Abstract—Complex flow simulations are conventionally performed on HPC clusters. However, the limited availability of HPC resources and steep learning curve of executing on traditional supercomputer infrastructure has drawn attention towards deploying flow simulation software on the cloud. We showcase how a complex computational framework—that can evaluate COVID-19 transmission risk in various indoor classroom scenarios—can be abstracted and deployed on cloud services. The availability of such cloud-based personalized planning tools can enable educational institutions, medical institutions, public sector workers (courthouses, police stations, airports, etc.), and other entities to comprehensively evaluate various in-person interaction scenarios for transmission risk. We deploy the simulation framework on the Azure cloud framework, utilizing the DENDRO-KT mesh generation tool and PETSc solvers. The cloud abstraction is provided by RocketML cloud infrastructure. We compare the performance of the cloud machines with state-of-the-art HPC machine TACC Frontera. Our results suggest that cloud-based HPC resources are a viable strategy for a diverse array of end-users to rapidly and efficiently deploy simulation software.

I. MOTIVATION

Current and next-generation high-performance computing (HPC) resources play a pivotal role in helping answer many of the fundamental questions related to science and technology. However, the limited availability of HPC resources, along with the steep learning curve needed to employ these resources effectively, have limited the utilization of HPC resources to academic and large industrial R&D groups. Additionally, standard HPC resource utilization policies restrict the number of nodes available to the user and the number of job instances per user. These limitations become a bottleneck, especially in risk assessment and design exploration, where parametric studies need to be performed that involve executing multiple instances of the simulation with different simulation inputs. In this context, the availability of on-demand cloud services suggests an attractive alternative for adaptive, interactive scientific computing workloads.

The ubiquitous availability of cloud computing suggests a path to the democratization of complex multi-physics simulations. Simulations—for example, parametric studies of ventilation rates and seating arrangements in critical public places such as courthouses, hospital waiting rooms, etc. to evaluate transmission risk—are currently limited to deployment on HPC clusters that require a steep learning curve to deploy, which disenfranchises non-computer savvy scientists, decision-makers, and designers. Additionally, the data visualization and analysis can be done in-situ without copying data from one place to another. In this work, we propose a public cloud-based framework with in-situ visualization and post-processing capabilities for scalable deployment of fluid dynamics simulation using FEM discretization. Our results suggest that the performance of such a framework is at par, if not better, with the framework executed on state-of-the-art HPC resources.

II. BACKGROUND

A large fraction of engineered and natural systems are analyzed using scientific simulation codes that involve numerical solutions of partial differential equations (PDE). The Finite Element Method (FEM) is a widely popular approach for the numerical solution of PDEs in complex domains, with multiple $ billions/year spent in CAD and CAM (computer-aided design and manufacturing) based FEM software alone [1]. The popularity arises from a compelling set of properties, including the ability (a) to model arbitrary geometries, (b) to change the order of representation (linear, quadratic, and higher-order), (c) to utilize variational arguments that guarantee monotonic convergence to the solution with improved discretization, and (d) to seamlessly utilize a posteriori error estimates to adapt the mesh. Over the past several years, we have developed a scalable FEM solver based on adaptive octree mesh and
deployed this framework for a diverse array of multi-physics simulations—primarily on HPC resources [2–8].

In this work, we explore the deployment (and assess performance) of this FEM framework on a public cloud computing facility. Cloud computing provides on-demand access to computing resources, data storage, development, and networking facilities via the internet. Cloud computing offers some key benefits compared to the traditional HPC settings, in particular:

- **Lower IT costs**: Cloud facilities enable the offloading of some or most of the costs and effort of purchasing, installing, configuring, and managing on-premises infrastructure. This can be especially significant for small research groups.

- **Scale on demand**: Instead of purchasing the capacity and waiting for the arrival of hardware, the cloud enables on-demand access and elastic scalability.

- **Choice of architecture**: Cloud allows choosing from different types of architecture, depending on the nature of the codes. Some computer architectures are tailored for compute-bound codes, whereas others provide good performance for memory-bound codes. A traditional HPC cluster usually has limited flexibility in terms of the availability of different types of machines. In contrast, cloud computing allows the flexibility to choose the machines based on the nature of the underlying code (compute-bound vs. memory-bound vs. communication-bound).

- **End-to-end deployment**: Cloud enables the full stack of software, starting from development to testing to execution to visualization. The latter aspect—visualization—is particularly appealing, as it allows for in-situ visualization and analysis without extensive data movement. We note that some federal HPC clusters, like TACC, have a dedicated visualization portal but are not accessible on all of them.

In this paper, we seek to address the following three components of the workshop:

- On-demand and interactivity with performance, scaling, and cost efficiencies;
- Application environment, integration and deployment technologies;
- Workflow orchestration using public cloud and HPC data center resources.

### III. Problem Definition

COVID-19 has irreversibly changed how we consider transmission risk in indoor environments. The Six-Foot Rule, for instance, is a guideline that does not account for small aerosol droplets that are continuously mixed through an indoor space. The distribution of aerosolized particles depends on a wide array of factors (ventilation, airflow patterns that are impacted by furniture, as well as the respiratory activity of the inhabitants. While conventional risk assessment tools may indicate that a particular room is low-risk on average, there may be specific locations in the room with significantly higher risk for transmission—for example, where there is local recirculation causing limited air exchange with the outside environment. Assessing the localized risk becomes especially important if individuals are seated in such locations for extended periods, increasing their cumulative exposure time.

We consider scenarios that impact first-responders, essential workers, and K-12 students. For instance, courthouse activities may indicate that a particular room is low-risk, impacting by furniture, as well as the respiratory activity of the inhabitants. While conventional risk assessment tools may indicate that a particular room is low-risk on average, there may be specific locations in the room with significantly higher risk for transmission—for example, where there is local recirculation causing limited air exchange with the outside environment. Assessing the localized risk becomes especially important if individuals are seated in such locations for extended periods, increasing their cumulative exposure time.

We consider scenarios that impact first-responders, essential workers, and K-12 students. For instance, courthouse activities require participants (judges, clerks, petitioners, jurors) to remain sedentary over long periods. Similarly, most classroom activities require K-12 students to stay seated for extended periods. In such cases, it becomes imperative to identify if specific locations have a higher risk and rank among alternate seating arrangements.

In this context, high-fidelity fluid/aerosol simulation is a viable approach to evaluate various seating and operating scenarios, identify risk factors, and rank order mitigation strategies. We aim to democratize such simulations on cloud-based resources by integrating four sophisticated concepts to provide very detailed estimates of aerosol distribution in indoor environments: (a) a detailed flow physics simulation (called Large Eddy Simulations) that produces significantly more accurate results while taking reasonable compute time, (b) accounting for the thermal plumes created by heat released by humans and electronics (which changes the flow in rooms), (c) immerso-geometric analysis that allows rapid exploration of a diverse array of configurations with complex geometries, (d) a passive scalar transport based model that tracks the spatial distribution of the aerosolized concentration. To our best knowledge, parametric simulations of such high-fidelity simulations have not been deployed on the cloud. The availability of such a cloud-based analysis approach will allow education administrators (K-12), federal/state/local government officials (courthouses, police stations), and hospital administrators to make informed decisions on seating arrangements and operating procedures.

In this work, we use DENDRO-KT for efficient and automatic mesh generation. DENDRO-KT is an extension of previous DENDRO versions to incomplete octrees. Tree-based grid generation (quadrees in 2D and octrees in 3D) is common in computational sciences [4, 9–16] largely due to its simplicity and parallel scalability. The ability to efficiently refine (and coarsen) regions of interest using tree-based data structures have made it possible to deploy them on large-scale multi-physics simulations [2–4, 17–21]. The use of incomplete octrees enables the handling of any arbitrary complex geometries, where the stated geometry is carved out from the domain (Fig. 1). The framework’s simplicity and robustness lie in the fact that complete mesh generation only needs a specification of IN or OUT (of the geometry) for any queried point. Furthermore, the efficient MATVEC computation is performed by traversing the trees rather than using any mesh-based data structure to prevent indirect memory access.

Once the mesh is created, we model the time-dependent transmission of the viral load as a scalar transport equation that is advected by a statistically steady-state flow field obtained from the solution of coupled Navier–Stokes and Heat transfer solver. The well-established Variational Multiscale (VMS) FEM [22] is used for the discretization. DENDRO-KT
interfaces with PETSc for solving the system of equations. We have recently extensively validated and demonstrated the capabilities of this framework [3, 8].

IV. WORKFLOW

In this section, we detail and demonstrate the deployment of our framework on the cloud infrastructure.

A. Cloud Abstraction

Our target user for a cloud orchestrator is a researcher who has experience using an HPC cluster at a university or a national lab and has never used cloud computing. Ease-of-use and flexibility of using public cloud computing resources are the two most important requirements. Specifically, a cloud orchestrator that provides a graphical user interface (GUI) to 1) deploy and manage CPU- and GPU-based HPC clusters pre-installed with MPI, BLAS/LAPACK, and PETSc libraries along with a Slurm scheduler to simplify job management, 2) access to different IDEs like Jupyter Lab, VSCode for coding, compiling, and debugging HPC software, and 3) access to visualization software like Paraview for in situ data visualization.

We have explored several options available on Microsoft Azure and AWS marketplaces [23, 24] and concluded that they either lack in ease-of-use or flexibility necessary for a researcher who has no exposure to cloud computing to conduct large-scale HPC experiments. In Figure 2, we show different components of RocketML's cloud orchestrator that has been purpose-built to address a researcher’s needs to take complete advantage of HPC on a public cloud for scientific computing applications.

B. Mesh Creation

We first begin with mesh creation. As mentioned in Section II, DENDRO-KT framework requires the specification of IN and OUT for a given queried point. The IN-OUT test is performed using Ray-Tracing. The input geometries are provided in the form of a .stl file format. Several instances of the same geometry, translated by some distance, can be grouped. An example of a config file for such an operation is:

```python
geometries = {
    mesh_path = "stl/human.stl"
    displacements = {
        position = [0.65, 0.0, 0.8125],
        position = [1.625, 0.0, 0.73125],
    }
    refine_lvl = 9
}
```

where displacements are the shift from original position of .stl and refine_lvl is the refinement level of the octree near the .stl. The octree generation starts with a coarse mesh and is subsequently refined until the required resolution is satisfied. We can similarly add multiple geometries. This approach gives an automated way to generate mesh for different scenarios. In the future, we plan to interface it with a GUI, where an application scientist can use the drag-and-drop feature to add multiple objects or to select the refine regions. Fig. 3 shows the adaptive mesh generated for the classroom case.

C. Solver

We use PETSc as the linear solver for solving the system of equation AX=b. PETSc is a very well validated scalable linear algebra solver, which has a rich collection of preconditioners and solvers. Different PETSc options are configurable from the config file itself. An example is shown below:

```python
solver_options_ht = {
    ksp_max_it = 500
    ksp_type = "bcgs"
    pc_type = "asm"
    ksp_atol = 1e-15
    ksp_rtol = 1e-15
    ksp_converged_reason = ""
}
```

D. File I/O

We use a parallel .vtu file format to write the data into the file at the user-defined interval. .vtu is an XML file format which enable a communication free write. Each processor writes its share of data in a separate file. All these files are later joined by another single .pvtu file, which is written by 0th rank processor.

E. Checkpointing

Machine failure is a common scenario when executing simulations on HPC servers. DENDRO-KT dumps the checkpoint file at user-defined intervals to keep a backup of the solutions. The checkpoint files are binary files that contain information about octree tree nodes and the solution vector. Similarly, to file I/O, each processor dumps its chunk of data in a communication - free fashion. If the code is killed for whatsoever reason, the user can restart the simulation by loading the checkpoint file.
Fig. 2. RocketML cloud orchestrator for large-scale CFD simulations.

Additionally, the user has the option to increase the number of nodes once the checkpoint is loaded. DENDRO-KT will repartition the domain if the number of processors is increased. At this point, we have not implemented the ability to restart with a decreased number of processors.

F. Visualization

We have deployed ParaView within the cloud framework for in-situ visualization and related data-analysis. ParaView comes with pvpython (serial version) and pvpbatch (parallel version) which provide a seamless python interface.

As noted earlier, we rely on the config.txt file to pass input operation to the code. In future, we plan to replace it with a GUI interface hosted on the cloud platform for better interaction with the application user.

V. RESULTS

A. Roofline Analysis

As the first step, we conducted a roofline analysis on Frontera using Intel advisor to check whether the framework is compute-bound or memory-bound. Identifying this is important to identify and specify the type of cloud machines for deploying the framework. Fig. 4 shows the single-core roofline plot for the elemental MATVEC computation using linear basis function on Frontera. We limit ourselves to using linear basis functions as this mimics our application problem. We can see that the code is memory-bound, which is along the lines of what we expect for any finite element code. Overall, we can achieve a performance of about 4 GFLOP/s with an Arithmetic Intensity of 0.072, which corresponds to a bandwidth of approximately 60 GB/s.
Remark. We have not used any hand-coded explicit vectorization to ensure the portability of the code across various platforms and have relied on compiler directives for vectorization.

B. Performance on the Cloud

Based on the roofline analysis discussed in the previous section, we chose Azure virtual machines that are suited for memory-bound computational problems. This is an additional benefit of using cloud, where we can opt for a virtual machine with specific hardware configuration based on computational complexity of the problem at hand (memory-bound vs. compute-bound). Specifically, we chose the Azure Da and HBv2 series virtual machines that are built for memory-bound codes and compared the time-to-solution with the state-of-the-art HPC cluster Frontera. Tab. I provides the functional specifications of Frontera and virtual machines on cloud.

**Scaling behavior comparison:** To study the scaling behavior and time comparison of our solver on these machines, we consider the classroom case (shown in Fig. 3) and solve the Navier–Stokes equation coupled with Heat Transfer. In both the azure machines (Da series and HBV2 series), we only used 50% of the available CPU cores per node (48 for Da series and 60 for HBV2 series) mentioned in the Tab. I. We note that the specs take the hyper-threading into account for Da-series with 2 threads per core. Since our code is only MPI parallelized, we are able to use only half of the available cores. On Frontera we used all 56 cores available per node. The overall mesh for scaling analysis consisted of around 790 K elements with higher refinement near the body (mannequin/tables) and inlet regions. We compared the three main components of the solver: matrix assembly, vector assembly, and the total solve time. Fig. 5 shows the parallel cost (defined as total time multiplied by the number of nodes used) and time taken on these three different machines. A straight line parallel to X-axis indicates ideal scaling. We can see that the cloud machines take significantly less time than Frontera, with the HBV2 series taking the least amount of time. In particular, 10 time steps of the solver took 432s on 1 node of Frontera, while it took 132s on 1 node of HBV2 series machine.

**Explanation of observed scaling behavior:** The two main architectural factors responsible for the difference between Azure machines and Frontera are memory bandwidth and cache size. The total peak memory bandwidth on the HBV2 node is about 340GB/s and around 210GB/s on a Frontera CLX node. For memory-bound workloads like CFD simulations, the higher memory bandwidth on the HBV2 node would result in a better solve time. Each AMD EPYC 7742 CPU in HBV2 VMs has 256MB of L3 cache divided across 16 core complexes with four cores each (4MB L3 per core, 4.5MB L2+L3 per core). Conversely, the Intel 8280 has only 38.5MB of L3 cache (1.4MB L3 per core, 2.4MB L2+L3 per core). Codes that benefit from high cache reuse would enormously benefit from the AMD Zen2 architecture, resulting in a much higher “effective” memory bandwidth. This indicates that an educated choice of the architecture for the code can lead to a significant boost in the performance. Such flexibility is offered...
only by cloud, and not on traditional HPC based clusters.

**Remark.** We note that our code is general-purpose and does not contain any explicit vectorization cache blocking or streaming operation, ensuring code portability. This approach is common in most of the FEM / fluid dynamics code. Explicit optimization such as vectorization/cache blocking can significantly boost the time to solve but comes at the cost of portability on different machines. Comparing the performance of hardware-specific optimized codes on these machines is beyond the scope of the work.

**Parallel efficiency:** We also see a good parallel efficiency on all these machines. Overall **Frontera** provides the best parallel efficiency (0.88) with the solve time decreasing from 432s on 1 node to 122s on 4 nodes. The cloud machines have a relatively low parallel efficiency of about 0.65. The deviation from ideal scaling can be attributed to the increased iteration count for the preconditioned solve. Interested readers are referred to [3] for detailed discussion. Briefly, it is expected that bare-metal HPC servers will have better parallel efficiency than virtual-machine-based cloud HPC servers due to increased communication costs across nodes. Optimizing multi-node VM-based HPC servers suggests the need to deploy new algorithms that minimizes communication at the cost of increased (and/or redundant) computations.

**Memory bandwidth effect:** Memory bandwidth plays a pivotal role in overall solve time, especially for memory bound codes. Therefore, we compare the effect of memory bandwidth on our solver. The number of processors is kept constant at 120, whereas the number of nodes is varied from 1 to 4 with a varying number of processors per node. Note that 4 nodes have 30 processors per node, and 1 node has 120 processors. With the increase in the number of processors per node, the memory bandwidth per core decreases. For instance, on the HB120rs_v2 (HBV series) machine, the memory bandwidth per core decreases from 11.50 GB/s when using 30 cores/node to 2.88 GB/s while using all 120 cores. [25]. Fig. 6 shows the effect of the memory bandwidth of our solver. We see a strong correlation between memory bandwidth and solve time. With the increase in the memory bandwidth, the solve time decreases from around 112 s with 30 cores/node to 80 s on 30 cores/node. However, we note that this reduction in time comes at an increased cost of utilizing more nodes.

**VI. Application Results**

In this section, we illustrate results of our application case. We consider a typical university classroom as illustrated in Fig. 3. The room has a length of 9 m, a width of 9 m, and a height of 3.5 m with a ventilation system installed on the ceiling. The occupants are modeled using mannequins. A representative resolved mesh is shown as an inset in this figure. The Reynolds number based on the inlet air velocity and classroom height is estimated to be $10^5$. The spatial discretization was performed using linear basis function and the time discretization was performed using second order BDF2. A dimensionless timestep of 1E-3 was used to carry out the simulation. Each scenario takes up to a final time of 2 dimensionless time units to reach the stationary steady state.
Without computer monitors

With computer monitors

Everyone wearing masks

Increased occupancy

Fig. 8. Classroom scenario: Evaluation of inhaled particles in different classroom scenarios. The mannequin marked in the red is infected with COVID and transmits the virus. The isocontours represent the regions of different inhaled particle concentrations in space.

(2K timestep), which required about a day of compute time on HBV2 series machine. This resulted in the total cost of $340 for simulating each scenario.

We evaluate the impact of one infected individual who coughs, releasing an aerosolized load of viral particles. Fig. 7 plots the time evolution of the viral load concentration of the air due to a simulated cough by a mannequin. Notice that the aerosolized concentration plume rises and is recirculated by the inlet air. The mixing with the inlet air dilutes the virus concentration, but the inlet air helps spread the concentration across the room, potentially impacting everyone in the classroom.

Finally, Fig. 8 shows the isocontours of viral particles concentration (at one time-point) for different operating scenarios. We compare the different scenarios in the classroom with (Fig. 8a) and without (Fig. 8b) monitors, increased occupancy (Fig. 8d) and the situation where everyone wears a mask (Fig. 8c). The mannequin colored in red is the infected person. The regions of contours with inhaled particles > 50 [26] are considered at a high risk of transmission. We can see different scenarios lead to significantly different regions of risk. Additionally, the wearing of masks significantly reduced the risk of transmission and should therefore be encouraged. We defer a detailed analysis and implication of these results to a companion publication.

VII. CONCLUSIONS

In this work, we deploy a scalable, adaptive framework on the cloud utilizing DENDRO-KT as a mesh generation tool and PETSc as a scalable linear algebra solver to analyze the risk of COVID-19 transmission in different classroom scenarios. The problem utilizes running different simulation instances, which are easily deployable in a cloud-based setting. Further, we analyzed the performance of the cloud-based resources to the state-of-the-art supercomputer, specifically Frontera. With the proper choice of cloud machines, we observe on par performance on the cloud, if not better, with the supercomputers. This suggests that cloud based VM HPC servers offer a viable approach to democratize and deploy complex simulation workflows. In the future, we would like...
to simplify the deployment of the use-cases with more GUI-friendly interfaces.

VIII. ACKNOWLEDGMENTS

We acknowledge partial support from NSF 1855902.

REFERENCES

[1] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs, *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[2] M. A. Khanwale, K. Saurabh, M. Fernando, V. M. Calo, J. A. Rossmanith, H. Sundar, and B. Ganapathysubramanian, “A fully-coupled framework for solving cahn-hilliard navier-stokes equations: Second-order, energy-stable numerical methods on adaptive octree based meshes,” arXiv preprint arXiv:2009.06628, 2020.

[3] K. Saurabh, B. Gao, M. Fernando, S. Xu, M. A. Khanwale, B. Khara, M.-C. Hsu, A. Krishnamurthy, H. Sundar, and B. Ganapathysubramanian, “Industrial scale large eddy simulations with adaptive octree meshes using immersogeometric analysis,” *Computers & Mathematics with Applications*, vol. 97, pp. 28–44, 2021.

[4] M. Fernando, D. Neilsen, H. Lim, E. Hirschmann, and H. Sundar, “Massively parallel simulations of binary black hole intermediate-mass-ratio inspirals,” *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. C97–C138, 2019.

[5] S. Xu, B. Gao, A. Lofquist, M. Fernando, M.-C. Hsu, H. Sundar, and B. Ganapathysubramanian, “An octree-based immersogeometric approach for modeling inertial migration of particles in channels,” *Computers & Fluids*, vol. 214, p. 104764, 2021.

[6] D. Neilsen, M. Fernando, H. Sundar, E. Hirschmann, and H. Lim, “Massively parallel simulations of binary black hole intermediate-mass-ratio inspirals,” *APS*, vol. 2018, pp. D14–005, 2018.

[7] D. Neilsen, M. Fernando, H. Sundar, and E. Hirschmann, “Dendro-gr: A scalable framework for adaptive computational general relativity on heterogeneous clusters,” in *APS April Meeting Abstracts*, vol. 2019, 2019, pp. G11–003.

[8] K. Saurabh, M. Ishii, M. Fernando, B. Gao, K. Tan, M.-C. Hsu, A. Krishnamurthy, H. Sundar, and B. Ganapathysubramanian, “Scalable adaptive pde solvers in arbitrary domains,” arXiv preprint arXiv:2108.03757, 2021.

[9] T. Tu, D. R. O’Hallaron, and O. Ghattas, “Scalable parallel octree meshing for terascale applications,” in *SC’05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE, 2005, pp. 4–4.

[10] M. Ishii, M. Fernando, K. Saurabh, B. Khara, B. Ganapathysubramanian, and H. Sundar, “Solving pdes in spacetime: 4d tree-based adaptivity, mesh-free and matrix-free approaches,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–61.

[11] H. Sundar, R. S. Sampath, and G. Biros, “Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel,” *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2675–2708, 2008.

[12] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander, “A generic grid interface for parallel and adaptive scientific computing, part i: abstract framework,” *Computing*, vol. 82, no. 2–3, pp. 103–119, 2008.

[13] D. M. Greaves and A. Borthwick, “Hierarchical tree-based finite element mesh generation,” *International Journal for Numerical Methods in Engineering*, vol. 45, no. 4, pp. 447–471, 1999.

[14] M. Bader, *Space-filling curves: an introduction with applications in scientific computing*. Springer Science & Business Media, 2012, vol. 9.

[15] C. Burstedde, L. C. Wilcox, and O. Ghattas, “p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees,” *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, 2011.

[16] S. Popinet, “ Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries,” *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, 2003.

[17] M. W. Akhtar and S. J. Kleis, “Boiling flow simulations on adaptive octree grids,” *International journal of multiphase flow*, vol. 53, pp. 88–99, 2013.

[18] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, “An extreme-scale implicit solver for complex pdes: highly heterogeneous flow in earth’s mantle,” in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2015, pp. 1–12.

[19] J. Bielak, O. Ghattas, and E. Kim, “Parallel octree-based finite element method for large-scale earthquake ground motion simulation,” *Computer Modeling in Engineering and Sciences*, vol. 10, no. 2, p. 99, 2005.

[20] F. Losasso, F. Gibou, and R. Fedkiw, “Simulating water and smoke with an octree data structure,” in *ACM SIGGRAPH 2004 Papers*, 2004, pp. 457–462.

[21] E. Kim, J. Bielak, and O. Ghattas, “Large-scale northridge earthquake simulation using octree-based multiresolution mesh method,” in *Proceedings of the 16th ASCE Engineering Mechanics Conference*. Seattle, WA, USA:, 2003.

[22] Y. Bazilevs, V. Calo, J. Cottrell, T. Hughes, A. Reali, and G. Scovazzi, “Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows,” *Computer methods in applied mechanics and engineering*, vol. 197, no. 1–4, pp. 173–201, 2007.

[23] “Microsoft Azure Marketplace.” [Online]. Available: https://azure.microsoft.com/en-us/marketplace/

[24] “Amazon Web Services Marketplace.” [Online]. Available: https://aws.amazon.com/marketplace/

[25] M. Azure. (2021) Optimal mpi process placement
for azure hb series vms. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-global/optimal-mpi-process-placement-for-azure-hb-series-vms/ba-p/2450663

[26] J. M. Kolinski and T. M. Schneider, “Superspreading events suggest aerosol transmission of sars-cov-2 by accumulation in enclosed spaces,” Physical Review E, vol. 103, no. 3, p. 033109, 2021.