# Team

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Administrative Role | Technical Role | Contact |
| Tong Qin | Industry Mentor |  | [tqinmath@amazon.com](mailto:tqinmath@amazon.com) |
| Varun Elango | Faculty Mentor |  | [varune98@uw.edu](mailto:varune98@uw.edu) |
| Jaidon Lybbert | Team Lead | Researcher / CUDA | [jlybbert@uw.edu](mailto:jlybbert@uw.edu) |
| Fulin Li | Finances | Python Developer | [fulin@uw.edu](mailto:fulin@uw.edu) |
| Shashank Shivashankar |  | CUDA C++ Lead Developer | [sham73@uw.edu](mailto:sham73@uw.edu) |
| Alice |  | Python Lead Developer | [alice615@uw.edu](mailto:alice615@uw.edu) |
| Yiming Du | Documentation | C++ / Python Developer | [yiming00@uw.edu](mailto:yiming00@uw.edu) |
| Mike Pao |  | Python Developer | [mikepao@uw.edu](mailto:mikepao@uw.edu) |

# Background

QR factorization is the decomposition of a matrix A into the matrix product A = QR, where Q is an orthogonal matrix, and R is an upper triangular matrix. The factorization is used for solving nonlinear systems, and has applications in robotics, virtual and augmented reality, and 3D structure reconstruction.

The algorithm is computationally expensive, which results in high latency ill-suited for real time applications when computed sequentially. For this reason, a parallel implementation is desired, such as using CUDA on a GPU.

Mixed-precision hardware such as the Nvida TensorCore on Volta and Turing GPU architectures, as well as Google TPUs, enable 16-bit arithmetic to reduce memory and compute workloads by sacrificing accuracy. Techniques exist to mitigate the effects of reduced accuracy while preserving the cost and energy efficiency of using mixed-precision hardware.

# Objectives and Deliverables

The goal of this project is to research and implement a fast and correct mixed-precision QR factorization algorithm using mixed-precision hardware on the GPU. The measure of success is defined by the accuracy and speed of our algorithm compared to a naïve implementation on an x86 CPU architecture.

The implementation will be open source and hosted on GitHub. A stretch goal as time allows is to integrate the algorithm into an existing open-source package.

# Milestones

1. Get clear idea of project scope
2. Full precision block QR implementation in Python
3. Full precision block QR implementation in CUDA
4. Prototype mixed-precision options in Python/C++
5. Fix the mixed-precision block QR algorithm
6. Fix the CUDA implementation design
7. Finish CUDA implementation
8. Test CUDA implementation
9. Final report

# Timeline and Sequence of Tasks

See Appendix A. for a preliminary Gantt chart representing the project schedule.

# Learning Goals

1. Develop Python skills
2. Develop CUDA C++ skills
3. Learn parallel programming concepts
4. Practice technical report writing
5. Learn to conduct and apply research

# Mentor Comments

<empty>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROJECT NAME | PROJECT LEAD | PROJECT START DATE | PROJECT END DATE | TODAY’S DATE |
| Mixed-precision Block QR | Jaidon Lybbert | 1/3/2023 | 6/1/2023 | 1/12/2023 |

**APPENDIX A - GANTT CHART**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MONTH** | | | | | **1** | | | | | **2** | | | | **3** | | | | **4** | | | | **5** | | |
| **WEEK STARTING** | | | | | **1** | **8** | **15** | **22** | **29** | **5** | **12** | **19** | **26** | **5** | **12** | **19** | **26** | **2** | **9** | **16** | **23** | **7** | **14** | **21** |
| **ACTIVITY** | **ASSIGNED TO** | **STATUS** | **START** | **END** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kickoff Meeting |  | Complete | 1/3 | 1/3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Develop Scope & Schedule | Jaidon L. | In Progress | 1/3 | 2/5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research Algorithm | Jaidon L. | In Progress | 1/3 | 2/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Full-precision block QR in Python | Alice L. | In Progress | 1/10 | 2/22 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Full-precision block QR in CUDA | Shashank S. | In Progress | 1/10 | 2/22 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Prototype mixed-precision in Python | Alice L. | Not Started | 1/22 | 2/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Prototype mixed-precision in CUDA | Shashank S. | Not Started | 1/22 | 2/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fix mixed-precision block QR algorithm | Jaidon L. | Not Started | 2/5 | 3/5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fix CUDA implementation design | Shashank S. | Not Started | 2/20 | 3/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fix x86 implementation design | Alice L. | Not Started | 2/20 | 3/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Finish CUDA implementation | Shashank S. | Not Started | 3/20 | 5/07 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Finish x86 implementation | Alice L. | Not Started | 3/20 | 5/07 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing | Yiming D. | Not Started | 5/07 | 5/14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Final Report | Yiming D. | Not Started | 5/14 | 5/20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |