`p2p_matrix`: Rust library for distributed matrix multiplication

Group F
Florian Edelmann
Markus Jürgens
Thomas Kagermeier
Michael Schmid
Yuhao Wang

Concept

```
// on client machine

extern crate matrix;
extern crate p2p_matrix;
use matrix::prelude::*;
use p2p_matrix::PeerToPeerClient;

let distributed = PeerToPeerClient();

let matrix1 = &*matrix![
    85, 83, 28;
    72, 72, 91;
];
let matrix2 = &*matrix![
    12, 13, 34, 67;
    32, 45, 56, 76;
    64, 22, 67, 67;
];

let result = distributed.matrix_mult(matrix1, matrix2);
```

Concept

Daemon running on all peers

Distributed multiplication of matrices

Any peer can start the computation

Automatic work distribution in the background

Parallel matrix multiplication on working peers

Discovery of other peers: constantly poll the network for specific open port

Crates

`matrix`
: Matrix representation

`tokio`
: High performance network communication

`serde`
: Serializing and deserializing data

`serde-json`
: JSON parsing

Matrix split strategy

Inputs: matrices A of size $n \times m$, B of size $m \times p$.

- If $\max(n, m, p) = n$, split A horizontally:

$$C = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} B = \begin{pmatrix} A_1 B \\ A_2 B \end{pmatrix}$$

- Else, if $\max(n, m, p) = p$, split B vertically:

$$C = A\begin{pmatrix} B_1 & B_2 \end{pmatrix} = \begin{pmatrix} AB_1 & AB_2 \end{pmatrix}$$

- Otherwise, $\max(n, m, p) = m$. Split A vertically and B horizontally:

$$C = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = A_1 B_1 + A_2 B_2$$

Work Distribution

Local Peer:

- splits matrix according to strategy, until `num_tasks >= num_peers`
- distributes smaller matrix multiplication tasks to other peers
- reassemble partial results to result matrix and return to client

Remote Peers:

- calculate their tasks on multiple threads
- send back results

Data Structures

```
enum SplitLocation {
  Horizontal_1, Horizontal_2, Vertical_1, Vertical_2, Both }
```

`daemon.result_matrix: Map<Int, Vec<Int>>`
Save incoming partial matrix results with their respective id

`daemon.split_map: Map<Int, (SplitLocation, Int, Int)>`
When splitting, save `<split_id, (direction, parent_id, other_id)>` for matrix reconstruction