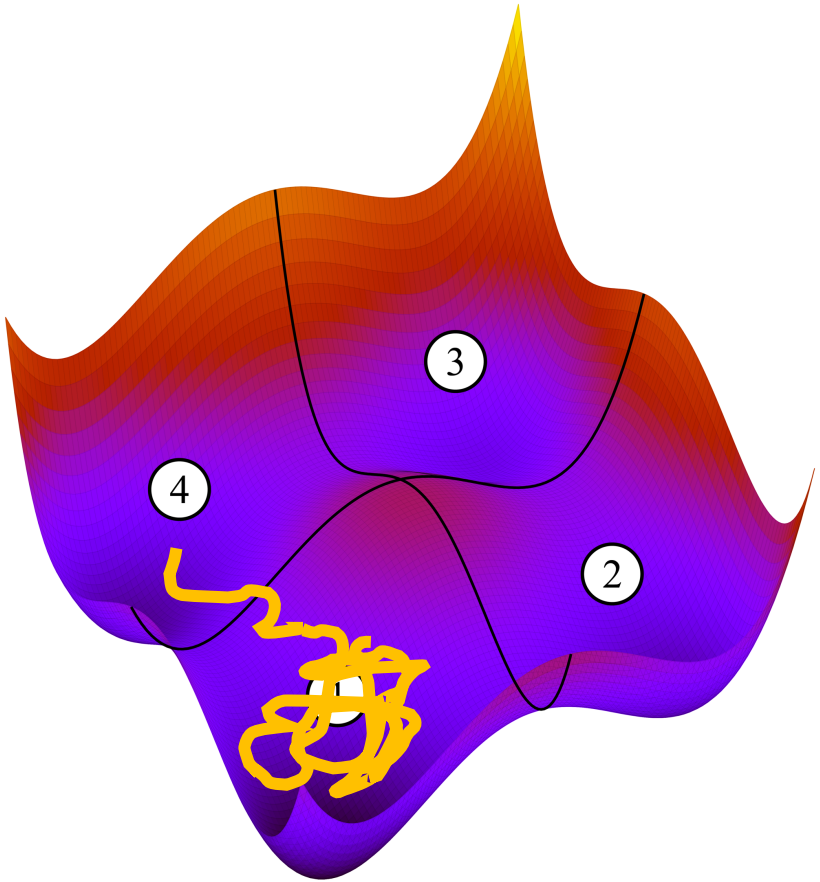
- First escape time is random and exponentially distributed with rate λ_1
- First escape point is random and uncorrelated with escape time

This is true for any state definition!

Does not depend on history before reaching the QSD

Overdamped Langevin: [Le Bris, Lelievre, Luskin, and DP, MCMA 18, 119 (2012)]

Langevin: [Lelievre, Ramil, Reygner, arXiv:2101.11999]



After only a short time in the state, the next escape time/location distribution is a complex function of the entry point

After spending $t_c > (\lambda_2 - \lambda_1)^{-1}$ in W, the next escape from W becomes *Markovian

All trajectories that spent $t_c > (\lambda_1 - \lambda_2)^{-1}$ in W are statistically equivalent with respect to how and when they will leave W*

* Up to an exponentially small error in t_c

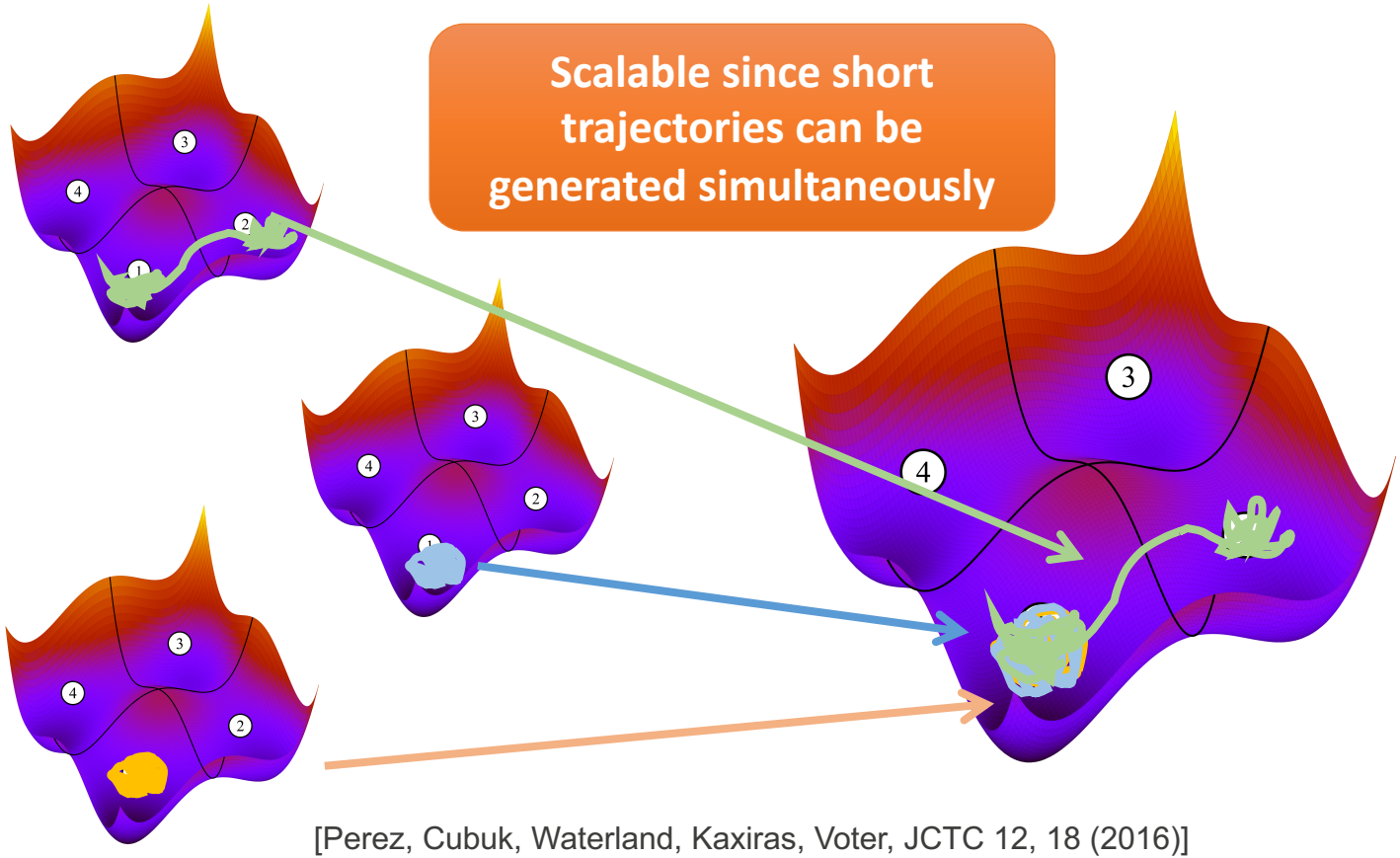
Trajectory building block

Define a **segment** as a trajectory that spent at least t_c in the same state before its

A valid state-to-state trajectory can be assembled by splicing independent segments end-to-end*

* Up to an exponentially small error in t_c

Parallel Trajectory Splicing (ParSplice)



[Perez, Cubuk, Waterland, Kaxiras, Voter, JCTC 12, 18 (2016)]
[Aristoff, SIAM/ASA Journal on Uncertainty Quantification 7, no. 2 (2019): 685-719]

Example of ParSplice use

- Shape fluctuations in nanoparticles:

  [Phys. Rev. Mat. 2, 126002 (2018)]

- Helium bubble transport in W:

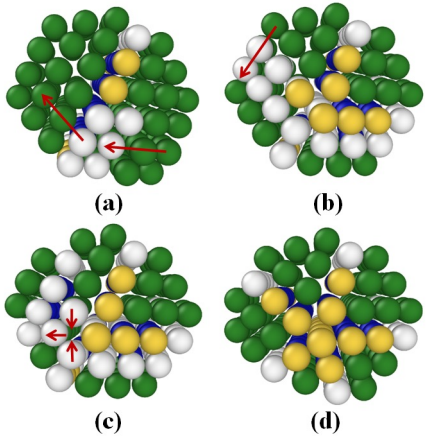
  [Sci. Rep. 7, 2522 (2017)]

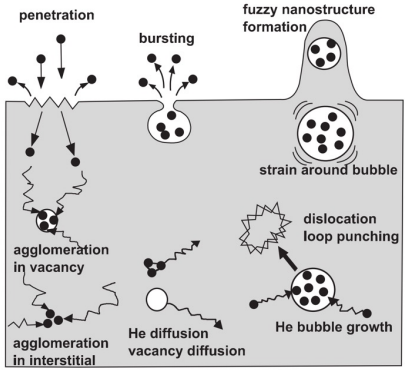
- Vacancy-mediated dislocation climb in Ni:

  [Phys. Rev. Mat. 5, 083603 (2021)]

- Segregation in CuNi:

  [JCTC 18, 4447 (2022)]





Shape fluctuations in nanoclusters

- Properties of nanoclusters are sensitive to shapes and sizes
- Some small nanoparticles don't have well defined shapes; continuously transform between different conformations
- This affects their physical/chemical properties
- How do these shape changes occur?

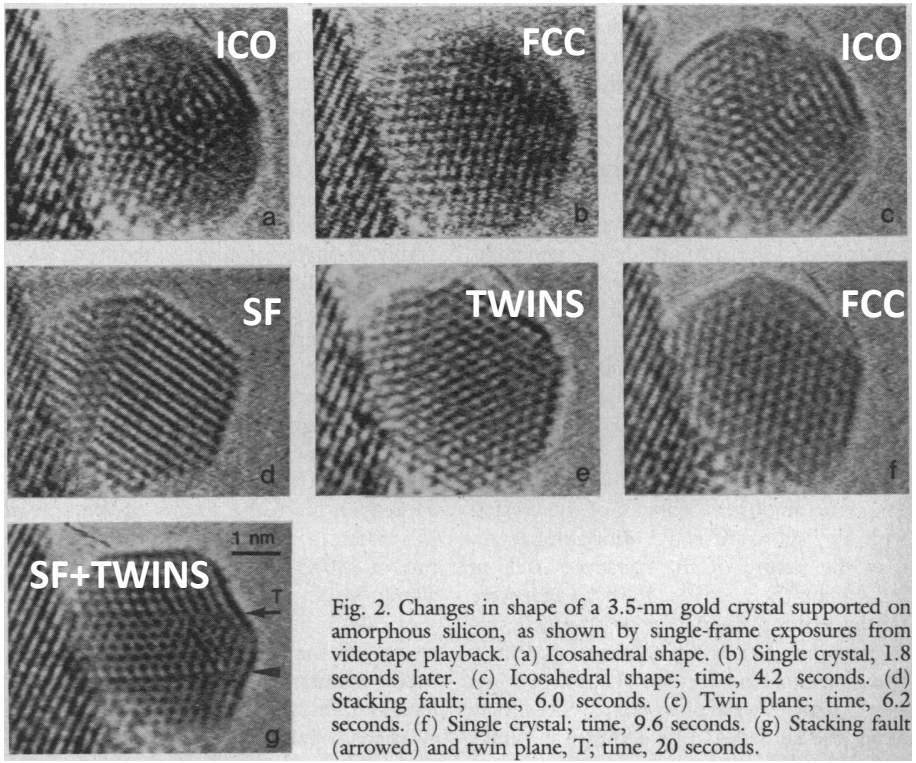


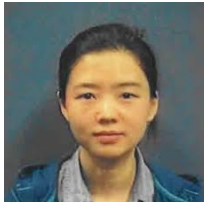
Fig. 2. Changes in shape of a 3.5-nm gold crystal supported on amorphous silicon, as shown by single-frame exposures from videotape playback. (a) Icosahedral shape. (b) Single crystal, 1.8 seconds later. (c) Icosahedral shape; time, 4.2 seconds. (d) Stacking fault; time, 6.0 seconds. (e) Twin plane; time, 6.2 seconds. (f) Single crystal; time, 9.6 seconds. (g) Stacking fault (arrowed) and twin plane, T; time, 20 seconds.

Smith *et al.*, Science 233, 872 (1986)

Shape Fluctuations in Nanoparticles

- Metallic nanoparticles (150-300 atoms)
- Between 3,600 and 36,000 cores

- **Long simulations:** up to 4 ms
- **Many transitions:** up to ~100M per run
- **Many states:** up to ~1M per run



Rao Huang
(Xiamen U.)

Huang, Lo, Wen, Voter, Perez, JCP 147, 152717 (2017)
Perez, Huang, Voter, JMR 33, 813 (2018)
Huang, Wen, Voter, Perez, Phys. Rev. Mat. 2, 126002 (2018)

Element	Number of Atoms	T (K)	Trajectory Length (ps)	Number of Transitions	Number of States	Description
Pt	146	900	70,257,528	162,965	6,246	fcc ⇒ deca ⇒ ico
	170	800	672,396,434	1,937,031	147,377	fcc ⇔ 5-fold caps ⇒ ico
		900	20,373,095	240,306	117,680	
	190	800	1,350,168,728	6,630,131	303,572	----------
		900	348,662,895	688,027	93,346	fcc ⇒ ico
	231	900	1,986,709,692	4,395,285	252,153	----------
		1000	92,171,602	955,401	42,383	
		1100	24,608,419	914,005	110,290	
Cu	146	550	301,832,137	3,942,180	237,293	fcc ⇒ ico
	170	500	4,156,073,707	6,160,286	240,594	----------
		550	23,712,165	656,202	241,491	fcc ⇔ 5-fold caps ⇒ ico
		600	21,690,608	1,039,065	144,713	fcc ⇒ deca ⇒ ico deca⇒ fcc ⇒ ico
	190	500	489,113,720	93,863,998	368,356	----------
		600	91,701,072	9,863,950	847,016	
	231	500	438,302,547	49,409	12,817	
		550	66,578,597	4,623,717	262,785	
		600	85,056,822	184,737	169,217	
		700	832,190	237,840	89,356	
Au	146	600	237,233,817	22,910,983	119,489	fcc ⇒ ico
	190	600	521,506,615	10,198,278	85,875	fcc ⇔ 5-fold caps
	231	800	774,813,889	795,678	159,743	fcc ⇒ 5-fold caps ⇒helical
Ag	146	500	122,897,307	2,558,937	71,357	----------
		550	21,613,546	1,988,646	136,297	fcc ⇔ off-centered 5-fold axis
	170	500	841,036,559	1,529,663	258,281	----------
		600	128,965,726	3,961,585	616,430	
	190	400	1,651,496,973	2,416,400	60,802	
		500	109,165,848	1,414,790	154,083	
		600	30,620,753	1,091,307	147,863	
	231	500	20,445,451	946,623	92,818	

Direct observation of shape fluctuations

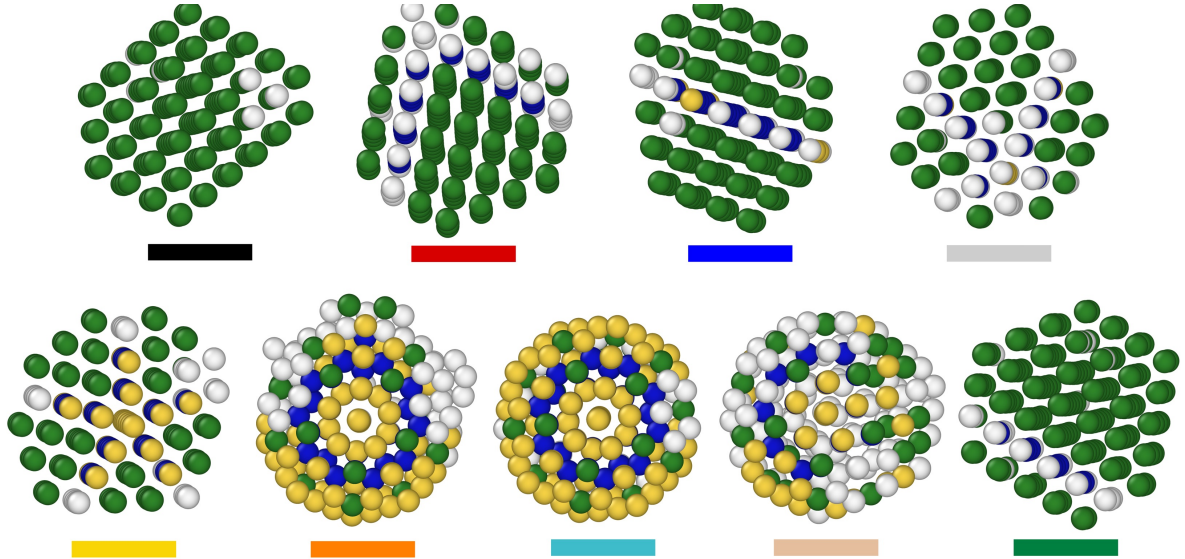
Cu-170@600K

- ~22 μs of simulation time
- ~10^6 transitions
- ~10^5 states

Icosahedral

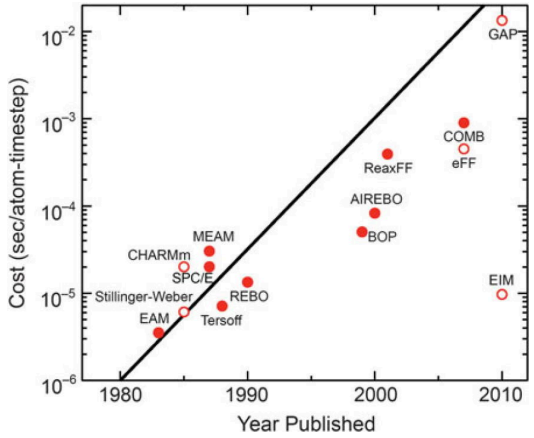
HCP

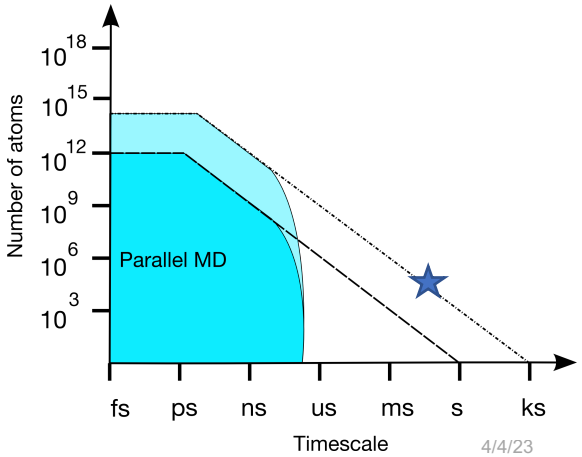
FCC



Where are we, and were do we need to go

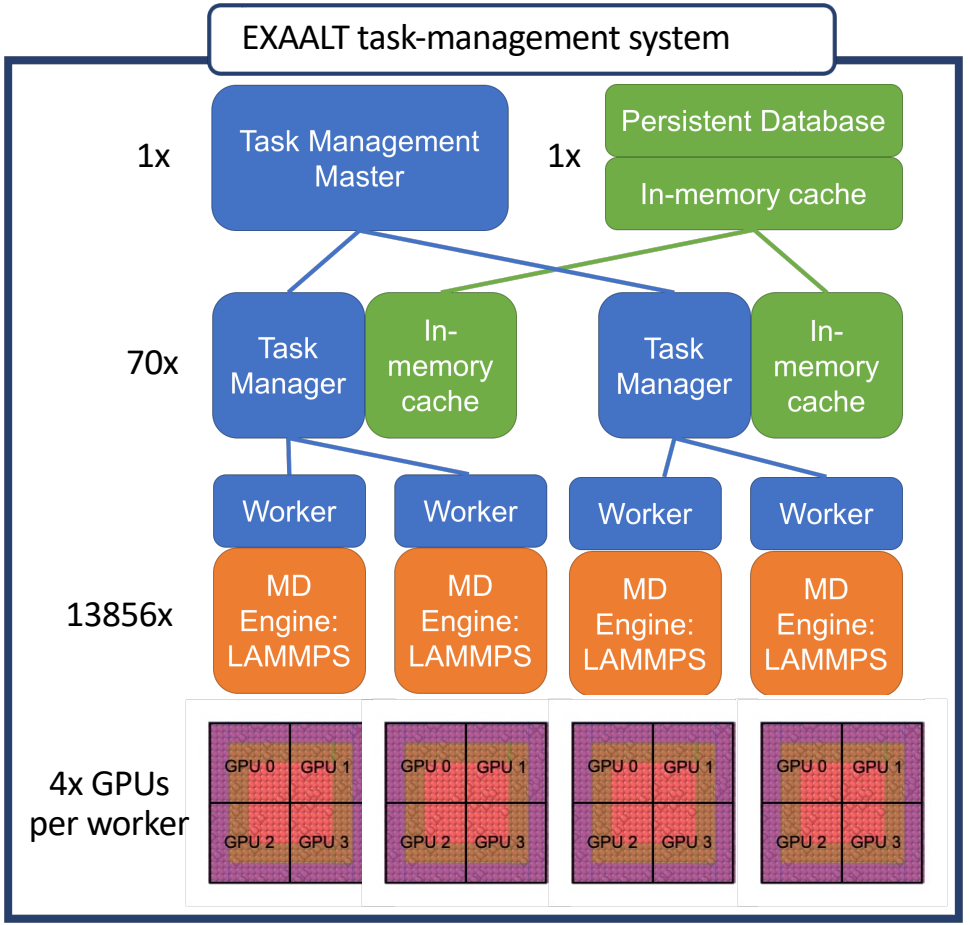
- Excellent performance on “expensive” ML potentials.
- The more expensive the potential, the easiest for ParSplice, as each replica strong-scales more.
- **The ultimate challenge is for “cheap” potentials for extremely long timescales.**





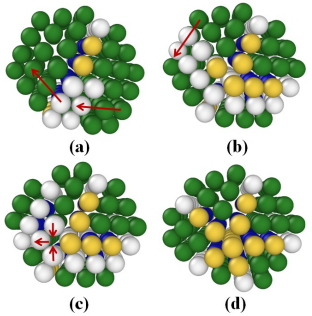
ParSplice at the exascale

- ParSplice executed using EXAALT on 7000 Frontier nodes (75% of machine)
- SNAP Potential
- 100,000 W atoms
- ~1% of resources for management
- ~99% of resources to simulation
- Infrastructure re-assigns MD tasks to workers every ~7 seconds
- 81 sub-domains
- ~170 instances of each sub-domain execute concurrently
- 4 GPU dies for every instance



72 nodes for data and task management
6928 nodes for MD simulations

Benchmark results



Rare events

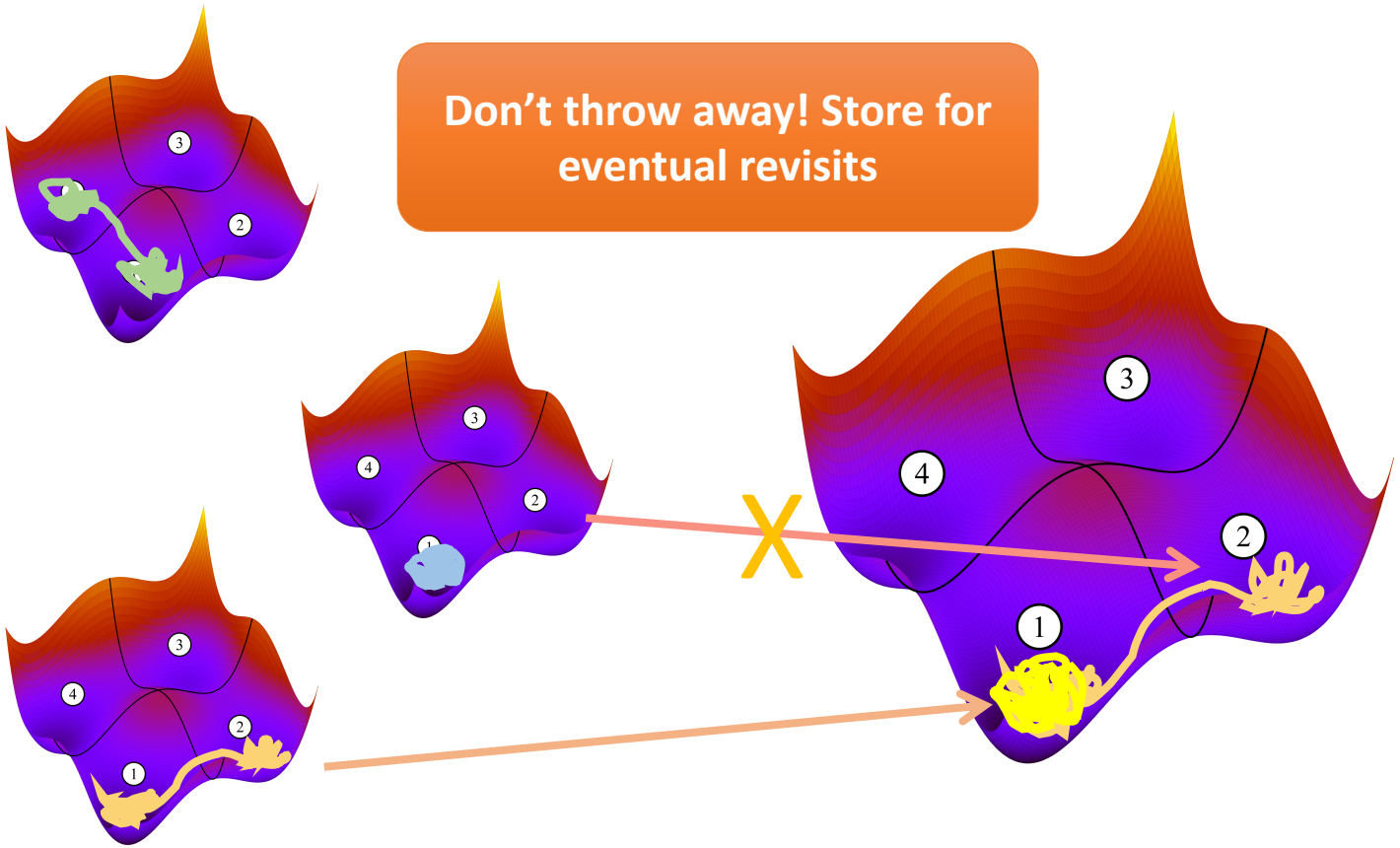
T=300K, LANL Grizzly, 4h runs

N_{cores}	Trajectory length (ps)	Generated segment time (ps)	#Transitions	#States	$\langle t_{trans}/N_c t_c \rangle$	<R>	Simulation rate (μs/hour)
9,000	556,093,988	556,539,980	4,614	28	13.19	166	139
18,000	1,315,941,923	1,346,516,503	24,610	64	2.97	384	333
27,000	2,209,432,238	2,214,868,608	13,479	47	4.15	294	552
36,000	2,291,027,808	2,318,254,470	50,258	60	1.26	909	592

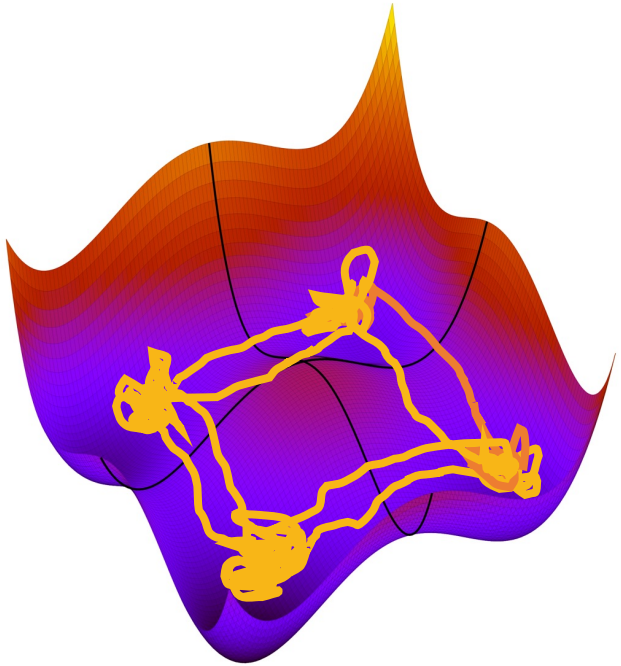
99% of generated segments were spliced

Peak simulation rate: 10 μs/min, 10 ms/day

Bookkeeping



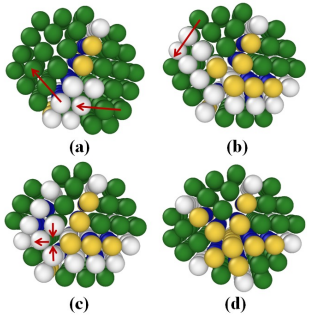
Super-basins



Revisits are extremely common!

Benchmark results

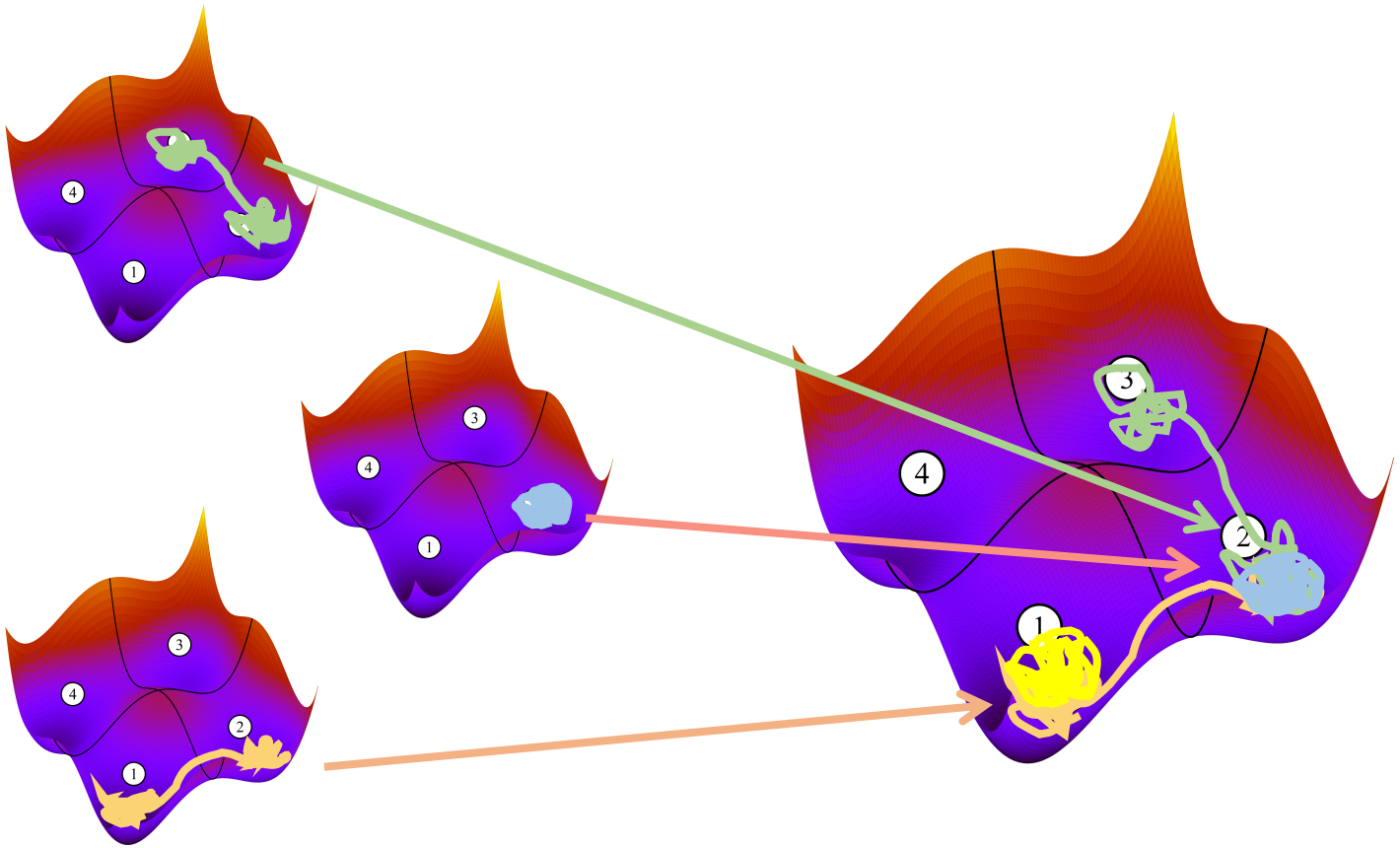
Revisits are common!



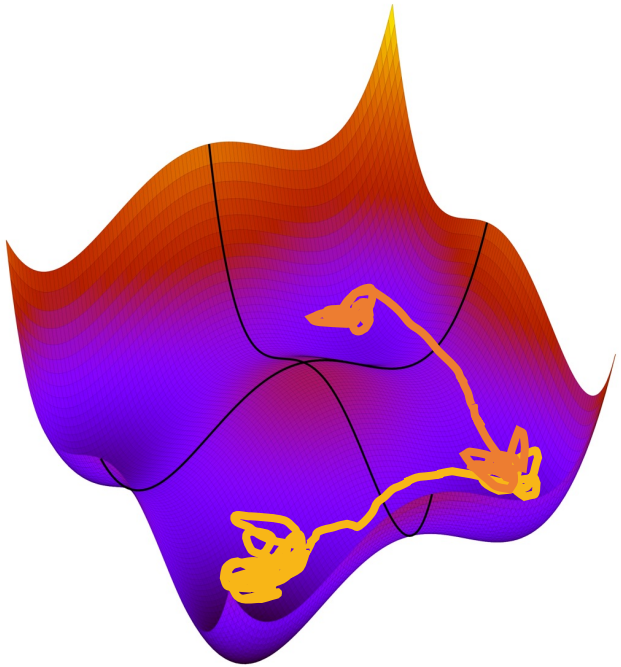
T=300K, LANL Grizzly, 4h runs

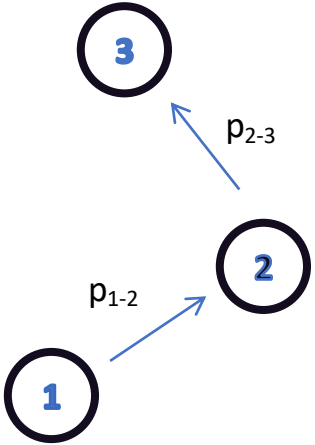
N_{cores}	Trajectory length (ps)	Generated segment time (ps)	#Transitions	#States	$\langle t_{trans}/M\ t_c \rangle$	<R>	Simulation rate (μs/hour)
9,000	556,093,988	556,539,980	4,614	28	13.39	166	139
18,000	1,315,941,923	1,346,516,503	24,610	64	2.97	384	333
27,000	2,209,432,238	2,214,868,608	13,479	47	4.55	294	552
36,000	2,291,027,808	2,318,254,470	50,258	60	1.26	909	592

Speculation



Statistical oracle

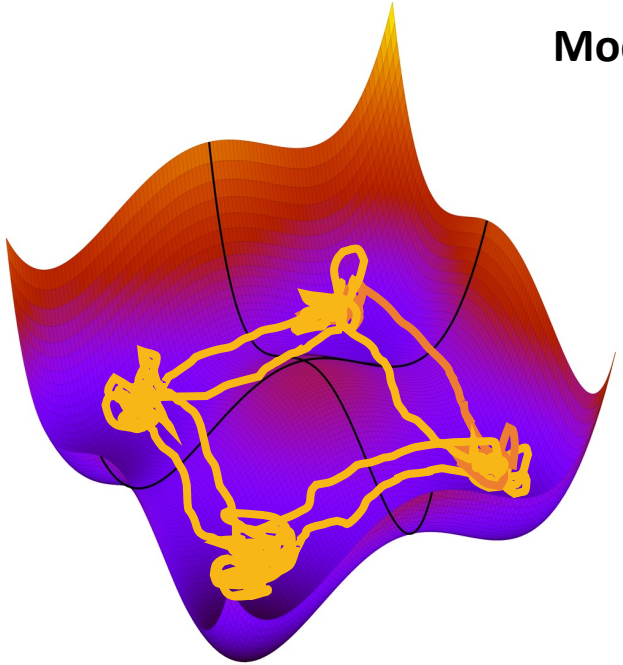


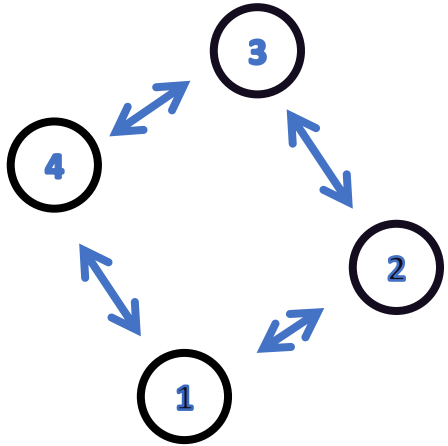


Statistical oracle

We use this model to speculate where the trajectory will be in the future

Model quality affects efficiency, but *not* accuracy





Statistical oracle

- Discrete time Markov chain: probability that a segment that starts in state *A* ends in state *B*

- V1: **MLE** on generated segments (simple!)

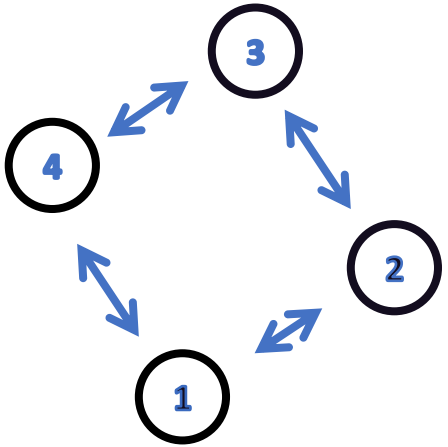
$$P_{AB} = \frac{Number\ of\ segments\ A \rightarrow B}{Number\ of\ segments\ generated\ in\ A}$$

- V2: MLE with **detailed balance constraint** (expensive)

$$P_{AB}\mu_A = P_{BA}\mu_B$$

[Noé et al., JCP 128, 244103(2008): 244103.]

- V3: MLE + DB + **Warp**



- **Warning:** The model are incomplete! **Contains only states and transitions that were observed before!**

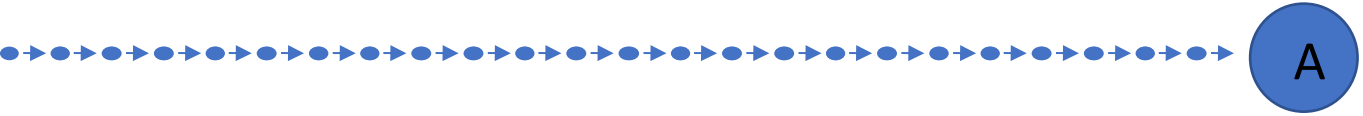
See A. Garmon, DP, MSMSE 28, 065015 (2020) for more detail on model construction

Segment scheduling

In which state should the **next segment** be generated?

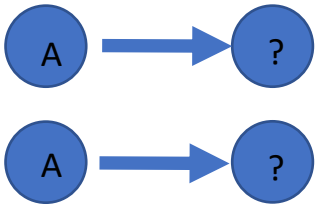
- In the state in which we are most likely to run out of segments!

Virtual-end scheduling



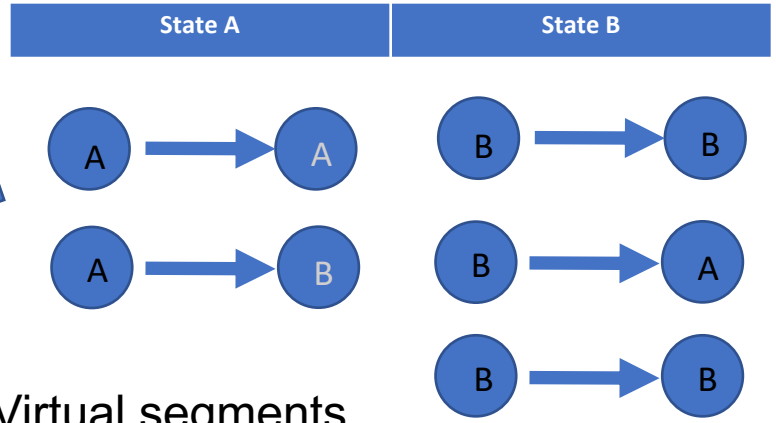
Virtual trajectory

Pending segments



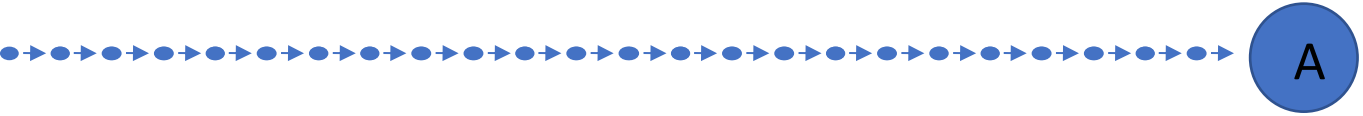
Statistical Model

$$\begin{bmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{bmatrix}$$

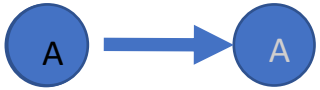


Virtual segments

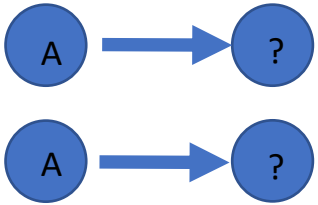
Virtual-end scheduling

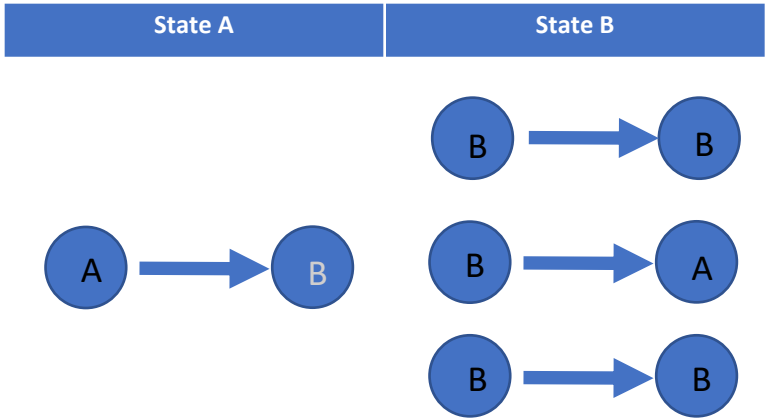


Virtual trajectory

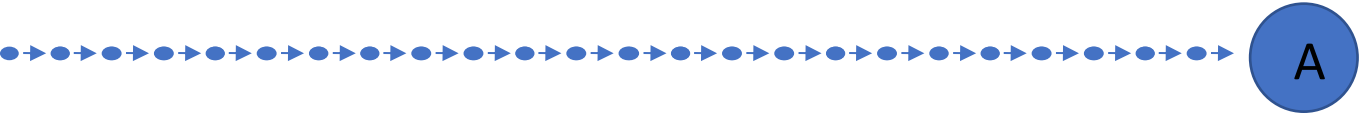


Pending segments



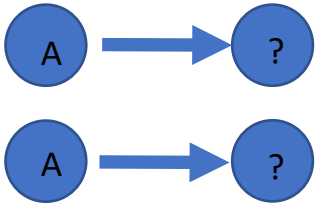


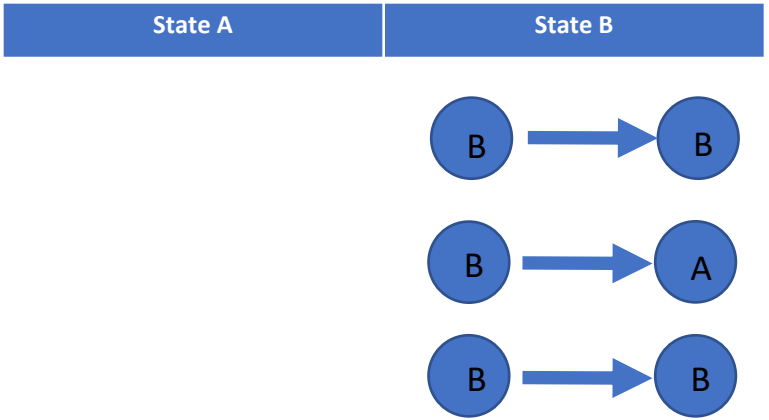
Virtual-end scheduling



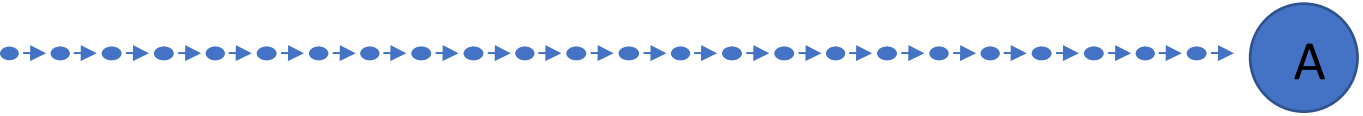
Virtual trajectory

Pending segments

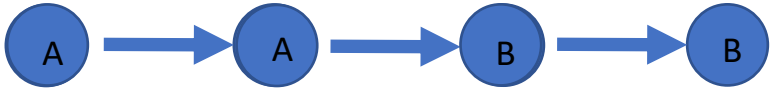




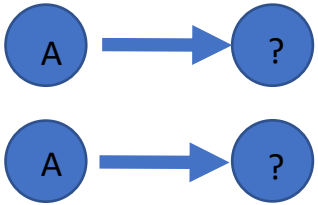
Virtual-end scheduling

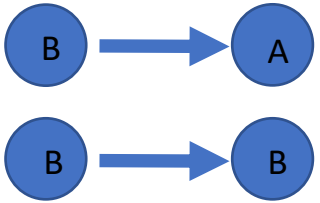


Virtual trajectory

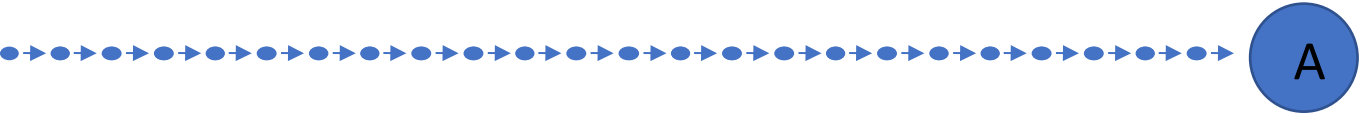


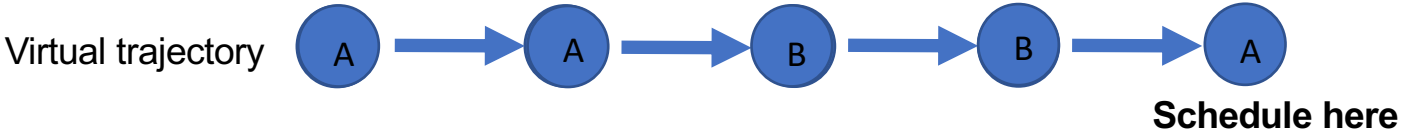
Pending segments



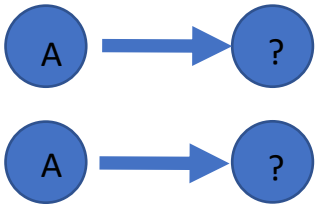


Virtual-end scheduling

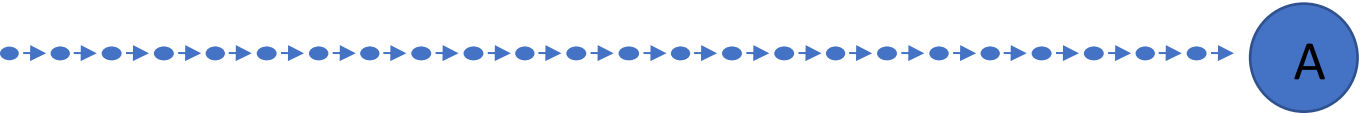


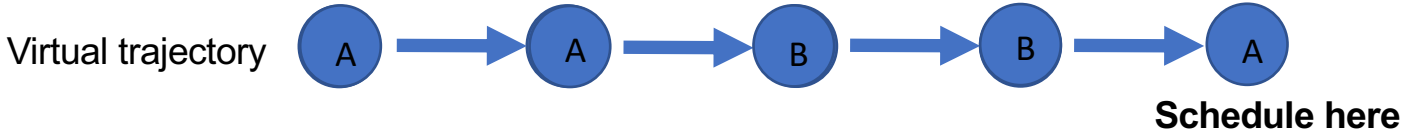


Pending segments

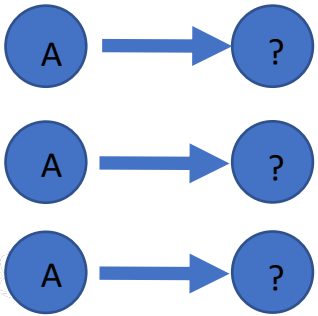


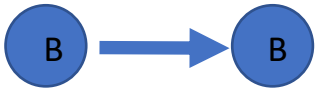
Virtual-end scheduling





Pending segments





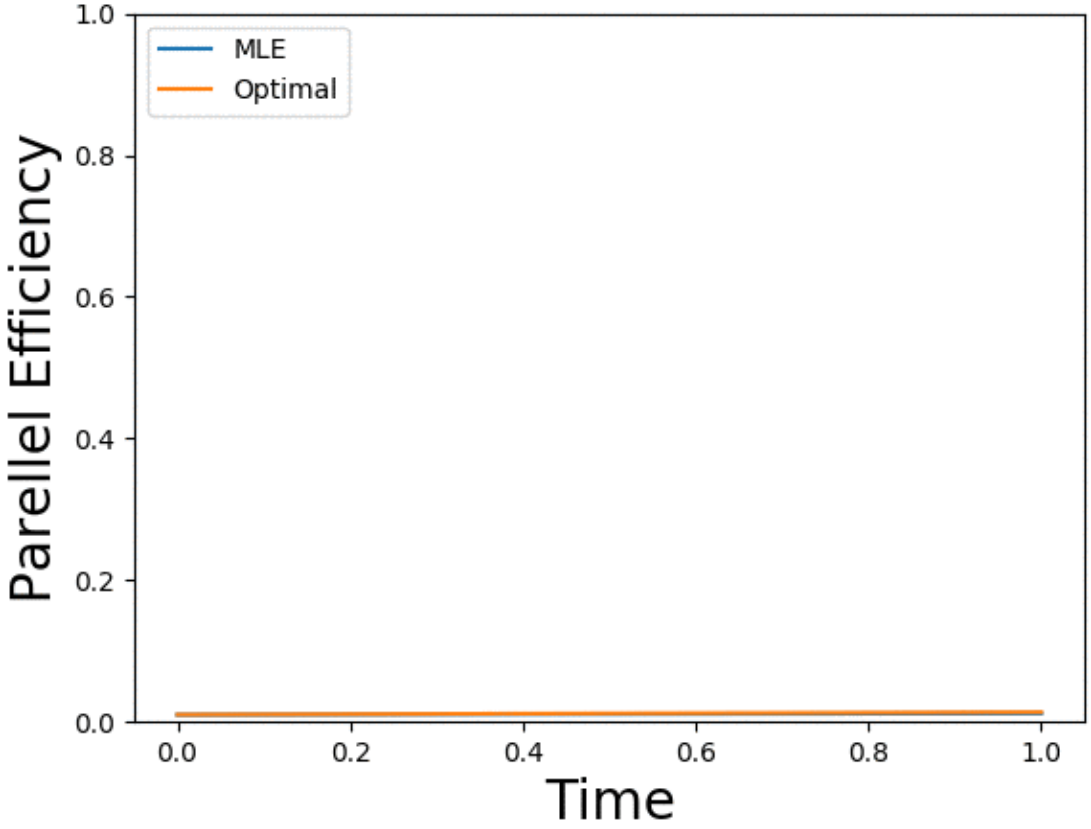
A ParSplice simulator

- To explore different strategies, we use a simulator with a known state-to-state dynamics model.
- In the following: **2D toy model**
  - 2500 states (50x50) with periodic boundary conditions
  - Rare events <n_escape> =10,000 segments
  - Resources: 1 million replicas

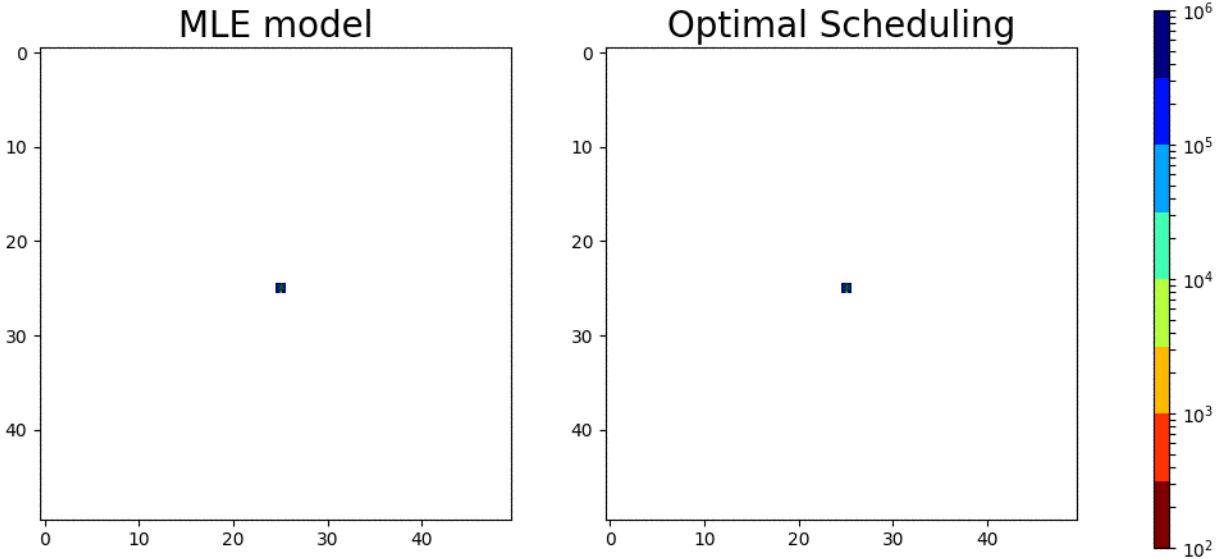
- Very hard test problem. Without any trick, you would get ~1% efficiency.

- Allows us to compare the data-driven models with truly optimal decisions taken with full information. Scheduling still only allowed in known states.

Predicted parallel efficiency

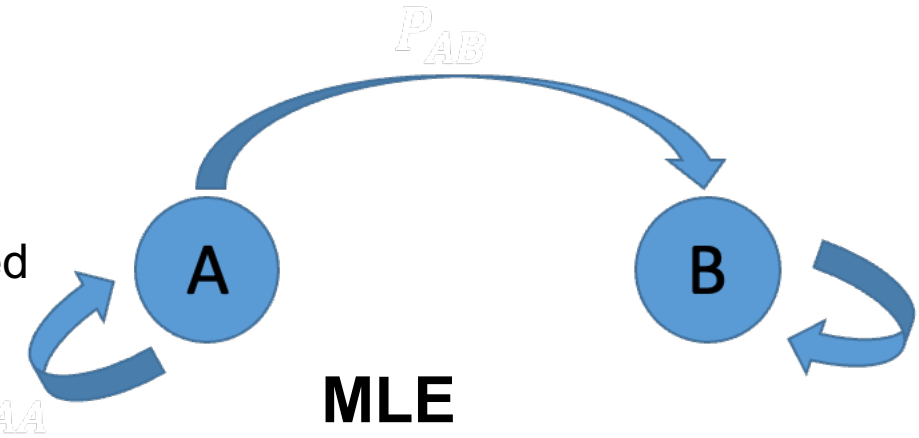


MLE Scheduling pattern



Imposing reversibility

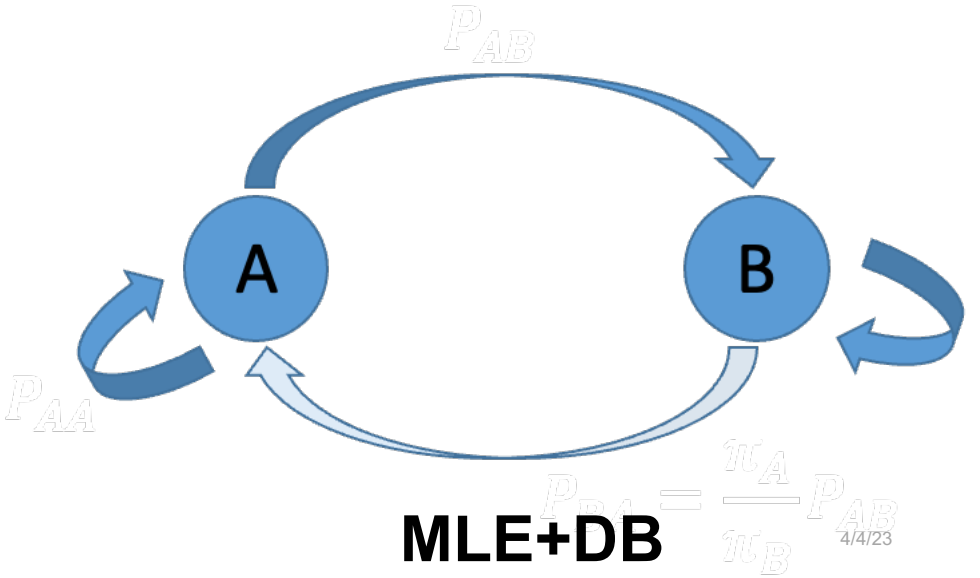
In MLE, these is **no escape** from newly discovered states. All segments scheduled in B until an escape is observed



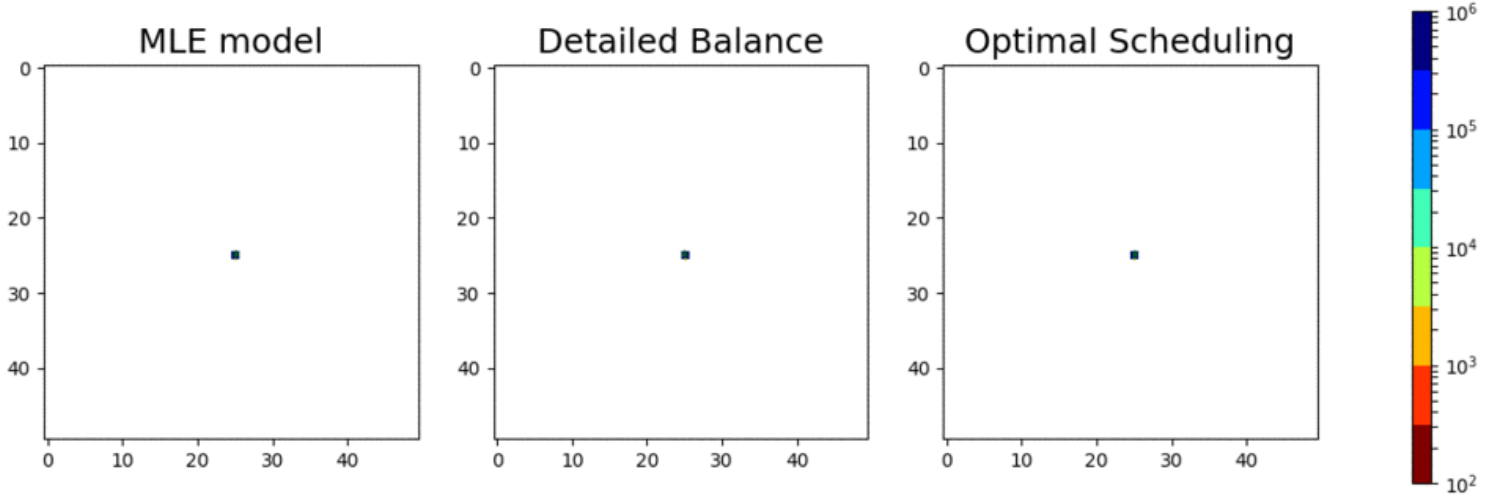
In MSE + DB, we impose reversibility since **the reverse pathway has to exist.**

$$P_{AB}\mu_A = P_{BA}\mu_B$$

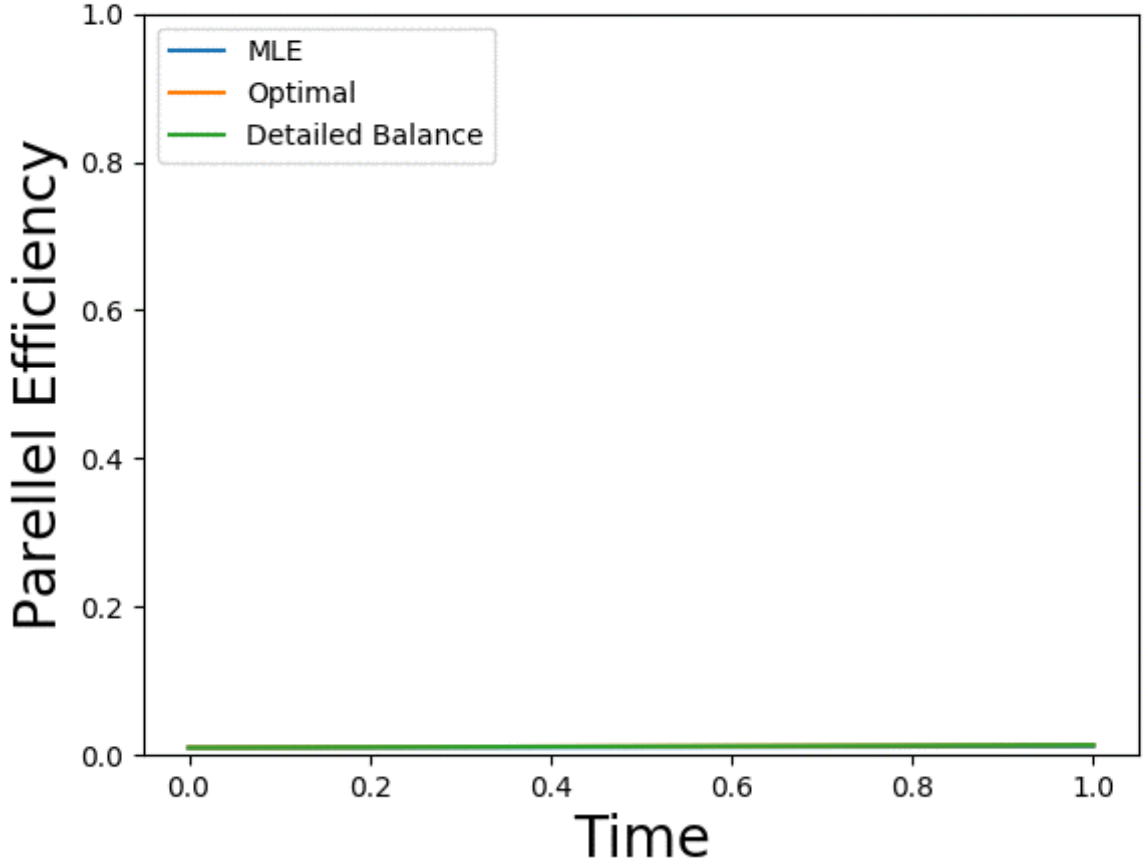
DB is not exact in general in this setting.



Detailed balance



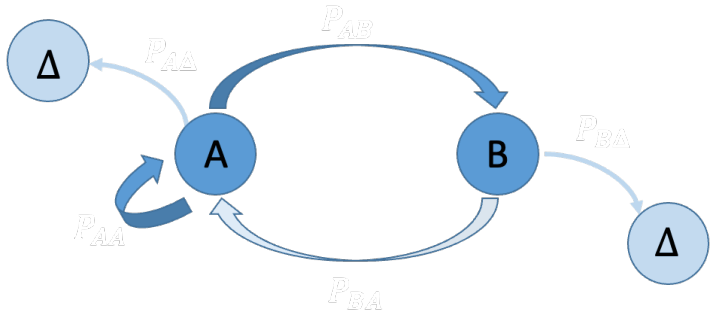
Parallel Efficiency



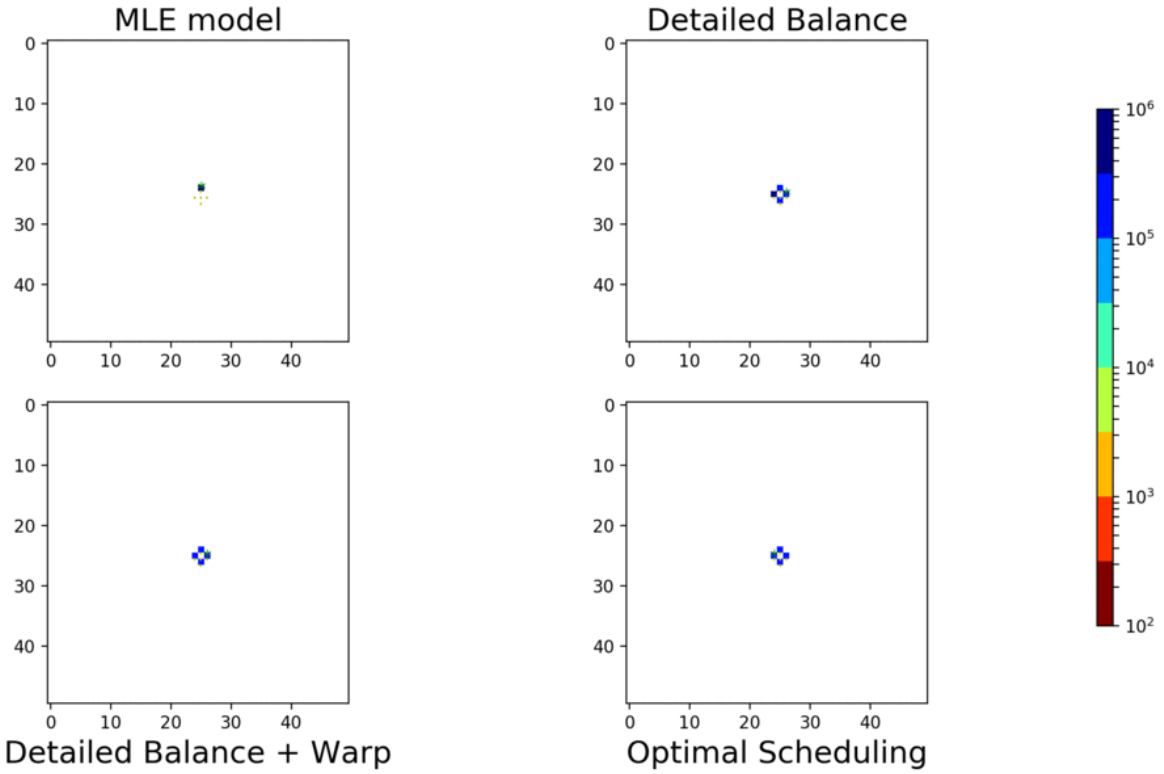
Accounting for uncertainty

- MLE+DB still produces allocations that are too local
- Caused by incompleteness: real trajectories would leave the known space and reenter in some other state

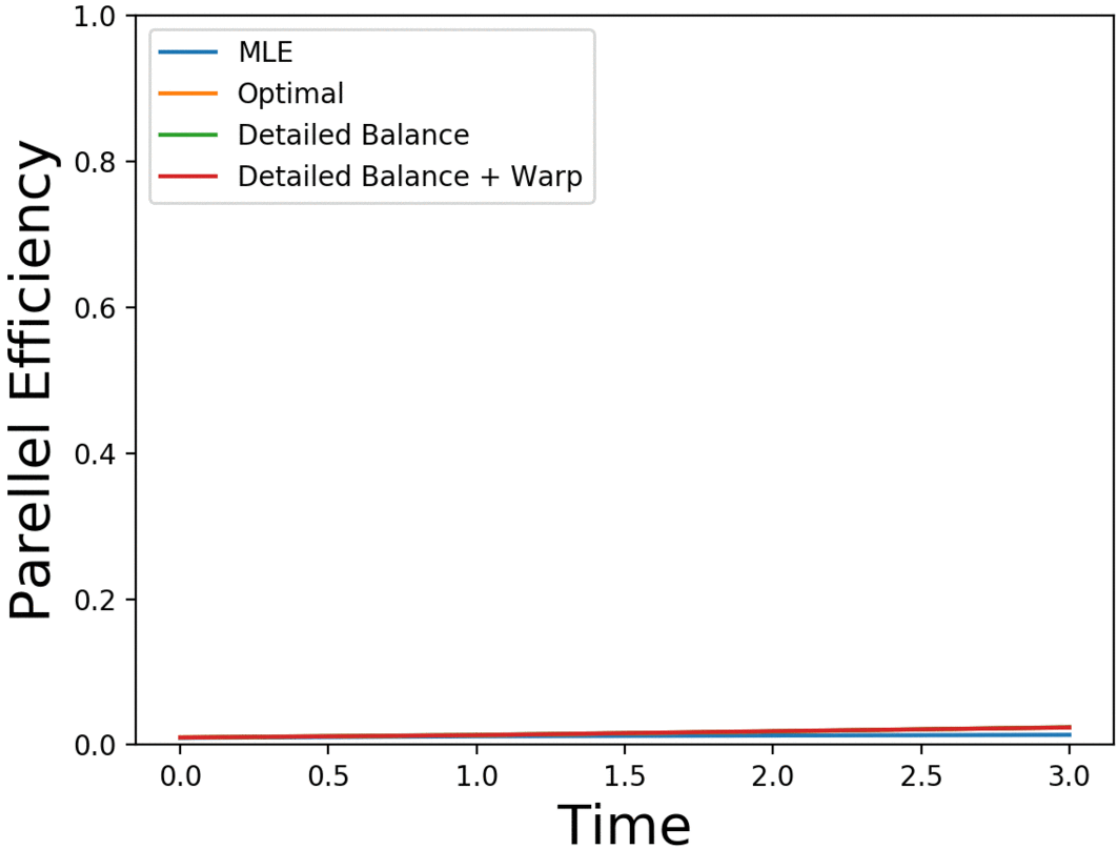
- Introduced heuristic **warp moves** to mimic this:
  - Bayesian formulation for observing a move that leaves the model
  - Upon leaving, random re-entry at any states connected by at most N hops from the departure state in the approximate model



Warping improves non-locality



Parallel Efficiency



Statistical oracle

- Discrete time Markov chain: probability that a segment that starts in state *A* ends in state *B*

- V1: **MLE** on generated segments (simple!)

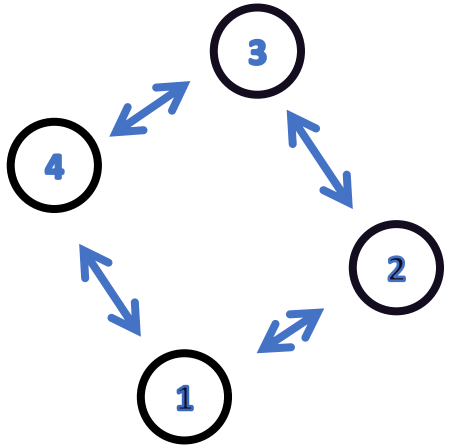
$$P_{AB} = \frac{Number\ of\ segments\ A \rightarrow B}{Number\ of\ segments\ generated\ in\ A}$$

- V2: MLE with **detailed balance constraint** (expensive)

$$P_{AB}\mu_A = P_{BA}\mu_B$$

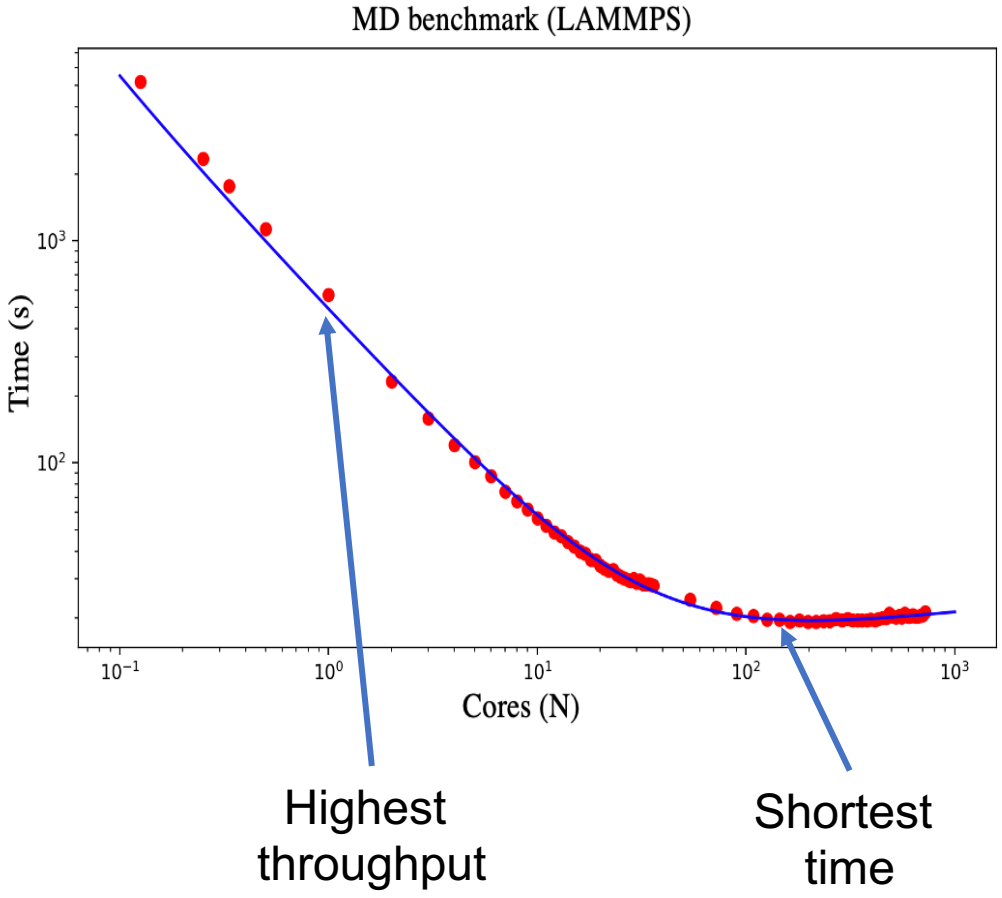
- V3: MLE + DB + **Warp**

- **V4: use ML for optimal scheduling (with the ECP/Exa-Learn project)**



Resource allocation

- Now we know where to run, but how to allocate resources to replicas?
  - Many cores/replica: **low MD efficiency**, **high ParSplice efficiency**
  - 1 core/replica: **high MD efficiency**, **low ParSplice efficiency**
- What is the **optimal allocation**?



Speculative resource allocation

- **Expected simulation throughput**:

$$R = \sum_{i}^{M} \frac{p_i}{T(w_i)}$$

- M: Number of tasks to be executed
- p_i: Probability that task i will be useful
- $T(w)$: Time to complete a task provided w resources
- w_i: Resources allocated to completing task i

Goal is to find the w_i that maximize *R*

[Garmon, Andrew, Vinay Ramakrishnaiah, and DP. *Parallel Computing* 112 (2022): 102936.]

Knowing the odds

- The direct utility of a segment is:
  - 1 if the segment is consumed before the end of the run
  - 0 if the segment is not consumed consumed before the end of the run

- The expected utility of an additional segment in state *j* is then

$$P_i\left(v_j > S_j \middle| H\right) \quad \forall j$$

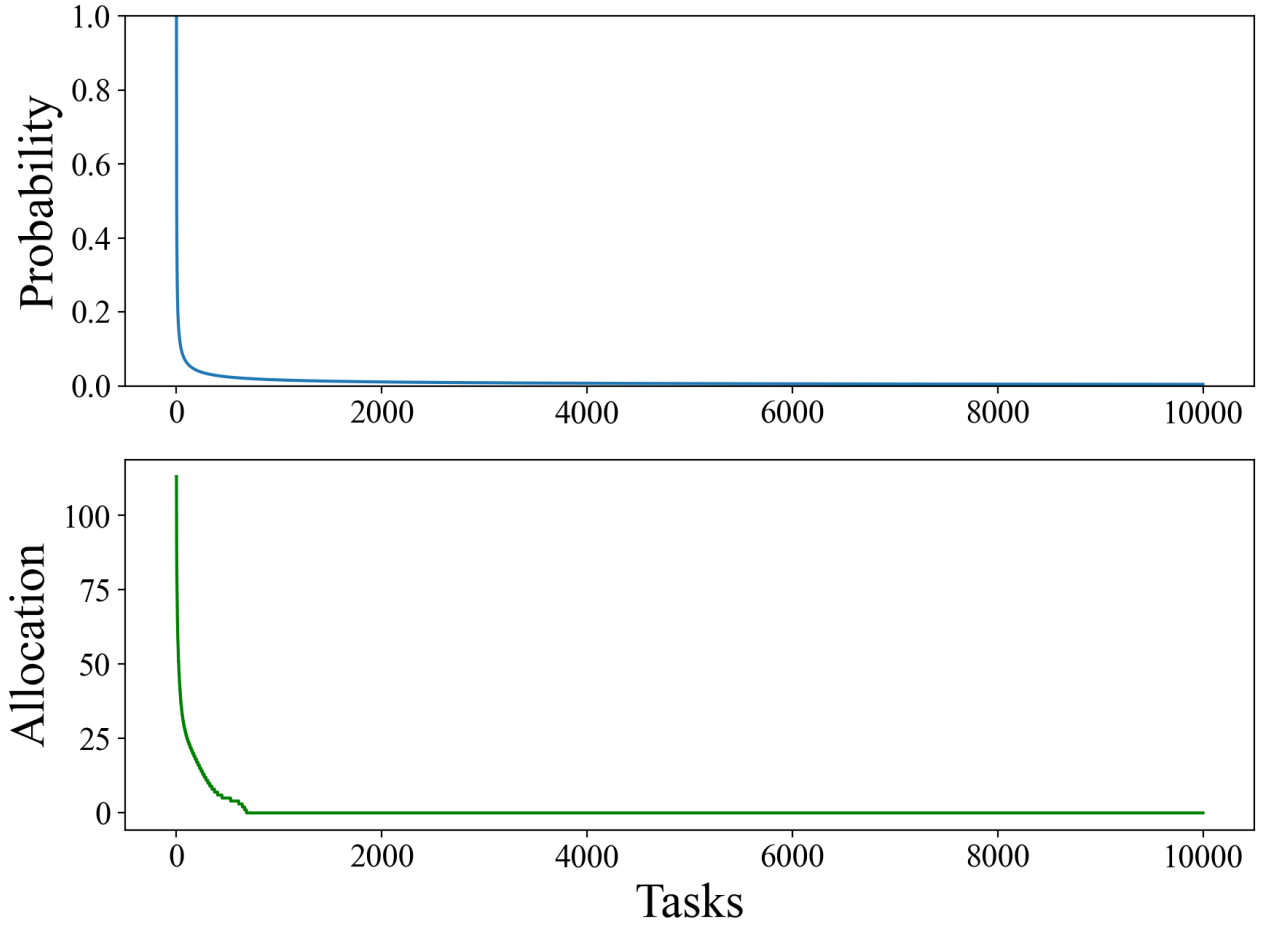
$$v_j\text{: } number\ of\ visits\ to\ state\ j$$

$$S_j\text{: } number\ of\ currently\ stored\ and\ pending\ segments$$

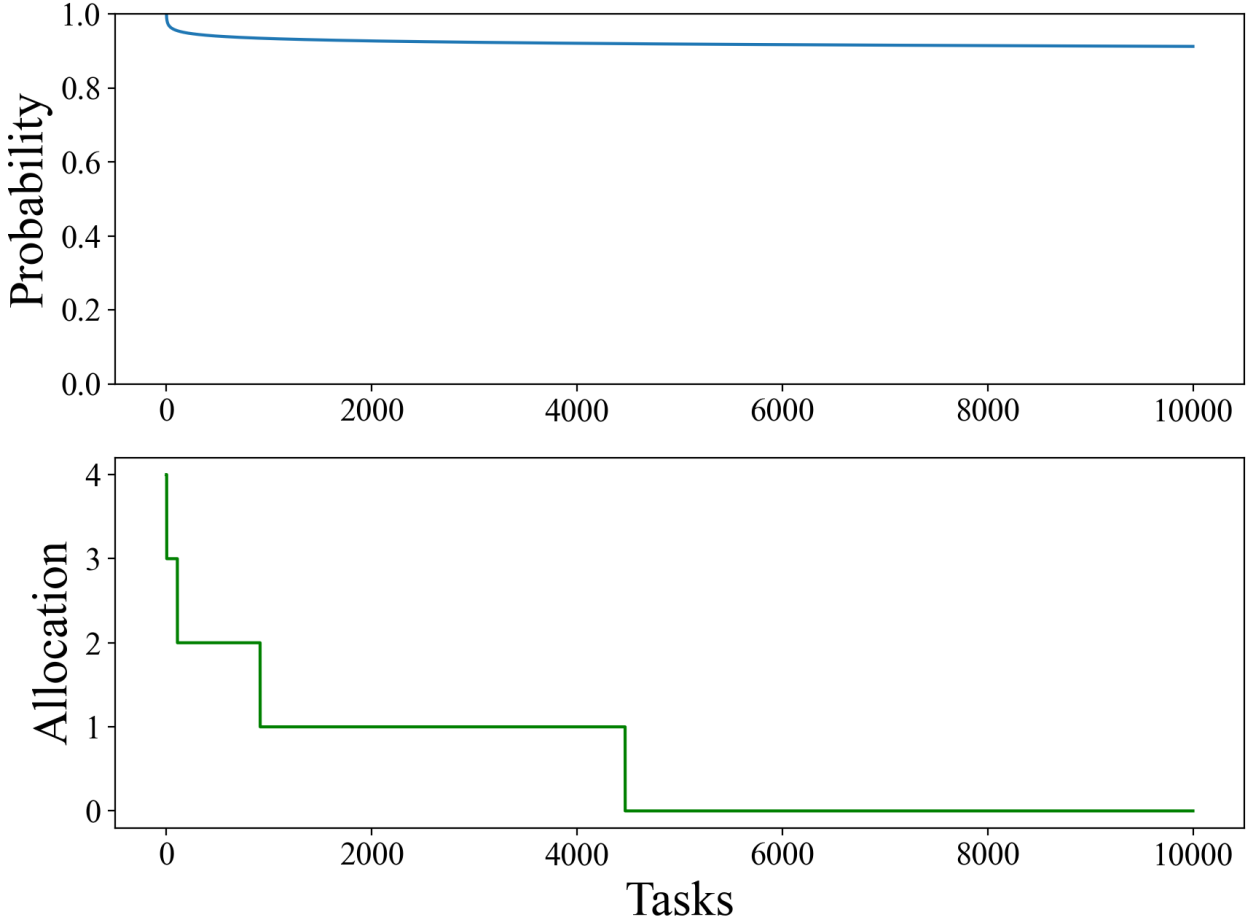
  Over some time horizon H (e.g., the end of the simulation)

- Can be estimated directly with MC theory or with **KMC**

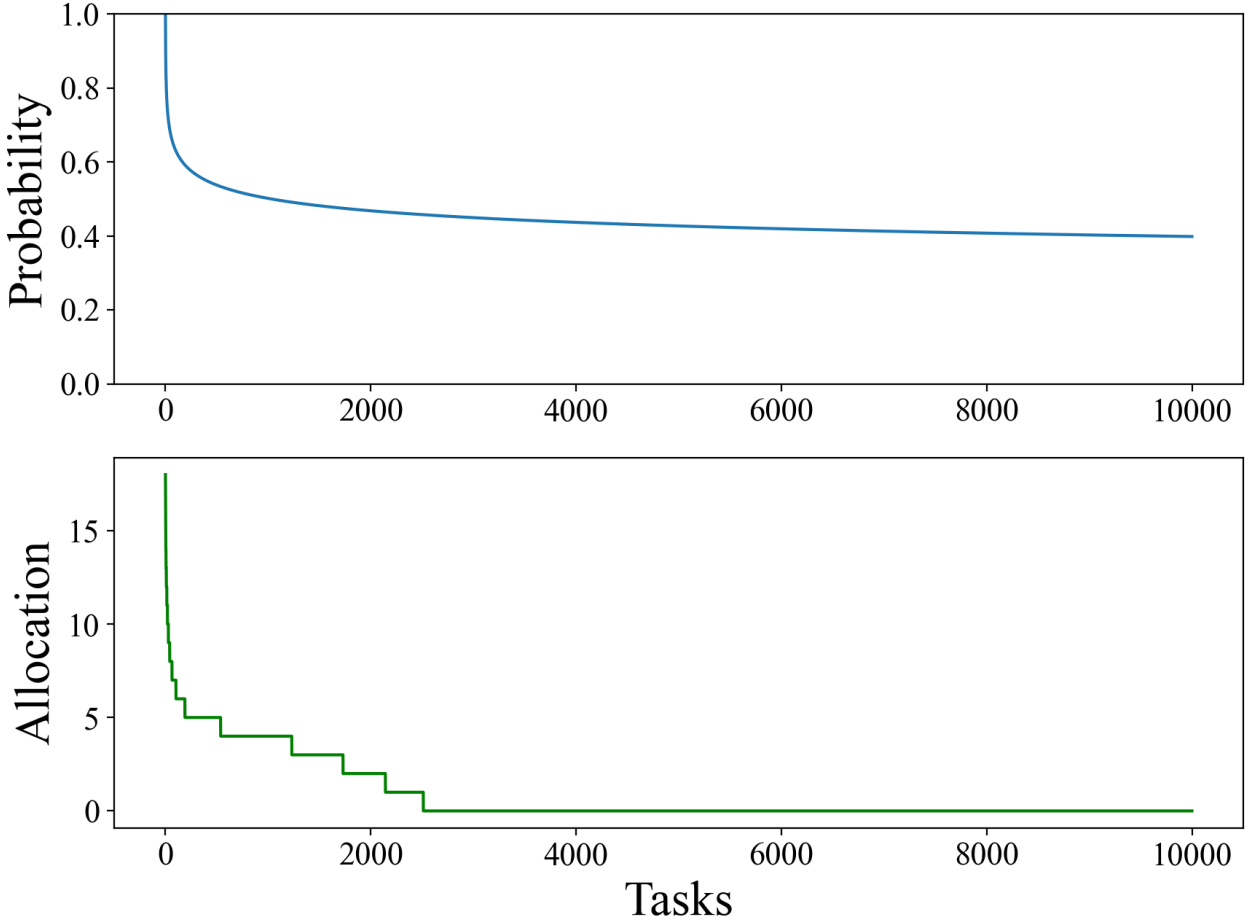
Optimal resource allocation



Optimal resource allocation

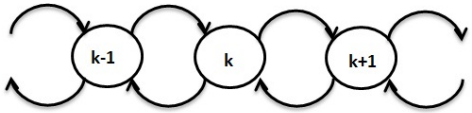


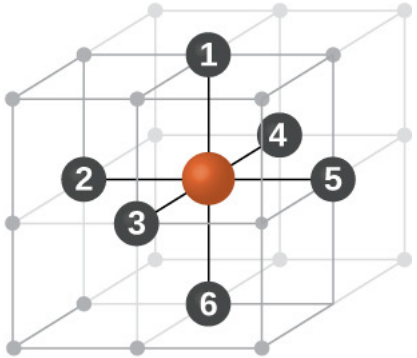
Optimal resource allocation

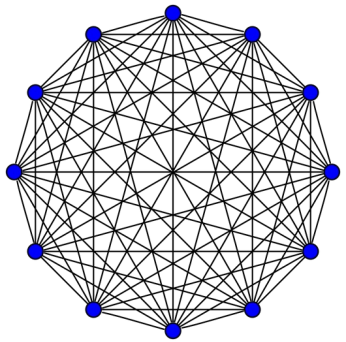


Simulator results

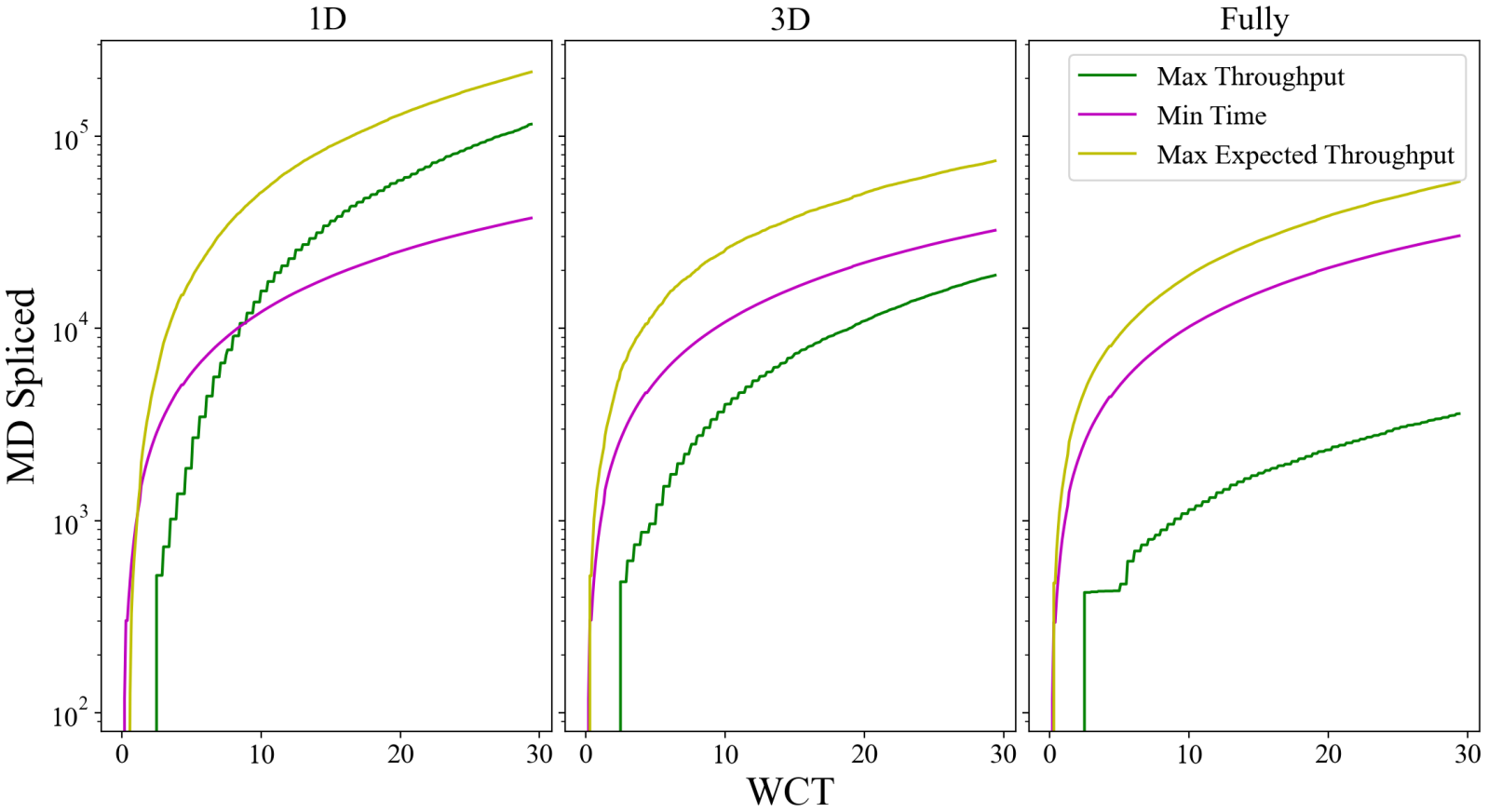
- Simulated optimal performance for 3 models:
  - 1D chain
  - 3D cubic lattice
  - Fully-connected graph
- $P_{ii} = 0.99, \quad P_{ij} = .01/N_j$
- 5000 CPUs
- Compared performance with two other strategies:
  - Max throughput: run as many as possible at the highest MD efficiency. **Good at high speculation confidence.**
  - Min time: run as many as possible at the highest MD speed. **Good at low speculation confidence.**







Simulator results



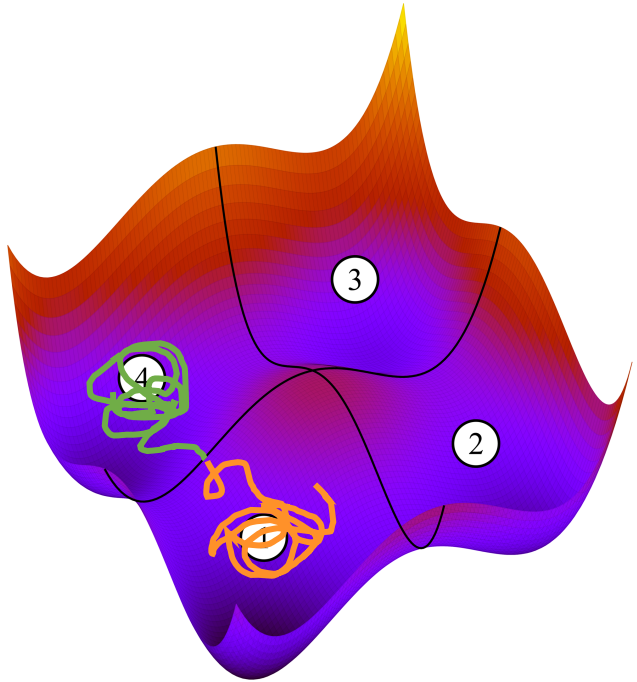
Practical implications

- Scheduling **heterogeneous** tasks is very complex
- Practical solution is to periodically **stop** all tasks **and restart** them with optimal resources. Easy to do with MD.
- We also observe that close-to-optimal **uniform** allocation almost always exists (>90% of peak). Much easier to deal with in practice!
- However, optimal uniform resource allocation can also change dramatically in time. Require constant adaptation during the run.

Mathematical Description of Rare Event Dynamics

Discretization of continuous dynamics

- The ParSplice formalism maps complex continuous dynamics into a simple, **arbitrarily accurate,** discrete framework
- Can it inform the development of accurate discrete state models?

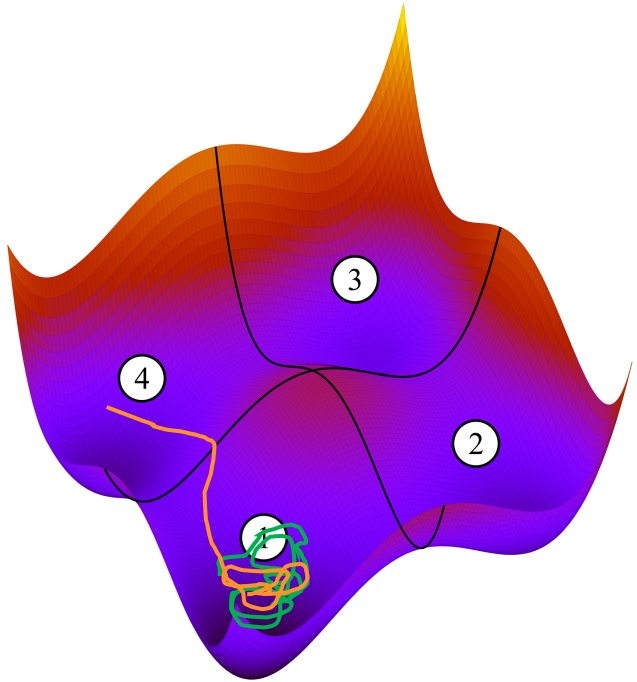


- Usual mapping is based on domains in configurations space
- Discrete model becomes a CTMC in the limit $(\lambda_2 - \lambda_1) \rightarrow \infty$ for all states. **This limit is often approached but never exactly reached.**
- No clear picture away from this limit

[T. Lelièvre, Handbook of Materials Modeling: Methods: Theory and Modeling, 773]

Markov Renewal Process representation

- ParSplice inspired mapping:
  - The “color” of a trajectory is the color of the last state it spent t_c in
- The color encodes the last domain the trajectory reached the QSD in.
- What is the appropriate representation of the color-to-color dynamics?



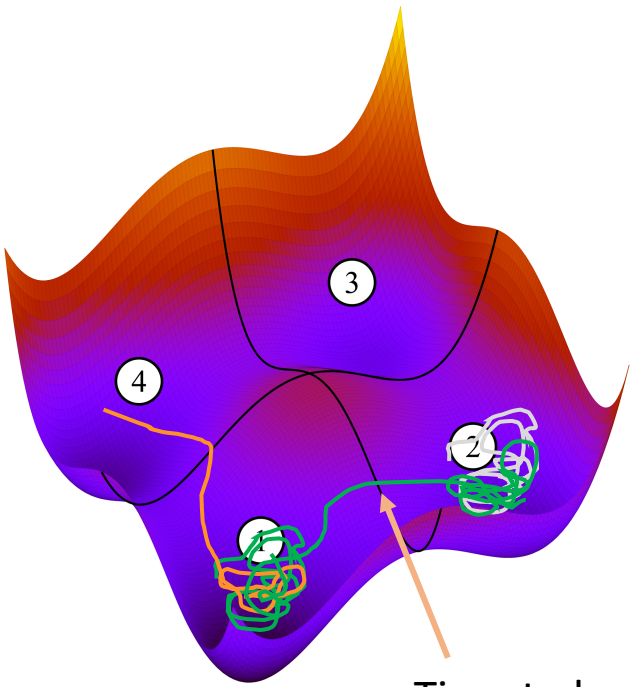
Markov Renewal Process representation

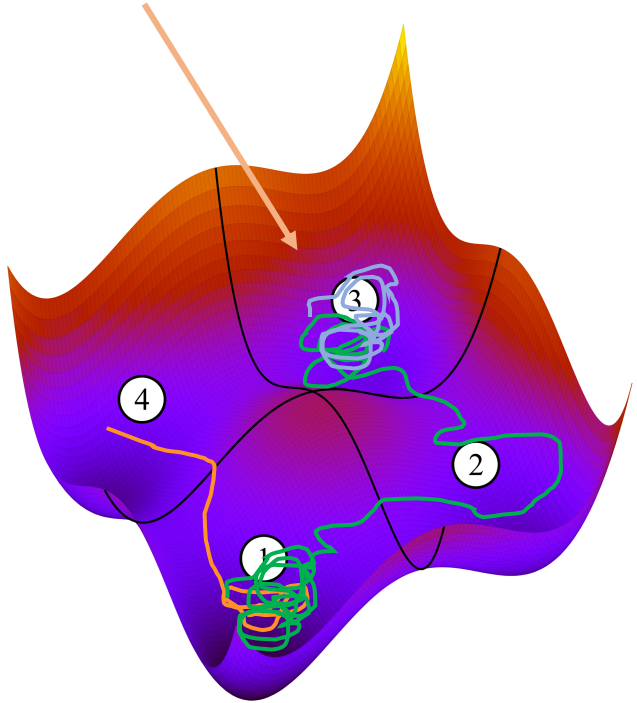
- Color changes when trajectory reaches QSD in a new state

- From the properties of the QSD:
  - Probability of next color can only depend on current color
  - Distribution of time to next color change cannot depend on previous colors
  - Distribution of time to next color change cannot depend on previous change times
  - Distribution of time to next color change can depend on next color

Agarwal, Gnanakaran, Hengartner, Voter, Perez, arXiv:2008.11623

Time to settle in new state and change color can depend on new color





Time to leave the state independent of past and future color

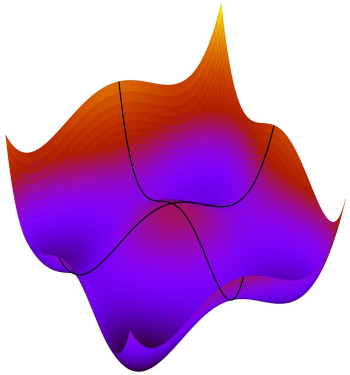
Markov Renewal Process

Color-to-color dynamics is described by a Markov Renewal Process*

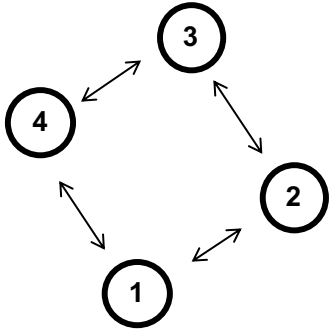
$$P(c_{n+1}, t_{n+1} < T | \, history \,) = p_{c_{n+1},c_n} \, F_{c_{n+1},c_n}(T - t_n)$$

for *any state definition*

* Up to an exponentially small error in t_c



QSD-to-QSD factorization



Continuous Trajectory

QSD-to-QSD factorization

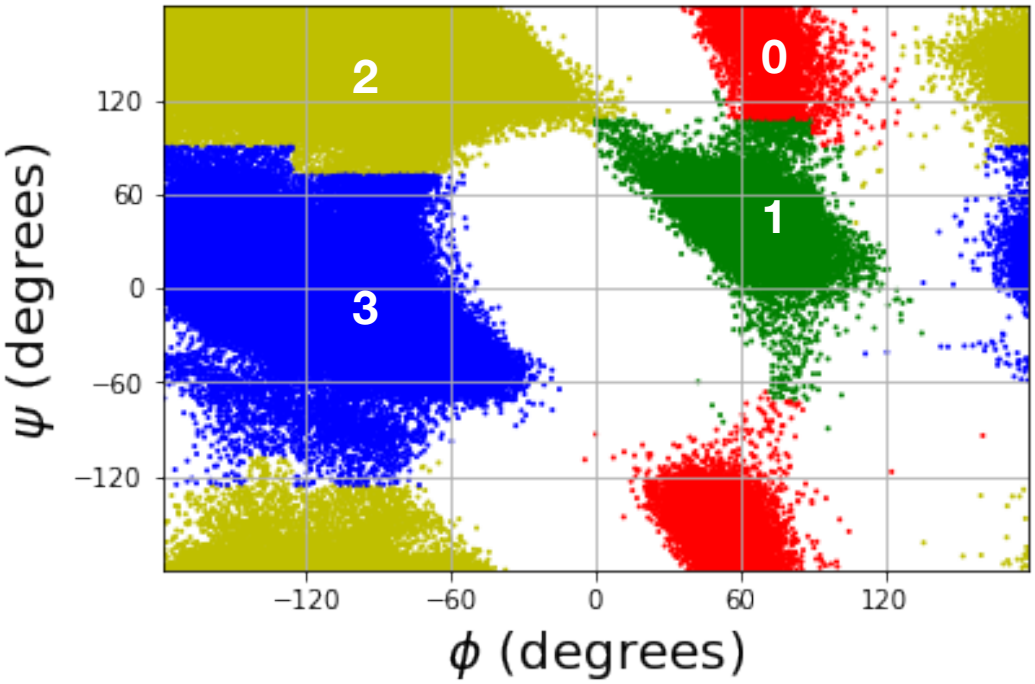
Markov renewal process

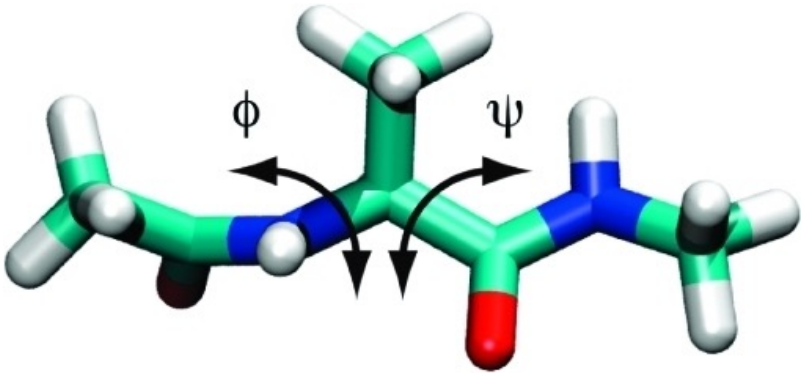
Fokker-Planck equation

QSD-to-QSD factorization

Renewal equations

Alanine dipeptide



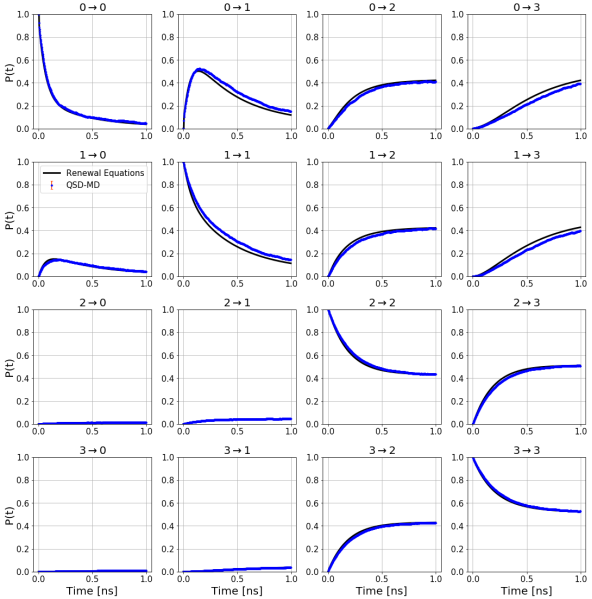


Carefully defined domains using PCCA

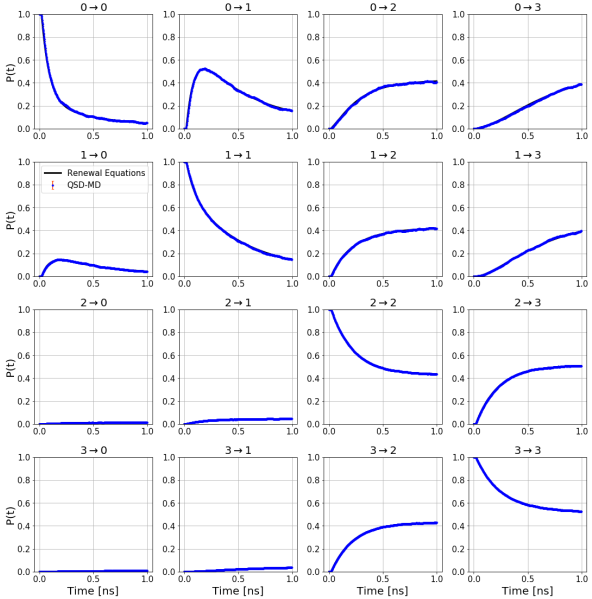
Alanine dipeptide

Direct MD

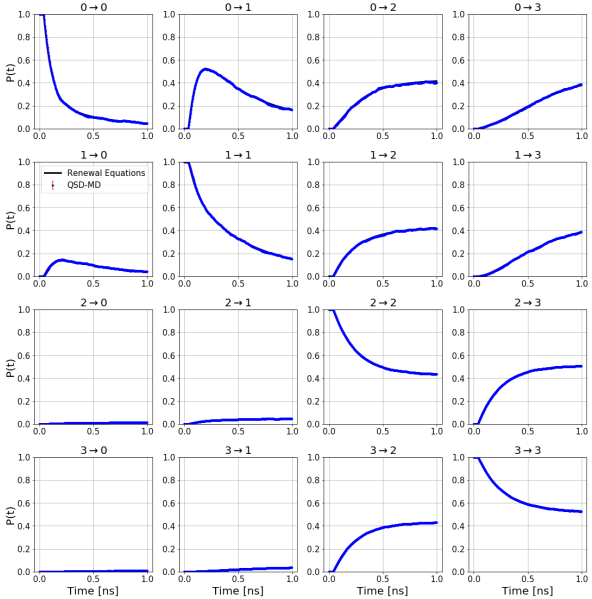
Renewal equations



t_c = 2 ps

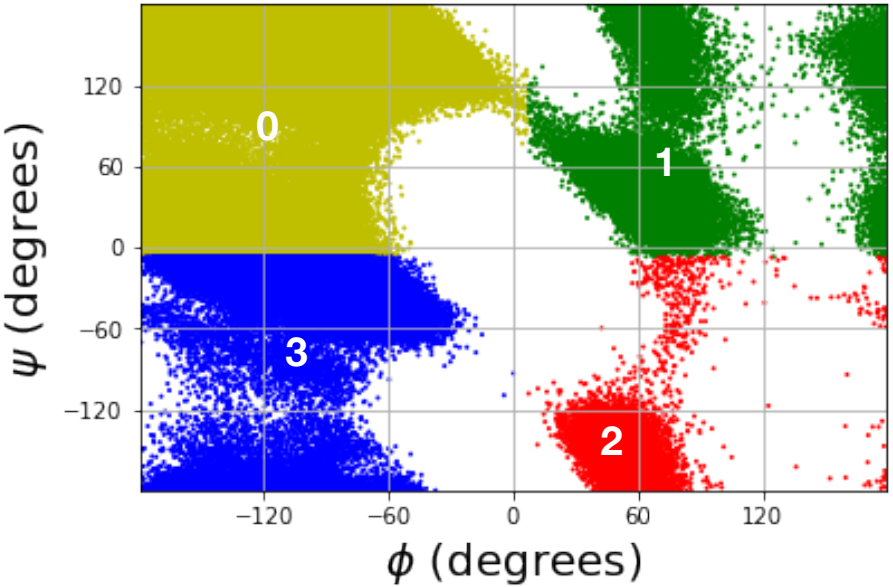


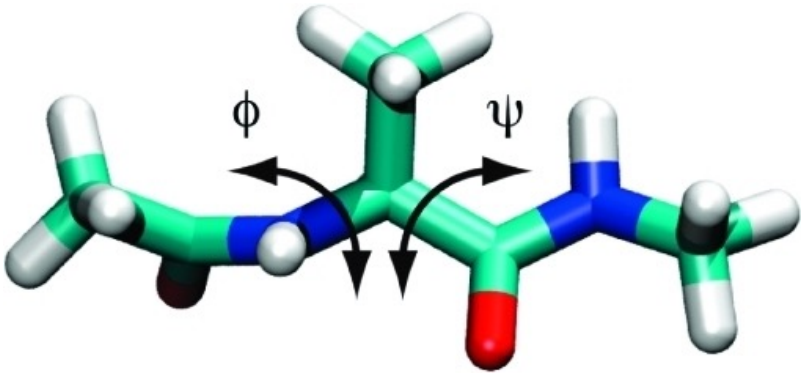
t_c = 20 ps



t_c = 40 ps

Alanine dipeptide



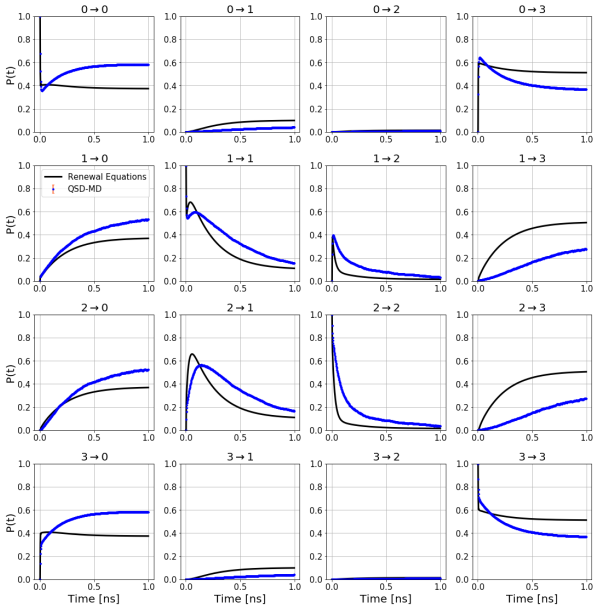


Intentionally poorly defined states

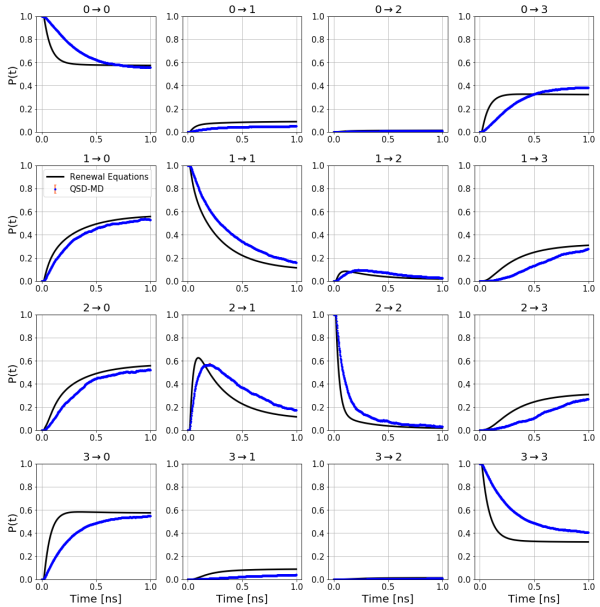
Alanine dipeptide

Direct MD

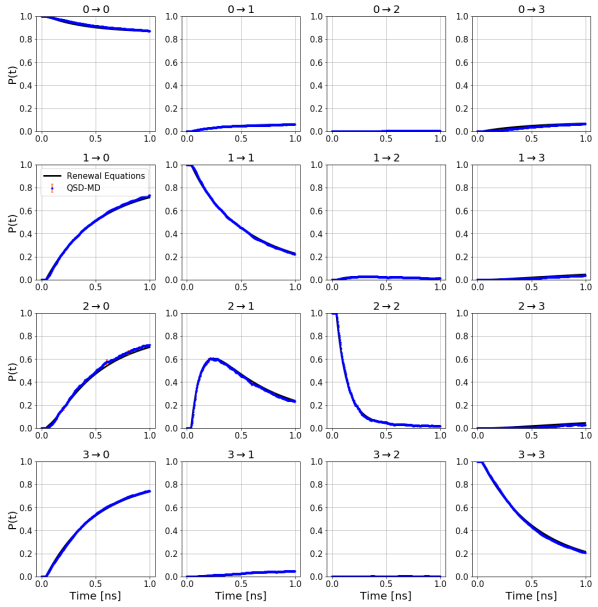
Renewal equations



t_c = 2 ps



t_c = 20 ps

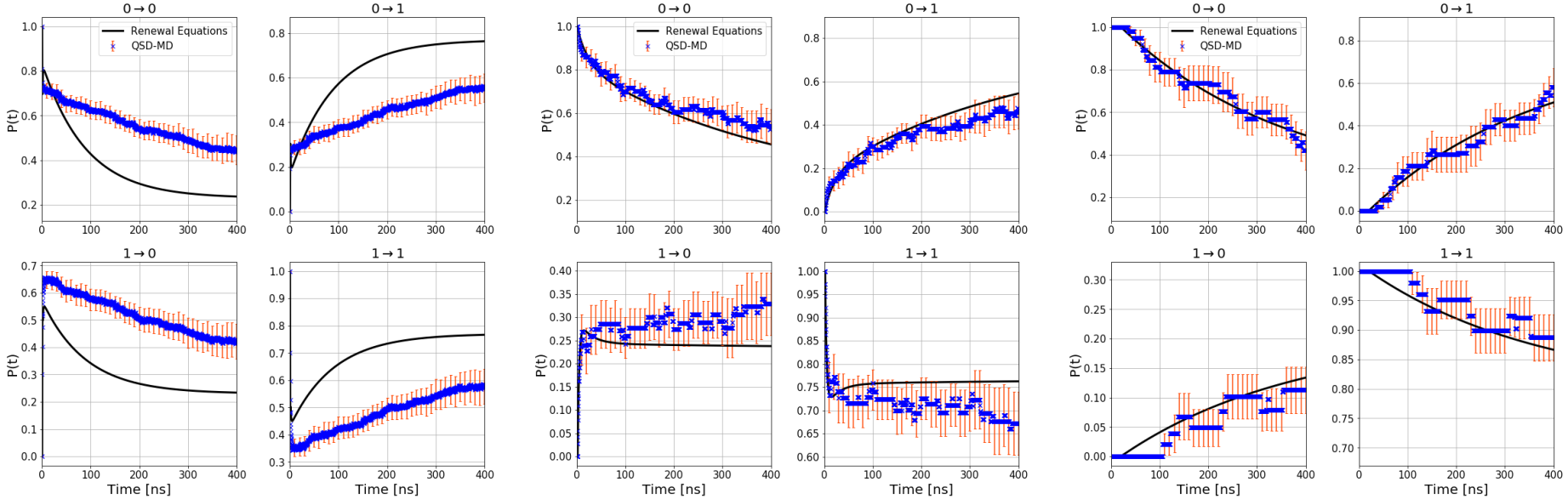


t_c = 40 ps

Villin headpiece

Direct MD

Renewal equations



t_c = 2 ps

t_c = 2 ns

t_c = 20 ns

Markov Renewal Process

- Not the only discretization scheme (CTMC, Hidden Markov Model, …)
- To our knowledge, simplest scheme that provides arbitrary accuracy *for any state definition*
- Easy to sample new trajectories from a MRP (modified BKL)

- **Caveat:**
  - not very informative if dynamics are not metastable and/or states are very badly defined. Leads to very long jumps.

- Next step: provide efficient numerical schemes to parameterize the MRP (ongoing work with D. Aristoff)

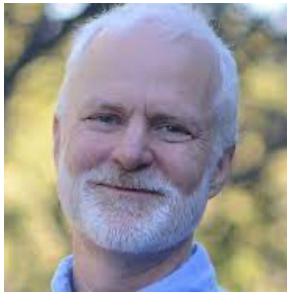
Agarwal, Gnanakaran, Hengartner, Voter, DP, arXiv:2008.11623

Conclusion

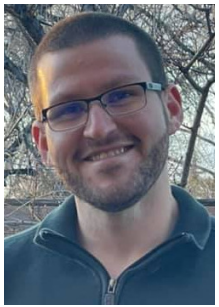
- MD is extremely powerful, but has a severe timescale limitations that cannot be cured by brute-force alone, even with exascale computing
- By leveraging insights from the theory of QSD, one can design rigorous parallel-in-time techniques that dramatically extend simulation times
- Progress in applied math, computer science, and domain science, was essential to address this problem.
- Careful resource allocation is especially important in “difficult” cases, and will be essential for challenging high efficiency simulations

Collaborators/Acknowledgements

- **Method development:** Arthur Voter, Andrew Garmon
- **Mathematics:** Tony Lelièvre, Claude Le Bris, Mitch Luskin, David Aristoff
- **Code:** The EXAALT ECP team
- **Funding:** DOE BES, ECP; LANL LDRD
- **Computing:** LANL IC, NERSC





Los Alamos NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT