

Computing Recommendations at Extreme Scale with Apache Flink



Till Rohrmann

Flink committer / data Artisans

trohrmann@apache.org

@stsffap

Recommendations: Collaborative Filtering

Recommendations



- Omnipresent nowadays
- Important for user experience and sales

amazon.com

Recommended for You

Amazon.com has new recommendations for you based on items you purchased or told us you own.

LOOK INSIDE! *Google Apps*
LOOK INSIDE! *Google Apps Administrator Guide*
LOOK INSIDE! *Googlepedia: The Ultimate Google Resource (3rd Edition)*

[Google Apps Deciphered: Compute in the Cloud to Streamline Your Desktop](#)
[Administrator Guide: A Private-Label Web Workspace](#)
[Googlepedia: The Ultimate Google Resource \(3rd Edition\)](#)

NETFLIX

Welcome,

Congratulations! Movies we think You will ❤️
Add movies to your Queue, or Rate ones you've seen for even better suggestions.

The Scarlet Letter
Unfaithful
Two can play that game
Indecent Proposal

Spotify Premium

Recommended Albums

MUMFORD & SONS - Babel
Sonic Boom Six
#3

Featured Spotify Apps

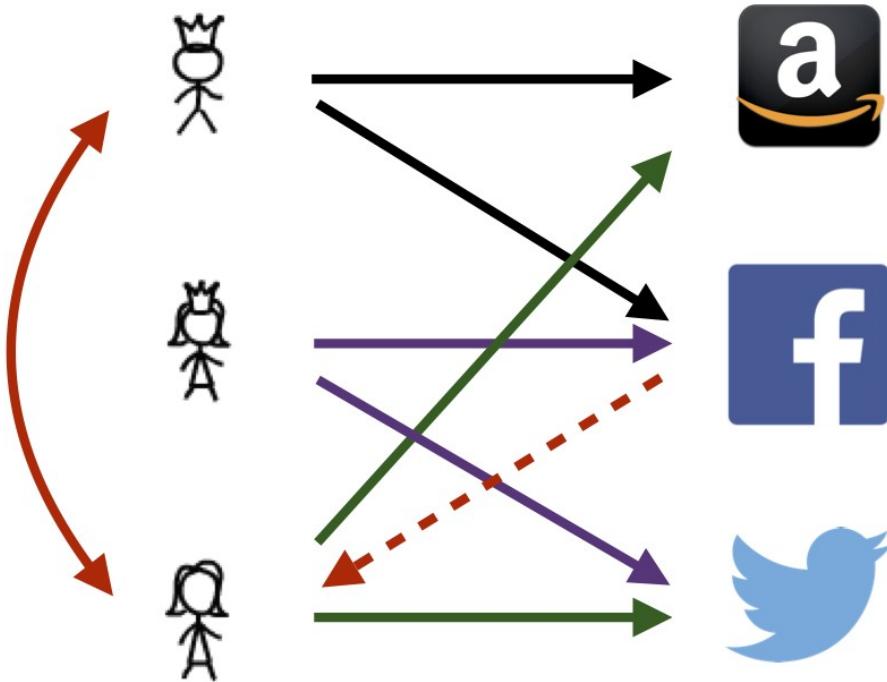
Antidote - Swedish House Mafia, Kefle Pa

History: 2007-2009

Mumford & Sons was formed in December 2007 by multi-instrumentalists Marcus Mumford, Ben Lovett, Winston Marshall, and Ted Dwane. The band's sound is heavily influenced by Mumford's folk guitar, and traditional folk instruments such as banjos, mandolin, and resonator guitar. The band's influences come from the folk duo Mumford and Sons and the post-punk/alternative rock group R.E.M.

3

Recommending Websites



Collaborative Filtering



- Recommend items based on users with similar preferences
- Latent factor models capture underlying characteristics of items and preferences of user
- Predicted preference: $\hat{r}_{ui} = \mathbf{x}_u^T \mathbf{y}_i$

Rating Matrix



- Explicit or implicit ratings
- Prediction goal: Rating for unseen items

		Items		
		10	5	?
Users	?	2	10	⋮
	7	?	5	⋮
		Princess		
		Facebook		



Matrix Factorization

- Calculate low rank approximation to obtain latent factors
Items

$$Users \begin{pmatrix} 10 & 5 & ? \\ ? & 2 & 10 \\ 7 & ? & 5 \end{pmatrix} \stackrel{\div}{\approx} X \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \bullet Y \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix}$$

Facebook Princess

The diagram illustrates matrix factorization. On the left, a 3x3 matrix labeled "Users" contains ratings: (10, 5, ?), (? , 2, 10), and (7, ?, 5). This matrix is divided by a symbol (÷) and approximated by the product of two matrices, X and Y. Matrix X has three rows, indicated by three vertical dots (⋮) under its columns. Matrix Y also has three rows, indicated by three vertical dots under its columns. To the right of the product, there is a dot (•) between X and Y. Above the product, the word "Facebook" is aligned with the third column of Y, and below it, the word "Princess" is aligned with the second column of X. A red dashed rectangle highlights the second column of X, and a blue dashed rectangle highlights the third column of Y.

$$\min_{X,Y} \sum_{r_{ui} \neq 0} (r_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u n_u \|x_u\|^2 + \sum_i n_i \|y_i\|^2 \right)$$

- r_{ui} rating of user u for item i
- x_u latent factors of user u
- y_i latent factors of item i
- λ regularization constant
- n_u number of rated items for user u
- n_i number of ratings for item i

Alternating Least Squares



- Hard to optimize since we have two variables X, Y
- Fixing one variables gives quadratic problem

$$x_u = \left(YS^u Y^T + \lambda n_u I \right)^{-1} Y r_u^T$$

$$S_{ii}^u = \begin{cases} 1 & \text{if } r_{u,i} \neq 0 \\ 0 & \text{else} \end{cases}$$



We only need the item vectors rated by user u

ALS Algorithm



- Update user matrix

Items Calculate update

Users $\begin{pmatrix} 10 & 5 & ? & \vdots \\ ? & 2 & 10 & \vdots \\ 7 & ? & 5 & \vdots \end{pmatrix} \approx \begin{pmatrix} X & \vdots \\ \vdots & Y \end{pmatrix}$

Keep fixed

ALS Algorithm contd.



- Update item matrix

$$\begin{matrix} & \text{Items} \\ \text{Users} & \left(\begin{array}{ccc} 10 & 5 & ? \\ ? & 2 & 10 \\ 7 & ? & 5 \end{array} \right) \end{matrix}$$

÷ ≈

Keep fixed

Calculate update

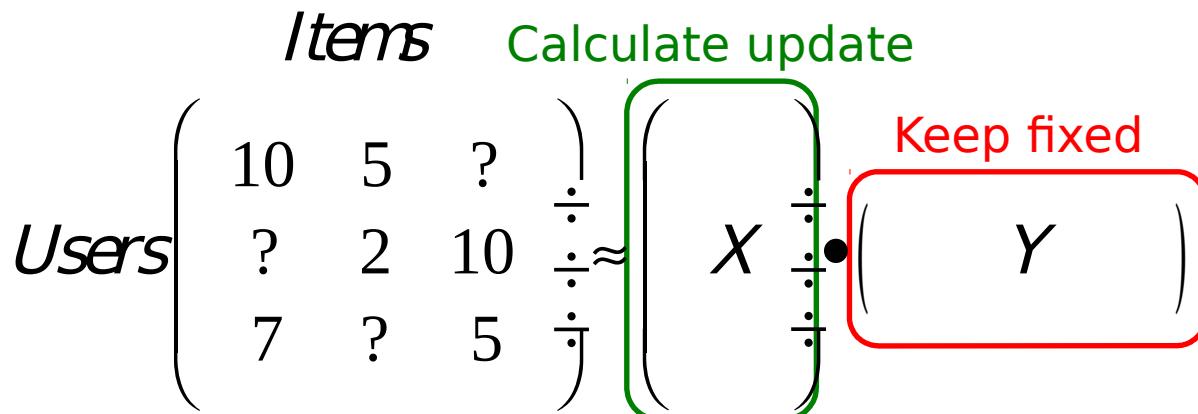
X \bullet Y

The diagram illustrates the Alternating Least Squares (ALS) algorithm for matrix factorization. It shows a user-item matrix with missing values represented by question marks. The columns are labeled 'Items' and the rows are labeled 'Users'. The matrix is divided into two main parts: a fixed part (the first column, labeled 'Keep fixed') and an updated part (the second column, labeled 'Calculate update'). The second column is further divided into a matrix X and a vector Y, with a multiplication symbol between them. An arrow points from the 'Keep fixed' box to the 'Calculate update' box, indicating the iterative nature of the algorithm where one column is fixed while others are updated.

ALS Algorithm contd.

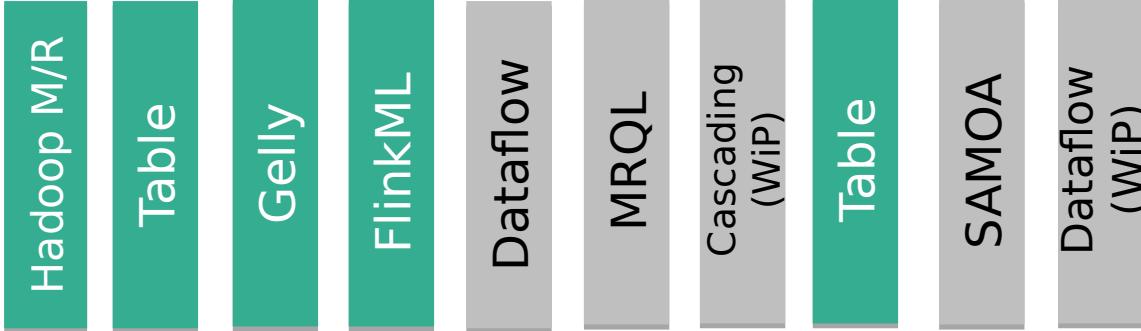


- Repeat update step until convergence

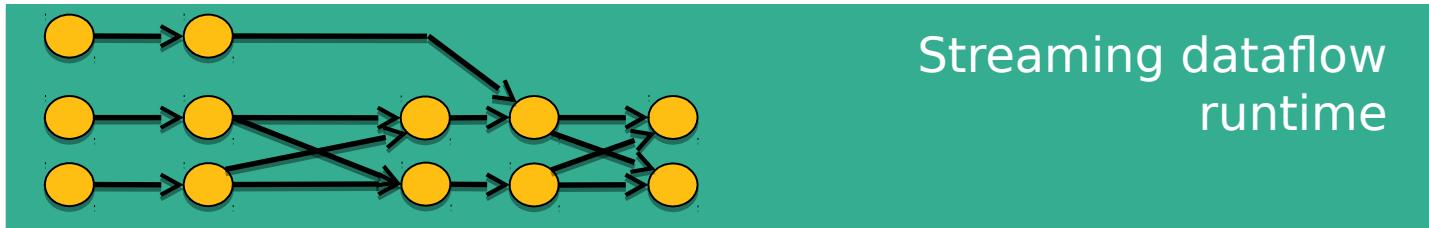


Apache Flink

What is Apache Flink?



Apache Flink deep-dive
by Stephan Ewen
Tomorrow, 12:20 - 13:00
on Stage 3



Why Using Flink for ALS?



- Expressive API
- Pipelined stream processor
- Closed loop iterations
- Operations on managed memory

Expressive APIs



- DataSet: Abstraction for distributed data
- Computation specified as sequence of lazily evaluated transformations

```
case class Word(word: String, frequency: Int)
```

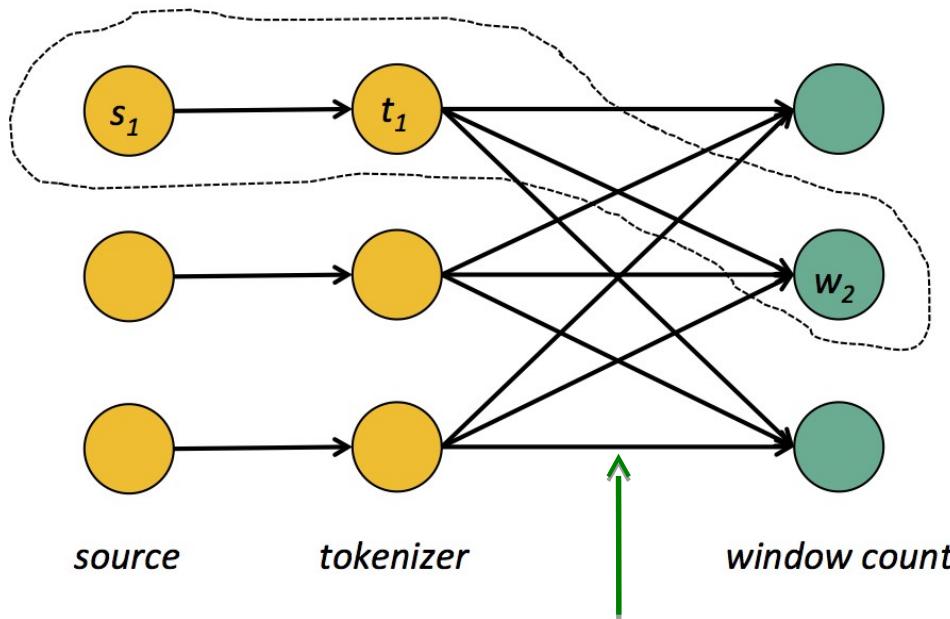
```
val lines: DataSet[String] = env.readTextFile(...)
```

```
lines.flatMap(line => line.split(" ")).map(word => Word(word, 1))  
    .groupBy("word").sum("frequency")  
    .print()
```

Pipelined Stream Processor

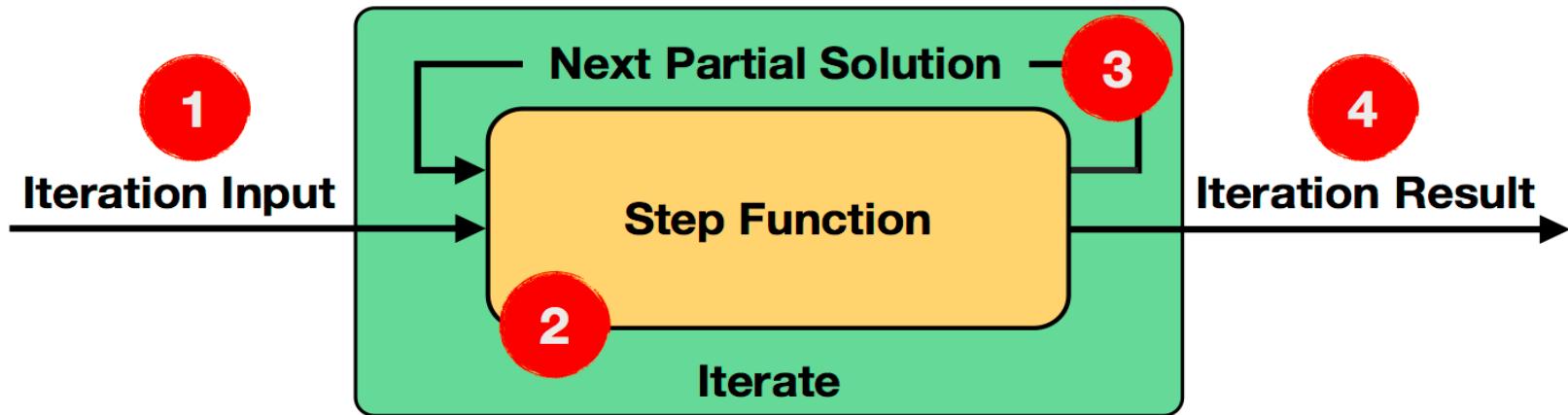


*Complete pipeline online
concurrently*



Avoiding materialization of
intermediate results

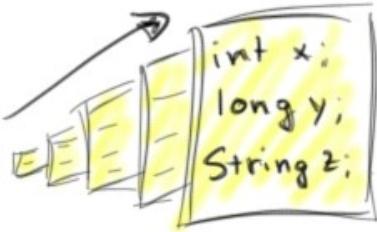
Iterate in the Dataflow



Memory Management



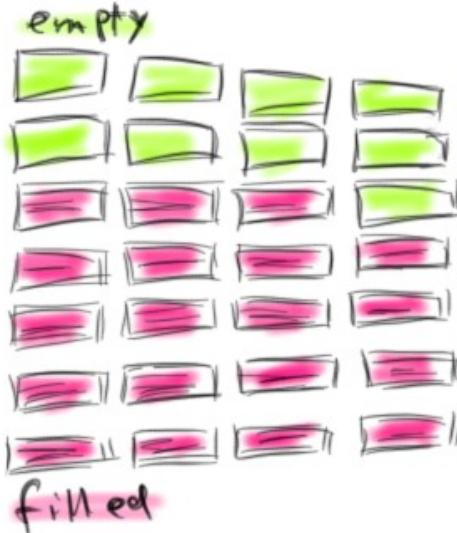
Custom
Data Objects



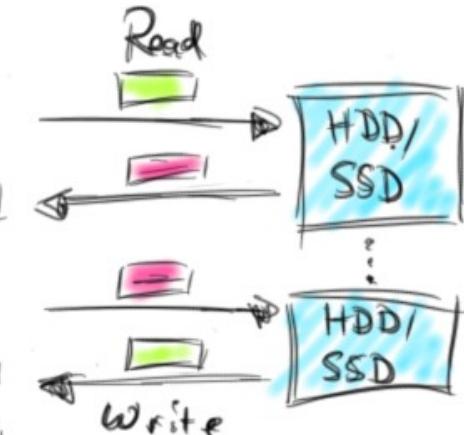
Efficient
De/Serialization



Managed Memory
32 KB memory segments



Destage to LocalFS
at Need

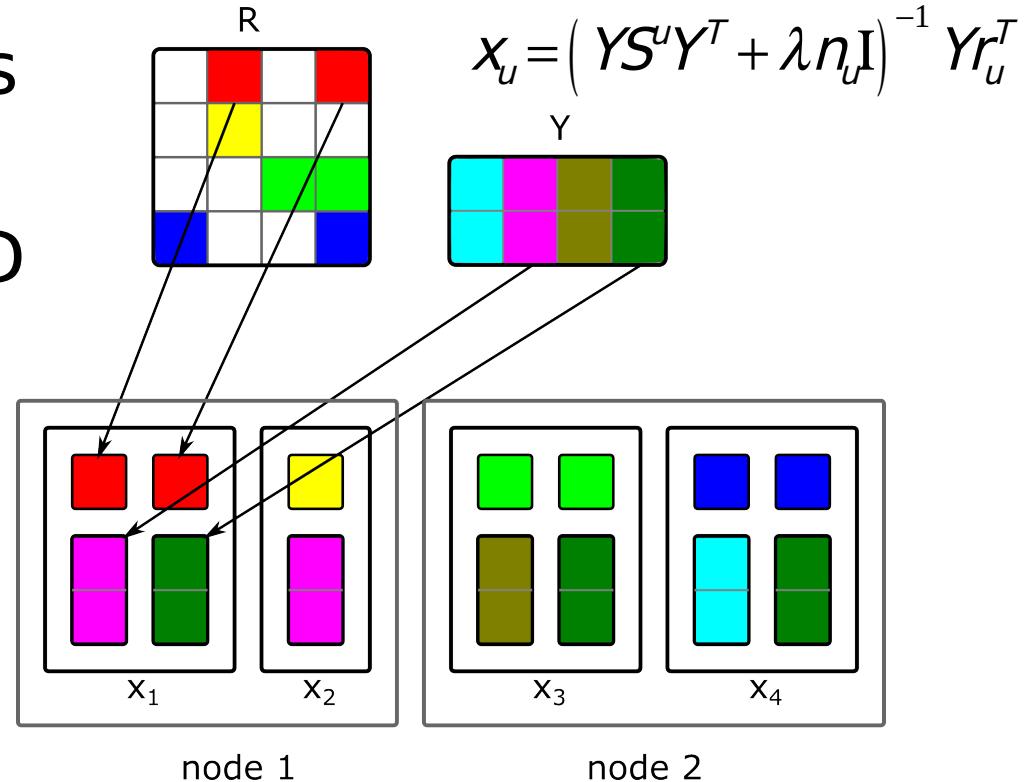


ALS implementations with Apache Flink

Naïve Implementation



1. Join item vectors with ratings
2. Group on user ID
3. Compute new user vectors



Pros and Cons of Naïve ALS

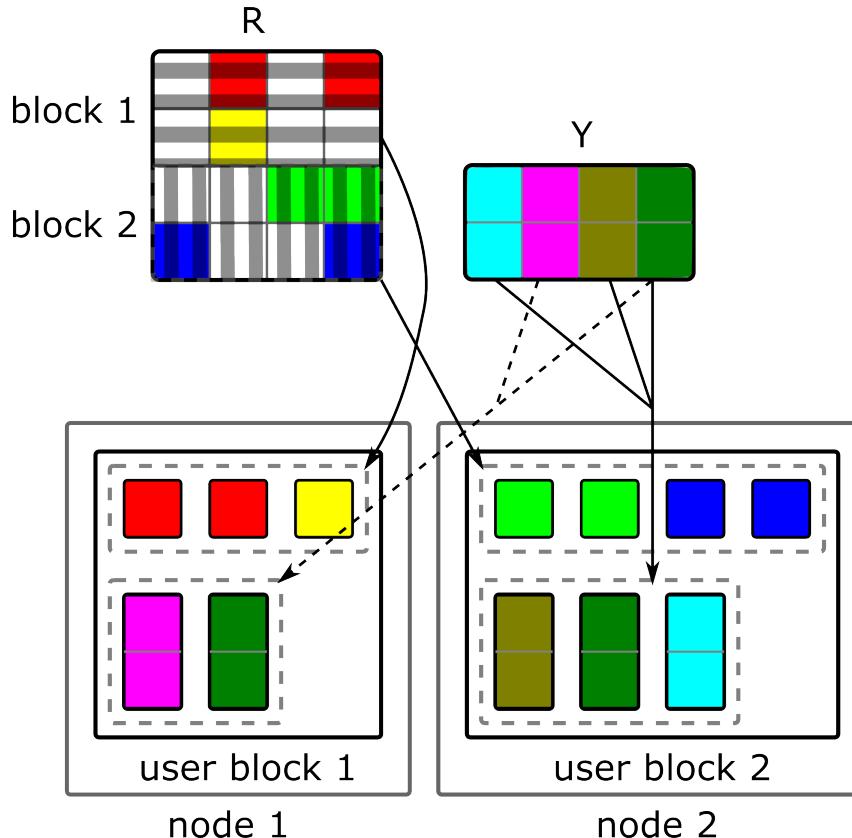


- Pros
 - Easy to implement
- Cons
 - Item vectors are sent redundantly to network nodes
 - Two shuffle steps make execution expensive

Blocked ALS Implementation



1. Create user and item rating blocks
2. Cache them on worker nodes
3. Send all item vectors needed by user rating block bundled
4. Compute block of item vectors



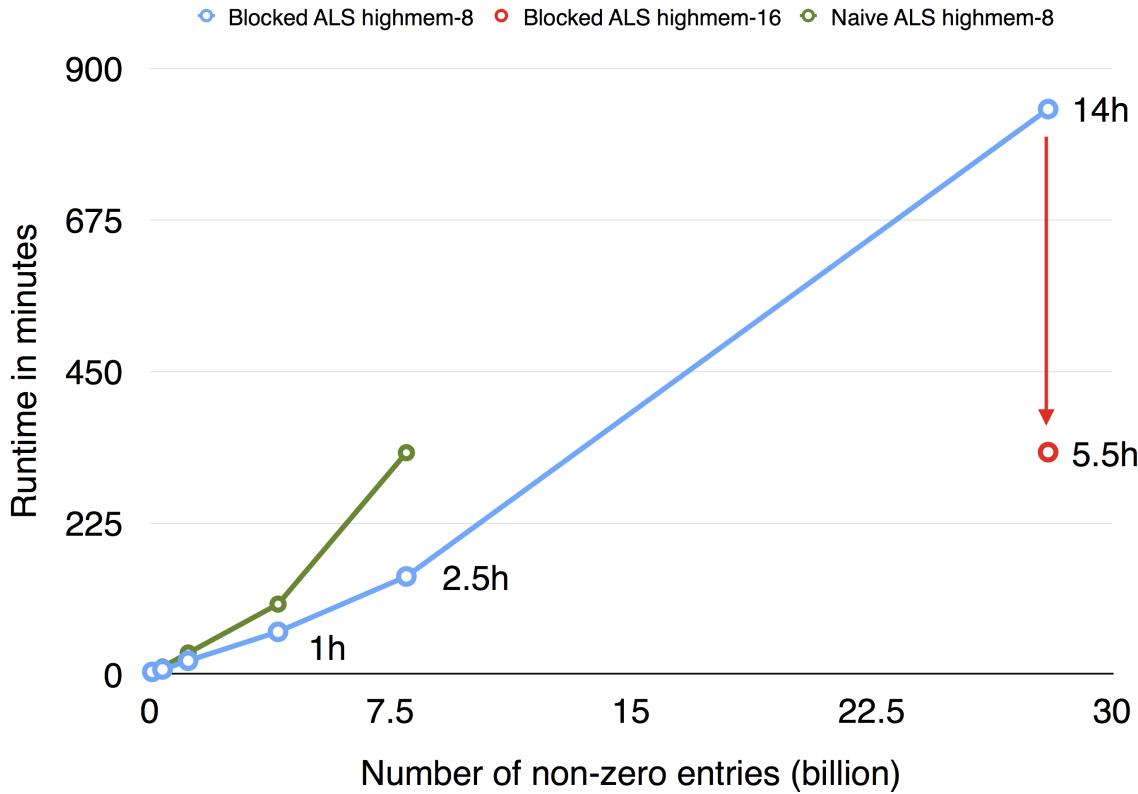
Based on Spark's MLLib implementation

Pros and Cons of Blocked ALS



- Pros
 - Reduces network load by avoiding data duplication
 - Caching ratings: Only one shuffle step needed
- Cons
 - Duplicates the rating matrix (user block/item block partitioning)

Performance Comparison



- 40 node GCE cluster, highmem-8
- 10 ALS iterations with 50 latent factors
- Rating matrix has 28 billion non zero entries: **Scale of Netflix or Spotify**

Machine Learning with FlinkML



- FlinkML contains blocked ALS
- Support for many other tasks
 - Clustering
 - Regression
 - Classification
- scikit-learn like pipeline support

```
val als = ALS()  
  
val ratingDS =  
    env.readCsvFile[(Int, Int, Double)](ratingData)  
  
val parameters = ParameterMap()  
    .add(ALS.Iterations, 10)  
    .add(ALS.NumFactors, 50)  
    .add(ALS.Lambda, 1.5)  
  
als.fit(ratingDS, parameters)  
  
val testingDS = env.readCsvFile[(Int, Int)](testingData)  
  
val predictions = als.predict(testingDS)
```

Closing

What Have You Seen?



- How to use collaborative filtering to make recommendations
- Apache Flink, a powerful parallel stream processing engine
- How to use Apache Flink and alternating least squares to factorize really large matrices



Flink Forward

BERLIN 12/13 OCT 2015



flink.apache.org
@ApacheFlink