

KAUST Supercomputing Laboratory (KSL)

Production Project Proposal (PPP) COVID-19

|  |  |
| --- | --- |
| Project Title | 3D Unstructured Mesh Deformation Using Rank Structured Matrix Computations:  A Testcase with the SARS-CoV-2 Virus |
| Principal Investigator (PI) | Prof. David Keyes |
| PI Signature | By submitting this proposal, I approve the entire content of this document and associated supporting documents. |
| Date of Proposal |  |
| Email: | david.keyes@kaust.edu.sa |
| Tel: |  |
| Organisation: | Extreme Computing Research Centre |
| Department: | Computer, Electrical and Mathematical Science and Engineering |
| Organisation Address: | KAUST |

|  |  |
| --- | --- |
| System | 🞏 Shaheen II 🞏 Ibex |
| Core Hours Requested | Shaheen II: \_\_\_\_\_\_\_\_\_\_\_\_\_ Ibex : \_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Storage TB Requested | Shaheen II: \_\_\_\_\_\_\_\_\_\_\_\_\_ Ibex : \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Available Systems:**

1. Shaheen II Supercomputer: 36-cabinets Cray XC40 system, comprising 6174 nodes, each with 32 Haswell cores and 128GB of memory, for a total of 197,568 cores along with 17.4 PB of Luster storage with a maximum data transfer bandwidth of 0.5 TB/s

<https://www.hpc.kaust.edu.sa>

1. Ibex Cluster:
   1. Heterogeneous computing cluster comprising of several differing architectures. There are about 500 computing nodes using Intel Skylake and Cascade Lake CPUs and Nvidia V100 GPUs.
   2. <https://www.hpc.kaust.edu.sa/ibex/computing>

Submission

Please send a scanned copy of the completed Project Proposal to:

[Projects@hpc.kaust.edu.sa](mailto:Projects@hpc.kaust.edu.sa)

**Terms and Conditions regarding Research Publications**

Whenever the results of research conducted on the HPC systems at KAUST are published, or the research involved personnel from KAUST Supercomputing Laboratory (KSL), Principal Investigators (PIs) are required to acknowledge the usage of the HPC systems at KAUST and/or the involvement of KSL personnel in their research in their publications. For example, the following statement could be used: “For computer time, this research used the resources of the Supercomputing Laboratory at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia.

# Additional Investigators and Users

|  |  |  |
| --- | --- | --- |
| 1 | Name: | Hatem Ltaief |
| Email: | Hatem.Ltaief@kaust.edu.sa |
| Tel: |  |
| Organisation: | Extreme Computing Research Centre |
| Org Address: | KAUST |
| 2 | Name: | Rabab Alomairy |
| Email: | Rabab.omairy@kaust.edu.sa |
| Tel: | 00966548884792 |
| Organisation: | Extreme Computing Research Centre |
| Org Address: | KAUST |

# Project Description:

Please describe the activities proposed, including current state of art, research work proposed, *expected milestones, and deliverables, as well as a summary description in the box below, and include the scientific field of the investigation as part of the description.*

* *Potential benefits for COVID-19 response*
* *Feasibility of the technical approach and its impact to tackle COVID-19 challenges*
* *Need for high-performance computing and justification for the required resources*

***Note****: Citations of the scientific literature are encouraged in order to show where the proposed simulations stand with respect to the ‘state of the practice’ in terms of such factors as model generality, resolution, and advantages of simulation versus experiment and theory.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Introduction:   Meshing the deformable contour of moving 3D bodies is a complex operation that leads to an enormous computational challenge. A mesh deformed to follow physics often exhibits poorer quality than the initial one, which may lead to inconsistent  solutions or numerical instability. Degradation over successive time steps may invalidate the mesh and bring the simulation to an end. Two different methods can improve the mesh quality of evolving geometries: mesh regeneration and mesh deformation. The first consists of regenerating the mesh frequently to ensure an accurate representation of the moving objects. This approach is computationally expensive for 3D objects and requires an interpolation stage to project the solution into the newly generated mesh. The second method deforms the mesh in order to track the moving geometries [14], using radaptation techniques.  The interpolation approach is a popular family of mesh deformation methods that consists in interpolating deformations between the moving structure and the boundaries of the computational domain. This approach permits to manage large deformations and arbitrary mesh topologies including adaptive unstructured meshes. In particular, we study the Radial Basis Function (RBF) [1], [2] scheme, one of the main interpolation methods that produces meshes with high fidelity. In this study, we design and implement a high performance software framework that calculates the mesh deformations of  3D moving bodies using the RBF interpolation technique. Our simulation framework takes advantage of the data sparsity of the symmetric positive-definite matrix operator by (1) performing low-rank approximations of its off-diagonal tiles and (2) using a Cholesky-based solver on the tile low-rank (TLR) compression data format. Exploiting the rank structured property of the matrix reduces the arithmetic complexity as well as the memory footprint and allows solving large 3D mesh deformation problems otherwise intractable. We highlight the robustness of our multiscale solver by assessing its numerical accuracy using realistic 3D geometries. In particular, we model the 3D mesh deformation and simulate the transport of the novel coronaviruses (i.e., SARS-CoV-2) inside a conduit of the upper respiratory system (S1 scenario) and within a droplet or air (S2 scenario).  Our framework depends on task-based solver that relies on dynamic runtime systems (e.g., StarPU [15], PaRSEC [16]) to orchestrate the various computational tasks of the TLR. Those tasks can be deployed on various shared and distributed-memory systems. The main idea is to break down the core computational phases into fine-grained tasks operating on the matrix tiles, as seen in Figure 1(a). Each tile is generated using the RBF kernel and compressed on-the-fly using the Randomized Singular Value Decomposition (RSVD) [17]. Once the matrix has been compressed with the TLR flat data compression format, the TLR algorithms can proceed by traversing each logical tile that is stored as two tall-and-skinny bases. The resulting computational tasks can be translated into a Directed Acyclic Graph (DAG), where nodes represent tasks and edges correspond to data dependencies, as seen in Figure 1(b).  dag-cholesky  (a) Tile low rank layout. (b) Directed Acyclic Graph.  Figure 1: Tile-based Cholesky factorization.   1. State of Art Hierarchical Low-rank Approaches:   Exploiting data sparsity of the underlying matrix operator has democratized direct dense methods for solving a broad class of large-scale scientific applications. Low-rank  matrix approximations in the form of hierarchical matrices (H-matrices) have enormously contributed toward reducing arithmetic complexity and memory footprint of direct dense factorizations and solvers. There are currently many state-of-the-art data compression formats for H-matrix approximation supporting weak (e.g., Hierarchically Semi- Separable (HSS) [10], Hierarchically Off-Diagonal Low- Rank (HODLR) [11]) and strong admissibility (e.g., H2-matrix [12], Block/Tile Low-Rank (BLR / TLR) [3]). A subset of these data compression formats may achieve almost linear arithmetic complexity and memory storage for some matrix kernels/operations and represent a game changer for scientific computing [13].   1. Radial Basis Function Interpolations (RBF)   Radial Basis Function (RBF) interpolations are used here to describe the displacement of the internal volume nodes given the displacement of the boundary nodes. As described in [1], an interpolation function describing the displacement in the whole domain, can be approximated by a sum of basis functions as follows:  Where are the boundary nodes at which the values are known, p a polynomial, the number of boundary nodes and a given basis function. The coefficients and the polynomial are determined by the interpolation conditions , where contains the known displacement values at the boundary. In our last paper [tech reference], we employ the Wendland basis function as a suitable RBF interpolant for unstructured mesh deformation. The Wendland basis function provides a flexible compact support to quantify the interactions between mesh points using a defined interaction radius.   1. 3D Unstructured Mesh Construction of the SARS-CoV-2 Virus:   This section describes the multiscale procedure employed to mesh bottom-up a single SARS-CoV-2 virus, up to a large population of viruses. While the geometry1 is employed  in many molecular dynamics simulations, to our knowledge, this is the first time a high resolution mesh is introduced for the SARS-CoV-2 virus in the context of applications in  mechanics.   1. The Meshing of the Spike Glycoprotein   We first generates a tetrahedral mesh of the volume surrounding the molecular structure of the Spike glycoprotein of the SARS-CoV-2 virus, a.k.a. the S protein, as shown in Fig. 2. The S protein is used by the virus as a key to unlock the human cells. Moreover, the 3D unstructured mesh of the S protein surface is perhaps one of the most challenging components of the virus structure to construct with high fidelity. Codenamed PDBID 6VXX at the Protein Data Bank (PDB) and available at (https://www.rcsb.org/structure/6VXX), the S protein has been generated with 2.8A ̊ resolution [18], i.e., 0:28nm. with a complex trimeric molecular structure. It is characterized by a complex trimeric molecular structure of an overall 10nm length [19].   |  |  |  | | --- | --- | --- | | Fig. 2: The molecular structure of the S protein. | Fig. 3: Convex hull of the S protein. | Fig. 4: The final S protein mesh structure. |   After reading the PDB file of atom coordinates and radii, we build a 3D isosurface data set that we discretize to construct a primitive 3D surface mesh. The quality of the 3D surface thus obtained is then improved by using local optimization operations, in particular weighted Laplacian smoothing [20]. The smoothed 3D surface is used to build an unstructured tetrahedral mesh using constrained Delaunay triangulation algorithms.  The resulting mesh has high resolution but is expensive to be used as a pattern into the complete virus geometry. An alternative approach consists in representing the spike geometry by its convex hull that we compute from the PDB data set using the molecular structure of the S protein, as shown in Fig. 3. The 3D isosurfaces are then discretized  and optimized to get high quality surface. Fig. depicts the final 3D surface mesh of the S protein that can be used in a Computer-Aided Design (CAD) software of choice toward assembling the whole virus mesh structure.   1. The Meshing of a Single Virus   Once the mesh of the S protein has been constructed, the next phase consists in stitching several S proteins, approximately evenly distributed around the spheric geometry of the virus main body. Based on the geometrical description of the virus structure [19], we set the virus diameter to 140nm and attach 80 S proteins. We build a CAD model which represents the building block for the simulation pipeline. The CAD model is then triangulated to generate a 3D surface mesh of the overall SARS-COV-2 virus geometry, as highlighted in Fig. 5. To capture the interactions between the mesh nodes at the surface of the virus, we define a flexible radius that expresses the interaction neighborhood when computing the Wendland basis function [21]. For taking into account all  mesh interactions within a single virus, the radius should be at least equal to the diameter of the minimum bounding box of the virus.   |  |  | | --- | --- | | Fig. 5: A closer look to a single SARS-COV-2 virus. | Fig. 6: Population of SARS-COV-2 viruses. |  1. The Meshing of a Virus Population   Instead of creating a population directly based on the CAD which may be tedious, a different approach is adopted for the meshing of a virus population. We build a high-quality unstructured 3D mesh for one virus at scale. Then, we use a homogeneous  transformation to generate the whole population of viruses based on uniform random translation and rotation. Fig. 6. demonstrates the resulting population. We confine the generated population within a cube with edge length proportional to the number of viruses. We maintain a decent concentration of viruses, while ensuring no interpenetrations occur by adopting a minimal distance between viruses. During time integration, collisions may happen and the viscosity of the virus membrane should allow the surface mesh to absorb the deformation before eventually invalidating the viruses mesh structure   1. Research Objectives and Milestones:   In our previous study [tech reference], we introduced a versatile, multiscale simulation framework for computing unstructured mesh deformations of 3D bodies using the RBF interpolations. In particular, we focus on the SARS-CoV-2 virus as a realistic test case, for which a realistic 3D geometry is newly available. The simulation tool exploits the data sparsity of the resulting RBF matrix operator using the Tile Low-Rank (TLR) Cholesky based solver from the HiCMA library [3]. The following are four milestones we accomplished in our last paper:   1. We introduce the 3D unstructured mesh generation of the SARS-CoV-2 virus with the Spike proteins from the realistic virus geometry, as discovered in [19], [21] and defined in the Protein Data Bank (PDB). 2. We implement the multiscale simulation involving a single virus and its Spike proteins (nanoscale), up to one million viruses (from microscale to few millimeters) with 10 billion mesh points, while leveraging the existing TLR Cholesky algorithm. 3. We evaluate the impact of various mesh sizes, the accuracy threshold for truncation, the reordering techniques, and the interaction radius on the overall performance. We study the numerical robustness of our solver. We launch a performance benchmarking campaign on various shared and distributed-memory systems based on three main vendor architectures (Intel, AMD, and ARM), and compare against state-of-the-art data-sparse and dense solvers. 4. We use StarPU [15] runtime systems to orchestrate computational tasks between shared- and distributed- memory systems.   In our subsequent research, we would like to extend this work by:   1. Studying the most common global and compact support RBF basis (i.e. Gaussian, Exponential, Inverse quadratic, inverse multi-quadratic, and continues thin plate splines), 2. Using a complex distribution between population of viruses contained within cube such as sphere backing approach 3. Integrating HiCMA RBF solver into the time integration workflow. An additional computational phase is required that consists in local re-meshing of the cavities formed around each vertex or edge in order to increase the quality of the cavity elements and maintain the mesh validity over time. 4. Exploring aspects of the binding of the S protein and the ACE-2 human protein (i.e., the host receptor) that are influenced by fluid dynamical interactions, before molecular dynamics (MD) models take over. 5. Employing PaRSEC [16] runtime system to provide strong and weak scalability studies by increasing the viruses within population. 6. Primarily Numerical and Performance Results:   In our paper [tech reference], We assess the impacts of the accuracy threshold, the mesh reordering, and the radius interaction on the TLR Cholesky solver. In addition, we carried several performance testing on shared- and distributed- memory systems.   1. Accuracy of the TLR Cholesky Solver:   Fig. 7 shows the residual defined by with A the RBF matrix based on the Wendland interpolation kernel, b the right hand side, and x the computed solution.  The residual is calculated at the end of the TLR Cholesky solver after computing the 3D mesh deformations of various resolutions of a single SARS-CoV-2 virus (i.e., 10370, 44932, 117715, and 142418 mesh points). The infinity norm decreases as we further truncate with a lower accuracy threshold. We noticed the RBF matrix loses its positive definiteness and we are able to restore it by shifting the main diagonal elements with a constant of an order magnitude higher than the accuracy threshold [22].    Fig. 7: Numerical accuracy of the TLR Cholesky solver using various mesh sizes of a single SARS-CoV-2 virus   1. Impact of Accuracy Threshold and Reordering on Rank Distributions   We study the impact of the accuracy threshold and the reordering technique on the rank distributions using 117715 mesh resolutions of the SARS-CoV-2 virus as seen Fig. 8.  The obtained ranks may be high when the tile low rankness is not being exposed due to the absence of reordering. However, when reordering is enabled, the Hilbert ordering approach seems to provide the best memory footprint compared to Morton ordering.   |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  | |  |  |  |   Fig. 8: Rank Distribution using single virus with 117715 mesh points. Each row represents , , and , truncation thresholds, respectively. Each column shows no ordering, Morton, and Hilbert, respectively.   1. Impact of Interaction Radius on Rank Distributions   We now consider the Hilbert ordering and fix the accuracy threshold to as this translates into the accuracy needed for the residual when using the RBF interpolation for 3D unstructured mesh deformations. Fig.9 shows the cross interactions between a population of 10 SARS-CoV-2 viruses contained inside a cube. As we increase the radius, the RBF matrix structures change from an S2 scenario (i.e., within a droplet) with a block (but low-rank) diagonal structure to an S1 scenario (i.e., within a confined conduit) with a regular full data-sparse structure.   |  |  |  |  | | --- | --- | --- | --- | |  |  | |  | |  | |  | |   Fig. 9: Rank distributions using ten viruses of 10370 mesh points with accuracy threshold and Hilbert ordering enabled when varying interaction radius to 400nm, 800nm, 1m, 2m, and 4m radius, respectively.   1. Performance Assessment When Using Single Virus   Fig. 10 shows the performance comparisons of TLR Cholesky solver against the dense Cholesky factorization, as implemented in the corresponding optimized vendor numerical libraries. We show the impact of various accuracy thresholds on the overall solver performance. We demonstrate performance improvement up to two orders of magnitude across the three hardware platforms, i.e., AMD, Intel and ARM, using various number of mesh points for a single SARS-CoV-2 virus.   |  |  |  | | --- | --- | --- | | (a) Intel Cascade Lake. | (b) AMD EPYC Rome. | (c) Marvell ThunderX2 ARM. |   Fig. 10: Performance of the TLR Cholesky HiCMA solver versus vendor optimized dense Cholesky on Intel Cascade Lake, AMD EPYC Rome, and Marvell ThunderX2 ARM shared-memory systems using single virus with various mesh sizes.  Fig. 11 compares the performance of the TLR Cholesky HiCMA solver and the symmetric factorization algorithm from the HODLR software library [23] on Intel Cascade Lake shared-memory system. To provide a comparison as fair as possible, Hilbert ordering is used for both methods. HiCMA always outperforms HODLR when using lower truncation thresholds for all mesh resolutions. Lower accuracy thresholds produce high ranks which is computationally expensive for weak-admissible conditions, as designed in HODLR. For the large accuracy threshold, HODLR is able to outperform HiCMA since the number of significant singular values is limited on the off-diagonal tiles. Such situations may be better handled within HiCMA thanks to the fine-granularity task scheduling, but require changes in the TLR Cholesky algorithm.    Fig. 11: Performance of HiCMA Cholesky versus HODLR on Intel Cascade Lake shared-memory system using single virus with various mesh sizes.   1. Performance Assessment When Considering Scenario S1   Fig. 12 considers interactions within a population of SARS-CoV-2 viruses, on Intel Cascade Lake, AMD EPYC Rome, and Marvell ThunderX2 ARM shared-memory systems. The corresponding RBF matrix structure is fully data-sparse with ranks on the off-diagonal tiles large enough to outperform HODLR. The figure reports up to two orders of magnitude performance improvement of TLR Cholesky HiCMA solver (with accuracy threshold ) against corresponding vendor optimized dense Cholesky. The three hardware systems run out of memory when addressing large number of viruses using dense Cholesky algorithms.   |  |  |  | | --- | --- | --- | | (a) Intel Cascade Lake. | (b) AMD EPYC Rome. | (c) Marvell ThunderX2 ARM. |   Fig. 12: Performance of TLR Cholesky HiCMA solver versus vendor optimized dense Cholesky when varying the number of viruses. The matrix size represents a population of 1, 2, 4, 8, and 10 viruses with 44932 mesh points per virus.  Fig. 13 shows the performance scalability of TLR Cholesky HiCMA solver when computing the 3D mesh deformations using RBF interpolations on two distributed-memory systems. As we increase the number of viruses, HiCMA scales up to 512 nodes on the Cray XC40 Intel Haswell system (Fig. 15a) and on the full scale of the Cray XC Intel Cascade Lake system, i.e., 128 nodes (Fig. 15b). While the scalability is decent, there is still room for improvement from the dynamic runtime’s perspective by mitigating the overhead of data motion, as suggested by the authors in [7].   |  |  | | --- | --- | | (a) Cray XC40 Intel Haswell. | (b) Cray XC Intel Cascade Lake. |   Fig. 13: Performance scalability of TLR Cholesky HiCMA factorization on distributed-memory systems when varying the number of viruses. The matrix sizes represent up to 48 viruses with 117715 mesh points per virus.  Performance Assessment When Considering Scenario S2  We investigate now the performance of the RBF matrix solver in the context of the transport of a population of SARS-CoV-2 viruses for S2, i.e., within a droplet. This scenario engenders a block diagonal matrix where each block contains the interactions  among virus belonging to the same cluster. This translates into running instances of TLR Cholesky HiCMA solver in an embarrassingly parallel fashion. Fig. 14 shows the performance of the TLR Cholesky HiCMA solver under the scenario S2, i.e., with a droplet containing independent clusters of viruses. The figure reports the time to solution to compute the mesh deformations of almost a million SARS-CoV-2 viruses on two different distributed-memory systems. This scenario translates into a simulation with 10 billion mesh points. Since the workload of S2 is embarrassingly parallel as we increase the number of clusters, this is similar to running weak scaling experiments. It takes 450 and 1200 seconds to process 1M viruses on 2048 and 512 nodes of the Cray XC40 Intel Haswell and the HPE Apollo AMD EPYC Rome systems, respectively. The throughput achieved in terms of millions of mesh points per second (Mega MPoints/s) turns out to be almost 2.5 greater on the Cray system compared to the HPE system. Although the same number of cores is used on both systems for each data point, we use four times  more nodes on the Cray system than the HPE system.    Fig. 14: Performance of TLR Cholesky HiCMA solver when considering interactions within independent clusters of up to 1M viruses with 10B mesh points overall on two distributed-memory systems. The curves reporting time to run 24 and 96 clusters of viruses per node are annotated with the increasing number of nodes.    [1] A. de Boer, M. van der Schoot, and H. Bijl, “Mesh Deformation Based on Radial Basis Function Interpolation,” Computers and Structures, vol. 85, no. 11, p. 784 795, 2007, fourth fMITg Conference on Computational Fluid and Solid Mechanics. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0045794907000223>  [2] T. Rendall and C. Allen, “Reduced Surface Point Selection Options for Efficient Mesh Deformation Using Radial Basis Functions,” Journal of Computational Physics, vol. 229, no. 8, pp. 2810 – 2820, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0021999109006779>  [3] K. Akbudak, H. Ltaief, A. Mikhalev, and D. Keyes, “Tile Low Rank Cholesky Factorization for Climate/Weather Modeling Applications on Manycore Architectures,” in 32nd International Conference on High Performance, Frankfurt, Germany. Springer, 2017, pp. 22–40.  [4] K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, A. Esposito, and D. Keyes, “Exploiting data sparsity for large-scale matrix computations,” in European Conference on Parallel Processing. Springer, 2018, pp. 721–734.  [5] K. Akbudak, H. Ltaief, A. Mikhalev, and D. Keyes, “Tile Low Rank Cholesky Factorization for Climate/Weather Modeling Applications on Manycore Architectures,” in International Supercomputing Conference. Springer, 2017, pp. 22–40.  [6] N. Al-Harthi, R. Alomairy, K. Akbudak, R. Chen, H. Ltaief, H. Bagci,and D. Keyes, “Solving Acoustic Boundary Integral Equations Using High Performance Tile Low-Rank LU Factorization,” in 35th International Conference on High Performance, Frankfurt, Germany. Springer, 2020.  [7] Q. Cao, Y. Pei, K. Akbudak, A. Mikhalev, G. Bosilca, H. Ltaief, D. Keyes, and J. Dongarra, “Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications,” in Proceedings of the Platform for Advanced Scientific Computing Conference. ACM, 2020.  [8] A. C. Walls, Y.-J. Park, M. A. Tortorici, A. Wall, A. T. McGuire, and D. Veesler, “Structure, Function, and Antigenicity of the SARS-CoV-2 Spike Glycoprotein,” Cell, 2020.  [9] D. Wrapp, N. Wang, K. S. Corbett, J. A. Goldsmith, C.-L. Hsieh, O. Abiona, B. S. Graham, and J. S. McLellan, “Cryo-EM Structure of the 2019-nCoV Spike in the Prefusion Conformation,” Science, vol. 367, no. 6483, pp. 1260–1263, 2020. [Online]. Available: <https://science.sciencemag.org/content/367/6483/1260>  [10] E. Corona, P.-G. Martinsson, and D. Zorin, “An O(N) direct solver for integral equations on the plane,” Applied and Computational Harmonic Analysis, vol. 38, no. 2, pp. 284–317, 2015.  [11] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O’Neil, “Fast direct methods for Gaussian processes,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, pp. 252–265, 2015.  [12] S. B¨orm, Efficient numerical methods for non-local operators: H2-matrix compression, algorithms and analysis. European Mathematical Society, 2010, vol. 14.  [13] W. Hackbusch, “Survey on the Technique of Hierarchical Matrices,” Vietnam  Journal of Mathematics, vol. 4, p. 71–101, 2016. [Online]. Available:  https://doi.org/10.1007/s10013-015-0168-5  [14] W. Huang and R. D. Russell, “A High Dimensional Moving Mesh Strategy,” Applied Numerical Mathematics, vol. 26, no. 1, pp. 63–76, 1998.  [15] Augonnet, C., Thibault, S., Namyst, R. & Wacrenier, P. A. (2011). StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. Concurrency and Computation: Practice and Experience, 23(2), 187-198.‏  [16] Sukkari, D., Ltaief, H., Keyes, D. & Faverge, M. (2019, September). Leveraging Task-Based Polar Decomposition Using PARSEC on Massively Parallel Systems. In 2019 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 1-12). IEEE.‏  [17] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions,” SIAM review, vol. 53, no. 2, pp. 217–288, 2011.  [18] A. C. Walls, Y.-J. Park, M. A. Tortorici, A. Wall, A. T. McGuire, and D. Veesler, “Structure, Function, and Antigenicity of the SARS-CoV-2 Spike Glycoprotein,” Cell, 2020.  [19] B. Huang, W. Shi, R. Lu, P. Niu, F. Zhan, X. Ma, D. Wang, W. Xu, G. Wu, G. F. Gao, and W. Tan, “A Novel Coronavirus from Patients with Pneumonia in China, 2019,” New England Journal of Medicine, vol. 382, no. 8, pp. 727–733, 2020, pMID: 31978945. [Online]. Available: <https://doi.org/10.1056/NEJMoa2001017>  [20] T. Zhou and K. Shimada, “An angle-based approach to two-dimensional mesh smoothing.” Proceedings of the 9th International Meshing Roundtable, vol. 2000, pp. 373–384, 2000.  [21] H. Wendland, “Scattered Data Approximation,” Cambridge monographs  on applied and computational mathematics, vol. 17, pp. 119–132, 2005.  [22] N. Higham and S. Pranesh, “Exploiting Lower Precision Arithmetic in Solving Symmetric Positive Definite Linear Systems and Least Squares Problems,” Technical Report, http://eprints.maths.manchester.ac.uk/id/eprint/2736, 2019.  [23] A. Aminfar, “A Fast Direct Solver for HODLR Matrices,” https://github.com/amiraa127/Dense HODLR, 2014. |
|  |

# Codes & Libraries:

* *Please provide the following information for each code or library that will be used.*

|  |  |
| --- | --- |
| Name of Code/Library: |  |
| Ownership / Licensing: |  |
| URL (*for Open Source codes*) |  |
| Function: |  |

# Resource Requirements:

|  |  |  |
| --- | --- | --- |
| Compute Resource | Requirement (core hours) | Duration (in Days) |
| Shaheen II |  |  |
| Ibex |  |  |

# Resource Requirement Justification:

*Please detail how the above requirements were calculated. The nodes are exclusive, therefore, even using only one core per node, 32 cores will be charged*

|  |
| --- |
|  |

# Storage Requirements:

|  |  |  |
| --- | --- | --- |
| Storage Resource | Requirement (TB) | Duration (in Days) |
| Shaheen II /project |  |  |
| Ibex |  |  |

# Storage Request Justification (beyond 20TB)

*Please describe both short-term and long-term storage requirements, including requirements for the number of files and data volume. How long the data needs to be stored after the project is completed? Will it be moved out from Shaheen parallel filesystem for subsequent analysis in other systems?*

|  |
| --- |
| 1. Please describe short-term (up to 1 year) storage requirements, including requirements for the number of files and data volume and corresponding justification. 2. Please describe long-term (beyond 1 year) storage requirements, including requirements for the number of files and data volume and corresponding justification. 3. How long the data needs to be stored after the project is completed? 4. Will it be moved out from Shaheen parallel filesystem for subsequent analysis in other systems? 5. Please provide your data management plan after completion of a given project. |

***Notes:***

1. Policy states that all project data stored on disk will be removed 6 months after the completion of the project.
2. Policy states that all project data stored on the tape archive will be removed upon the completion of the project *unless* special arrangements have been requested and granted.

# Consultancy Support Required

*Please indicate the number of man-days and type of any support required from staff (e.g. 0.2FTE for 3 months), which can include:*

* *Code development*
* *Code porting*
* *Code performance tuning*
* *Algorithm development*
* *Pre- and Post-Processing code development*
* *Data analysis and visualisation support*
* *Research program development*
* *Project management support*

*Please note that KAUST may be entitled to a share in the Intellectual Property Rights to any research results produced as a result of support provided by KAUST.*

|  |
| --- |
|  |

# Confidentiality and Legal Issues:

Please provide details of any potential confidentiality or legal issues, e.g.*:*

* *Is the project proposal confidential? If so, how? Does it contain human data?*
* *Is the data confidential? If so, how?*
* *Are any other aspects of the project confidential? If so, how?*
* *If the project is successful, could it be the subject of publicity?*
* *Do any third parties have ownership of any codes or data being used?*

|  |
| --- |
|  |

**Curriculum Vitae of Principal Investigator(s)**

*To aid reviewers in their scientific evaluation, please attach a 1-page (or at most 2-page) C.V., updated to within the past year, for each principal investigator. For KAUST-based investigators, this typically would be the C.V. that is kept on file for KAUST Office of Sponsored Research purposes.*

**☐ attached**

**Additional Information**

|  |
| --- |
|  |