HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Collaborative Filtering
Scalable Data Analysis Algorithms

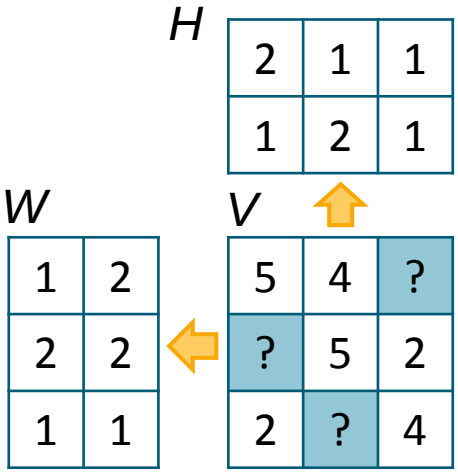
Claudia Lehmann, Andrina Mascher

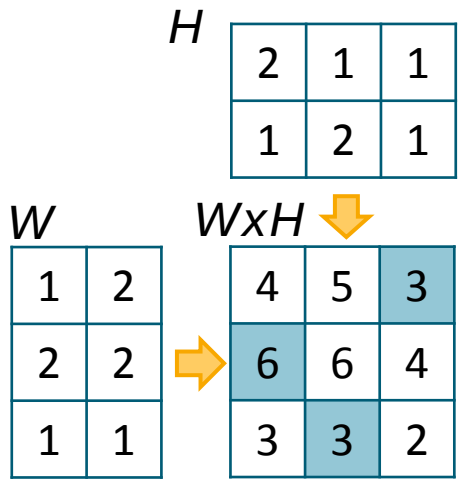
Outline

1. Retrospection
2. Stratosphere Plans
3. Comparison with Hadoop
4. Evaluation
5. Outlook

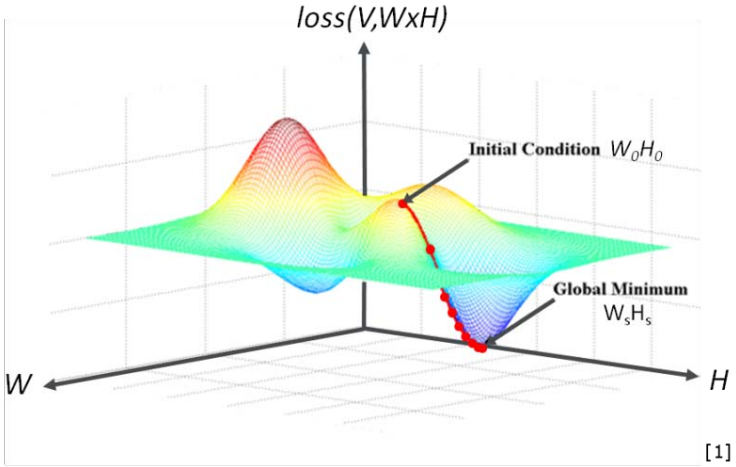
Retrospection

Matrix factorization:
Find optimal factors



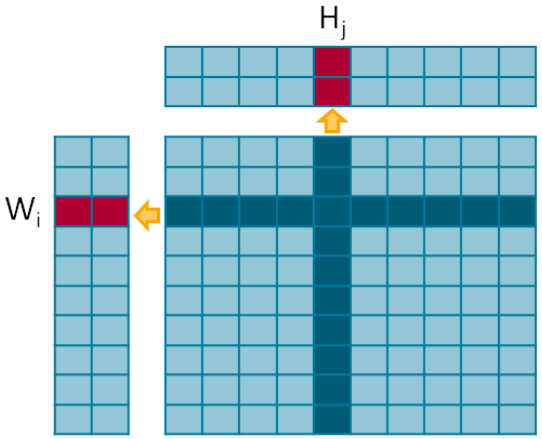


Stochastic Gradient Descent

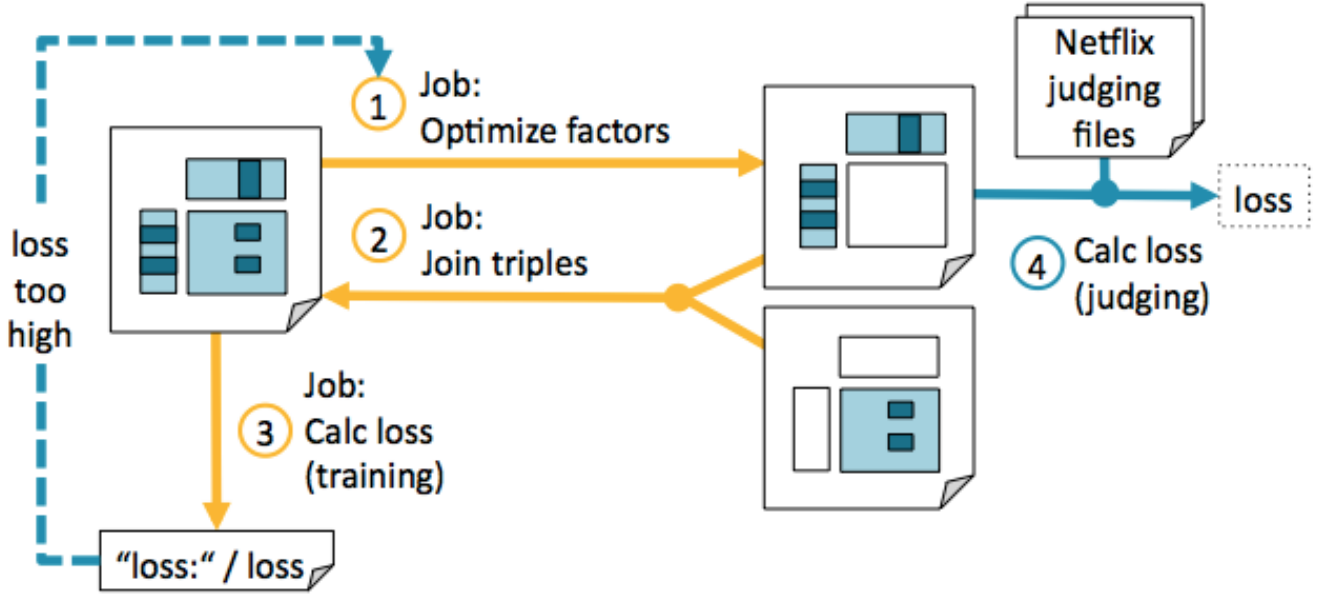


Retrospection

Result of SGD step



Hadoop jobs with judging loss



Stratosphere Plans
Multiple Iterations

Hasso Plattner Institut

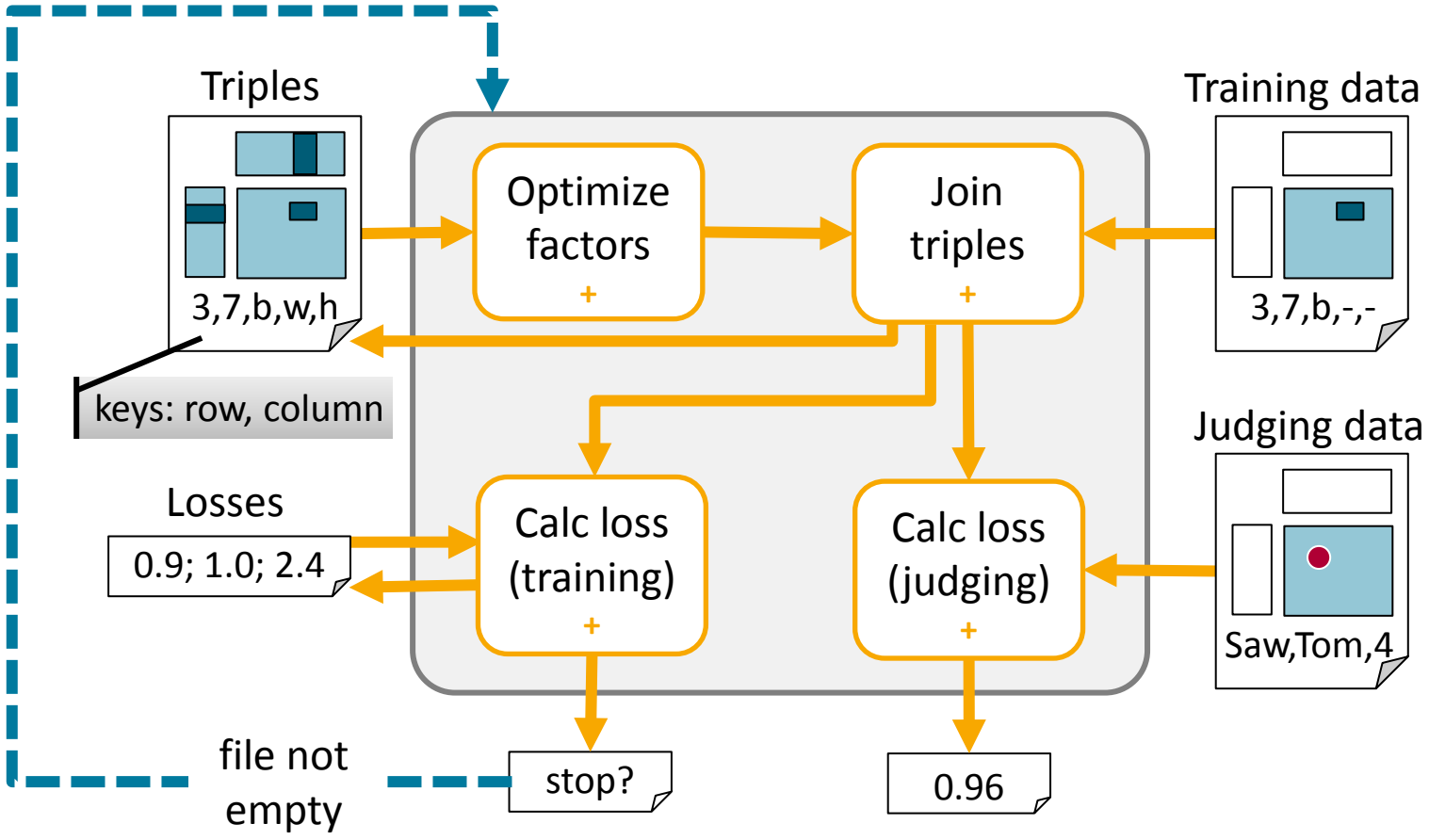

```
for each starting point:
  while loss too high:
      MapReduce      optimize factors
      MapReduce      join triples
      MapReduce      calculate loss (training)
      calculate loss (judging)
```

hadoop

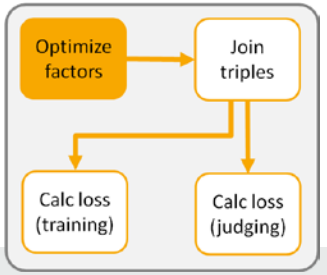
StratoSphere

```
for each starting point:
  while stop file not empty:
      MapMapReduce   optimize factors
      CrossMatch     join triples
      MapReduceCross calculate loss (training) and decide
      MapMatchReduce calculate loss (judging)
```

Stratosphere Plans
Subplans in One Iteration



Stratosphere Plans
Optimize Factors



Triples

Map: SGD

Reduce: average

Factors

hadoop

StratoSphere

not materialized

Factor W

Triples

3,7,b,w,h
5,7,b,w,h

Map: SGD

3,-,-,w,-
-,7,-,-,h

5,-,-,w,-
-,7,-,-,h

Map: filter w

Reduce: average

3,-,-,w,-
5,-,-,w,-

pacts configured to read fields

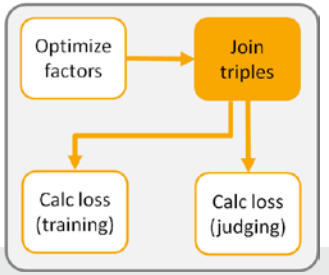
Factor H

Map: filter h

Reduce: average

-,7,-,-,h

Stratosphere Plans
Join Triples (1/2)



Factors

Training data

needs matrix dimensions

Map: replicate factors, keep training data

Reduce: group

Triples

hadoop

StratoSphere

Factor W

3,-,-,w,-

Factor H

-,7,-,-,h

Cross

3,7,-,w,h

Training data

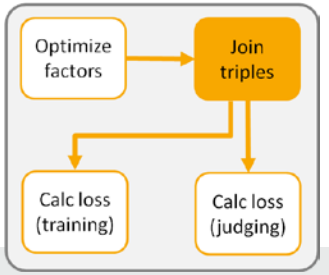
3,7,b,-,-

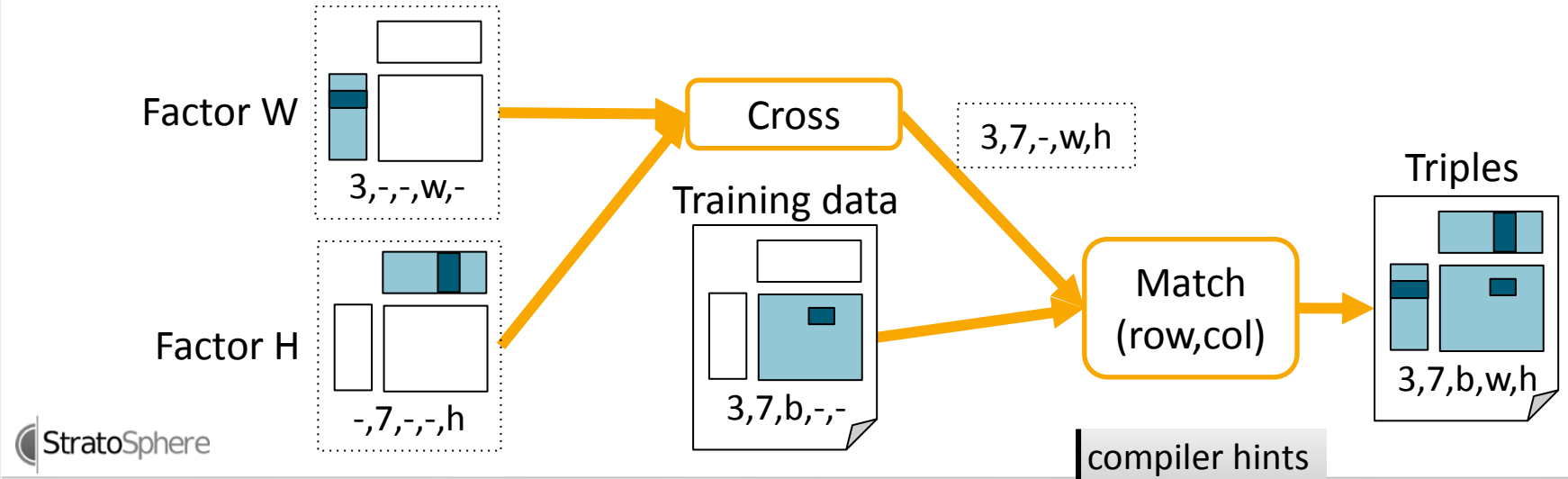
Match (row,col)

Triples

3,7,b,w,h

Stratosphere Plans
Join Triples (2/2)

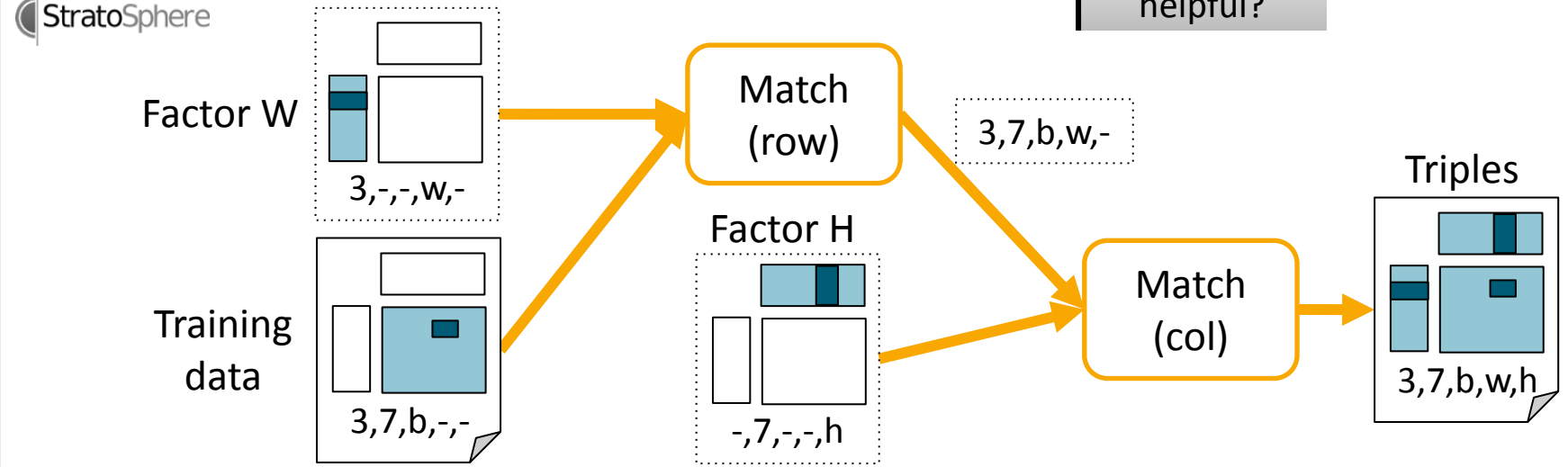




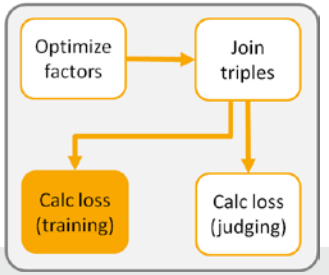
StratoSphere

compiler hints helpful?

StratoSphere



Stratosphere Plans
Calculate Loss (Training)



Triples

Map: local loss

Reduce: RMSE

0.9

driver class knows loss history and decides on stopping

hadoop

dummy, loss, #points

StratoSphere

Triples

Map: local loss

Reduce: RMSE

1

Cross: loss history

1

OutputFormat decide stop

stop?

1

1

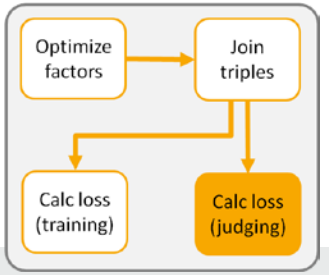
Losses (epoch e)

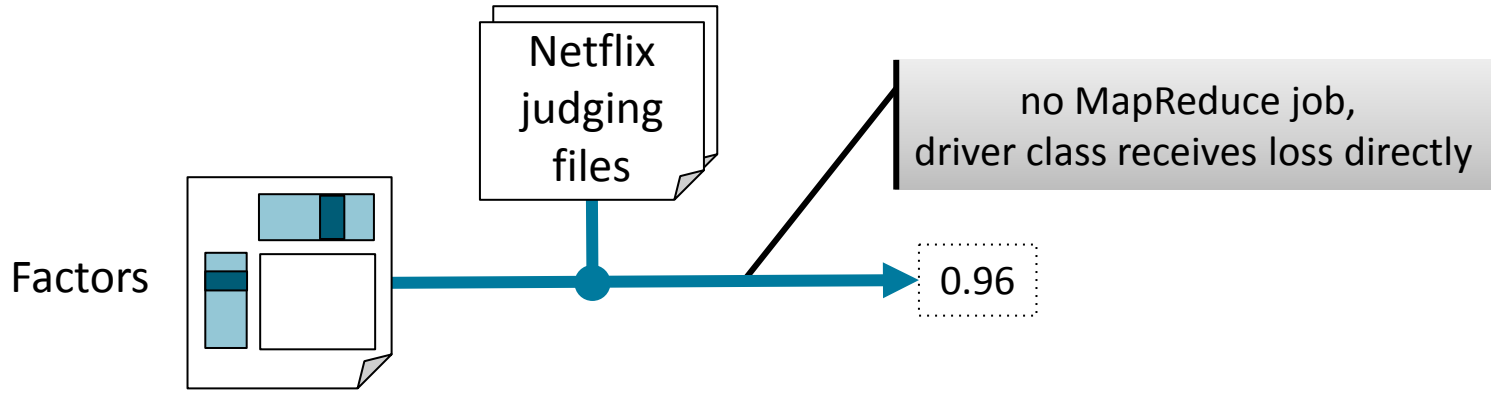
1.0; 2.4

0.9; 1.0; 2.4

Losses (epoch e+1)

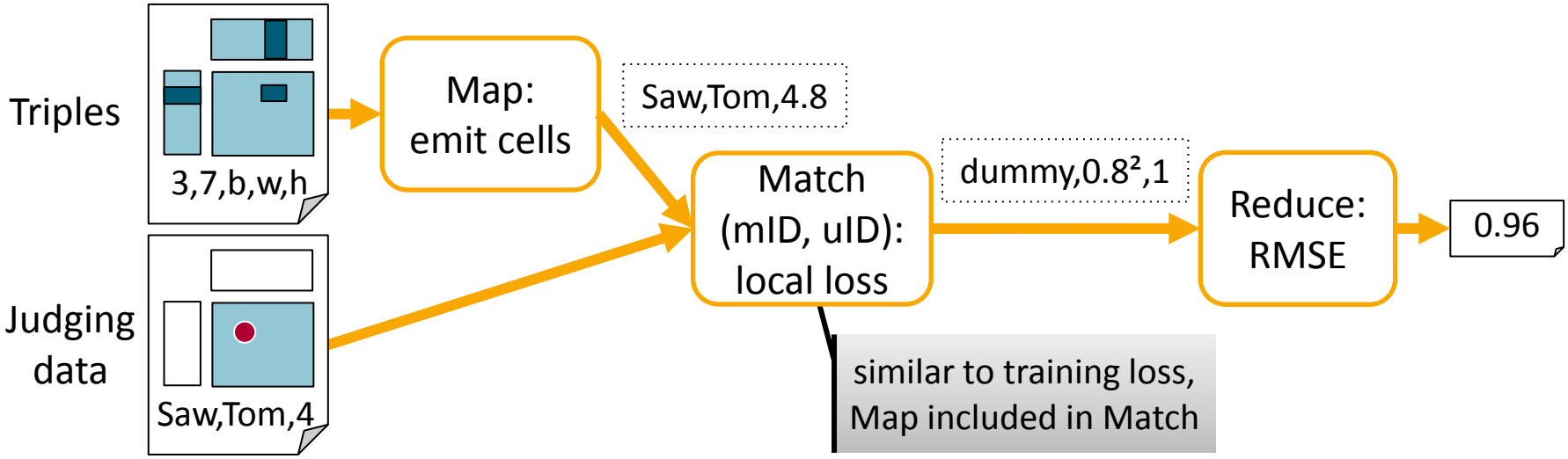
Stratosphere Plans
Calculate Loss (Judging)





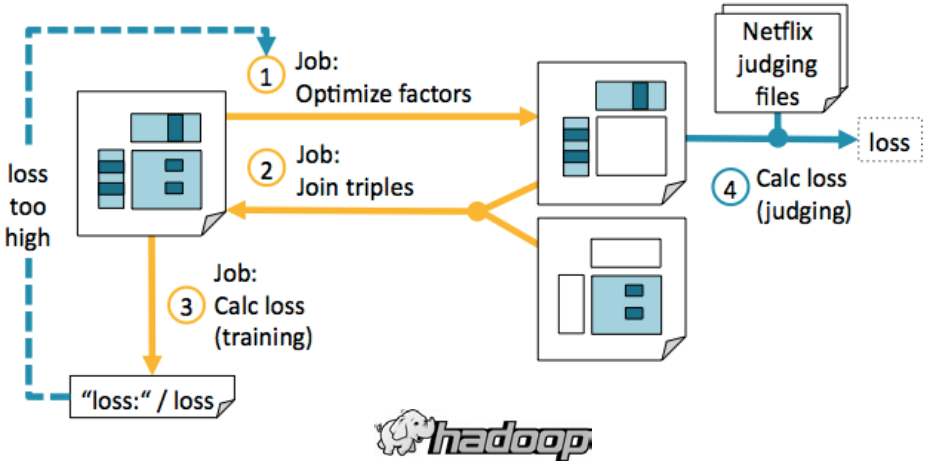
hadoop

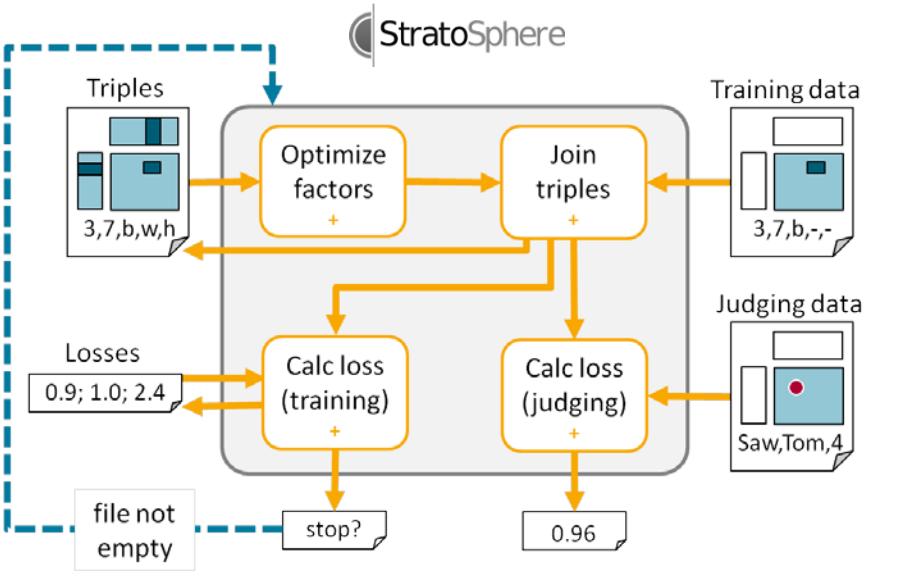
StratoSphere



Comparison
Jobs vs. Plans

- Equal results without random...
  - Starting points
  - Training sequence
- Files
  - Factors not materialized
  - Separate factor files possible
  - Create new file for each iteration
- No efficient serialization between plans (yet)
  - Either parsing text file
  - Or use sequence file single-threaded

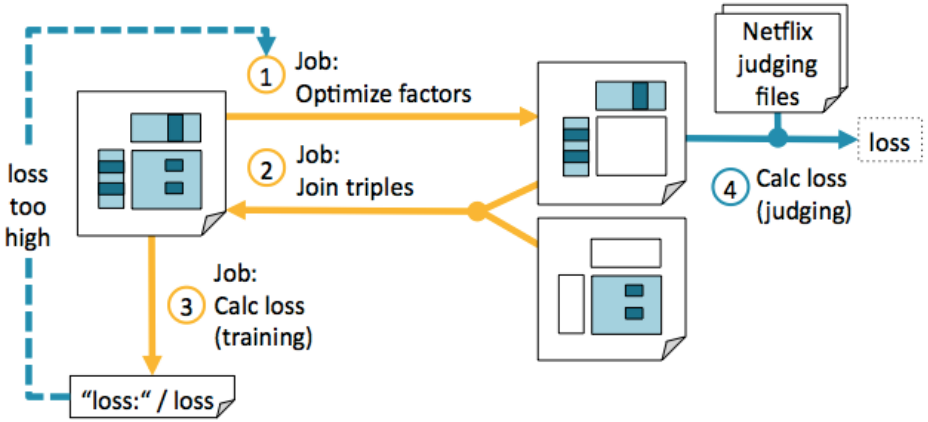




Comparison
Data Schema

- 3#7 / TripleStorage

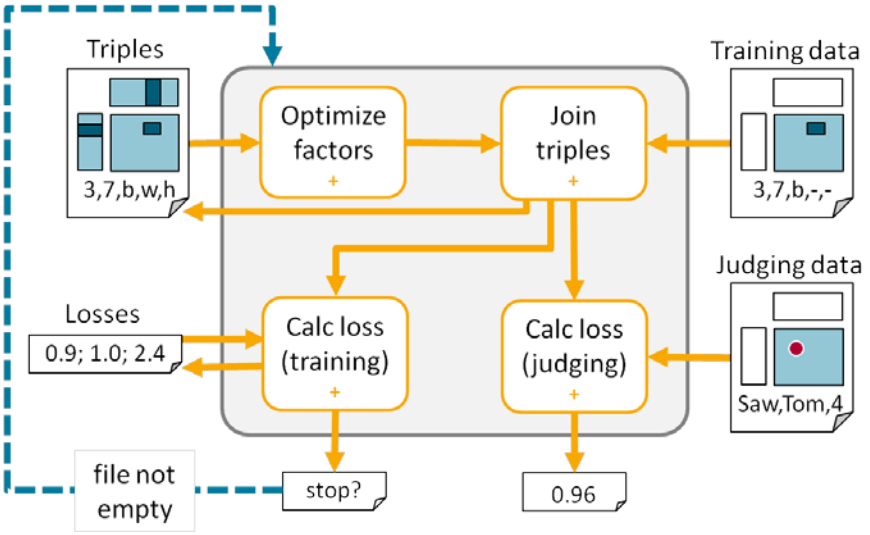
  (Storage class is tagged union)
- Dummy key / LossStorage
- → Getter, setter, toString()



hadoop

StratoSphere

- 3,7,b,w,h
- Dummy key, loss, #points
- → Remember key places
- → Reuse pacts with configurations for different keys
- → Composite keys possible



Comparison Preprocessing

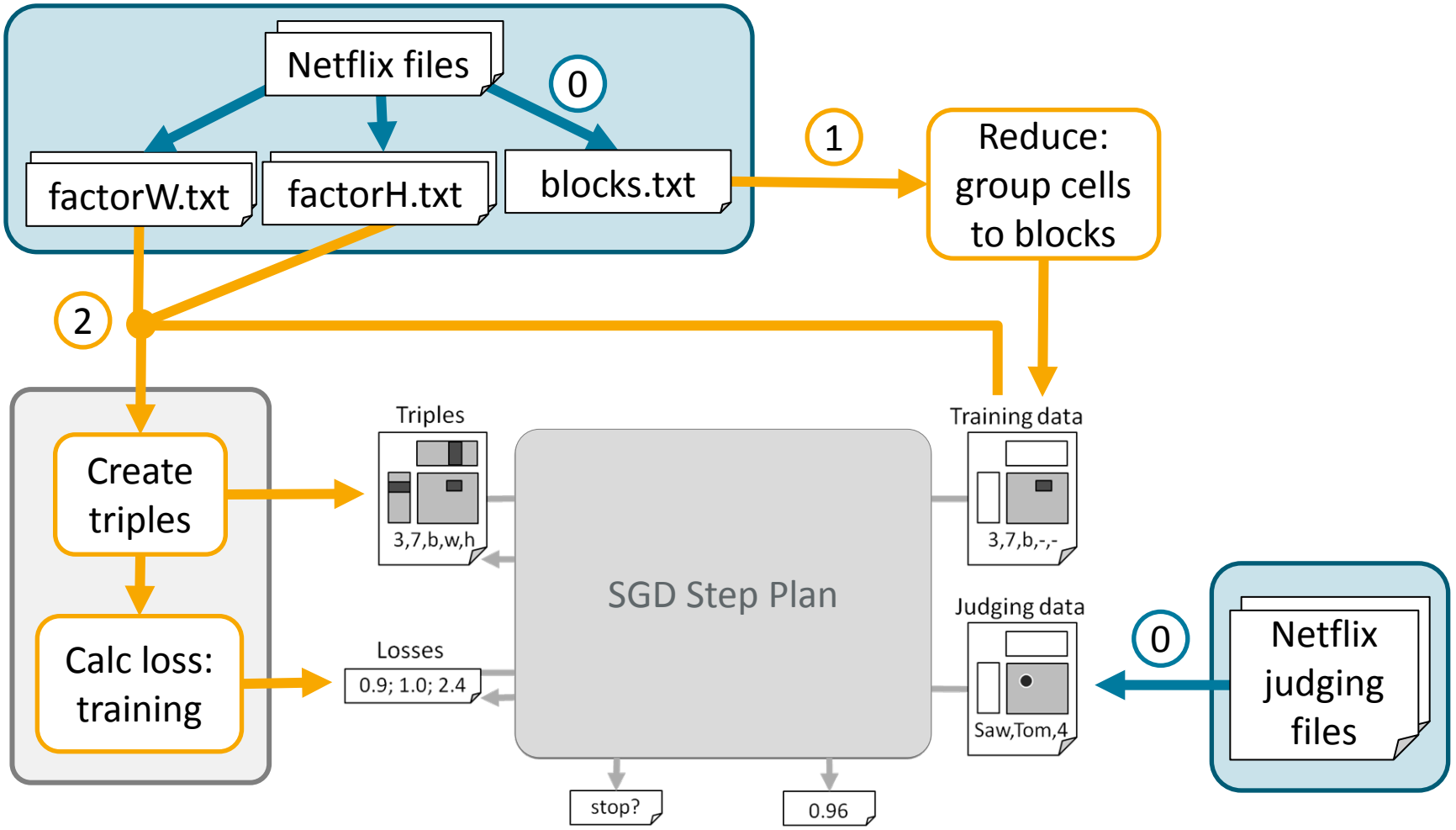
Requires:

- Line format according to parameters
- Copy to HDFS
- Serialize factors and blocks

- Use Map file to write serialized values to HDFS

- Java process to define lines
- Shell script to move to HDFS
- Extra pacts to parse lines

Stratosphere Preprocessing
Define Line Format



Suggestions

- Global aggregation of loss with dummy key:
  - Reducer with no key
  - Reducer with compiler hint nrOfKeys = 1
  - → No sorting needed
- Provide toString() in PactRecord
- Provide getter, setter in PactRecord to encapsulate field numbers and class types
  - Keep configuration options (e.g. reduce: average W or H)
- Sequence files for sinks and sources
- Move log files to master node

Evaluation
Parameters for Netflix Data

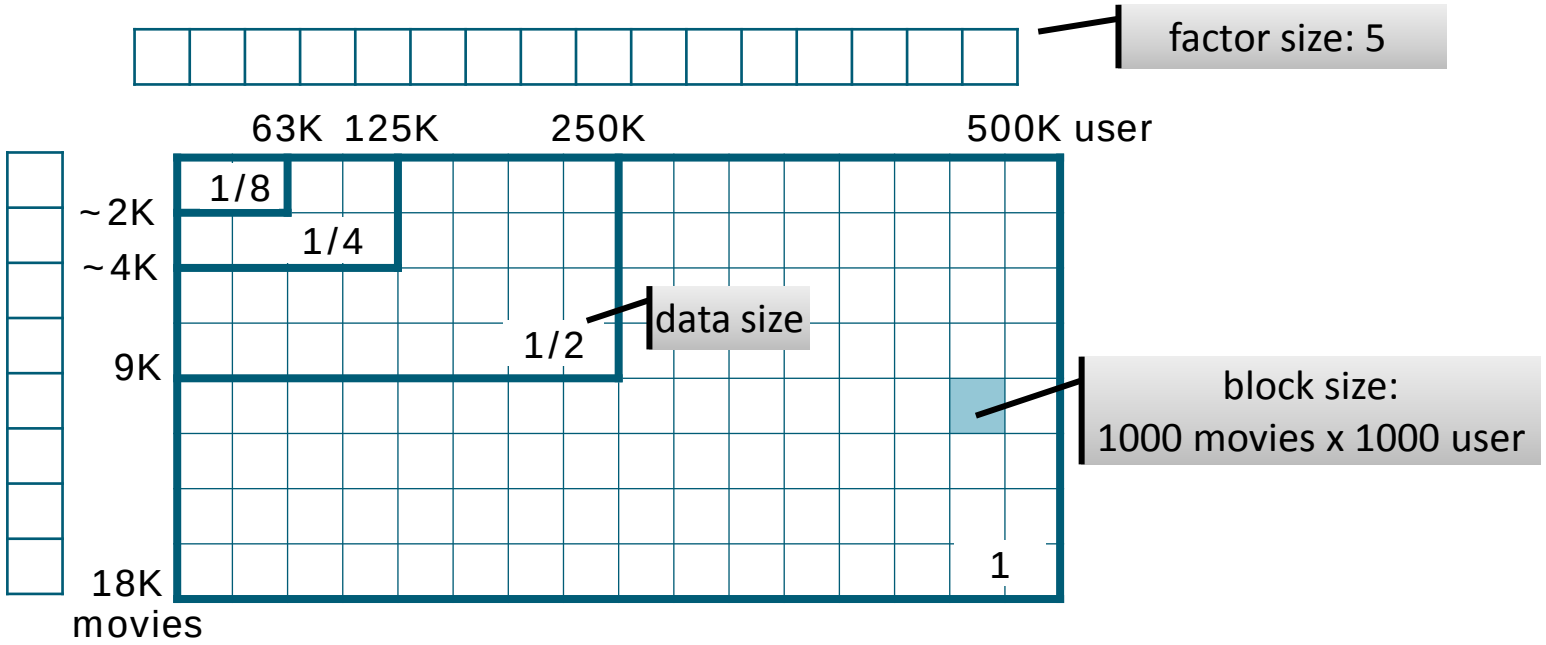
starting points: 1

max. iterations: 1

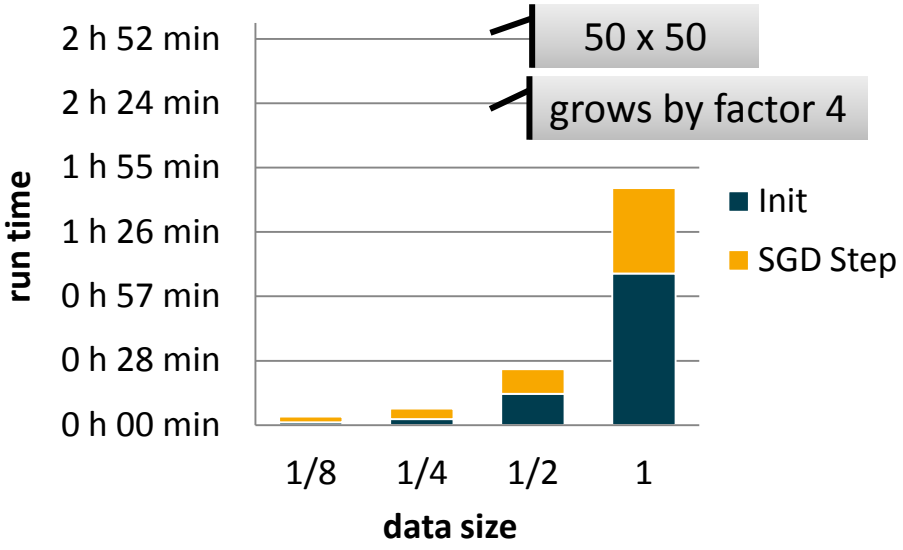
degree of parallelism: 1, 2, 5, 10

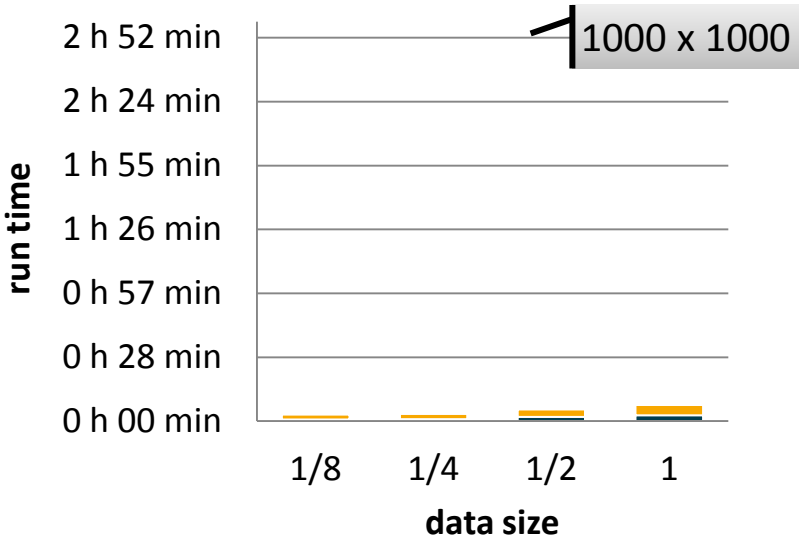
```
for each starting point:
while stop file not empty:
    MapMapReduce      optimize factors
    CrossMatch        join triples
    MapReduceCross    calculate loss (training) and decide
    MapMatchReduce    calculate loss (judging)
```

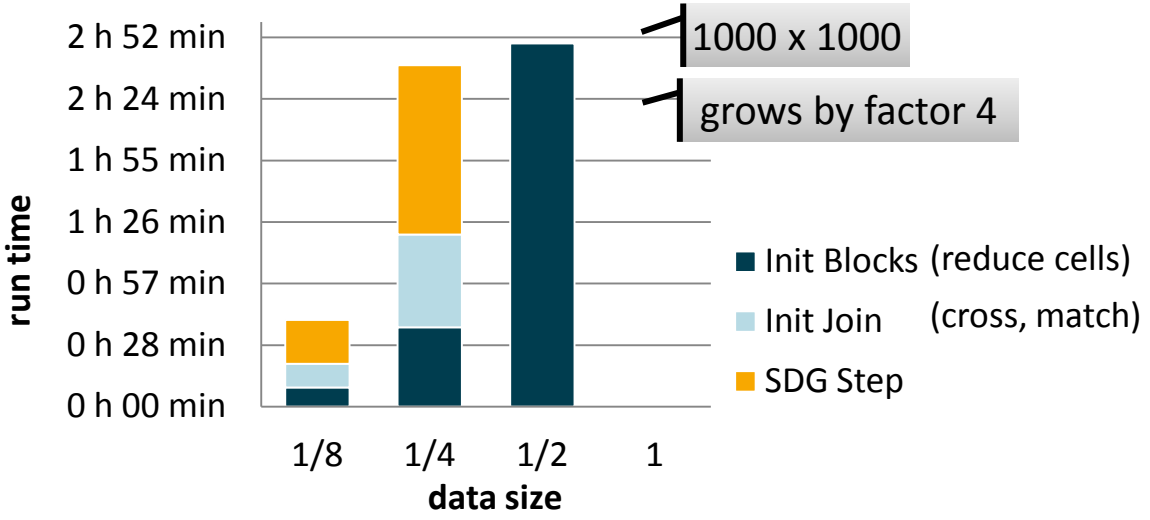
step size: $\frac{1}{\text{iteration} + 8}$



Evaluation
Run Time for Variable Data Size



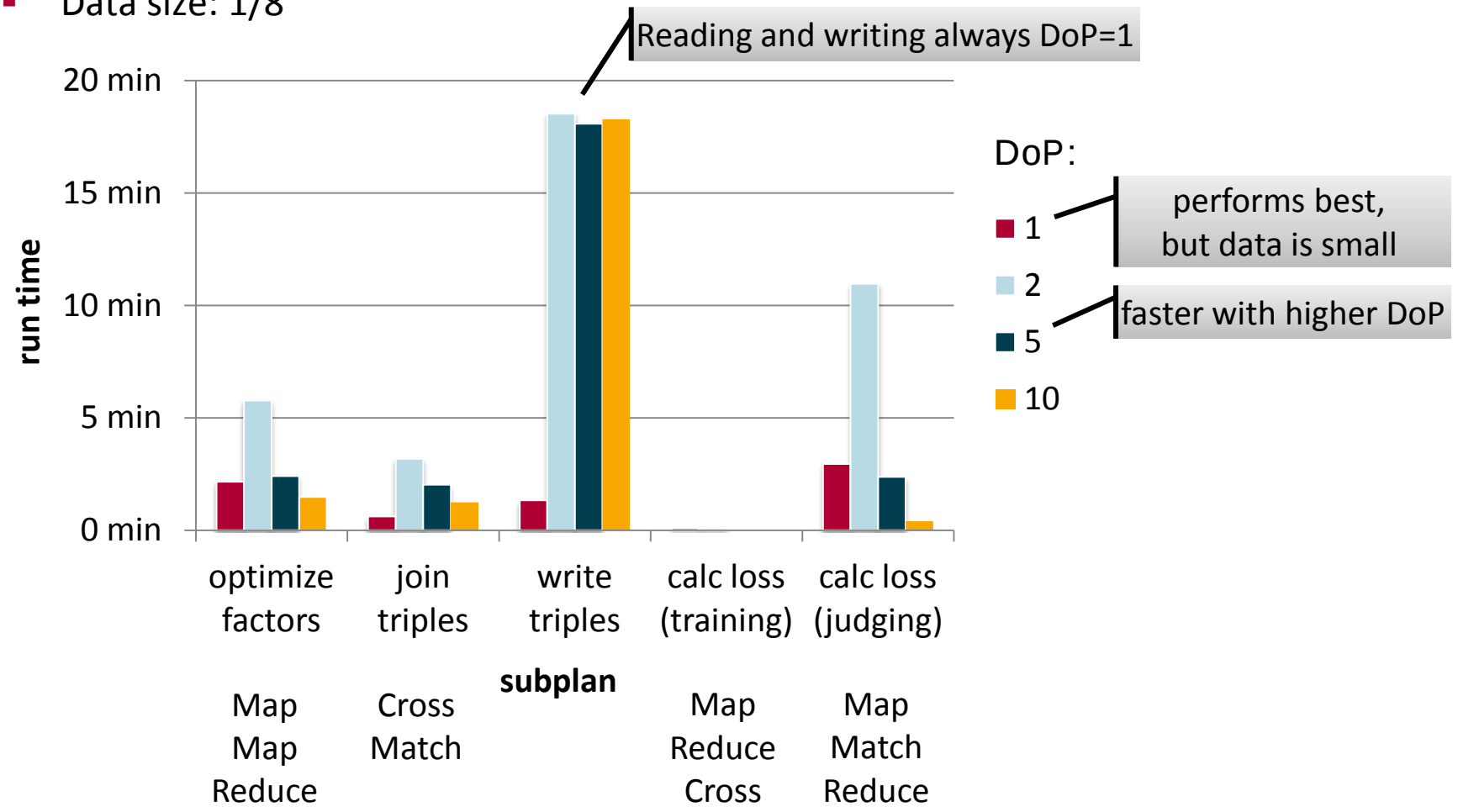




- 10 nodes vs. DoP = 8
- Usually 10 - 30 iterations
- No sequence file between plans

Evaluation
Run Time for Variable Degree of Parallelism

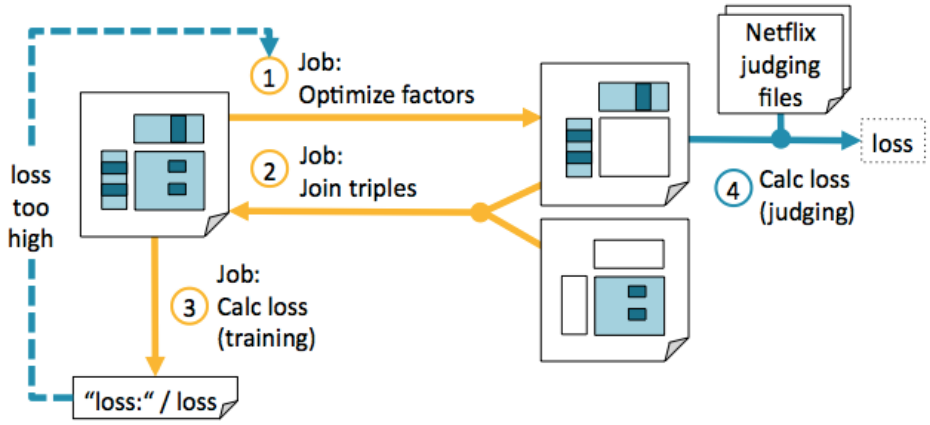
- Data size: 1/8

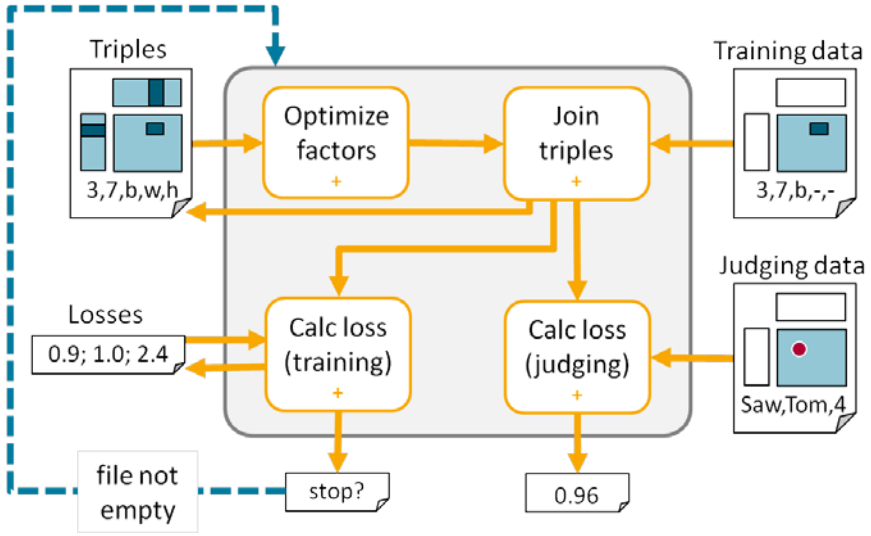


Outlook

- Join triples with Cross-Match vs. Match-Match
- Degree of parallelism: 10
- Inspect judging outcome: RMSE should be equal
- Evaluate DoP with bigger data

Summary


Netflix judging files
loss
1 Job: Optimize factors
2 Job: Join triples
4 Calc loss (judging)
loss too high
3 Job: Calc loss (training)
"loss:" / loss


Triples
3,7,b,w,h
Optimize factors
Join triples
Training data
3,7,b,-,-
Judging data
Saw,Tom,4
Losses
0.9; 1.0; 2.4
Calc loss (training)
Calc loss (judging)
file not empty
stop?
0.96

References

- Anand Rajaraman and Jeff Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2010.
- Rainer Gemulla, Peter J. Haas, Erik Nijkamp, and Yannis Sismanis. *Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent.* IBM Research Report RJ10481, March 2011.

[1] http://www.mathworks.de/matlabcentral/fx_files/27631/1/fff.png, November 2011