Long-Time Atomistic Simulations with Parallel Trajectory Splicing

Danny Perez

Theoretical Division T-1

Los Alamos NATIONAL LABORATORY EST.1943

NNSA National Nuclear Security Administration

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Molecular Dynamics

- Molecular Dynamics, the numerical integration of atomistic equations of motion, is the gold standard to investigate the dynamical evolution of atomistic systems
- Essentially a numerical experiment
- MD can be used to compute “any” atomistic dynamic or thermodynamic property

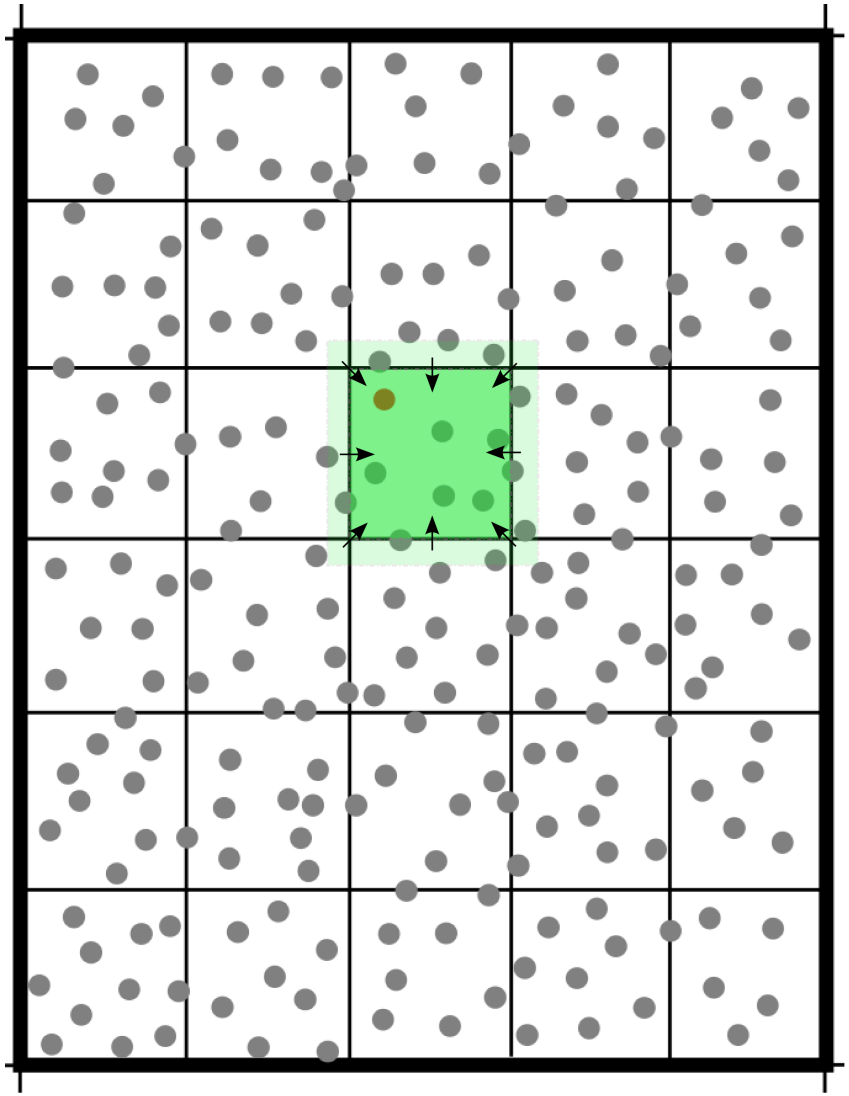
MD is extremely widely used: about 3M hits on Google scholar…

Molecular Dynamics

- MD is computationally expensive
  - Every atom has to be resolved
  - The fastest vibrational motion has to be resolved (dt ~ 1 fs)
- Limits size and time scales that can be directly simulated
- Parallel computers help, but only partially address the problem

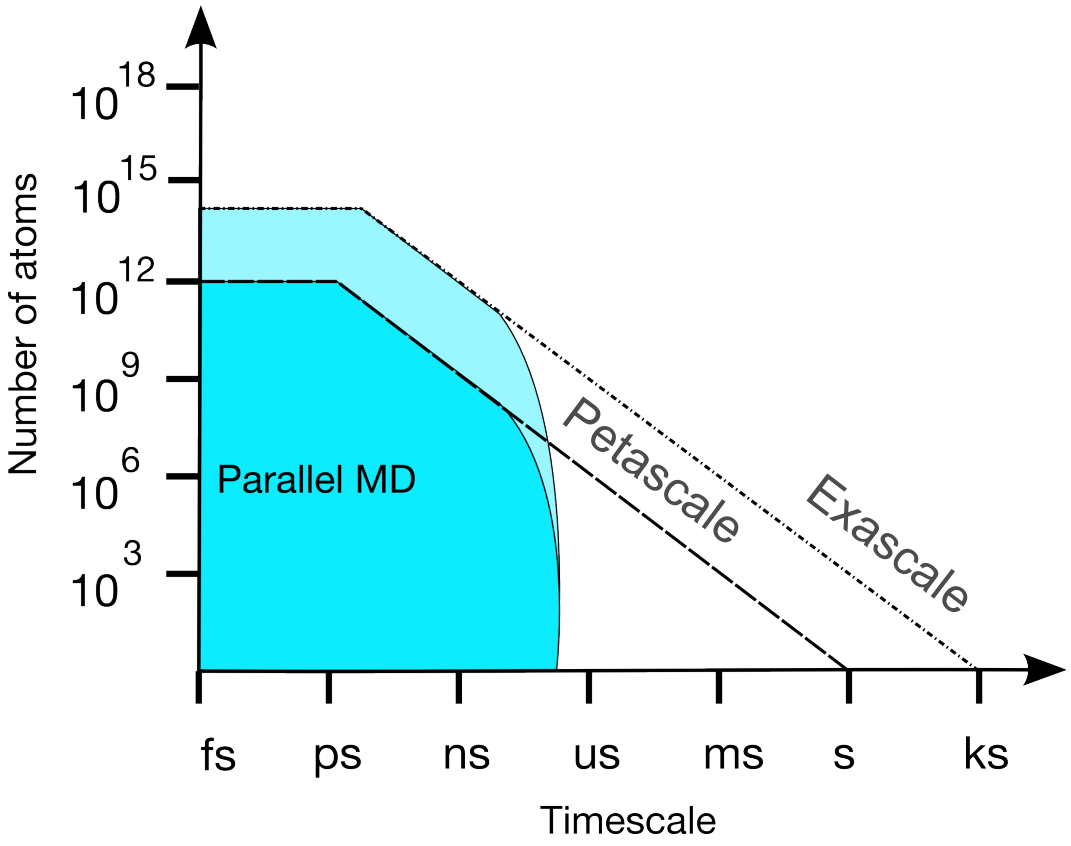
Molecular Dynamics on Parallel Platforms

- Spatial decomposition weak-scales very well. Trillions of atoms can be simulated.
- Strong-scaling is comparatively poor. Temporal reach of MD is limited to μs or less.
- This will remain so for the foreseeable future.



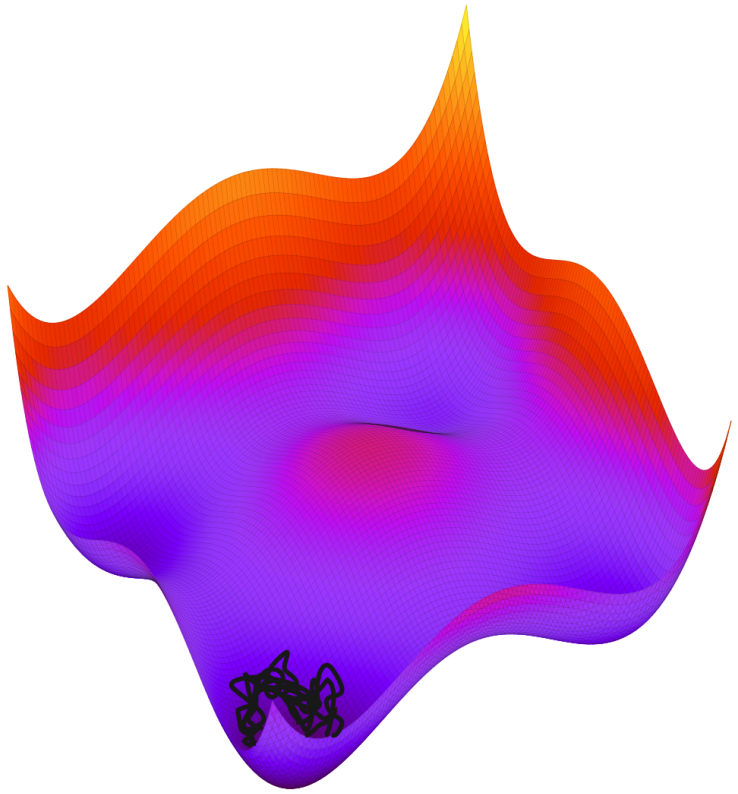
Molecular Dynamics on Parallel Platforms

- Spatial decomposition weak-scales very well. Trillions of atoms can be simulated.
- Strong-scaling is comparatively poor. Temporal reach of MD is limited to μs or less.
- This will remain so for the foreseeable future.



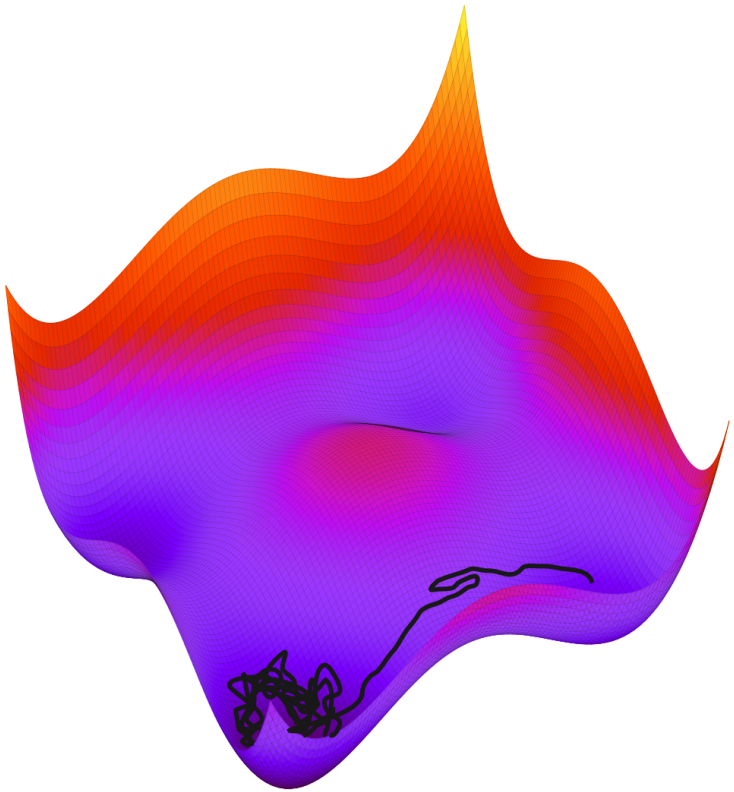
Molecular Dynamics

- The timescale limitation is especially crippling for systems that evolve through rare, activated events.
- These systems are characterized by long periods of vibrational dynamics…



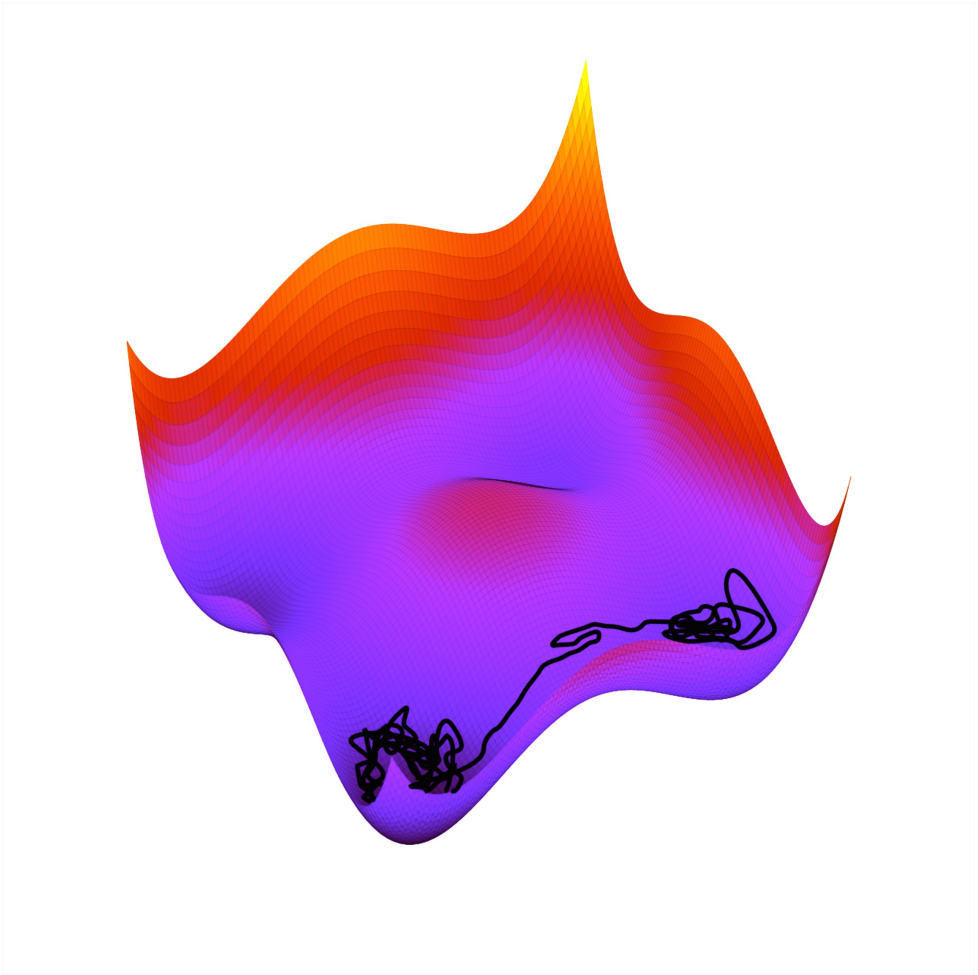
Molecular Dynamics

- …punctuated by rare, but fast, transitions…



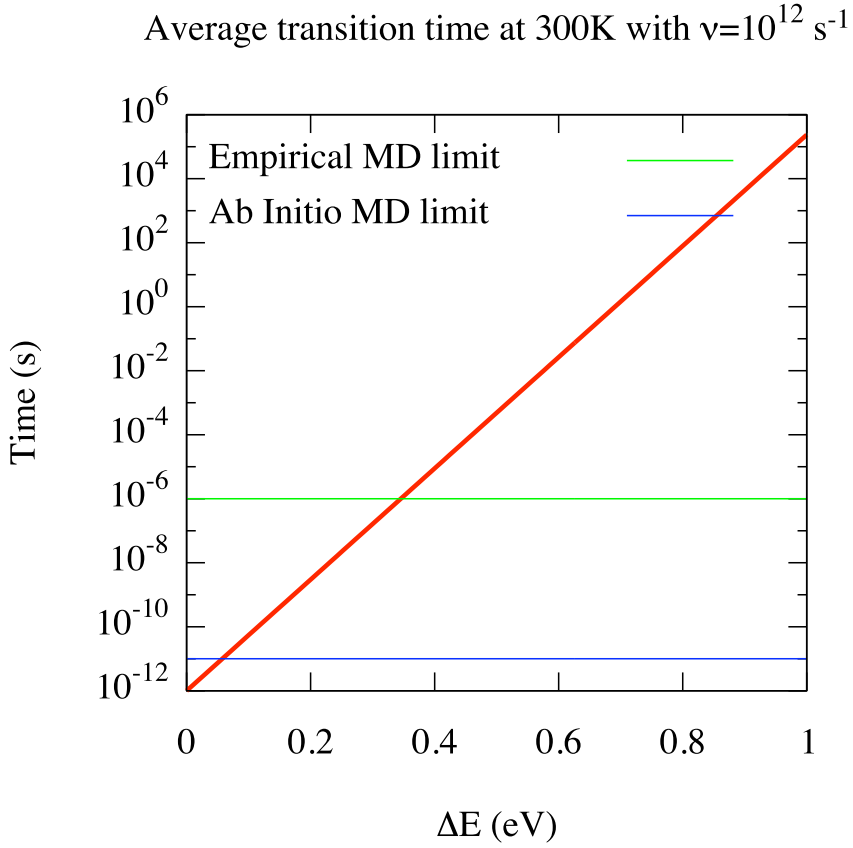
Molecular Dynamics

- …to a another long-lived state…



The timescale problem of MD

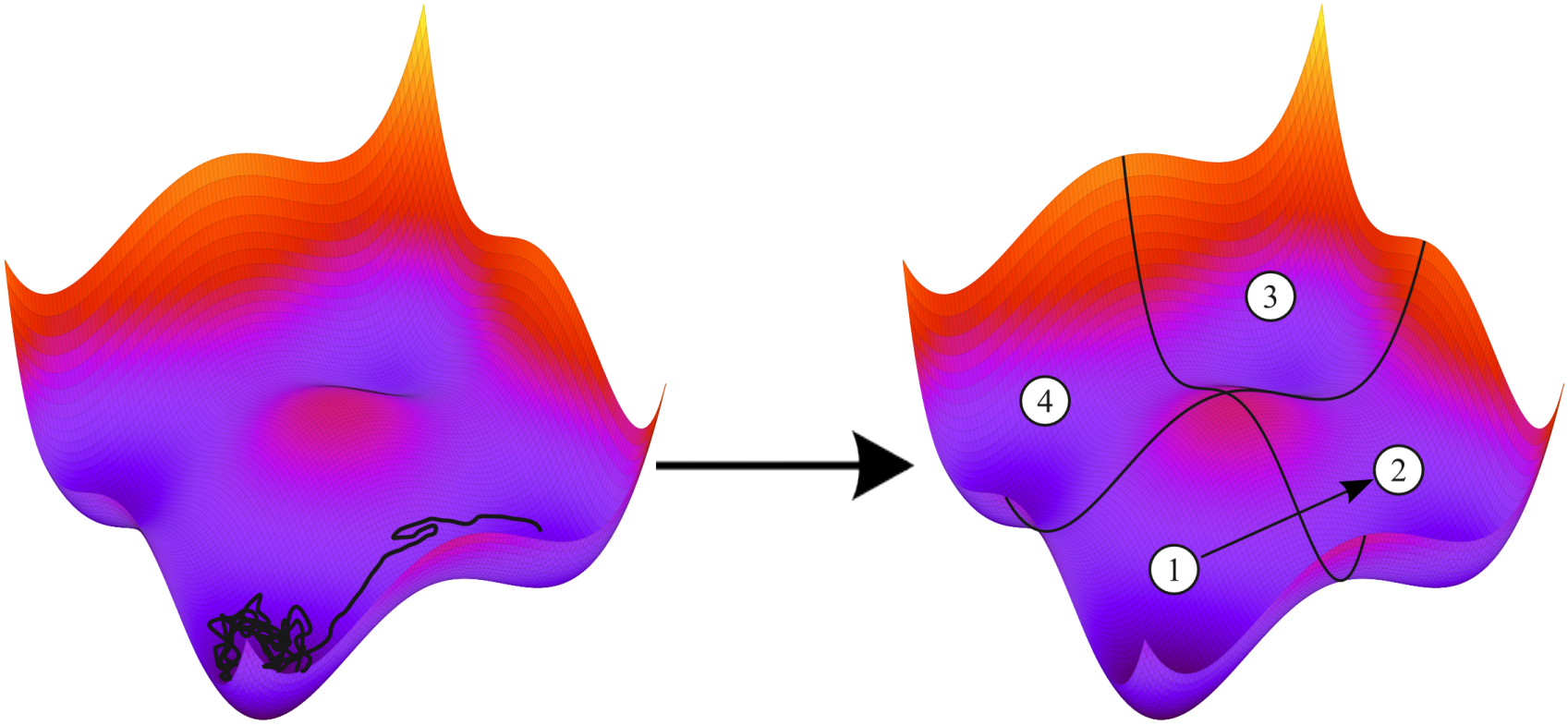
- If you cannot afford to some number of transitions, you learn ***nothing*** about long time evolution of the system



Accelerated Molecular Dynamics

- Accelerated MD methods (A.F. Voter *et al*, LANL) are designed to address this limitation
- The goal is to mimic MD, but at a coarser level: to obtain long, statistically correct, state-to-state trajectories
- The basic idea is to let MD trajectories find appropriate transitions but to coax them into doing so faster
- Use statistical mechanics concepts (primarily rate theories) to make sure trajectories are unbiased

State-to-state dynamics



Today

- Discuss a new strategy to harness parallel computers to reach long timescales: **Parallel Trajectory Splicing** [Perez, Cubuk, Waterland, Kaxiras, Voter, JCTC 12, 18 (2016)]

- The idea is to parallelize the dynamics in time, instead of in space. We also parallelize through speculative execution.

- Generalization of the Parallel Replica Dynamics method [Voter, PRB 57, 13985 (1998)]

Problem Statement

How can we use parallel computers to generate proper state-to-state trajectories over very long times?

Strategy:

1. Factorize the problem into independent tasks

2. Decide which task should be executed at what time

Trajectory Factorization

- Integrating MD trajectories is inherently serial: you have to finish a timestep before you start the next one
- That makes parallel-in-time methods challenging if one insists on continuity in phase space
- If one only wants state-to-state trajectories, things become much easier

Trajectory Factorization

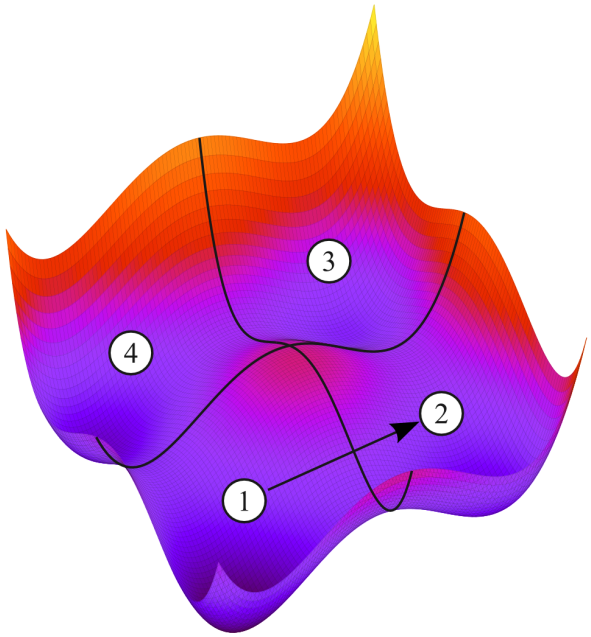
Markovian limit:

$$P(X_{t+1}=C \mid X_t=A, X_{t-1}=B, \ldots) = P(X_{t+1}=C \mid X_t=A)$$

- Future only depends on present

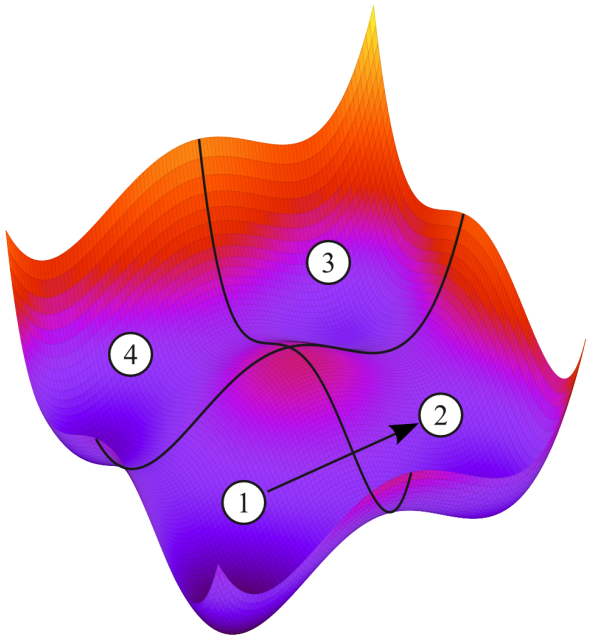
- All trajectories that begin in a given state are proper futures of a trajectory that ends in this state.

- A long Markovian trajectory can be constructed by splicing independently generated trajectories end-to-end.



Trajectory Splicing

A long Markovian trajectory can be constructed by **splicing independently generated trajectories end-to-end.**



Trajectory Factorization

MD state-to-state dynamics is *not* strictly Markovian. What then?

Consider a Fokker-Planck operator with absorbing boundary conditions around a state. Let $\{-\lambda\}$ be its eigenvalues.

Define :

λ_1 : quasi-stationary inter-state transition rate

λ_2 : slowest intra-state relaxation rate

Then :

First escape **becomes** (approximatively) Markovian after $\tau_c > 1/(\lambda_2 - \lambda_1)$

[Le Bris, Lelievre, Luskin, and Perez, MCMA 18, 119 (2012)]

Trajectory factorization: Physical interpretation

- Conditional on not escaping, a trajectory gradually looses its memory of how it entered a state

- After a time $\tau_c > 1/(\lambda_2 - \lambda_1)$, the end of the trajectory is (approximately) a sample from the **unique** Quasi-Stationary Distribution (QSD) of that state

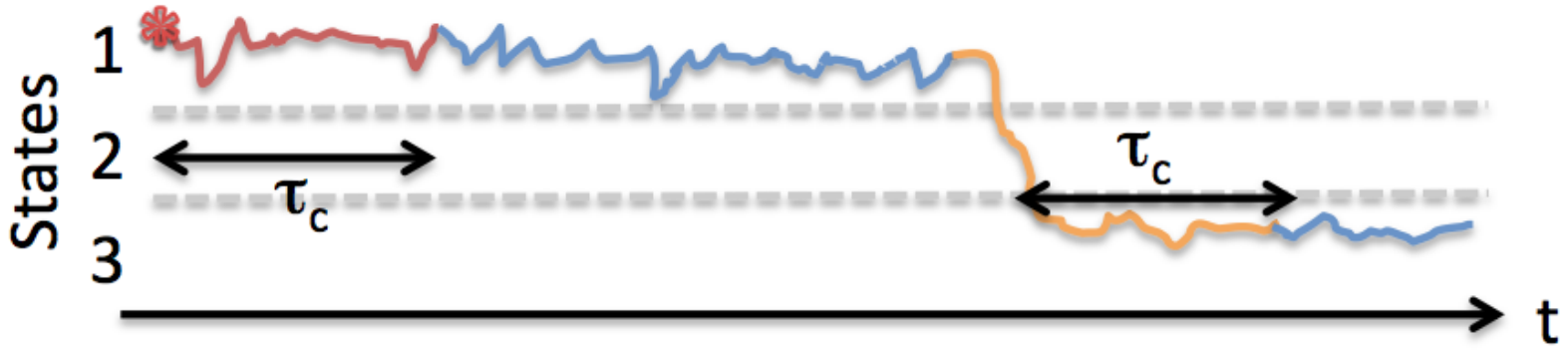
- First escape from the QSD is a Markovian process

- Samples from the QSD are equivalent starting points w.r.t future state-to-state evolution

[Le Bris, Lelievre, Luskin, and Perez, MCMA 18, 119 (2012)]

Trajectory splicing

You can splice sections of trajectories as long as their ends are samples for the QSD in their respective state.



Segment: section of trajectory that remained **within the same state** for at least τ_c before its beginning and before its end.

Trajectory splicing

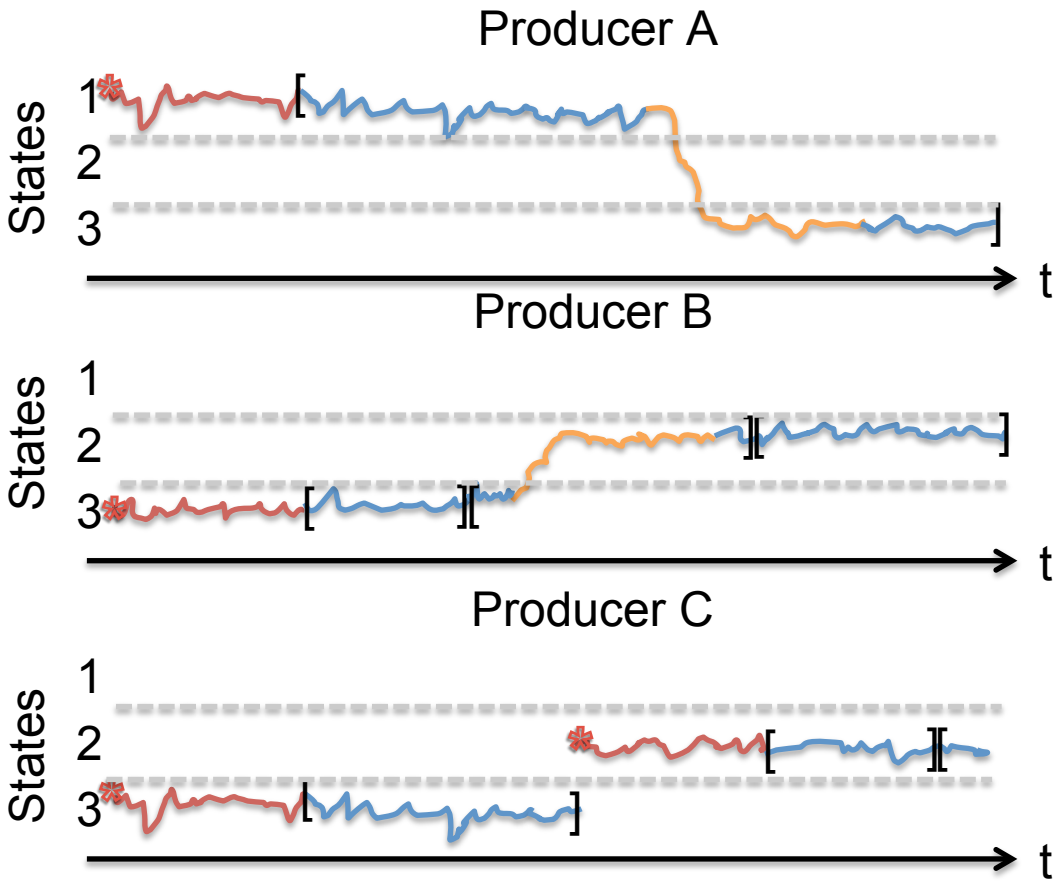
Non-Markovian Trajectory splicing:

With error $O(\exp[-(\lambda_2 - \lambda_1)\tau_c])$, a trajectory can be constructed by **splicing independently generated segments end-to-end**.

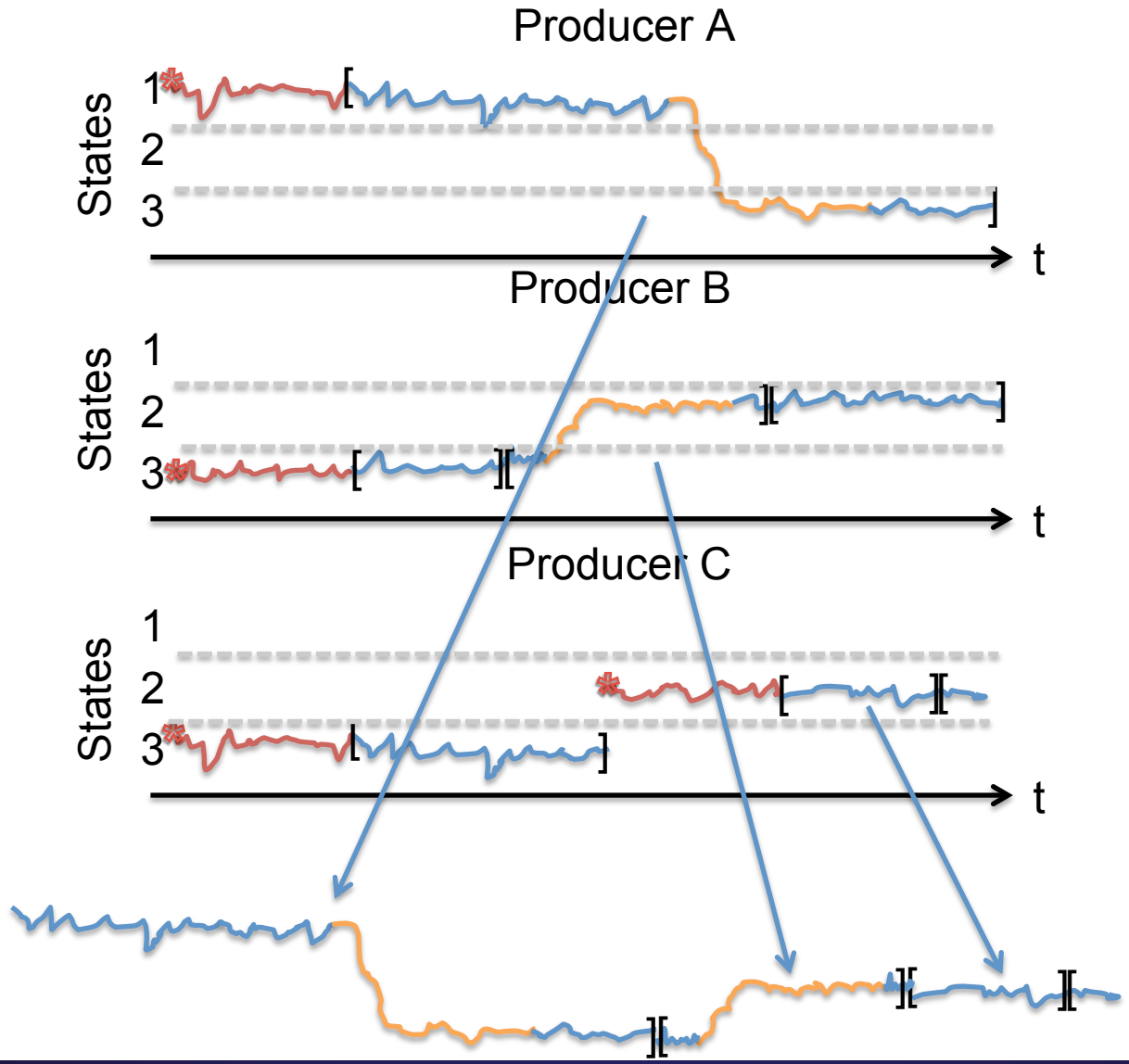
Segments are the basic work units in ParSplice. They can be generated concurrently and stored in a database until they are spliced into the trajectory.

[Perez, Cubuk, Waterland, Kaxiras, Voter, JCTC 12, 18 (2016)]

Trajectory splicing



Trajectory splicing



Task execution

Which segments should we generate? State space is much too vast to be covered. Decisions have to be made **at run-time.**

–Conservative approach (ParRep): run short segments at the **current end** of the trajectory**. Efficient only if $N < \tau_{esc} / \tau_c$.**

–Speculative approach: run short segments at the **expected end** of the trajectory *when the segment is complete*. **Perfectly scalable given an oracle.**

Efficiency of ParSplice

- We don't have an oracle, but trajectories are often statistically predictable, especially when trapped in a super-state

- Scalability is then limited by *super-state escape time*:

$$N \sim \tau^{ss}_{esc} / \tau_c >> \tau^{i}_{esc} / \tau_c$$

- If prediction of the future is impossible and the trajectory never revisits states: ParSplice == ParRep.

Speculative execution

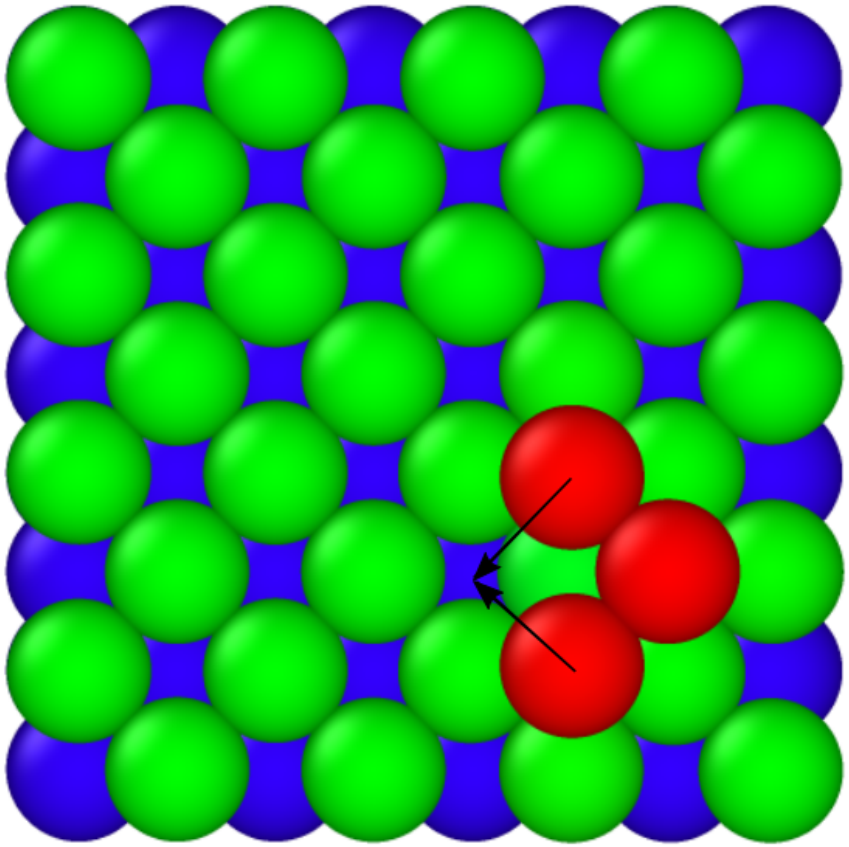
How to speculate?

- Build a Markovian model of state-to-state dynamics on-the-fly by analyzing completed segments.
- **Use this model to carry out meta-simulations, i.e., simulations of the future execution of the code.**
- Takes into account the expected behavior of the system and the internal state of the code (i.e., the content of segment database)

ParSplice procedure

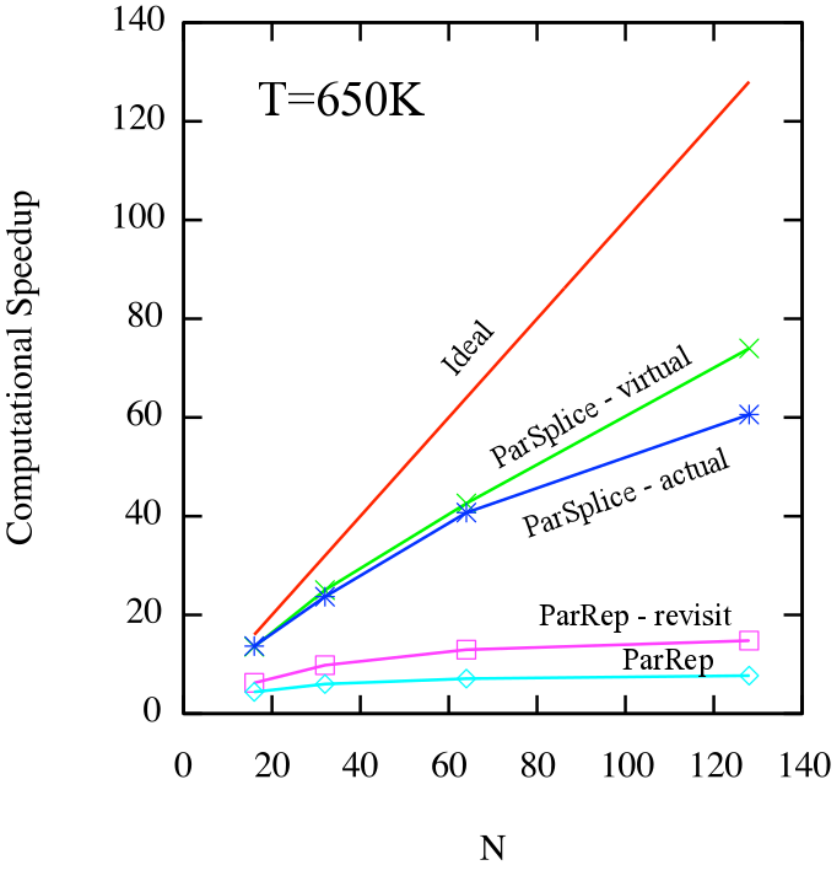
- **Each time a segment is complete:**
  - –Insert into segment database
  - –Update model of state-to-state dynamics
  - –Extend trajectory by splicing until running out of completed segments
  - –Sample endpoints of incomplete segments using Markov model
  - –Virtually extend the trajectory by splicing segments (now including incomplete ones) until running out.
  - –Request additional segment at end of virtual trajectory

Example: Ag on Ag (100)

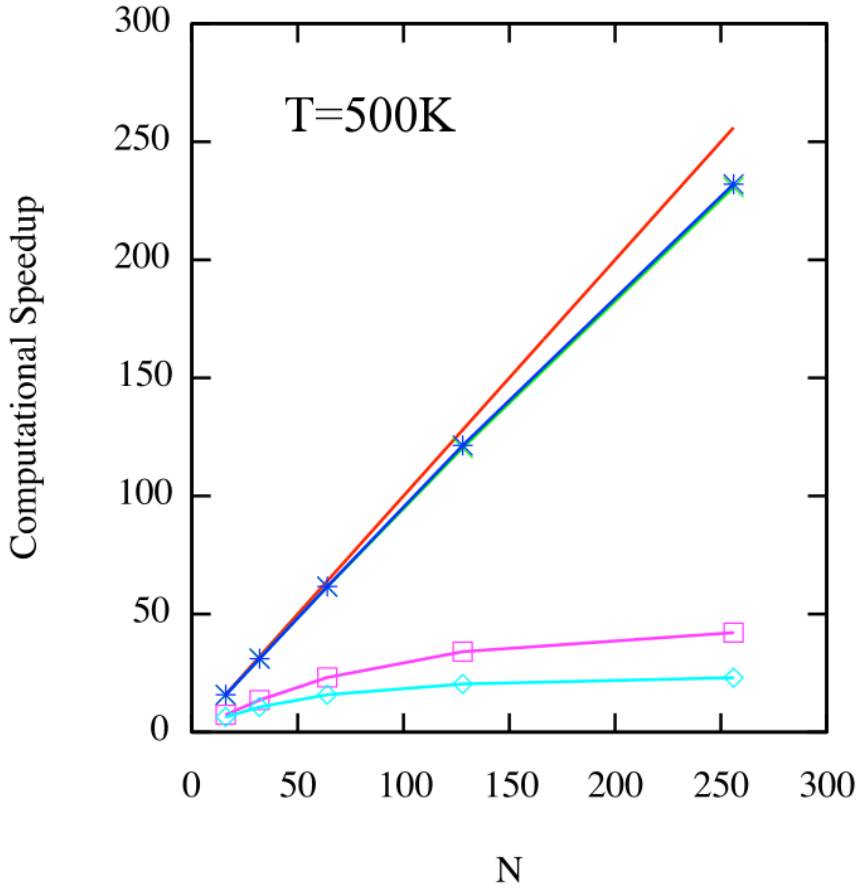


Ag trimer on Ag (100)

Example: Ag on Ag (100)



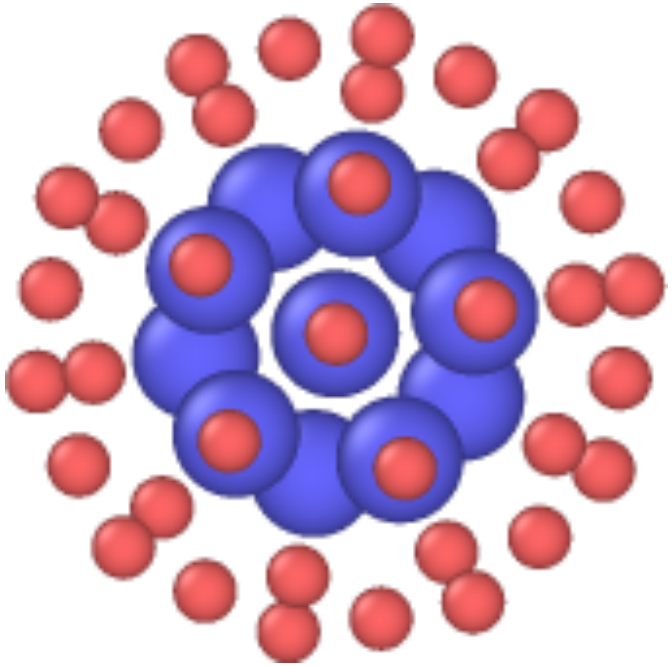
~4 revisits per state



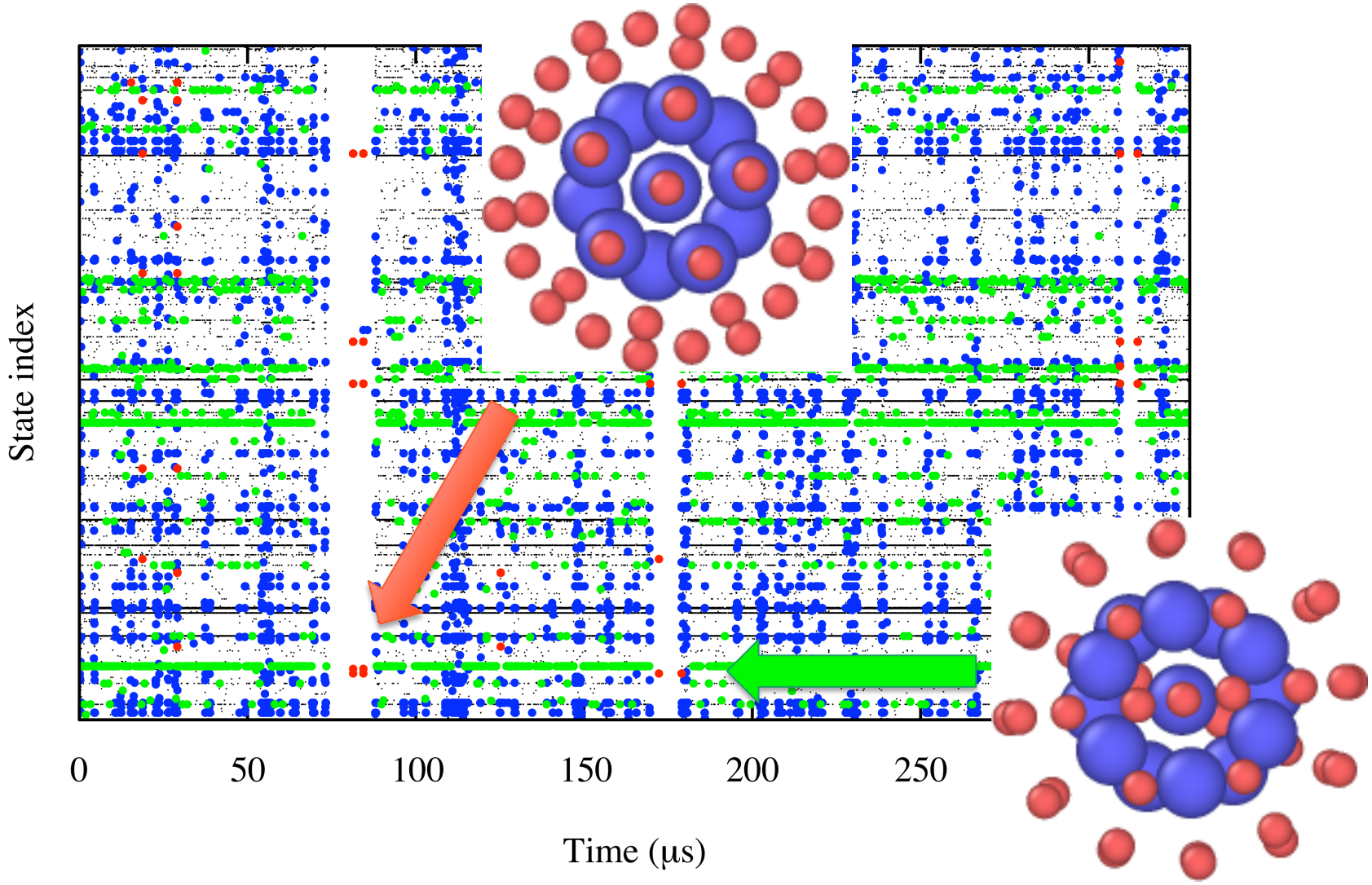
~50 revisits by state

Application to $Ag_{42}Cu_{13}$ nanoparticles

- Magic size: perfect icosahedral Cu core and Ag shell
- Studied with global optimization methods, but not with dynamical ones
- Started simulations from a completely disordered state (quenched liquid)



Application to $Ag_{42}Cu_{13}$ nanoparticles



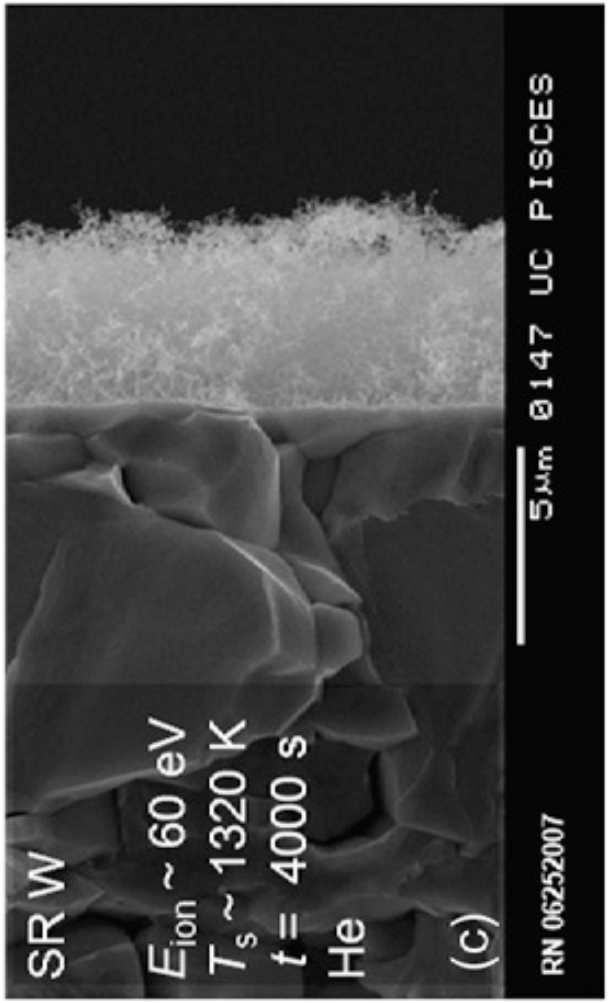
Application to $Ag_{42}Cu_{13}$ nanoparticles

At 500K :

- 334 μs of simulation time
- 10^8 transitions
- 10^5 different states visited
- Average visit durations : 2.5 ps
- Six independent visits to the ground state
- Six-fold symmetric core is lower in free energy for T>400K
- 89% efficiency on 180 replicas
- ParRep would have had ~ 50% efficiency on 2 replicas...

ParSplice at the Peta-Scale

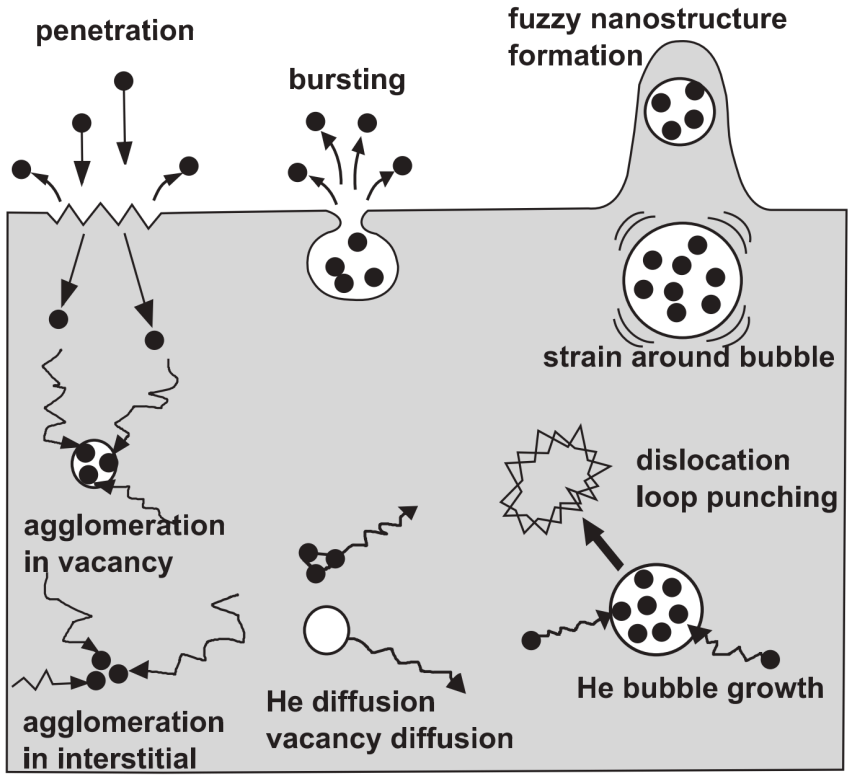
- Tungsten is a leading candidate plasma-facing material for fusion reactors
- Incoming He from the plasma is however known to adversely affect the microstructure, e.g., to lead to the formation of “fuzz”
- Fuzz formation mechanisms are unknown, but He is essential



Baldwin et al.

ParSplice at the Peta-Scale

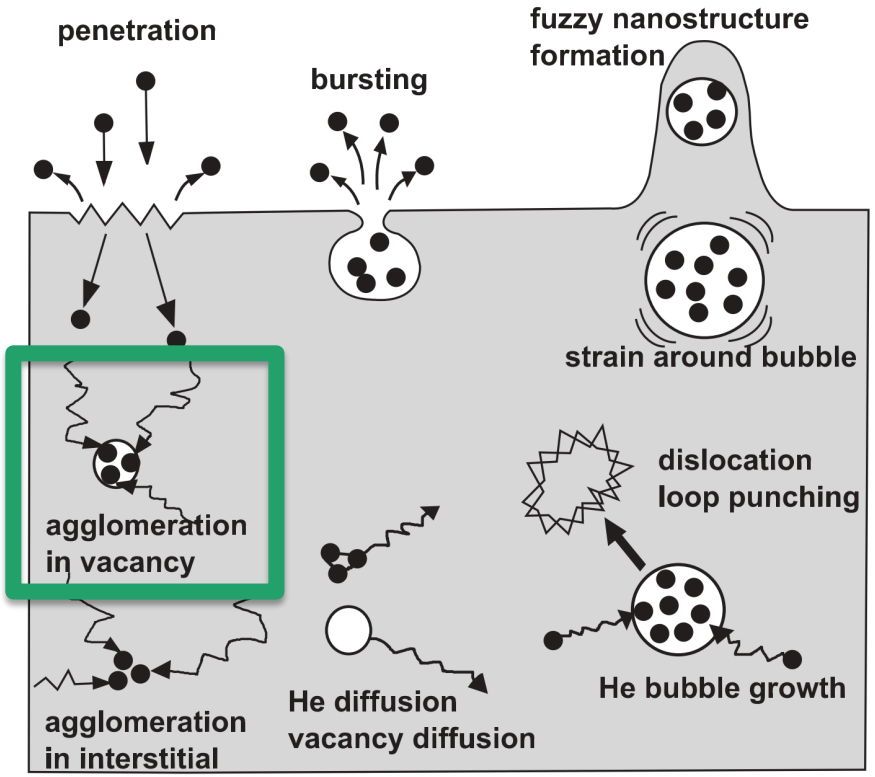
- Tungsten is a leading candidate plasma-facing material for fusion reactors
- Incoming He from the plasma is however known to adversely affect the microstructure, e.g., to lead to the formation of “fuzz”
- Fuzz formation mechanisms are unknown, but He is essential



Ito et al.

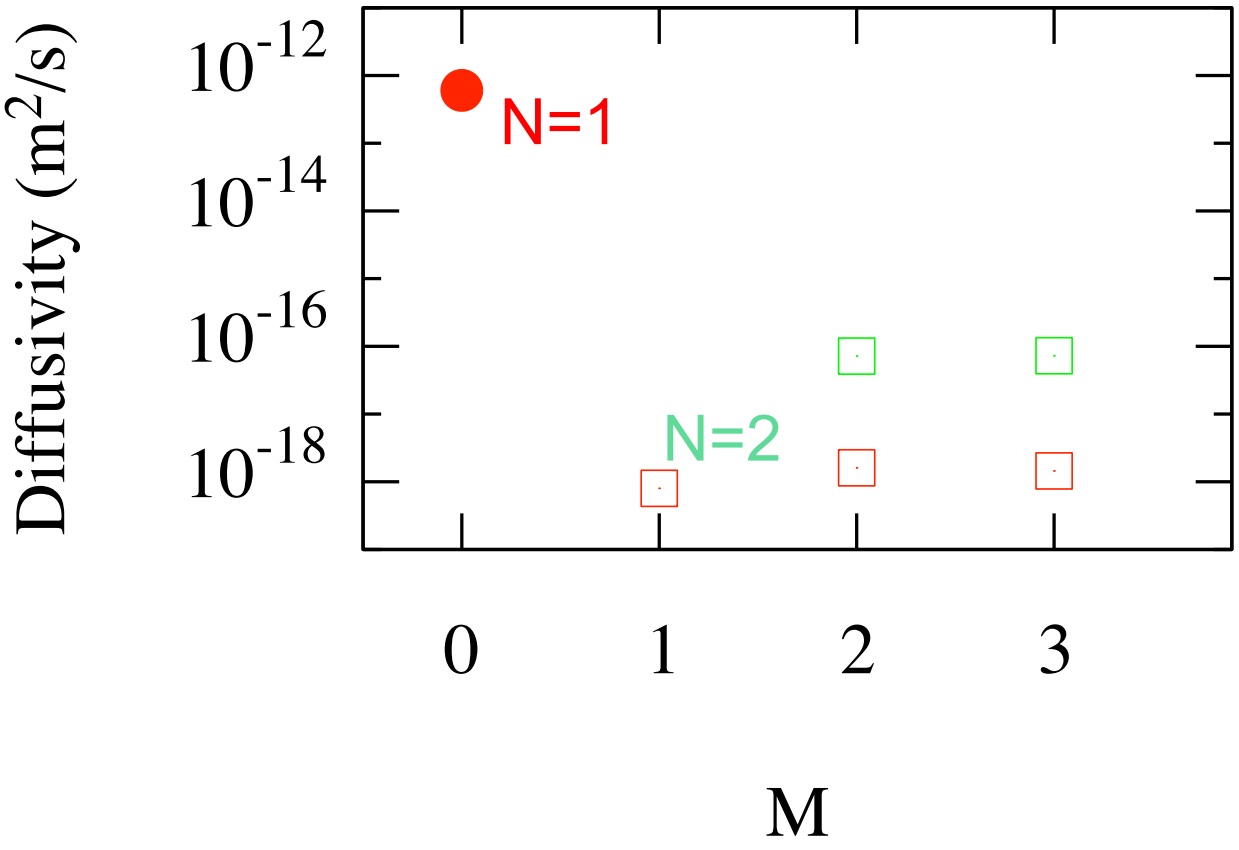
ParSplice at the Peta-Scale

- We study the behavior of small He_M/V_N cluster (N=1,2) in fusion conditions.
- Nuclei of He bubbles
- Conflicting assumptions on the mobility of these defects
- We simulate this system with ParSplice on Trinity at LANL, using up to 200,000 workers
- This gives access to **ms** timescales



Ito et al.

Low He content



One He per vacancy is sufficient to immobilize the defects on ms timescales.

High He content

Tungsten vacancies **Tungsten intersitials**

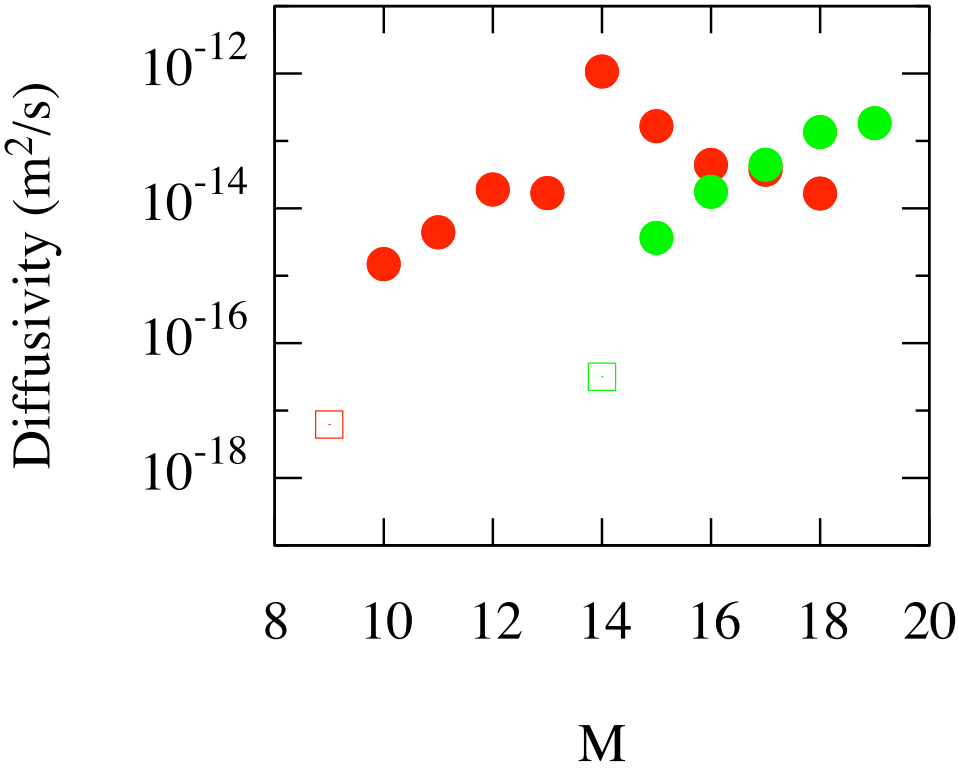
High He content

- As the He pressure increases, W Frenkel pair nucleation occurs, followed by He filling up the vacancy.
- Given enough time, nucleation is reversible and Frenkel pair annihilation can follow.
- Leads to interconversion between different $I_L V_{N+L} He_M$ variants.
- Repeated interconversions lead to net motion.

High He content

Mobility requires **both** fast nucleation **and** annihilation

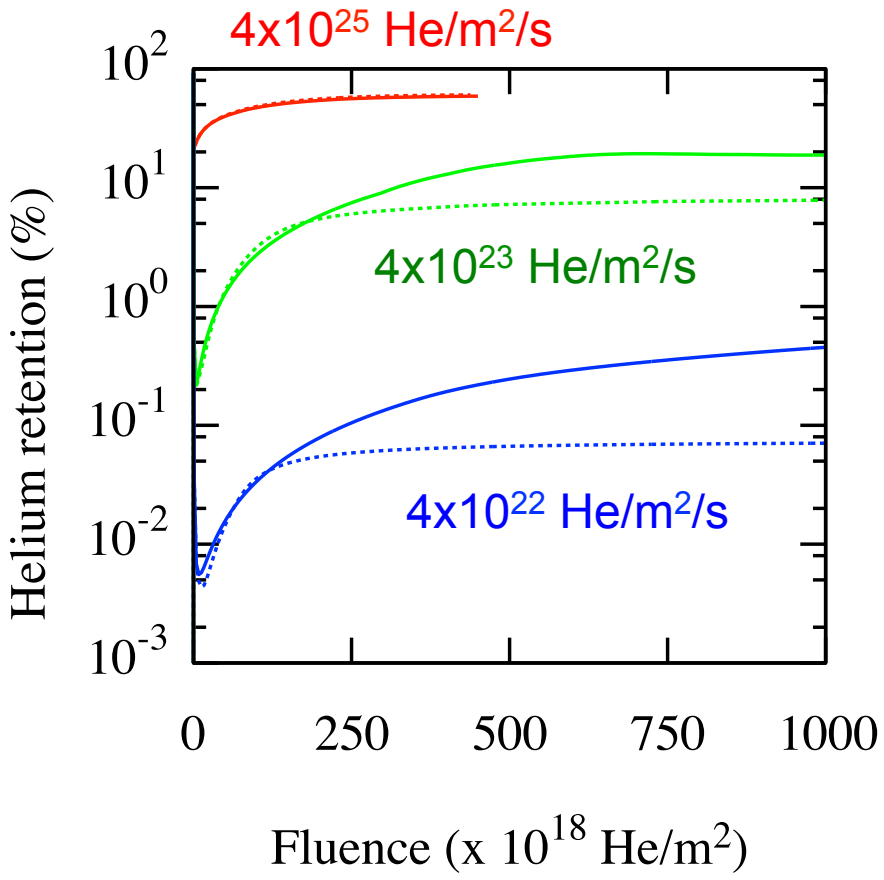
- Low M: nucleation-limited regime. Clusters are immobile on ms timscales
- Large M: nucleation is faster but annihilation is slower. Variant distribution also changes. Leads to strong mobility variations.
- **At the peak, mobility is close to that of bare vacancy,** but it is usually much slower.



Does this matter for the evolution of the wall?

We use the Xolotl cluster dynamics code (UTK) to assess the effect of the mobility of these clusters. We assume that only N=1,2 are mobile.

- At high flux (MD), effect is negligible. Complexes grow before they can move very far.
- At low fluxes typical of real reactors, He retention decreases significantly (~5-fold). ***Diffusion of the complexes opens up an efficient channel for He out-take.***



Pros and cons

Pros:

- Can be made arbitrarily accurate for *any* definition of states
- Speculation only gambles on excess capacity
- Extremely scalable when trajectories are predictable
- No waste: every segment can potentially be used
- Well suited to large scale platforms (shown scalability up to 200,000 producers)

Cons:

- Speedup limited by number of workers
- Making good predictions can be hard at large scales (have to predict further into the future)
- Implementation is complex compared to ParRep

Conclusion

- ParSplice is a parallel-in-time method that leverages massively-parallel computers to reach long timescales.
- It uses modern tools and concepts to significantly improve scaling relative to its predecessors.
- On current machines, it allows for direct simulations of small systems for ~1000 atoms over ms.

Acknowledgements

Collaborators: E. Cubuk, A. Waterland, E. Kaxiras

Funding: DOE/BES (method), LANL/LDRD (code)