# Distributed Matrix Computation
# Team Milestone Retrospective

Ryan Bazzell & Jiwoon Yim

March 7, 2025

## 1 Team Reflection

Overall, the project coordination has been great for the team. We have been communicating well with each other and ensured deadlines are met for each project deliverable. We both realized our project topic aligned well with the previous homework assignments, allowing us to transfer multiple concepts and implementations from the homework assignments to our project space.

## 2 Revisions to Proposal

Upon initial reflection on the first draft of our project, we realized implementing a distributed Straussen's algorithm would be particularly difficult. The biggest bottleneck here would be the distribution of each sub-multiplication generated at each level. For each $2^n$ by $2^n$ matrix multiplication, 7 additional $2^{n-1}$ by $2^{n-1}$ multiplications would be generated. For matrices of size 8 by 8, this would leave us with 57 total multiplications (and function calls). If we do not parallelize the issue, we would need some serious architectural changes (mostly worker to worker communication) to prevent the coordinator from bottle-necking the system. If we do parallelize, that is a change in architecture itself and brings a variety of problems of its own. Currently, we are attempting to implement the parallelized solution, as to emphasize the performance of our distributed system, being careful to make sure data remains thread-safe, so we do not encounter dead-locking issues.

Initially, we had discussed using RPyC for our remote procedure calls, similar to the implementation of Homework 2. However, due to the vast number of problems encountered by Ryan in Homework 3 (and the relatively few problems encountered by Jay), we have decided to forgo using RPyC in exchange for a simpler in-house design using socket connections. We have also begun using a Python library called Flask, which helps manage remote connections.

Furthermore, and much more substantially, we have decided to make the architectural shift away from a pure middleware system to using a worker-coordinator architecture, akin to the architecture in Homework 3. Our coordinator will act like the original middleware we intended, as we are not building in the capability for the workers to switch to coordinator mode, which was required in Homework 3. This should simplify our design and our troubleshooting process.

With this new architecture, the message passing between nodes has increased in complexity. Currently, the coordinator receives a job from a client, sends that job to a worker (which further breaks that job down into more smaller jobs), and then returns those jobs to the coordinator. The coordinator has special logic to ensure it can take these smaller jobs and recombine them in the future. Then the coordinator sends the smaller tasks down until we are left with 2x2 matrices. At this point, computed results are returned, compiled into a larger matrix, returned, compiled into a larger matrix, and so on until we have the final matrix at the end.