Trigger Smart Data Saving Applied to CO_2 Capture in Metal-Organic Frameworks

Estelle Dirand
estelle.dirand@totalenergies.com
TotalEnergies SE
Palaiseau, France

Ali Asad
ali.asad@totalenergies.com
TotalEnergies SE
Palaiseau, France

Yann Magnin
yann.magnin@totalenergies.com
TotalEnergies SE
Pau, France

ABSTRACT

Facing the need for carbon emission reduction, processes such as CO_2 capture in nanoporous Metal-Organic Frameworks (MOFs) have emerged. However, such processes still need to be improved, by understanding the dynamic properties of CO_2 molecules when confined in MOF nanopores. To do so, molecular dynamics (MD) simulations are run for several millions of iterations, enabling to accurately compute the CO_2 residency time. Nevertheless, this dynamical parameter remains challenging to compute by standard post-processing approaches and may require terabytes of memory when data are saved after each iteration. To tackle this issue, we developed a trigger-based in situ approach that saves only the relevant data. We implement it by instrumenting the LAMMPS MD code with the SENSEI/Python in situ API. We show that this approach reduces the quantity of data saved by 4 orders of magnitude and can be up to 14% faster than traditional MD simulations without in situ processing.

CCS CONCEPTS

• **Information systems → Data management systems**; • **Human-centered computing → Visualization systems and tools**; • **Applied computing → Chemistry**.

KEYWORDS

in situ analysis, triggers, molecular dynamics, carbon capture

ACM Reference Format:
Estelle Dirand, Ali Asad, and Yann Magnin. 2023. Trigger Smart Data Saving Applied to CO_2 Capture in Metal-Organic Frameworks. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3624062.3624156

1 INTRODUCTION

The abatement of greenhouse gases, responsible for deleterious effects on climate change is an urgent issue of our century. Along with better energy efficiency and the replacement of fossil fuels with renewable energies, carbon capture and storage (CCS) is called to play a key role in the energy transition [20, 33]. In it, the first step is the CO_2 capture that represents the most expensive brick in this chain [15]. Among different techniques, CO_2 capture from synthetic nanoporous solids like Metal-Organic Frameworks (MOFs) is currently attracting considerable interest [32]. MOF structures are comparable to molecular "interlocking building bricks" (molecular Lego), presenting a quasi-infinite tunability with respect to their pore sizes and reactivity [9, 29]. While equilibrium properties from thermodynamics are well documented in the literature, dynamic properties of CO_2 molecules passing through MOF nanopores (also called cages) are less studied. Nonetheless, such studies are key for optimizing industrial processes as well as process cost reductions [25]. To gain insight into CO_2 molecular diffusion for practical applications, diffusion properties of CO_2 molecules (called guests) are determined at the atom scale thanks to molecular dynamics simulations where CO_2 molecules diffuse through the MOF structure following a Levy's jump process [28]. In other words, CO_2 molecules get stuck in a MOF cage for a while, before promptly hopping to a neighboring one. A dynamical property of interest in such dynamics is the guest's residency time (τ) [7], defined as the averaged characteristic time between successive hoppings.

Molecular simulations are based on atomistic 3D configurations of a MOF structure made of several thousands of atoms. To reach robust statistics, simulations need to be run for a large number of iterations (several millions). Accurately computing τ from a traditional post-processing approach would require storing the CO_2 positions after each iteration. The output of the positions and additional parameters of a system of $\sim$ 10,000 atoms for one iteration corresponding to more than 1MB of data, the output of several millions of iterations requires several terabytes of data, most of them being unnecessary because of the jump process. To limit the quantity of data saved to only dynamical events of interest (i.e., CO_2 hopping events), we propose to compute in situ the location of a CO_2 molecule among the MOF cages and to trigger data output only when the molecule hops from one cage to another. Hence, the post-processing is facilitated and the computation of τ for various CO_2 concentrations is accelerated with a reasonable memory footprint.

We demonstrate the benefit of this approach by instrumenting the LAMMPS simulation package [30, 34] with the SENSEI [4] in situ infrastructure and by implementing the in situ analysis and trigger with Python. We apply it to the prototypical UiO-66(Zr) MOF structure, however, the proposed methodology is versatile and can be applied to any other nanoporous materials (other MOFs structures, zeolites, porous carbons, ...).

2 RELATED WORK

In the last decades, in situ processing has gained interest among computational science, high-performance computing, and scientific

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624156

visualization communities [8, 24]. In particular, triggers are gaining popularity offering automation in the analysis of simulation data, reducing human-in-the-loop needs [5], and enabling the saving of only interesting data at a low cost [14, 18]. In situ processing and triggers can be directly inserted in the time loop of the simulation [22, 23, 27]. However, the use of these techniques is greatly facilitated by emerging in situ infrastructure that provides generic APIs to tackle a large variety of simulations.

As an example, the first version of Catalyst [2] provides synchronous in situ capabilities, extracting simulation data through the VTK [31] data model and using ParaView [1] for visualization. The second version of Catalyst [3] proposes a model based on Conduit [12] with the possibility to add new implementations. The most common one remains ParaView Catalyst, but ADIOS [11] implementation emerges to add in transit capabilities [26]. Some features also exist in ParaView Catalyst for triggering data output.

Ascent [17] is a flyweight in situ infrastructure that aims at minimizing execution time, memory usage, and integration effort, using the Conduit data model for ease of integration and zero-copy features. It provides tools for in situ analysis and visualization, including in situ exploration of data via Jupyter Notebooks [13], and recent efforts include the integration and use of triggers [19].

Each of the in situ infrastructure provides its own set of tools for in situ analysis but comes with different APIs. As a consequence, switching to another in situ infrastructure implies a modification of the code instrumentation, which may be a source of limitation. To solve this issue, the SENSEI project [4] proposes a "write once, run everywhere" concept by providing a bridge between several in situ infrastructures, including Libsim [16], Catalyst, Ascent, and ADIOS. SENSEI uses VTK as a conversion between the data models of the simulations and the in situ infrastructures and provides a Python interface for in situ analysis [6].

In this study, we used SENSEI to instrument LAMMPS, enabling effortless switch between different in situ infrastructures. Prior study already instrumented LAMMPS with SENSEI to visualize data with LibIS [35]. In this paper, we propose to use the Python interface of SENSEI to develop a trigger-based in situ analysis of CO_2 dynamics inside a MOF.

3 INSTRUMENTATION OF LAMMPS WITH SENSEI API

The instrumentation of the LAMMPS simulation code with SENSEI in situ infrastructure can be done in two ways, an intrusive way that directly modifies the simulation source code and a non-intrusive way that uses a third-party code for the instrumentation. This non-intrusive instrumentation is made possible because the LAMMPS simulation package can be compiled as a dynamic library [34]. In this work, we decided to instrument LAMMPS in a non-intrusive way for ease of use for non-expert users and developers. To do so, we developed a C++ driver that uses LAMMPS API to manage the simulation and access data and we instrumented this driver with SENSEI API for in situ analysis.

Our driver takes as input:

- The number of iterations to execute and the frequency of in situ analysis;
- The input file used for traditional LAMMPS simulations where the `run` command used by LAMMPS to determine the number of iterations to process, is replaced by the `run 0` command. This way the time loop management is transferred to the driver;
- An XML file used by SENSEI to calibrate the in situ analysis.

The driver is organized as a standard LAMMPS simulation. An initialization step is first performed to initialize LAMMPS and SENSEI with the simulation input file and the XML file, respectively. In this latter, a data adaptor is created to convert LAMMPS data in the SENSEI data model. This is done with VTK by converting atoms and molecules in a `vtkPolyData` mesh where points correspond to the atoms and lines correspond to interatomic bonds.

The driver then instantiates a time loop with the requested number of iterations. During this time loop, each iteration of LAMMPS is executed using the `one("run 1 pre no post no")` command provided in LAMMPS API. At the required in situ frequency, data are extracted from LAMMPS using the `lammps_extract_atom` function. To create the `vtkPolyData` mesh, the positions of the atoms, as well as the interatomic bonds are extracted. Per-atom and per-molecule attributes are also extracted and saved in the `vtkPolyData` as Point Data and Cell Data respectively. By default, the variables of interest are the atom's velocity, their chemical types, and an index that determines on which molecule the atom belongs. Other variables can be requested through the SENSEI XML file such as the atomic mass, the partial charges, or the kinetic and potential energies of the atoms. Once the `vtkPolyData` mesh has been filled with appropriate data, the SENSEI bridge is executed and the analysis predefined by the user in the XML file begins.

Once the number of iterations set by the user is reached, a finalization phase occurs for the simulation and the in situ framework and some time logs are exported.

4 DEVELOPMENT OF A TRIGGERING SYSTEM FOR SMART DATA SAVING

We are interested here in computing the residency time τ of CO_2 molecules inside MOF cages, namely the averaged time between two successive hoppings of CO_2 molecules in MOF cages. To do so, we first need, based on the position of a CO_2 molecule diffusing in the MOF, to determine in which cage the molecule belongs. Once this identification is done, the location is computed after each iteration and is used as a trigger for data output. Doing so, only iterations corresponding to CO_2 hoppings are recorded, greatly accelerating the computation of τ.

4.1 Background of UiO-66(Zr) MOF

The algorithm for the identification of the cage in which a CO_2 molecule belongs can be applied to any MOF or nanoporous materials but we illustrate it with the prototypical UiO-66(Zr) MOF. This crystal is made of zirconium oxide nodes (Figure 1a), bridged by terephthalic acid ligands (Figure 1b). The overall UiO-66(Zr) structure is made of a periodic repetition of these metal nodes and ligands throughout the 3D space (Figure 1c), resulting in a nanoporous structure of 3×2×2 UiO-66(Zr) unit-cells of about 6×4×4 nm^3. Each unit-cell contains tetrahedral and octahedral cages, with diameters of ~0.75 nm and ~1.1 nm, respectively (Figure 1d). The total specific

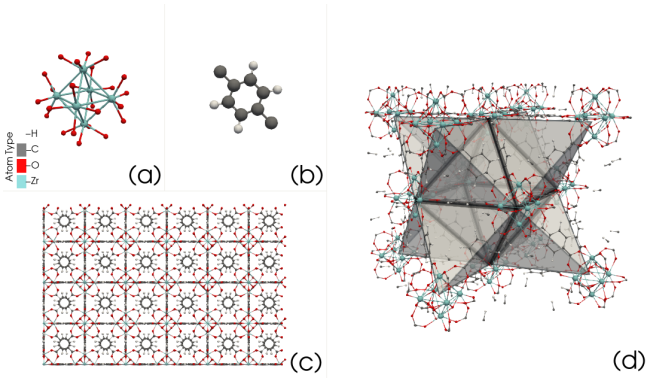

Figure 1: Atomistic structure of the UiO-66(Zr) MOF. The metal oxide nodes are composed of zirconium metal (a), connected by terephthalic acid ligands (b). The overall UiO-66(Zr) structure is made of a periodic repetition of the metal nodes and ligands. The full structure consists of 3×2×2 UiO-66(Zr) unit-cells of about 6×4×4 nm^3 (c). Zoom on the octahedral and tetrahedral cages formed in one unit cell of the UiO-66(Zr) structure (d).

pore volume of the structure is around ~0.77 cm^3/g, with a specific surface area of about ~1160 m^2/g, making it a fully nanoporous MOF (pores <2nm). Such properties and the large experimental and theoretical literature available on UiO-66(Zr) MOF make it an ideal structure to gain insights into CO_2 mobility in highly confined pores, as well as an excellent use case to develop an in situ approach applied to the CO_2 capture in MOFs.

4.2 Development of a Python script for in situ analysis

As already mentioned in Section 2, we use the SENSEI in situ infrastructure that allows an easy switch between different in situ infrastructures. We use in this work the Python interface to develop our in situ analysis and triggering system. We chose Python because it provides a large number of tools to manipulate data (NumPy, SciPy, pandas, ...) and it enables us to develop complex analyses in a few lines of code (less than 800 for this work). In this work, we use the Python interface of VTK to access the data, the NumPy package to manipulate data arrays, and the Python interface of MPI to exchange data among processes.

To develop a Python analysis to be used with SENSEI, we need to define 3 functions: an `Initialize` function executed at initialization of SENSEI to check that the requested atom attributes are present in the driver, an `Execute` function executed after each iteration that contains the code executed at the in situ frequency set by the user, and a `Finalize` function executed at SENSEI finalization to clean the different data structures and to print time information.

We detail in the following the development of the `Execute` function that is split into two parts: the construction of the MOF geometry followed by the extraction of the cage positions executed only at the first in situ iteration (Section 4.2.1), and the computation of the CO_2 location that triggers data output executed at the in situ frequency set by the user (Section 4.2.2).

4.2.1 Construction of the MOF geometry. To identify in which cage a CO_2 molecule belongs, the first step is to construct the geometry of the MOF and to compute the location of the different cages. To do so, the atomistic data is converted into a `vtkUnstructuredGrid` mesh where points correspond to the centers of mass of the metal nodes, and cells correspond to the MOF pores. The edges of the cells correspond to the position of the ligands. For the UiO-66(Zr) MOF, the cells are octahedra or tetrahedra (Figure 1d).

The centers of mass of the metal nodes are computed by extracting the Zr atoms thanks to the type attribute exported from the driver. For each Zr atom not already flagged as seen, the algorithm searches the Zr atoms belonging to the same metal nodes, namely Zr atoms located in a radius of 6Å. We then use VTK to compute the center of mass of these atoms and mark the corresponding atoms as seen. As this code is executed in a synchronous in situ way, it inherits from the parallelization of the simulation, meaning that data are split among the different processes. This step is thus performed in parallel, with data owned by the different processes, and the computed centers of mass are then gathered on the rank 0 process. This gathering is possible because the systems at stake are small (< 10,000 atoms) and these data are thus small enough not to disturb the simulation. Centers of mass are then sorted on the rank 0 process based on their positions, with increasing x-coordinate, then y-coordinate, and finally z-coordinate. A (i,j,k) tuple is created for each point to account for their (x,y,z) coordinates and to ease the creation of the cells. The sorted centers of mass are finally used to define the points of the mesh describing the MOF geometry.

The cells of the mesh are defined thanks to the periodic structure of the MOF (Figure 2). In the case of UiO-66(Zr) MOF, we can create from each point of the `vtkUnstructuredGrid`, at most 4 octahedra in the xz-plan, 2 octahedra in the yz-plan and 8 tetrahedra to fill the gaps between the octahedra. The algorithm scans each point in the mesh to check which cells can be constructed based on the location of the point in the mesh and adds them in the `vtkUnstructuredGrid` only if it does not already exist. The tetrahedra are defined as `vtkTetra` cells and the octahedra are defined as two `vtkPyramid` with the same base because VTK does not easily handle octahedra. Once the cells are constructed for the entire dataset, the position of the MOF cages is computed as the center of mass of the different cages. For tetrahedral cages, the position corresponds to the center of mass of the vertices of `vtkTetra` cells. For octahedral cages, the position corresponds to the center of mass of the vertices of the two `vtkPyramid` cells sharing the same base. The positions of the different cages are finally stored in a global variable so that they can be kept in memory from one iteration to the other.

4.2.2 Development of the trigger. The second part of the `Execute` function computes which cage is occupied by a CO_2 molecule and triggers data output only when a CO_2 molecule has hopped from one cage to another. In this study, we decided to track only one CO_2 molecule. This molecule is randomly chosen at the in situ initialization and its molecule ID is kept as a global variable to be accessible at each time step. Tracking one CO_2 molecule simplifies the development of the trigger without interfering with the physics behind it. However, tracking several molecules would lead to better

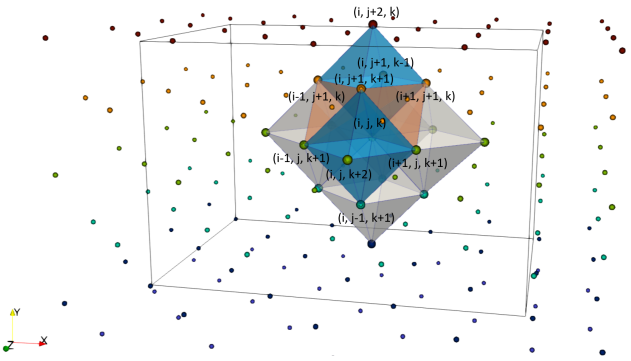

Figure 2: Creation of the 6 octahedra and 8 tetrahedra from one point of the MOF mesh. We highlight the (i,j,k) indices that form 2 octahedral cells (in blue) and 2 tetrahedral cells (in orange).

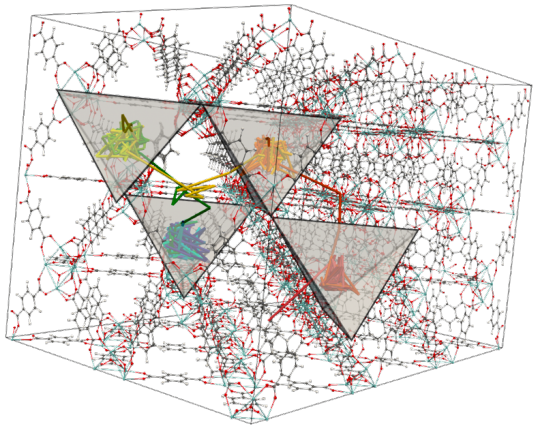

Figure 3: Visualization of the trajectory of a CO_2 molecule (colored line) in a rigid UiO-66(Zr) MOF (no MOF atoms vibrations) for 5 million iterations.

statistics for the computation of the residency time and is thus left for future work.

After each iteration, each MPI process searches whether the tracked CO_2 molecule belongs to its process. The process that owns this CO_2 molecule sends the position of the corresponding carbon atom to the rank 0 process that is then in charge of computing which cage's center of mass is closer to this atom. In the case where the corresponding cage ID is different than the one stored at the previous iteration, a data output is triggered. This change of the closest cage's center of mass is what we call *trigger criteria* in the following. The user can choose to export an xyz file (as proposed by LAMMPS) for later post-processing, or to write only the information necessary to compute the residency time, namely a text file with the iterations of hopping and the IDs of the visited cages along with time.

In the case where a flexible MOF is considered i.e., the MOF atoms are allowed to oscillate around their positions due to the thermal excitation, computing the MOF geometry only at the first iteration may cause errors in the detection of the cage containing the CO_2 molecule. To solve this issue, the extraction of the centers of mass of the metal nodes can be done periodically during the simulation, at a frequency set by the user. As the atoms are only oscillating around their equilibrium positions, it is possible to skip the re-generation of the cells of the mesh to save computing time.

5 RESULTS

5.1 Performance evaluation

The evaluation of this triggering system is performed by running simulations with one CO_2 molecule inside a rigid UiO-66(Zr) MOF composed of 5,184 atoms on 5 million iterations. We run 3 simulations in this evaluation: a traditional LAMMPS simulation without any data output, a traditional LAMMPS simulation that writes an xyz file after each iteration, and our LAMMPS driver with our triggering system executed after each iteration that writes an xyz file when the trigger is fired. The data output or in situ analysis frequency is set to 1 in this study because there is no a priori knowledge of the frequency of molecule hopping and we want to compute the residency time the more accurately as possible. The simulations are executed on 32 MPI processes of the Pangea 2 supercomputer composed of Intel®Haswell nodes of 24 cores and 128GB of memory each, with the stable_3Mar2020 version of LAMMPS and the version 3.2.1 of SENSEI.

First, our triggering system enables us to easily analyze the trajectory of the CO_2 molecule, and in particular to extract the visited cages over time (Figure 3). We highlight that the diffusion mechanisms occur by successive hoppings, where a CO_2 molecule is trapped for a while in a tetrahedral cage, then diffuses to the next one by promptly crossing an octahedral cage. For this simulation, the in situ algorithm triggers data output only 94 times over the 5 million iterations, showing that CO_2 molecules get trapped for a long time before hopping to another cage.

As a consequence, our driver outputs 130MB of data, while the traditional LAMMPS simulation with output performed after each iteration generates 1.4TB of data. This corresponds to a gain of 4 orders of magnitude in the size of the data saved between the two approaches. Moreover, this also has an effect on the total execution time. In Figure 4 we show the total execution time of the 3 simulations, divided into the time to run the simulation, the time to write files, and the time to compute the in situ analysis and evaluate the trigger criteria. We thus show that our driver with in situ analysis has an overhead of 32.85% compared to the traditional LAMMPS simulation without output. This comes from the fact that the driver introduces an overhead on the simulation execution time and the in situ analysis adds the computation time needed to evaluate the trigger criteria. On the other hand, the in situ analysis scenario is 14% faster than the traditional LAMMPS simulation with output performed at each iteration. This gain is mostly attributed to the smaller time needed to evaluate the trigger criteria than to write data to the filesystem. Finally, we do not evaluate the gain in the post-processing of the data itself because, to our knowledge, no tool exists to compute, in a post-processing way, the residency time from LAMMPS output. However, as LAMMPS appends all the data in the same file, we safely assume that the time to read and analyze

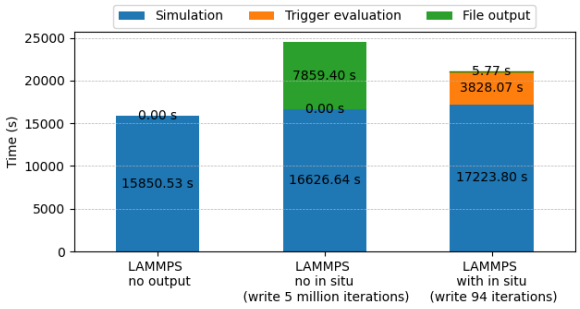

Figure 4: Comparison of the execution time for three simulations: traditional LAMMPS without any data output, traditional LAMMPS with data output after each iteration, and the LAMMPS driver with our triggering system executed after each iteration. For each simulation, we show the time taken by LAMMPS itself, the time spent to evaluate the trigger if applicable, and the time necessary to write data.

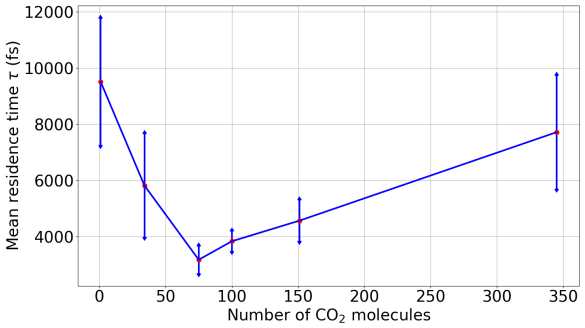

Figure 5: Residency times (τ) in UiO-66(Zr) MOF for various CO_2 concentrations. Each point corresponds to the mean and standard deviation of τ over 10 simulations. Time is in femtoseconds (10^{-15} s) where 1 iteration corresponds to 1 fs.

1.4TB of data is way higher than the time to process the 130MB data exported by our driver.

5.2 Computation of the residency time for various CO_2 concentrations

We finally use our in situ analysis and triggering system to compute the averaged CO_2 residency time τ for several CO_2 concentrations. In this case, we run simulations with different numbers of CO_2 molecules (1, 34, 75, 100, 151, and 345) in flexible UiO-66(Zr) MOF for 10 million iterations. For one simulation, τ is computed as the mean residency time of a randomly chosen CO_2 molecule. To account for the trajectory of different CO_2 molecules, each simulation is executed 10 times.

Figure 5 shows the evolution of τ with the number of CO_2 molecules in the simulation. For a given number of CO_2 molecules, we compute the mean and standard deviation of τ over the 10 simulation runs. We show that τ has a nonlinear behavior, decreasing until a minimum for simulations with 75 CO_2 molecules, followed by an increasing behavior at larger CO_2 concentrations. The molecular mechanistics behind such a behavior can be highlight as follow. At low molecular loading, corresponding to a few guest molecules in the overall MOF pores, CO_2 molecules strongly bond to the most favorable energy sites of the MOF, reducing their mobility and thus, increasing τ. When the guest loading increases and that all favorable energy sites are occupied by one CO_2, additional guests molecules locate on less favorable sites increasing their mobility, and in turn decreasing τ. At large loading (above 75 CO_2), the available free pore volume in the MOF is low and the probability of CO_2 collisions increases with guests number. As a consequence collisions decrease the overall CO_2 mobility and an increase of τ.

Averaging τ is challenging but is important to provide input to assess general diffusion theories from statistical physics or process modeling. Our in situ analysis and triggering system enables us to perform this study with a low overhead on the simulation execution time and with a small amount of data saved in the filesystem. Indeed, by exporting only the information about residency time (as seen in Section 4.2.2), we run the 60 simulations required to create Figure 5 with a memory footprint smaller than 100MB.

6 CONCLUSION AND FUTURE WORK

Carbon capture, including CO_2 capture processes based on MOF materials, is called to play an important role in the energy transition. To understand and upscale the dynamics of CO_2 molecules inside MOF, the residency time of CO_2 molecules is an important parameter that can be used as an input in various theoretical models for multiscale modeling. However, its accurate computation requires a huge amount of data to be saved and calls for more efficient analysis methodologies. In this study, we propose an in situ analysis to extract the geometry of a MOF to compute the CO_2 kinetics in MOF porosity and to trigger data output only when a CO_2 molecule has hopped from one cage to another. We demonstrate the benefits of this approach by implementing a LAMMPS C++ driver instrumented with SENSEI in situ infrastructure and with an in situ analysis developed in Python. We show that the triggering system is up to 14% faster than the traditional approach without in situ, with a memory footprint reduction by 4 orders of magnitude that enables to perform studies of residency time computations of 60 simulations with a memory footprint smaller than 100MB.

As future work, we intend to compute the residency time for various MOFs, such as ZIF-8 [10] and CALF-20(Zn) [21, 36], and for various mixtures (different concentrations of CO_2, blends with H_2O, N_2, ...). We will also adapt our trigger algorithm to track several CO_2 molecules to compute better statistics of residency time. In the next steps, we would like to test other in situ frameworks such as Ascent or Catalyst through the SENSEI interface and to mix triggers and bi-directional approaches. For example, we would like to rewind the simulation when a molecule hops to another cage to gain insights into hopping mechanisms in different MOFs. This way, we could relay the role played by the local pore surface chemistry in molecular diffusion that is crucial for designing the next generation of MOFs materials that can efficiently capture CO_2 at a reduced cost.

ACKNOWLEDGMENTS

This work was supported by TotalEnergies S.E. through OneTech, Sustainability, CCUS R&D program. We would like to thank TotalEnergies High Performance Computing Center for the CPU time provided on its supercomputer Pangea 2. We would also like to thank Burlen Loring and Wes Bethel for their valuable help in using the SENSEI in situ infrastructure.

REFERENCES

[1] Utkarsh Ayachit. 2015. *The paraview guide: a parallel visualization application*. Kitware, Inc.

[2] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the first workshop on in situ infrastructures for enabling extreme-scale analysis and visualization*. 25–29.

[3] Utkarsh Ayachit, Andrew C Bauer, Ben Boeckel, Berk Geveci, Kenneth Moreland, Patrick O'Leary, and Tom Osika. 2021. Catalyst revised: rethinking the paraview in situ analysis and visualization api. In *High Performance Computing: ISC High Performance Digital 2021 International Workshops, Frankfurt am Main, Germany, June 24–July 2, 2021, Revised Selected Papers 36*. Springer, 484–494.

[4] Utkarsh Ayachit, Brad Whitlock, Matthew Wolf, Burlen Loring, Berk Geveci, David Lonie, and E Wes Bethel. 2016. The SENSEI generic in situ interface. In *2016 second workshop on in situ infrastructures for enabling extreme-scale Analysis and visualization (ISAV)*. IEEE, 40–44.

[5] Janine C Bennett, Ankit Bhagatwala, Jacqueline H Chen, Ali Pinar, Maher Salloum, and C Seshadhri. 2016. Trigger detection for adaptive scientific workflows using percentile sampling. *SIAM Journal on Scientific Computing* 38, 5 (2016), S240–S263.

[6] E Wes Bethel, Burlen Loring, Utkarsh Ayachit, David Camp, Earl PN Duque, Nicola Ferrier, Joseph Insley, Junmin Gu, James Kress, Patrick O'Leary, et al. 2022. The SENSEI Generic In Situ Interface: Tool and Processing Portability at Scale. In *In Situ Visualization for Computational Science*. Springer, 281–306.

[7] Colin Bousige, Pierre Levitz, and Benoit Coasne. 2021. Bridging scales in disordered porous media by mapping molecular dynamics onto intermittent Brownian motion. *Nature Communications* 12, 1 (2021), 1043.

[8] Hank Childs, Janine Bennett, Christoph Garth, and Bernd Hentschel. 2019. In situ visualization for computational science. *IEEE Computer Graphics and Applications* 39, 6 (2019), 76–85.

[9] Sean P Collins, Thomas D Daff, Sarah S Piotrkowski, and Tom K Woo. 2016. Materials design by evolutionary optimization of functional groups in metal-organic frameworks. *Science advances* 2, 11 (2016), e1600954.

[10] D Fairen-Jimenez, SA Moggach, MT Wharmby, PA Wright, S Parsons, and T Duren. 2011. Opening the gate: framework flexibility in ZIF-8 explored by experiments and simulations. *Journal of the American Chemical Society* 133, 23 (2011), 8900–8902.

[11] William F Godoy, Norbert Podhorszki, Ruonan Wang, Chuck Atkins, Greg Eisenhauer, Junmin Gu, Philip Davis, Jong Choi, Kai Germaschewski, Kevin Huck, et al. 2020. Adios 2: The adaptable input output system. a framework for high-performance data management. *SoftwareX* 12 (2020), 100561.

[12] Cyrus Harrison, Matthew Larsen, Brian S Ryujin, Adam Kunen, Arlie Capps, and Justin Privitera. 2022. Conduit: A Successful Strategy for Describing and Sharing Data In Situ. In *2022 IEEE/ACM International Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. IEEE, 1–6.

[13] Seif Ibrahim, Thomas Stitt, Matthew Larsen, and Cyrus Harrison. 2019. Interactive in situ visualization and analysis using ascent and jupyter. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 44–48.

[14] Yuya Kawakami, Nicole Marsaglia, Matthew Larsen, and Hank Childs. 2020. Benchmarking in situ triggers via reconstruction error. In *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 38–43.

[15] D Koutsonikolas, G Pantoleontos, M Mavroudi, S Kaldis, A Pagana, ES Kikkinides, and D Konstantinidis. 2016. Pilot tests of CO2 capture in brick production industry using gas–liquid contact membranes. *International Journal of Energy and Environmental Engineering* 7, 1 (2016), 61–68.

[16] T Kuhlen, R Pajarola, and K Zhou. 2011. Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*, Vol. 10. Eurographics Association Aire-la-Ville, Switzerland, 101–109.

[17] Matthew Larsen, Eric Brugger, Hank Childs, and Cyrus Harrison. 2022. Ascent: A flyweight in situ library for exascale simulations. In *In Situ Visualization for Computational Science*. Springer, 255–279.

[18] Matthew Larsen, Cyrus Harrison, Terece L Turton, Sudhanshu Sane, Stephanie Brink, and Hank Childs. 2021. Trigger happy: Assessing the viability of trigger-based in situ analysis. In *2021 IEEE 11th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 22–31.

[19] Matthew Larsen, Amy Woods, Nicole Marsaglia, Ayan Biswas, Soumya Dutta, Cyrus Harrison, and Hank Childs. 2018. A flexible system for in situ triggers. In *Proceedings of the workshop on in situ infrastructures for enabling extreme-scale analysis and visualization*. 1–6.

[20] Fabrice Lecomte, Paul Broutin, and Etienne Lebas. 2010. *CO2 capture: technologies to reduce greenhouse gas emissions*. Editions Technip.

[21] Jian-Bin Lin, Tai TT Nguyen, Ramanathan Vaidhyanathan, Jake Burner, Jared M Taylor, Hana Durekova, Farid Akhtar, Roger K Mah, Omid Ghaffari-Nik, Stefan Marx, et al. 2021. A scalable metal-organic framework as a durable physisorbent for carbon dioxide capture. *Science* 374, 6574 (2021), 1464–1469.

[22] Julia Ling, W Philip Kegelmeyer, Konduri Aditya, Hemanth Kolla, Kevin A Reed, Timothy M Shead, and Warren L Davis. 2017. Using feature importance metrics to detect events of interest in scientific computing applications. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 55–63.

[23] Yingjie Liu, Ge Chen, Miao Sun, Shuai Liu, and Fenglin Tian. 2016. A parallel SLA-based algorithm for global mesoscale eddy identification. *Journal of Atmospheric and Oceanic Technology* 33, 12 (2016), 2743–2754.

[24] Kwan-Liu Ma. 2009. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications* 29, 6 (2009), 14–19.

[25] Yann Magnin, Estelle Dirand, Alejandro Orsikowsky, Mélanie Plainchault, Véronique Pugnet, Philippe Cordier, and Philip L Llewellyn. 2022. A step in carbon capture from wet gases: understanding the effect of water on CO2 adsorption and diffusion in UiO-66. *The Journal of Physical Chemistry C* 126, 6 (2022), 3211–3220.

[26] François Mazen, Lucas Givord, and Charles Gueunet. 2023. Catalyst-ADIOS2: In Transit Analysis for Numerical Simulations Using Catalyst 2 API. In *International Conference on High Performance Computing*. Springer, 269–276.

[27] Kary Myers, Earl Lawrence, Michael Fugate, Jon Woodring, Joanne Wendelberger, and Jim Ahrens. 2014. *An in situ approach for approximating complex computer simulations and identifying important time steps*. Technical Report. Citeseer.

[28] Albrecht Ott, Jean-Philippe Bouchaud, Dominique Langevin, and Wladimir Urbach. 1990. Anomalous diffusion in "living polymers": A genuine Levy flight? *Physical review letters* 65, 17 (1990), 2201.

[29] Calogero Giancarlo Piscopo and Stefan Loebbecke. 2020. Strategies to Enhance Carbon Dioxide Capture in Metal-Organic Frameworks. *ChemPlusChem* 85, 3 (2020), 538–547.

[30] Steve Plimpton. 1995. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics* 117, 1 (1995), 1–19.

[31] Will Schroeder, Kenneth M Martin, and William E Lorensen. 1998. *The visualization toolkit an object-oriented approach to 3D graphics*. Prentice-Hall, Inc.

[32] Kenji Sumida, David L Rogow, Jarad A Mason, Thomas M McDonald, Eric D Bloch, Zoey R Herm, Tae-Hyun Bae, and Jeffrey R Long. 2012. Carbon dioxide capture in metal–organic frameworks. *Chemical reviews* 112, 2 (2012), 724–781.

[33] John Frederick D Tapia, Jui-Yuan Lee, Raymond EH Ooi, Dominic CY Foo, and Raymond R Tan. 2018. A review of optimization and decision-making models for the planning of CO2 capture, utilization and storage (CCUS) systems. *Sustainable Production and Consumption* 13 (2018), 1–15.

[34] Aidan P Thompson, H Metin Aktulga, Richard Berger, Dan S Bolintineanu, W Michael Brown, Paul S Crozier, Pieter J in't Veld, Axel Kohlmeyer, Stan G Moore, Trung Dac Nguyen, et al. 2022. LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications* 271 (2022), 108171.

[35] Will Usher, Silvio Rizzi, Ingo Wald, Jefferson Amstutz, Joseph Insley, Venkatram Vishwanath, Nicola Ferrier, Michael E Papka, and Valerio Pascucci. 2018. libIS: A lightweight library for flexible in transit visualization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 33–38.

[36] Magnin Yann, Dirand Estelle, Maurin Guillaume, and Llewellyn Philip. 2023. Abnormal CO2 and H2O Diffusion in CALF-20 (Zn) Metal-Organic Framework Angstropores. *arXiv preprint arXiv:2307.09200* (2023).