

User

write a general matrix multiply (GEMM) algorithm in Rust

Bing

Sure, here's a simple implementation of a General Matrix Multiply (GEMM) algorithm in Rust. This code multiplies two matrices *a* and *b* and stores the result in *c*.

```
fn gemm(a: &Vec<Vec<f64>>, b: &Vec<Vec<f64>>, c: &mut Vec<Vec<f64>>) {
    let n = a.len();
    let m = b[0].len();
    let p = a[0].len();

    for i in 0..n {
        for j in 0..m {
            let mut sum = 0.0;
            for k in 0..p {
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
    }
}
```

Please note that this is a basic implementation and does not include any optimizations. For large matrices, you might want to consider using a library like *ndarray* or *nalgebra* which are optimized for numerical computations. Also, this code assumes that the input matrices are well-formed and have compatible dimensions for multiplication. In a production setting, you would want to add checks for these conditions.

Let me know if you need further assistance! 😊

User

create a rust function to initialize a randomized m-by-n matrix to satisfy the parameters of this function header `fn gemm(a: &Vec<Vec>, b: &Vec<Vec>, c: &mut Vec<Vec>) { let n = a.len(); let m = b[0].len(); let p = a[0].len();`

```
for i in 0..n {
    for j in 0..m {
        let mut sum = 0.0;
        for k in 0..p {
            sum += a[i][k] * b[k][j];
        }
        c[i][j] = sum;
    }
}

}
```

Bing

Sure, here's a function that creates a randomized m-by-n matrix in Rust. This function uses the *rand* crate, so you'll need to add `rand = "0.8"` to your *Cargo.toml* file.

```
use rand::Rng;
```

```
fn create_random_matrix(m: usize, n: usize) -> Vec<Vec<f64>> {
    let mut rng = rand::thread_rng();
    let mut matrix = vec![vec![0.0; n]; m];
    for i in 0..m {
        for j in 0..n {
```

```

        matrix[i][j] = rng.gen::();
    }
}
matrix
}

```

This function generates a matrix of size m by n filled with random floating-point numbers between 0 and 1. You can use this function to initialize the matrices a , b , and c for the `gemm` function. Please note that the dimensions of these matrices should be compatible for the multiplication operation in `gemm`. For instance, if a is an m by p matrix and b is a p by n matrix, then c should be an m by n matrix.

Let me know if you need further assistance! 😊

User

write a Rust function which creates two m -by- n matrices using the function `create_random_matrix(m: usize, n: usize) -> Vec<Vec>`, then passes them into the function `gemm(a: &Vec<Vec>, b: &Vec<Vec>, c: &mut Vec<Vec>)` and returns the execution time in milliseconds

Bing

Sure, here's a function that does that. This function uses the `Instant` struct from the `std::time` module to measure the execution time.

```

use std::time::Instant;

fn measure_gemm(m: usize, n: usize, p: usize) -> u128 {
    let mut a = create_random_matrix(m, p);
    let mut b = create_random_matrix(p, n);
    let mut c = vec![vec![0.0; n]; m];

    let start = Instant::now();
    gemm(&a, &b, &mut c);
    let duration = start.elapsed();

    duration.as_millis()
}

```

This function first creates two random matrices a and b of dimensions m by p and p by n respectively using the `create_random_matrix` function. It also initializes a zero matrix c of dimensions m by n . It then measures the time taken to execute the `gemm` function on these matrices and returns this time in milliseconds.

Please note that the actual execution time can vary depending on the system's load and the specific hardware used. Also, this function assumes that the `gemm` and `create_random_matrix` functions are defined in the same scope.

Let me know if you need further assistance! 😊

User

write a function that iterates over 100 matrix sizes from 10×10 to 1000×1000 and calls this function `fn measure_gemm(m: usize, n: usize, p: usize) -> u128` which returns the execution time in ms, using the values returned by `measure_gemm`, plot the results with execution time on the y-axis and matrix size on the x-axis

Bing

Sure, here's a function that does that. However, please note that Rust doesn't have built-in support for plotting. You would need to use a library like `plotters` or export the data to a CSV file and use a separate tool like Python's `matplotlib` or Excel to create the plot.

```

fn benchmark_gemm() {
    // Prepare a file to write the results
    let mut file = std::fs::File::create("gemm_benchmark.csv").unwrap();
    writeln!(file, "Matrix Size,Execution Time").unwrap();
}

```

```
// Iterate over matrix sizes from 10x10 to 1000x1000
for size in (10..=1000).step_by(10) {
  let time = measure_gemm(size, size, size);
  println!("Size: {}x{}, Time: {} ms", size, size, time);
  writeln!(file, "{},{},{},", size, time).unwrap();
}

println!("Benchmark results written to gemm_benchmark.csv");
}
```

This function writes the matrix size and corresponding execution time to a CSV file named `gemm_benchmark.csv`. You can then use this file to create a plot with any tool you prefer. The x-axis should represent the matrix size and the y-axis should represent the execution time.

Let me know if you need further assistance! 😊

User

write a python script which reads a CSV with values formatted as “matrix size, execution time” and plots the results with matrix size on the x-axis and execution time on the y-axis

Bing

Sure, here’s a Python script that does that. This script uses the pandas library to read the CSV file and matplotlib to create the plot.

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv('gemm_benchmark.csv')

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(data['Matrix Size'], data['Execution Time'], marker='o')
plt.title('GEMM Execution Time')
plt.xlabel('Matrix Size')
plt.ylabel('Execution Time (ms)')
plt.grid(True)
plt.show()
```

This script assumes that the CSV file is named `gemm_benchmark.csv` and is located in the same directory as the script. The CSV file should have two columns named ‘Matrix Size’ and ‘Execution Time’. The script reads the CSV file into a pandas DataFrame, then creates a line plot with ‘Matrix Size’ on the x-axis and ‘Execution Time’ on the y-axis.

Please make sure to install the required libraries by running `pip install pandas matplotlib` in your terminal before running the script.

Let me know if you need further assistance! 😊

User

what row delimiter is used for the `pandas.read_csv()` function?

Bing

The `pandas.read_csv()` function in Python uses a newline (`\n`) as the row delimiter by default. However, you can specify a custom row delimiter using the `lineterminator` parameter. Please note that the `lineterminator` parameter does not allow a string longer than 1 character.

Here is an example of how to use the `lineterminator` parameter:

```
import pandas as pd
```

```
# Read a CSV file with a custom row delimiter
data = pd.read_csv('filename.csv', lineterminator=':')
```

In this example, the colon (:) is used as the row delimiter. Please replace 'filename.csv' with the path to your CSV file.

Let me know if you need further assistance! 😊

User

I don't see any output from the plotting function which reads the csv. Here is the format of the csv file Matrix Size,Execution Time 10,0, 20,0, 30,1, 40,4, 50,7, 60,13, 70,22, 80,33, 90,46, 100,63, 110,81, 120,92, 130,124, 140,149, 150,191, 160,229, 170,293, 180,351, 190,401, 200,556, 210,548, 220,706, 230,714, 240,819, 250,921, 260,1091, 270,1182, 280,1320, 290,1451, 300,1557, 310,1953, 320,2277, 330,3491, 340,2674, 350,3033, 360,3042, 370,3240, 380,3533, 390,3719, 400,4265,

Bing

The CSV file format you provided seems correct. If you're not seeing any output from the plotting function, it could be due to a few reasons:

1. **Library Import Issues:** Make sure you have the necessary libraries (pandas and matplotlib) installed and imported correctly in your Python environment.
2. **File Path:** Ensure that the CSV file is located in the correct path that your Python script is referring to. If the file is in the same directory as your Python script, you can just use the filename. Otherwise, you need to provide the full path to the file.
3. **Plot Display:** In some environments, plots created by matplotlib don't show up automatically. You might need to call `plt.show()` explicitly to display the plot.
4. **Data Reading:** Make sure that pandas is reading the CSV file correctly. You can check this by printing out the DataFrame after reading the CSV file. If the data is not being read correctly, you might need to adjust the parameters of the `pandas.read_csv()` function.

Here's the adjusted Python script with explicit plot display and data checking:

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv('gemm_benchmark.csv')

# Check the data
print(data)

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(data['Matrix Size'], data['Execution Time'], marker='o')
plt.title('GEMM Execution Time')
plt.xlabel('Matrix Size')
plt.ylabel('Execution Time (ms)')
plt.grid(True)
plt.show()
```

If you're still having issues, could you please provide more details about the error or the issue you're facing? 😊

User

pandas seems to be interpreting the first value as a row label, here is the printed output Matrix Size Execution Time 10 0 NaN 20 0 NaN 30 1 NaN 40 4 NaN 50 7 NaN 60 13 NaN 70 22 NaN 80 33 NaN 90 46 NaN 100 63

NaN 110 81 NaN 120 92 NaN 130 124 NaN 140 149 NaN 150 191 NaN 160 229 NaN 170 293 NaN 180 351 NaN 190 401 NaN 200 556 NaN 210 548 NaN 220 706 NaN 230 714 NaN 240 819 NaN 250 921 NaN 260 1091 NaN 270 1182 NaN 280 1320 NaN 290 1451 NaN 300 1557 NaN 310 1953 NaN 320 2277 NaN 330 3491 NaN 340 2674 NaN 350 3033 NaN 360 3042 NaN 370 3240 NaN 380 3533 NaN 390 3719 NaN 400 4265 NaN

Bing

The issue you're experiencing is due to the default behavior of `pandas.read_csv()`. By default, it uses the first row of the CSV file as the header. If your CSV file doesn't have a header row, the first row of data will be used as the header, which seems to be the case here.

To fix this, you can use the `header=None` argument when calling `pandas.read_csv()`. This tells pandas that your CSV file does not have a header row, and the column names will be assigned automatically as integers.

Here's how you can modify your code:

```
import pandas as pd
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv('gemm_benchmark.csv', header=None, names=['Matrix Size', 'Execution Time'])

# Check the data
print(data)

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(data['Matrix Size'], data['Execution Time'], marker='o')
plt.title('GEMM Execution Time')
plt.xlabel('Matrix Size')
plt.ylabel('Execution Time (ms)')
plt.grid(True)
plt.show()
```

In this modified script, I've added `header=None` to the `read_csv()` function and also provided a `names` parameter to manually set the column names. Now, pandas should correctly interpret your CSV file.

Let me know if you need further assistance! 😊

User

how do I profile in microseconds using Rust?

Bing

In Rust, you can use the `std::time::Instant` struct to measure the execution time of a code block. Here's an example:

```
use std::time::Instant;

fn main() {
    let start = Instant::now();

    // Code block to profile
    // ...

    let elapsed = start.elapsed();
    let elapsed_micros = elapsed.as_micros();

    println!("Elapsed: {} microseconds", elapsed_micros);
}
```

In this example, `start` is the time at which the measurement begins. After the code block to profile, `elapsed` is the `Duration` since `start`. The `as_micros` method converts this `Duration` into microseconds.

Please note that the actual precision depends on your system's hardware and operating system.

Let me know if you need further assistance! 😊

User

create a function in Rust to add two m-by-n matrices

Bing

Sure, here's a simple function in Rust that adds two m-by-n matrices. This function assumes that the two input matrices have the same dimensions.

```
fn add_matrices(a: &Vec<Vec<f64>>, b: &Vec<Vec<f64>>) -> Vec<Vec<f64>> {
    let m = a.len();
    let n = a[0].len();
    let mut result = vec![vec![0.0; n]; m];

    for i in 0..m {
        for j in 0..n {
            result[i][j] = a[i][j] + b[i][j];
        }
    }

    result
}
```

This function iterates over each element in the matrices a and b, adds the corresponding elements together, and stores the result in a new matrix. The new matrix is then returned as the result.

Please note that this code does not include any error checking. In a production setting, you would want to add checks to ensure that the input matrices are well-formed and have the same dimensions.

Let me know if you need further assistance! 😊

User

Create a Python function that reads in two .csvs and plots them on the same figure as scatter plots. The two .csv files are named “tiled_gemm_benchmark.csv” and “gemm_benchmark.csv”. They each contain two columns: “Matrix Size” and “Execution Time”

Bing

Sure, here's a Python function that does that. This function uses the pandas library to read the CSV files and matplotlib to create the scatter plots.

```
import pandas as pd
import matplotlib.pyplot as plt

def plot_csvs(file1, file2):
    # Read the CSV files
    data1 = pd.read_csv(file1)
    data2 = pd.read_csv(file2)

    # Create the scatter plots
    plt.figure(figsize=(10, 6))
    plt.scatter(data1['Matrix Size'], data1['Execution Time'], color='blue', label=file1)
    plt.scatter(data2['Matrix Size'], data2['Execution Time'], color='red', label=file2)
    plt.title('Execution Time Comparison')
    plt.xlabel('Matrix Size')
    plt.ylabel('Execution Time (ms)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
# Call the function with your file names
plot_csvs('tiled_gemm_benchmark.csv', 'gemm_benchmark.csv')
```

This function reads the two CSV files into pandas DataFrames, then creates a scatter plot for each DataFrame on the same figure. The `Matrix Size` column is plotted on the x-axis and the `Execution Time` column is plotted on the y-axis. The `label` parameter in the scatter function is used to create a legend for the plot.

Please make sure to install the required libraries by running `pip install pandas matplotlib` in your terminal before running the script.

Let me know if you need further assistance! 😊