September 26 Thursday

- Experiment

1. Compare rust_linag and netlib-java
    - Use linux code to fix number of core while experiment
    - Read csv file from dist to ram and do calculation
    - Try different serialization (Rust has serialization library)
2. Perform gradient descent to big matrix data which can feed up memory
    - Use command below to set max memory
        - Java yourapp.class -- Xms 2GB(max load) --Xms 2GB (initial load)
    - Use row number data (with 1000 rows to start)
    - And increase by 1000 rows to see when the memory is feed up
    - The proper one can be selected one before the memory crashed
3. Try to see mutable object vs imutable object

September 25 Wednesday:

- Search
    - Spark(specifically MLlib) depends on linear algebra library called “Breeze” which is based on “netlib-java” which is based on “Netlib”.
    - We can configure to “Intel MKL” and “OpenBLAS”.
    - Read from disk and calculate it and write it to disk
        - Forcing serialization
    -

September 23 Monday:

- Progress
    - Implemented add function for matrix operation
- Need to do
    - Implement slicing fucntion for 2 dimentional array
    - Strassen algorithm
    - Create repo for thesis template

September 20 Friday: Meeting with Kia

- Statement
    - Do not use external library
    - Use only pure vector and array

- Experiment condition
    - Use pure rust and java
    - Implement simple test
        - Transpose matrix and

        - Dot product
    - Get small machine (one core one RAM)
        - To force java and rust not to compute multiprocess
            - Java might perform multi-thread
        - To force java to perform garbage collection
            - Garbage collection will perform when the memory is filed up.
    - We could use task command to make the program to stay on one core
    - Java matrix computation might run fast enough to beat rust for raw number data
        - We would like to experiment for complicated object matrix (twitter data)
    - Fix the experiment process
        - Load from csv file
        - Number of core
        - Number of Ram
    - Use Ganglia to monitor the experiment

- Thesis paper
    - Use effective and scientifically correct word
        - Ex: host language is language working on Virtual machine
        - We should define abbreviation before using it
        - System language is good for memory management
    - Create template of thesis
        - Find temple for Ratex BU thesis on google
- How to find papers related to our research
    - Clash of the titans : MapReduce vs spark for large scale data analytics
        - See references
    - Find papers accepted in VLBD and SIGMOID
- Task for next week
    - Implement simple matrix operation in Rust (primarily), Java and spark.
        - Using only pure Rust and Java
        - We can implement similar algorithm used for spark, because it can be best algorithm for java
        - `org.apache.spark.mllib.linalg.{Vector, Vectors}`
        - `https://spark.apache.org/docs/2.2.0/mllib-data-types.html`
    - Create a repository for Thesis template
        - Find Ratex template for BU thesis
        - Write chapters
        - Write abstract
    - Read related papers and review to Kia

September 16 Monday:

- Ndarray package can be useful

- Ndarray-linalog package needs to use BLAS / LAPACK backend
    - If we want to use BLAS / LAPACK backend we need to use it for java implementation as well.
    - Otherwise implement linear algebra feature from scruch
    -

September 10 Tuesday: Meeting with Kia

- Progress plan
    - Implement small problems in Rust
        - Implementation step 1
            - Use small data set
                - Generate random variable
                - And create data set
            - Reading data
            - Dump into memory
            - Do some calculation (matrix multiplication, multiple linear) regression)
            - Output result in file
        - Implementation plan 2
            - Use large data text set
            - Do tfidf
            - Logistic regression using gradient descent
    - Concept to learn
        - Deeply understand memory management in Rust (ownership and borrowing)
        - LLVM (Rust is restricted version of C++)
        - Memory management for machine learning task