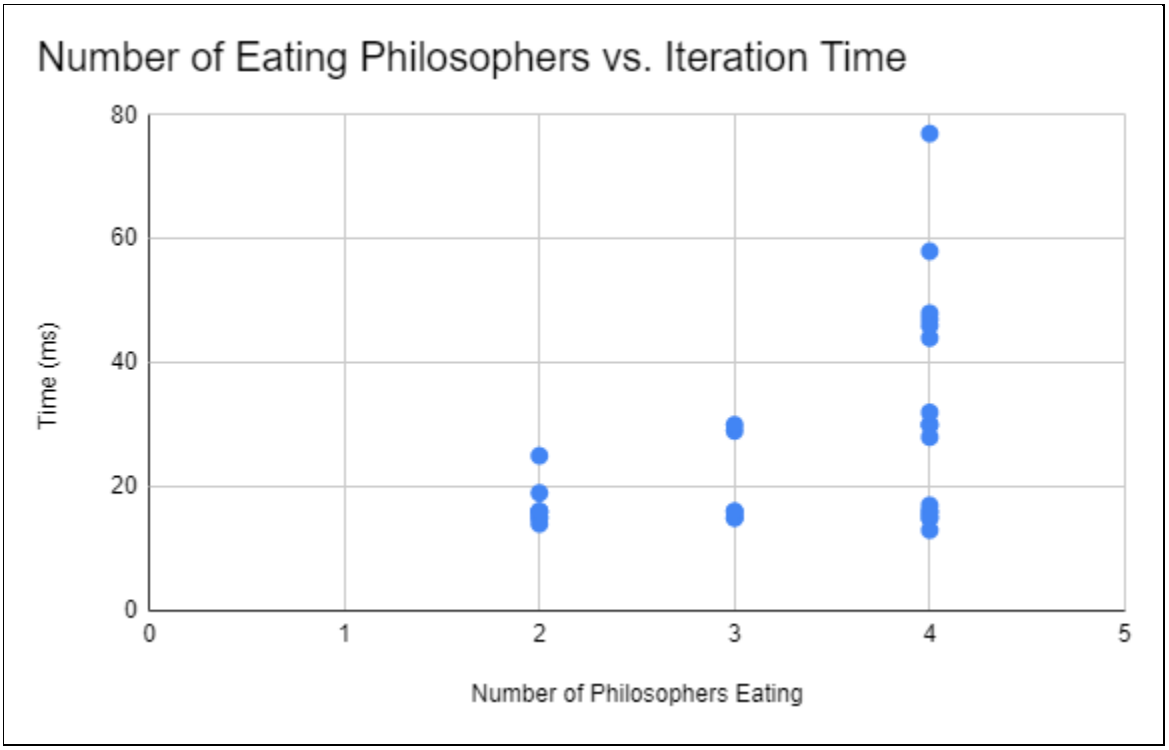
Erik Hynes
Maggie Butterfield
Bradley Kolar

Operating Systems Simulation HW 2
https://github.com/maggiez-z/Simulator2.git



The data shows an interesting variance as the number of philosophers eating in a step increases. While a step that executed in a short amount of time could have any number of philosophers eating, the steps with few eating always resulted in a shorter iteration time. However, we are completely aware that there should never be more than 3 philosophers eating at a time (we did the #1 variation so philosophers 1, 3, and 5 could eat simultaneously due to the extra chopstick between 1 and 5).

We think the problems resulted from threads releasing the resources (forks) while the “philosopher” was still “eating”. In this case, it could be immediately grabbed by another thread, allowing for more to be “eating”. However, we are not sure what causes the threads to release, because we tried using the “synchronized” keyword in Java to take total control of each Fork object, while checking for use by other threads. Even after switching to if-else statements instead, we experienced the same kind of resource dropping.