Week 6 exercises: MapReduce

EPFL SaCS and DIAS

EPFL

March 24, 2025

Exercise 1

You are given a symmetric social network (like Facebook) where a is a friend of b implies that b is also a friend of a.

- The input is a dataset D (sharded) containing such pairs of identifiers (a, b).
- Find the last names of those users whose first name is "Kanye" and who have at least 300 friends.

Exercise 1 Solution

MapReduce Pseudocode

procedure Map(a: User, b: User)

Exercise 1 Solution

MapReduce Pseudocode

```
procedure Map(a: User, b: User)
    if firstname(a) == "Kanye" then
        emit(a, b)
    end if
    if firstname(b) == "Kanye" then
        emit(b, a)
    end if
end procedure
```

Exercise 1 Solution

MapReduce Pseudocode

procedure Reduce(u: User, friends: List[User])

Exercise 1 Solution

MapReduce Pseudocode

```
procedure Reduce(u: User, friends: List[User])
    if length(friends) ≥ 300 then
        emit(lastname(u))
    end if
end procedure
```

Exercise 2

For an asymmetrical social network, you are given a dataset D where lines consist of (a, b) which means user a follows user b.

- Output the list of all users U who:
  1. Have at least 2 million followers,
  2. Follow fewer than 20 other users,
  3. Are followed back by all the users they follow.

Exercise 2 Solution

Map Function Pseudocode

procedure Map(a: User, b: User)

Exercise 2 Solution

Map Function Pseudocode

procedure Map(a: User, b: User)
    emit($a, \langle b, 1 \rangle$) ▷ a follows b
    emit($b, \langle a, 0 \rangle$) ▷ b is followed by a
end procedure

Exercise 2 Solution

Reduce Function Pseudocode

procedure Reduce(u: User, list: List[$\langle$`User`, `Int`$\rangle$])

Exercise 2 Solution

Reduce Function Pseudocode

```
procedure Reduce(u: User, list: List[⟨User, Int⟩])
    follows ← ∅
    count_0 ← 0
    count_1 ← 0
    for all pair in list do
        if pair.value = 0 then
            count_0 ← count_0 + 1                ▷ Count followers
        end if
        if pair.value = 1 then
            count_1 ← count_1 + 1                ▷ Count follows
            follows ← follows ∪ pair.key
        end if
    end for
end procedure
```

Exercise 2 Solution

Reduce Function Pseudocode

procedure Reduce(u: User, list: List[$\langle$`User`, `Int`$\rangle$])
    ...
    if $\texttt{count}_0 \geq 2M$ **and** $\texttt{count}_1 < 20$ **then**
        for all user **in** follows **do**
            if $\langle \texttt{user}, 0 \rangle \notin \texttt{list}$ **then**
                return ▷ User not followed back
            end if
        end for
        emit(u)
    end if
end procedure

Exercise 3

Matrix multiplication is a fundamental operation in machine learning. Design a Map-Reduce program for computing the product $\boldsymbol{M} = \boldsymbol{AB}$.

- $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times m}$.
- Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are represented through mn pairs.
- Each pair corresponds to (A, i, j,A[i, j]) or (B, i, j,B[i, j]).

Matrix Multiplication Example

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{pmatrix}$$

$$\text{General Form:} \quad M_{i,j} = \sum_{l=0}^{n-1} A_{i,l} \times B_{l,j}$$

Exercise 3 Solution: Map Function

MapReduce Pseudocode for Map

procedure Map($\langle X, i, j, X_{i,j} \rangle$: $\langle \texttt{String}, \texttt{String}, \texttt{String}, \texttt{Float} \rangle$)

Exercise 3 Solution: Map Function

MapReduce Pseudocode for Map

procedure Map($\langle X, i, j, X_{i,j} \rangle$: $\langle \texttt{String}, \texttt{String}, \texttt{String}, \texttt{Float} \rangle$)
    if $X =$ "A" **then**
        for k in {0, 1, 2, ..., m-1} **do**
            emit($\langle i, k \rangle, \langle X, j, X_{ij} \rangle$)
        end for
    end if

Exercise 3 Solution: Map Function

MapReduce Pseudocode for Map

procedure Map($\langle X, i, j, X_{i,j} \rangle$: $\langle \texttt{String}, \texttt{String}, \texttt{String}, \texttt{Float} \rangle$)
    if $X =$ "A" **then**
        for k in {0, 1, 2, ..., m-1} **do**
            emit($\langle i, k \rangle, \langle X, j, X_{ij} \rangle$)
        end for
    end if
    if $X =$ "B" **then**
        for k in {0, 1, 2, ..., m-1} **do**
            emit($\langle k, j \rangle, \langle X, i, X_{ij} \rangle$)
        end for
    end if
end procedure

Exercise 3 Solution: Reduce Function

MapReduce Pseudocode for Reduce

procedure Reduce($\langle i, j \rangle$: $\langle \texttt{String}, \texttt{String} \rangle$, list: List[$\langle \texttt{String}, \texttt{String}, \texttt{Float} \rangle$])

Exercise 3 Solution: Reduce Function

MapReduce Pseudocode for Reduce

procedure Reduce($\langle i, j \rangle$: $\langle \texttt{String}, \texttt{String} \rangle$, list: List[$\langle \texttt{String}, \texttt{String}, \texttt{Float} \rangle$])
    $\texttt{Avec} \leftarrow \emptyset$
    $\texttt{Bvec} \leftarrow \emptyset$
    $M_{i,j} \leftarrow 0$
    for all $\langle X, k, X_k \rangle$ **in** list **do**
        if $X =$ "A" **then**
            $\texttt{Avec} \leftarrow \texttt{Avec} \cup \langle k, X_k \rangle$
        end if
        if $X =$ "B" **then**
            $\texttt{Bvec} \leftarrow \texttt{Bvec} \cup \langle k, X_k \rangle$
        end if
    end for
end procedure

Exercise 3 Solution: Reduce Function

MapReduce Pseudocode for Reduce

procedure Reduce($\langle i, j \rangle$: $\langle \texttt{String}, \texttt{String} \rangle$, list: List[$\langle \texttt{String}, \texttt{String}, \texttt{Float} \rangle$])

    ...
    sort(Avec)
    sort(Bvec)
    for k in {0, 1, 2, ..., n-1} **do**
        $M_{i,j} \leftarrow M_{i,j} + \texttt{Avec}[k] \times \texttt{Bvec}[k]$
    end for
    emit($\langle M, i, j, M_{i,j} \rangle$)

end procedure

Exercise 4: Word Count

- Word count for a dataset comprising W total words with d distinct words.
- The mappers receive a single word as input.
- Compute the total communication cost between the mappers and the reducers.

Choices

- d
- W
- $\frac{W}{d}$
- dW
- $2W$

Exercise 4: Word Count Solution

MapReduce Pseudocode for Word Count

```
procedure Map(word: String)
    emit(word, 1)
end procedure
procedure Reduce(word: String, counts: List[Integer])
    total ← 0
    for all count in counts do
        total ← total + count
    end for
    emit(word, total)
end procedure
```

Exercise 4: Word Count

Choices

- d
- W – *The algorithm will emit one key-value pair per word.*
- $\frac{W}{d}$
- dW
- $2W$

Exercise 4: Matrix Multiplication Communication Cost

- Matrix multiplication of two matrices of size $m \times n$ and $n \times p$.
- Mappers read input tuples in the form $\langle$Matrix identifier, row index, column index, value$\rangle$.
- What is the communication cost between the mappers and the reducers?

Choices

- mp
- $n(m+p)$
- $2n(m+p)$
- mnp
- $2mnp$

Exercise 4: Matrix Multiplication Communication Cost Solution

MapReduce Pseudocode for Matrix Multiplication

```
procedure Map(MatrixID: String, i: Integer, j: Integer, value: Float)
    if MatrixID = "A" then
        for k ← 1 to p do
            emit((i, k), (A, j, value))
        end for
    else if MatrixID = "B" then
        for k ← 1 to m do
            emit((k, j), (B, i, value))
        end for
    end if
end procedure
```

Exercise 4: Matrix Multiplication Communication Cost

Choices

- mp
- $n(m+p)$
- $2n(m+p)$
- mnp
- $2mnp$ – *Each element (i,j) will require n elements of the first matrix and n elements of the second matrix.*

Exercise 4: Part (c1) - INNER JOIN

- Compute the INNER JOIN of two relations.
- $R1(X, Y)$ with 4 tuples {(5,21), (7,16), (15,3), (3,21)}
- $R2(Y, Z)$ with 3 tuples {(3,1), (4,8), (21,28)}.
- Mappers read input tuples in the form ⟨Relation identifier, X, Y⟩.

Question

How many key-value pairs are emitted by the mappers?

Choices

- 12
- 7
- 4
- 3
- 2

Exercise 4: INNER JOIN

MapReduce Pseudocode for INNER JOIN

```
1: procedure Map(RelationID: String, X: Integer, Y: Integer)
2:     if RelationID = "R1" then
3:         emit(Y, (RelationID, X))
4:     else
5:         emit(X, (RelationID, Y))
6:     end if
7: end procedure
```

Exercise 4: Part (c1) - INNER JOIN

Choices

- 12
- 7 – *Mappers will emit rows of both relations with a tag.*
- 4
- 3
- 2

Exercise 4: Part (c2) - INNER JOIN Output

- Compute the INNER JOIN of two relations.
- $R1(X, Y)$ with 4 tuples {(5,21), (7,16), (15,3), (3,21)}
- $R2(Y, Z)$ with 3 tuples {(3,1), (4,8), (21,28)}.
- Mappers read input tuples in the form ⟨Relation identifier, X, Y⟩.

Question

How many output tuples are produced by the reducers?

Choices

- 12
- 7
- 4
- 3
- 2

Exercise 4: INNER JOIN

MapReduce Pseudocode for INNER JOIN

```
1: procedure Reduce(Key: Integer, Values: List[Tuple])
2:     R1_list ← []
3:     R2_list ← []
4:     for all (RelationID, Value) in Values do
5:         if RelationID = "R1" then
6:             append R1_list with Value
7:         else
8:             append R2_list with Value
9:         end if
10:    end for
11:    for all v1 in R1_list do
12:        for all v2 in R2_list do
13:            emit((v1, Key, v2))
14:        end for
15:    end for
16: end procedure
```

Exercise 4: Part (c2) - INNER JOIN Output

Choices

- 12
- 7
- 4
- 3 – *Outputs are (5, 21, 28), (15, 3, 1), (3, 21, 28).*
- 2

Exercise 4: Part (d) - Set Difference

- Compute the difference of two sets X and Y with x and y elements respectively.
- Mappers read input tuples in the form $\langle$Set identifier, Value$\rangle$.
- Recall: The difference of two sets X and Y is a set that contains those elements of X that are NOT in Y.

Question

Compute the total communication cost between the mappers and the reducers.

Choices

- $x + y$
- $x - y$
- x
- y
- xy

Exercise 4: Part (d) - Set Difference Solution

MapReduce Pseudocode for Set Difference

```
procedure Map(SetId: String, Value: Integer)
    emit(Value, SetId)
end procedure
procedure Reduce(Value: Integer, SetIds: List[String])
    if "X" in SetIds and "Y" not in SetIds then
        emit(Value)
    end if
end procedure
```

Exercise 4: Part (d) - Set Difference

Choices

- $x + y$ – *All values are emitted once from both sets.*
- $x - y$
- x
- y
- xy