Evaluating the Parallel Simulation of Dynamics of Electrons in Molecules on AWS Spot Instances

Vanderlei Munhoz[1], **Márcio Castro**[1], **Luis G. C. Rego**[2]

[1] Federal University of Santa Catarina (UFSC)
Department of Informatics and Statistics (INE)

[2] Federal University of Santa Catarina (UFSC)
Department of Physics (FSC)

vanderlei.filho@proton.me, marcio.castro@ufsc.br
luis.guilherme@ufsc.br

Abstract. *In this paper, we evaluate the cost-effectiveness and performance of simulating the dynamics of electrons in molecules on AWS and investigate the implications of using various types of storage solutions, contrasting the results with those obtained on a traditional HPC cluster. Our findings reveal key insights into the computational efficiency and cost-effectiveness of these diverse platforms, contributing to the critical discourse on how to optimally harness the power of modern computing infrastructures for complex molecular simulations.*

1. Introduction

The dynamics of electrons in molecules is a critical area of study that has implications for various fields, from fundamental science to practical applications in technology and industry [Marcus 1993, Evers et al. 2020]. Understanding changes in the electronic structure of the atoms and molecules can help predict the outcome of chemical reactions, significantly aiding pursuits in drug discovery and materials science [De Vivo et al. 2016, Carter 2008]. Moreover, deeper knowledge in this domain can produce groundbreaking data processing and transmission methods. In certain quantum computing architectures, electrons play a crucial role as information carriers. Therefore, a thorough understanding of their dynamics becomes indispensable in fostering advancement in this promising and rapidly evolving field [Yu et al. 2022, Ollitrault et al. 2021].

Computer simulations play a vital role in this field of study, offering a practical and efficient method for exploring complex quantum phenomena that are otherwise difficult to observe directly. Using computational models, researchers can investigate how electrons behave under various conditions, providing an invaluable tool for understanding the intricacies of molecular behavior [Ciccotti et al. 2022]. As computational power continues to grow, simulations will allow for examining increasingly complex molecular systems, enabling breakthroughs that might not be achievable through traditional experimental methodologies alone.

Public cloud platforms have effectively democratized access to computational power, extending their benefits to millions of users worldwide. They are versatile tools, especially for small organizations and independent scientists, who may lack the access or the financial means to run intensive simulations on their own [Buyya et al. 2019]. However, navigating these platforms can be riddled with challenges. One significant hurdle is the need for more standardization among providers, causing users to struggle with making fair and informed comparisons. The pricing models, frequently opaque and subject to continual change, further complicate matters, adding another layer of complexity and making it difficult for users to anticipate costs and budgets effectively. The hardware abstraction inherent in cloud platforms presents another sizable obstacle, specifically for High Performance Computing (HPC) applications that often require precise performance characteristics [Netto et al. 2018]. This level of abstraction damps the optimization process for these applications, making it hard for users to fully leverage their performance capabilities.

In prior research, we proposed the *HPC@Cloud* tool to assist users in creating HPC clusters and executing HPC applications in public clouds while reducing costs by employing transient Virtual Machines (VMs) from Amazon Web Services (AWS) known as *Spot instances* [Wang et al. 2018]. These instances represent idle infrastructure that cloud providers offer at a significantly discounted rate, although they are subject to revocation at any time [Munhoz et al. 2022]. It also provides a comprehensive suite of tools to monitor potential Spot failures and promptly recover parallel applications based on the Message Passing Interface (MPI) standard, thereby offering users an effective way to maximize the benefits of their cloud resources.

In this paper, we dissect the significant issues of running parallel simulations of dynamics of electrons in molecules using the AWS cloud infrastructure. We provide a thorough cost-effectiveness assessment of AWS Spot instances and different storage solutions, comparing their performance and costs with a traditional on-premise HPC cluster. We focus on the *DynEMol* application, which is capable of describing the nonadiabatic excited state of molecules adsorbed on extended solid surfaces as well as charge transfer processes.

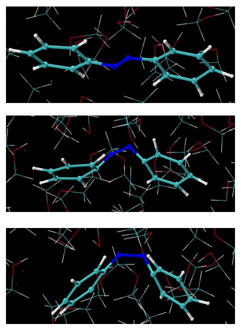
In summary, we bring the following contributions to the HPC and Cloud Computing convergence:

1. A comprehensive evaluation of the efficacy of the *HPC@Cloud* tool when deployed for a real-world HPC application (*DynEMol*) on public cloud platforms.
2. A series of experiments that illustrate the advantages and disadvantages of utilizing public cloud infrastructure for molecular dynamics simulations, thereby providing a lucid overview of the potential costs and the scale of experiments that can be efficiently conducted within these environments.
3. An examination of the various compute and storage options provided by AWS, focusing on differentiating their unique features and potential applications.

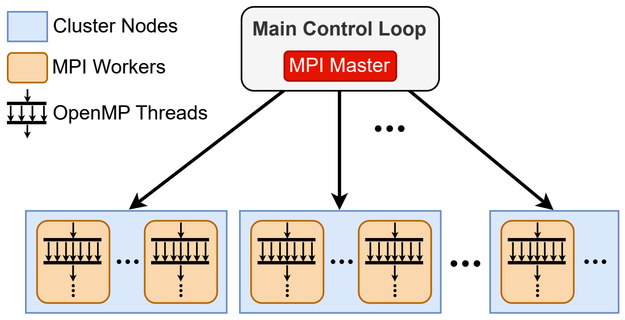
The remainder of this work is organized as follows. In Section 2, we briefly present the *DynEMol* simulation tool. In Section 3, we present our method for migrating the *DynEMol* application to the public cloud, with a detailed discussion regarding the major trade-offs and the reasoning behind our decisions. In Section 4, we present our applied evaluation method and gathered experimental data. In Section 5, we analyze and discuss the results. In Section 6, we discuss related research. Finally, we draw our conclusions and discuss future work in Section 7.

2. The *DynEMol* Simulation Tool

Charge Transfer (CT) and Electronic Excitation Dynamics (EED) are ubiquitous in photochemistry. They constitute the underlying mechanisms for electron transfer reactions, light-harvesting in natural and artificial molecular structures, energy transduction phenomena, and many other processes. They are often influenced by the non-adiabatic coupling between electronic and nuclear degrees of freedom.



(a) Simulation results.



(b) Parallelization strategy.

Figure 1. Overview of *DynEMol* simulation results and parallelization.

The Dynamics of Electrons in Molecules (*DynEMol*[1]) is a sophisticated simulation tool for accurately representing how molecules behave when they are attached to large solid surfaces and during processes where electrical charge is transferred. It performs simulations of the excited-state dynamics of molecular systems in the solid and liquid phases.

The method combines tight-biding Quantum Mechanical (QM) with classical Molecular Mechanics (MM) formalisms in a semi-empirical hybrid quantum-classical model capable of simulating the non-adiabatic dynamics of large atomistic models at the lowest computational cost.

This hybrid method provides the tools for studying a variety of photoinduced effects, including the charge and energy transfer dynamics in large molecular and nanostructured materials subject to complex structural deformations [Torres et al. 2018, Abraham et al. 2019]. Simulations are carried out within the framework of the self-consistent Ehrenfest method and the Coherent Switching with Decay-of-Mixing (CSDM) method [Shu et al. 2020].

DynEMol is written mainly in Fortran and leverages OpenMP and Message Passing Interface (MPI) to execute compute-intensive tasks in parallel. The dynamics simulation consists of a main loop (the *time loop*) that executes fifty to a hundred thousand time-steps. Parallel computations are restricted to interaction calculations in a given time slice. A master MPI process (*rank 0*) undergoes the entire time evolution of the simulation and exchanges data with MPI worker processes distributed across the cluster. It also executes a few compute-intensive tasks, such as matrix diagonalizations. Most MPI worker processes are set to dwell in specific subroutines, particularly the ones that calculate the forces on the atoms. A smaller number of MPI worker processes are used for eigenvalue and eigenvector calculations as well as matrix inversions.

Figure 1a shows an example of a simulation output by *DynEMol* (in this case, an azobenzene molecule in ethanol), whereas Figure 1b shows an overview of the hybrid MPI/OpenMP parallelization strategy adopted in *DynEMol*. Parameters of the parallel solution include the number of: (i) HPC cluster nodes; (ii) MPI worker processes per cluster node; and (iii) OpenMP threads per MPI worker process. A variety of communications are used to send tasks and receive solutions to/from MPI processes. *DynEMol* achieves better performance when the number of MPI workers and OpenMP threads are fine-tuned

[1]`https://luisrego.sites.ufsc.br/`

to the target system based on its atomic size. Linear algebra calculations, such as matrix diagonalization and matrix inversion, benefit from OpenMP threads, whereas the atomic force calculations are efficiently parallelized via MPI processes.

DynEMol also employs a data fault tolerance mechanism that preserves the state of simulations in a set of files. The frequency of these backups, determined by the number of iterations, can be adjusted to suit the needs of the simulation. We have marginally adapted this existing mechanism to save backup files only when an eviction alarm for a Spot instance is detected. This optimized approach helps conserve computational resources, allowing them to be wholly dedicated to running the simulation, thus enhancing overall efficiency and reducing costs.

3. The Proposed Cloud-based HPC Cluster Architecture

Leading public cloud providers, such as Amazon Web Services (AWS), offer services and hardware infrastructures to a broad audience over the Internet, eliminating the need for long-term commitments or direct engagements with the service provider [Mell and Grance 2011]. With hardware strategically located in multiple data centers, AWS extends computing capabilities to users in the form of *instances*. An instance, often typified as a VM, represents a virtual allocation on shared physical infrastructure and remains a prevalent and cost-effective computing choice. AWS provides a diverse spectrum of instances, each differing in attributes such as hypervisor type, CPU design, number of vCPUs, memory capacity, and more, enabling users to pinpoint the ideal configuration for their unique requirements. Besides the VMs, AWS offers APIs to create and manage other infrastructure elements, such as networking and storage.

To facilitate the migration of legacy HPC applications to public cloud platforms, we leverage the *HPC@Cloud* toolkit[2], introduced by [Munhoz and Castro 2022]. *HPC@Cloud* offers a suite of tools that can be executed on the user's machine. These tools enable users to configure cloud infrastructure, execute jobs, monitor performance, predict costs, and interact with the provisioned resources in an automated and provider-agnostic manner. *HPC@Cloud* is open-source software.

In this section, we detail the proposed HPC cluster architecture built on top of AWS using *HPC@Cloud*. We first discuss its topology in Section 3.1. Then, we give the details about its network infrastructure in Section 3.2.

3.1. Cluster Topology and Storage Devices

We build a cluster architecture in AWS akin to a classical on-premise homogeneous HPC cluster. Given its ephemeral character — being instantiated solely for a singular task before termination — there is no distinction between master and worker nodes. As such, every node will be dedicated to workload execution.

We leverage Spot instances, a cost-efficient option that utilizes spare EC2 compute capacity at significant discounts. However, these instances carry the risk of interruptions (failures), with AWS providing a two-minute notification should the capacity be required elsewhere. This makes Spot instances an ideal solution for applications that are adaptable, can tolerate faults, or can manage potential disruptions, which is the case for *DynEMol*.

The cluster configuration is based on a *shared storage topology*, where multiple EC2 nodes access a shared storage system, ensuring they can all read and write data to the

[2]`https://github.com/lapesd/hpcac-toolkit/`

same set of files concurrently. A centralized storage approach simplifies data management and provides a consistent data view to all nodes. The performance, however, relies heavily on the network, depending on the type of device used to implement the shared storage system. In this paper, we discuss two options: Elastic Block Storage (EBS) and Lustre.

At a low level, EBS volumes operate like raw, unformatted block devices, which can be individually attached to the cluster's EC2 instances. Once attached, they are formatted with a file system. EBS volumes can also be configured in terms of Input/Output Operations Per Second (IOPS), and we test a variety of setups in our experiments. Our cluster design allocates a distinct EBS volume to each node, maintaining a consistent IOPS configuration across all volumes. To establish a shared file system, we implement a Linux Network File System (NFS) directory accessible throughout the entire cluster. The NFS server is hosted on an on-demand instance, ensuring consistent availability and eliminating the risk of unexpected disruptions arising from a Spot instance eviction, thereby preventing a single point of failure.

Lustre is a file system tailored for rapid processing, prevalent in HPC settings. For our cloud-based clusters, we utilize Amazon FSx for Lustre, a managed version of the Lustre file system. Unlike EBS volumes, which can only be attached to a single EC2 instance at once, FSx supports concurrent attachment to multiple instances, providing a ready-to-use, high-performance shared file system. A notable distinction between EBS-based shared storage and FSx, beyond the financial implications, is that the latter is equipped with its dedicated computational infrastructure, not requiring the allocation of resources and management of an NFS server in one of the cluster nodes, further improving performance. EC2 costs can also be slightly reduced when using Lustre, as all nodes can be based on Spot instances.

3.2. Network Infrastructure

AWS provides a robust and scalable network infrastructure for EC2 instances. At its core is the Virtual Private Cloud (VPC), a logically isolated section of the underlying AWS cloud infrastructure where resources, including EC2 instances, can be launched. It enables configurations of resources like subnets, route tables, and network gateways, mimicking conventional network elements. We have opted to deploy a *single Availability Zone VPC* for our cloud cluster architecture. This ensures all nodes are housed within the same physical data center, minimizing geographic spread and, consequently, reducing latency.

Moreover, when launching multiple EC2 instances, AWS strategically positions them to ensure a spread distribution across the underlying hardware, mitigating the risk of simultaneous failures. Although ideal for enterprise web applications focused on fault-tolerance through redundancy, this default behavior results in high network latency, damping the performance of tightly-integrated node-to-node communications commonly seen in HPC applications, including *DynEMol*. To improve this, we configured the EC2 service to pack instances within a singular Availability Zone using the AWS placement groups feature, thus further improving the network performance.

The basic network interface provided by AWS is called Elastic Network Adapter (ENA), which provides bandwidth capabilities above 100 Gbps for some specific instances. AWS also offers an improved network interface to be attached to EC2 instances named Elastic Fabric Adapter (EFA). EFA provides all of the functionality of an ENA but is specifically designed to boost network performance by allowing instances to bypass the typical TCP/IP stack when communicating with each other, resulting in lower network latency. EFA is an optional EC2 networking feature that can be enabled at no additional cost. A downside is that a limited number of operating systems and EC2 instance types are compatible with EFA adapters.

4. Evaluation Method

4.1. Experimental Environment

In this study, we employed a conventional HPC cluster (named ON-PREMISE in our experiments) housed within the Department of Physics at the University of Santa Catarina (UFSC). We used four homogeneous nodes, each powered by Intel Xeon E5–2687W CPUs equipped with 16 physical cores with hyperthreading enabled (32 virtual cores). These nodes are interconnected via a high-speed InfiniBand network. We leverage InfiniBand's Remote Direct Memory Access (RDMA) capabilities, which allow direct data transfers with the RAM, achieving high-speed data transfers. The cluster operating system is based on the Linux Ubuntu 20.04.4 LTS distribution.

We also assessed a variety of AWS public cloud clusters, all configured to run with the same MPI distribution and compilers from the Intel OneAPI package, mirroring the setup of the ON-PREMISE cluster. Although having two extra physical cores, we opted for the `c5n.9xlarge` instance type due to its close resemblance to the ON-PREMISE nodes. Table 1 shows the cluster configurations used in the experimental evaluation. In AWS, we considered clusters composed of on-demand instances (named ON-DEMAND in our experiments) and Spot instances (named SPOT in our experiments). We also considered different storage technologies (EBS and FSx) and interconnection technologies (ENA and EFA) provided by AWS.

We executed preliminary tests varying the number of MPI ranks per cluster node and number of OpenMP threads per rank, which revealed that *DynEMol* attained the best performance when executed with four MPI ranks and four OpenMP threads per cluster node. We thus kept this configuration as the standard one for all of the remaining experiments. To investigate the performance implications of *DynEMol*'s checkpointing mechanism, we measured the time required to save checkpoint files and restart execution from a checkpoint.

4.2. Evaluated Scenarios

We carried out simulations using *DynEMol* that describe the vibrational relaxation of photoexcited molecular systems, whereby a photon excites the molecular system from the ground quantum-state to an unoccupied quantum-state of higher energy, thus leaving the ground state unoccupied (with an excitation known as a "hole"). As the simulation

Table 1. Cluster configurations evaluated.

Platform	Nodes	Cores* per Node	Infrastructure Type	Storage	Network Adapter
AWS	4	36	VM – On-demand	EBS io2	ENA (TCP/IP)
	4	36	VM – Spot	EBS io2	ENA (TCP/IP)
	4	36	VM – Spot	EBS io2	EFA
	4	36	VM – Spot	FSx	EFA
On-Premise	4	32	Baremetal	SSD	InfiniBand

*Virtual (logical) cores with hyperthreading enabled.

Table 2. Workload sizes.

Workload Size	Quantum Atoms	Orbitals	Force Pairs	Time-Steps (t_s)	Simulated Time
Small	35	95	595	$100,000$	1 ps
Medium	451	$2,351$	$4,656$	100	10^{-3} ps
Large	628	$3,412$	$4,005$	100	10^{-3} ps

progresses, the photo-excited high-energy electron decays back to the ground state and eventually annihilates the hole. In these simulations, electronic decay occurs through the generation of vibrational states on the molecular arrangement. Therefore, electronic energy is converted into vibrational energy of the molecular system.

We executed three different molecular simulations using the parameters described in Table 2. Only one isolated molecule is in the gas phase in the small system. The medium and large systems comprise a molecular dye attached (anchored) to the surface of a titanium dioxide (TiO2) anatase cluster. In these cases, in addition to generating vibrations in the molecule and the cluster arrangements, the photoexcited electron is transferred from the molecular dye into the TiO2 cluster (the process is called interfacial electron transfer). This type of inorganic substrate sensitized with molecular dyes is used in photoelectrochemical fuel cells.

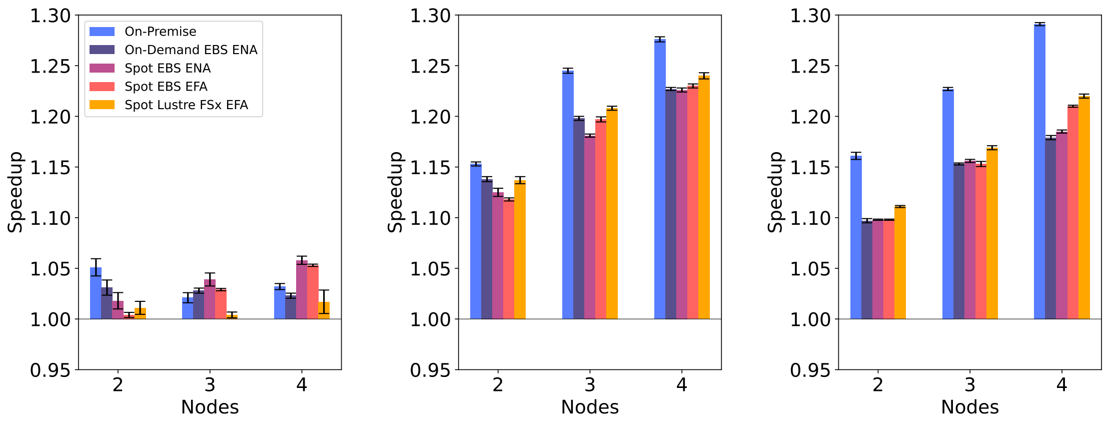
We pursue the following primary objectives when running *DynEMol* across diverse cluster architectures in AWS (ON-DEMAND and SPOT) as well as on the on-premise HPC cluster (ON-PREMISE): (i) assess both the performance enhancements and cost-effectiveness of FSx for Lustre relative to EBS; (ii) investigate the performance benefits derived from employing ENA and EFA to optimize network communications; and (iii) gauge the performance overhead associated with *DynEMol*'s fault tolerance mechanism when a variable number of simulated Spot evictions occur during the execution of the application.

For each scenario, we compute the performance improvements (speedups) achieved by *DynEMol* when multiple nodes of the cluster are used. These speedups are relative to the parallel execution of *DynEMol* using a single node of the cluster. All speedups of SPOT clusters are relative to the non-optimized SPOT cluster variant (EBS/ENA). To evaluate the cost-effectiveness of each cluster configuration, we vary the number of nodes and induce spot instance evictions, analyzing the number of executed time-steps per USD spent.

The frequency of checkpoints, denoted by f, is determined for each SPOT cluster according to the time-steps t_s, as detailed in Table 2. The frequency is calculated using the function $f(t_s) = \frac{t_s}{10}$. We do not make checkpoints when running *DynEMol* on ON-DEMAND and ON-PREMISE clusters. All results were derived from the average of 3 executions per experiment.

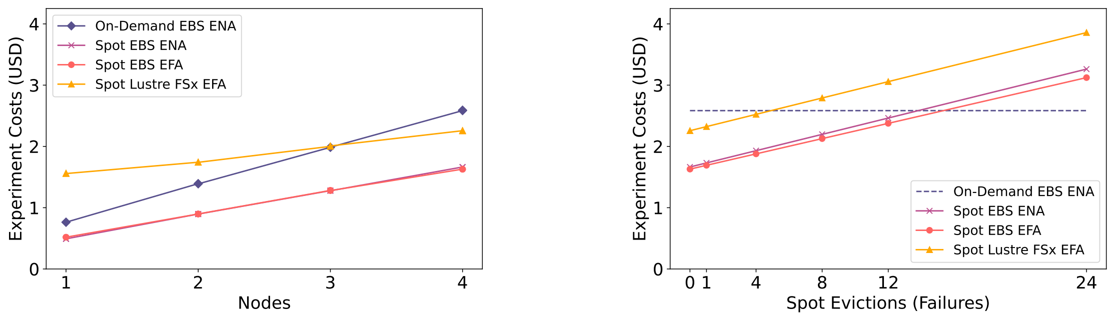
5. Experimental Results

Figure 2 showcases the results obtained from the three workload sizes (Table 2) across the five cluster configurations (Table 1). This initial analysis does not consider the impact of spot instance evictions. As noted in Figure 2a, horizontal scaling performed poorly for small workloads, producing a negligible speedup on all clusters. The enhanced networking and storage solutions on the SPOT cluster (EFA and FSx) did not yield significant improvements for small workloads either. Results in Figure 2 presents that a better horizontal



(a) Small Workload **(b)** Medium Workload **(c)** Large Workload

Figure 2. Speedups achieved with different cluster configurations.



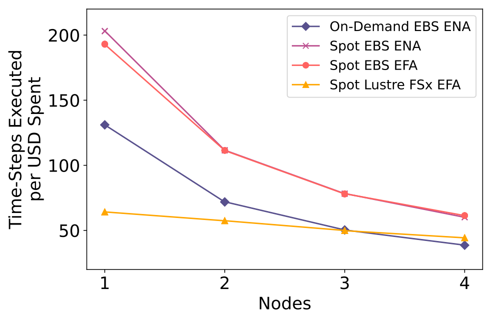
(a) Increasing Nodes (0 failures) **(b)** Increasing Failures (4 nodes)

Figure 3. Execution costs (large workload).

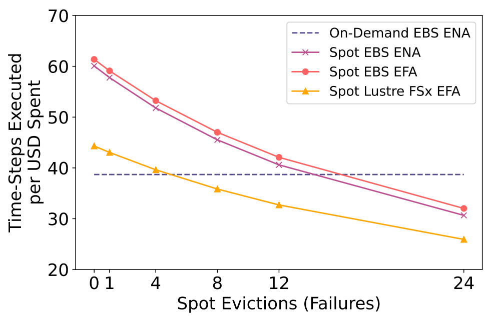
scaling was observed with medium and large workloads for all cluster configurations, achieving speedups of up to 1.22x (FSx) and 1.29x (ON-DEMAND clusters) over 4-node clusters. The advantages of using EFA are more apparent for larger workloads, producing up to 9% increase in performance compared to ENA.

Figure 3 illustrates the monetary costs (in US dollars) of running *DynEMol* on AWS with the large workload across a variable number of nodes with no spot evictions (Figure 3a) and with four nodes and a variable number of spot evictions (Figure 3b). Based on the results in Figure 3a, we can conclude that FSx becomes cost-effective when running larger workloads in large clusters. Although spot instances can be much cheaper than on-demand infrastructure, the performance impact of the fault tolerance strategy can easily translate to higher infrastructure costs (Figure 3b). The checkpoint restoration process employed in *DynEMol* is CPU-intensive, demanding little I/O throughput for performance. This is reflected in the steep cost increase for experiments with FSx (Figure 3b), indicating that experiencing more than four spot evictions could lead to costs higher than simply opting for on-demand instances.

Figure 4 presents the ratio between executed time-steps and each US dollar spent on cloud resources. Overall, SPOT clusters offer a much more cost-effective alternative



(a) Increasing Nodes (0 failures)



(b) Increasing Failures (4 nodes)

Figure 4. Time-steps per USD spent (large workload).

than ON-DEMAND clusters when there are no spot evictions (Figure 4a), executing up to 54% more time-steps for the same price. However, the cost-efficiency of SPOT clusters undergoes a steep decline with a rising number of failures (Figure 4b), potentially rendering the utilization of spot instances less advantageous. Still, SPOT clusters with EBS were cheaper than ON-DEMAND clusters with up to 12 failures.

Finally, we noticed that FSx is only better than EBS when employed in larger clusters (four nodes or more). Given that FSx is approximately 233% costlier than EBS, it may only be worth for heavy I/O applications.

6. Related Work

Several recent studies have examined the cost-effectiveness of utilizing Spot instances for HPC. Notably, *Zhou et al.* proposed *FarSpot*, a framework that centers on forecasting cloud infrastructure expenses using Machine Learning ensemble methods, such as *xgboost* [Zhou et al. 2022]. While aligning with the broader theme of cost optimization, our study serves a distinct purpose and offers a complementary perspective. Rather than developing a generalized model for predicting costs, we specifically explore and evaluate the application of a real-world HPC solution using fault-tolerant Spot instances, thus focusing on practical implementation and performance.

Teylo et al. conducted an in-depth assessment of AWS Spot instances in the context of scheduling bag-of-tasks applications [Teylo et al. 2021]. While their research is complementary to ours, our study extends the exploration into a different dimension. We present a comprehensive analysis of *DynEMol*, a tightly-coupled, real-world HPC application, which is a context that presents notable variances from bag-of-tasks applications.

Sharma and Jadhao conducted a comprehensive survey of the main challenges of executing molecular dynamics simulations using public cloud infrastructure [Sharma and Jadhao 2021]. Their work also touched on the integration of TensorFlow and other Machine Learning frameworks in the field of molecular dynamics simulation. While their investigation offers valuable insights, our study goes a step further, actively engaging in the migration and execution of molecular simulations using AWS Spot instances. This practical approach allows us to directly address and overcome some of the infrastructure management challenges highlighted by *Sharma and Jadhao*, thereby providing a more hands-on perspective.

Sena et al. presented a comprehensive exploration of the potential advantages users might extract from the diversity of instances and contract models offered by public cloud providers, aiming to reduce financial expenditure. Their research delineates a methodology for dynamically scheduling applications subject to deadline constraints across both Spot and persistent instances [Sena et al. 2023]. Rather than emphasizing scheduling strategies and pricing model analyses, we concentrate on the practical nuances of running real-world applications using the aforementioned infrastructure. Thus, our work offers valuable insights into the application side of leveraging such resources.

Brum et al. offers a comprehensive review of the fault tolerance techniques most commonly employed by cloud and HPC applications operating within these environments. Their focus primarily revolves around checkpoint-rollback and replication strategies, in addition to exploring fault detection approaches and existing reliable storage solutions within the cloud [Brum et al. 2023]. In our study, we empirically examine the checkpoint-rollback method specifically adapted for less reliable AWS Spot clusters. This analysis contributes practical insights to the existing theoretical landscape, enhancing our understanding of how these fault tolerance techniques perform in real-world cloud computing scenarios.

Finally, studies such as [Dancheva et al. 2023] and [Fernandez 2022] present a broad analysis of IaaS from public cloud providers, using micro and macro benchmarks to assess the performance of MPI operations across various vendors and architectures. Our research differs in several key aspects: (i) our research is centered explicitly around *DynEMol*, real-world HPC application, which is a distinct departure from the use of MPI benchmarks in the referenced studies; (ii) we address the challenges of migrating and managing such HPC applications in a public cloud environment, which is not explicitly addressed by *Dancheva et al.* and *Fernandez*; (iii) we focus on strategies for further reducing costs associated with cloud-based HPC, such as leveraging transient virtual machines like AWS Spot instances, rather than just relying on on-demand pricing models; and (iv) our study also delves into a practical application of a fault tolerance technique tailored for AWS Spot clusters, taking advantage of a eviction notification system to minimize checkpoint frequency and overhead. Therefore, our research offers a more applied perspective on cloud-based HPC, addressing both the performance and operational challenges associated with such an approach.

7. Conclusion and Future Work

In this paper, we migrated the *DynEMol* simulation tool to the cloud using the *HPC@Cloud* framework. We tested *DynEMol* over various cluster configurations, evaluating different available technologies, such as spot and on-demand instances for computational infrastructure; EBS and FSx for storage; and EFA and ENA for networking.

Our experimental results suggest that while cloud-based resources configured optimally exhibit slightly inferior scalability compared to the on-premise infrastructure tested, they still offer substantial benefits. The existing fault tolerance mechanism in *DynEMol*, specifically its checkpointing feature, is sufficiently lightweight and its performance footprint is virtually undetectable for medium to large workloads. This feature enables us to increase checkpointing frequency, thereby circumventing unnecessary rework when restoring from these checkpoints. Moreover, limiting the number of instances enhances the cost-effectiveness of spot infrastructure due to the associated reduction in failure probability. Utilizing spot instances and maintaining a finely-tuned parallel composition for the workload is crucial for achieving cost-effectiveness. However, if users need immediate access or a specific workload lacks a fault tolerance mechanism, they can utilize on-demand instances instead of spot instances, albeit at a higher cost.

Regarding the available storage in AWS, although FSx can be costly, the benefits outweigh the costs when a high number of nodes (at least four) perform parallel I/O. Furthermore, advances in networking technologies have paved the way for high-bandwidth, low-latency communications in the public cloud. When testing networking technologies, we observed better performance when using EFA, attesting to this fact. Furthermore, EFA can be enabled at no additional cost, further improving the speedup efficiency.

Future research includes a deeper analysis of *DynEMol*'s parallelization methods to further improve the obtainable speedups through code improvements and fine-tuning of the number of MPI ranks per node and OpenMP threads per MPI rank. We also intend to test the GPU-accelerated version of *DynEMol* using cloud resources, a different range of instance types, and cloud providers. Future objectives also include the development of a cloud-native version of *DynEMol* for easy deployment by any researcher in the field.

Acknowledgements

The authors would like thank the National Laboratory for Scientific Computing (LNCC/MCTI), whose resources have contributed to this research. This work was partially funded by the National Council for Scientific and Technological Development (CNPq) and Amazon Web Services (AWS) through the CNPq/AWS call Nº 64/2022 — Cloud Credits for Research.

References

Abraham, B., Rego, L. G. C., and Gundlach, L. (2019). Electronic–vibrational coupling and electron transfer. *The Journal of Physical Chemistry C*, 123(39):23760–23772.

Brum, R., Teylo, L., Arantes, L., and Sens, P. (2023). *Ensuring Application Continuity with Fault Tolerance Techniques*, pages 191–212. Springer International Publishing, Cham.

Buyya, R. et al. (2019). A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. *ACM Computing Surveys*, 51(5).

Carter, E. A. (2008). Challenges in modeling materials properties without experimental input. *Science*, 321(5890):800–803.

Ciccotti, G., Dellago, C., Ferrario, M., Hernández, E., and Tuckerman, M. (2022). Molecular simulations: past, present, and future (a topical issue in epjb). *The European Physical Journal B*, 95.

Dancheva, T., Alonso, U., and Bartoň, M. (2023). Cloud benchmarking and performance analysis of an hpc application in amazon ec2. *Cluster Computing*, pages 1–18.

De Vivo, M., Masetti, M., Bottegoni, G., and Cavalli, A. (2016). Role of molecular dynamics and related methods in drug discovery. *Journal of Medicinal Chemistry*, 59(9):4035–4061. PMID: 26807648.

Evers, F., Korytár, R., Tewari, S., and van Ruitenbeek, J. M. (2020). Advances and challenges in single-molecule electron transport. *Rev. Mod. Phys.*, 92:035001.

Fernandez, A. (2022). Evaluation of the performance of tightly coupled parallel solvers and mpi communications in iaas from the public cloud. *IEEE Transactions on Cloud Computing*, 10(4):2613–2622.

Marcus, R. A. (1993). Electron transfer reactions in chemistry. theory and experiment. *Rev. Mod. Phys.*, 65:599–610.

Mell, P. M. and Grance, T. (2011). SP 800-145. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, USA.

Munhoz, V. and Castro, M. (2022). HPC@Cloud: A provider-agnostic software framework for enabling hpc in public cloud platforms. In *Anais do Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)*, pages 157–168, Florianópolis. Brazilian Computer Society.

Munhoz, V., Castro, M., and Mendizabal, O. (2022). Strategies for fault-tolerant tightly-coupled hpc workloads running on low-budget spot cloud infrastructures. In *IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 263–272, Bordeaux. IEEE Computer Society.

Netto, M., Calheiros, R., Rodrigues, E., Cunha, R., and Buyya, R. (2018). HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges. *ACM Computing Surveys*, 51.

Ollitrault, P. J., Miessen, A., and Tavernelli, I. (2021). Molecular quantum dynamics: A quantum computing perspective. *Accounts of Chemical Research*, 54(23):4229–4238. PMID: 34787398.

Sena, A. C., Boeres, C., Teylo, L., Drummond, L. M. A., and Rebello, V. E. F. (2023). *Harnessing Low-Cost Virtual Machines on the Spot*, pages 163–189. Springer International Publishing, Cham.

Sharma, P. and Jadhao, V. (2021). Molecular dynamics simulations on cloud computing and machine learning platforms. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 751–753.

Shu, Y., Zhang, L., Sun, S., and Truhlar, D. G. (2020). Time-derivative couplings for self-consistent electronically nonadiabatic dynamics. *Journal of Chemical Theory and Computation*, 16(7):4098–4106.

Teylo, L., Arantes, L., Sens, P., and Drummond, L. M. d. A. (2021). Scheduling Bag-of-Tasks in Clouds using Spot and Burstable Virtual Machines. *IEEE Transactions on Cloud Computing*, pages 1–1.

Torres, A., Prado, L. R., Bortolini, G., and Rego, L. G. C. (2018). Charge transfer driven structural relaxation in a push–pull azobenzene dye–semiconductor complex. *The Journal of Physical Chemistry Letters*, 9(20):5926–5933.

Wang, C., Liang, Q., and Urgaonkar, B. (2018). An empirical analysis of amazon ec2 spot instance features affecting cost-effective resource procurement. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 3(2).

Yu, Q., Alonso, A. M., Caminiti, J., Beck, K. M., Sutherland, R. T., Leibfried, D., Rodriguez, K. J., Dhital, M., Hemmerling, B., and Häffner, H. (2022). Feasibility study of quantum computing using trapped electrons. *Phys. Rev. A*, 105:022420.

Zhou, A. C., Lao, J., Ke, Z., Wang, Y., and Mao, R. (2022). Farspot: Optimizing monetary cost for hpc applications in the cloud spot market. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2955–2967.