

# LSML #3

Рекомендательные системы

# Рекомендательные системы

Вам также могут понравиться



Tefal GV7485

от 13 790 руб.

0 отзывов 6 предложений

Утюг

- с парогенератором
- мощность 2000 Вт
- мощный паровой удар
- мощность подачи пара до 120 г/мин



Цены



Philips GC 1026

от 1 448 руб.

0 отзывов 94 предложения

Утюг

- мощность 2000 Вт
- мощность подачи пара до 25 г/мин
- вес 0.9 кг
- паровой удар



Цены



Tefal FV2352E0

от 1 800 руб.

1 отзыв 9 предложений

Утюг

- мощность 2000 Вт
- мощность подачи пара до 30 г/мин
- вес 1.4 кг
- паровой удар



Цены



Sinbo SSI-2872

от 900 руб.

1 отзыв 94 предложения

Утюг

- мощность 2000 Вт
- керамическая подошва
- паровой удар
- вертикальное отпаривание



Цены

# User-based CF

- $n$  пользователей,  $m$  товаров
- $r_{ui}$  — оценка пользователя  $u$  товару  $i$
- $r_u$  — средняя оценка пользователя  $u$
- Похожесть пользователей  $u$  и  $v$ :

$$\text{corr}(u, v) = \frac{\sum_{i=1}^m (r_{ui} - r_u)(r_{vi} - r_v)}{\sqrt{\sum_{i=1}^m (r_{ui} - r_u)^2 \sum_{i=1}^m (r_{vi} - r_v)^2}}$$

- Предсказание:

$$\widehat{r_{ui}} = r_u + \frac{\sum_{v=1}^n \text{corr}(u, v)(r_{vi} - r_v)}{\sum_{v=1}^n |\text{corr}(u, v)|}$$

# Item-based CF

- Похожие формулы
- Более уверенные оценки похожести товаров
- Медленнее устаревают похожести товаров
- Как работает:
  - Посчитали заранее Item-Item похожести
  - Приходит пользователь, делает новый клик
  - По его кликам можно найти похожие товары в real-time

# Параллелим Item-based CF












- $n$  пользователей,  $m$  товаров,  $n \sim m \sim 10^6$
- Нужно посчитать заранее Item-Item похожести:

$$corr(i, j) = \frac{\sum_{u=1}^n (r_{ui} - r_u)(r_{uj} - r_u)}{\sqrt{\sum_{u=1}^n (r_{ui} - r_u)^2 \sum_{u=1}^n (r_{uj} - r_u)^2}}$$

adjusted cosine similarity

- Наивный подход –  $O(m^2n)$
- Матрица оценок сильно разрежена, можно лучше?
- Вклад в похожест двух товаров ненулевой только от пользователей, которые оценили оба этих товара

# Инвертированный индекс

						
	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

$$(1, 3) \rightarrow 5 * 4$$

$$(1, 6) \rightarrow 5 * 1$$

$$(3, 6) \rightarrow 4 * 1$$

# Как посчитать все числители на Spark

```
def emit_pairs(x):
    user, items = x
    items = sorted(items)
    if len(items) < 300:
        for i in range(len(items) - 1):
            for j in range(i + 1, len(items)):
                yield (
                    (items[i][0], items[j][0]),
                    items[i][1] * items[j][1]
                )

dot_product = (
    ratings
    .map(lambda x: (x.user, (x.product, x.rating)))
    .groupByKey()
    .flatMap(emit_pairs)
    .reduceByKey(lambda x, y: x + y)
)
```

$$corr(i, j) = \frac{\sum_{u=1}^n (r_{ui} - r_u)(r_{uj} - r_u)}{\sqrt{\dots}}$$

# Решение на Spark SQL

```
SELECT
    ic1.itemId,
    ic2.itemId AS jointItemId,
    SUM(ic1.val * ic2.val) AS cosine
FROM ic AS ic1
    JOIN ic AS ic2 ON ic1.clientId = ic2.clientId
WHERE ic1.itemId < ic2.itemId
GROUP BY ic1.itemId, ic2.itemId;
```



# Content-based рекомендации

- Используем свойства товаров и пользователей
  - Возраст, пол, ...
  - Жанр, режиссер, ...
- Можем рекомендовать товары, не имея статистики

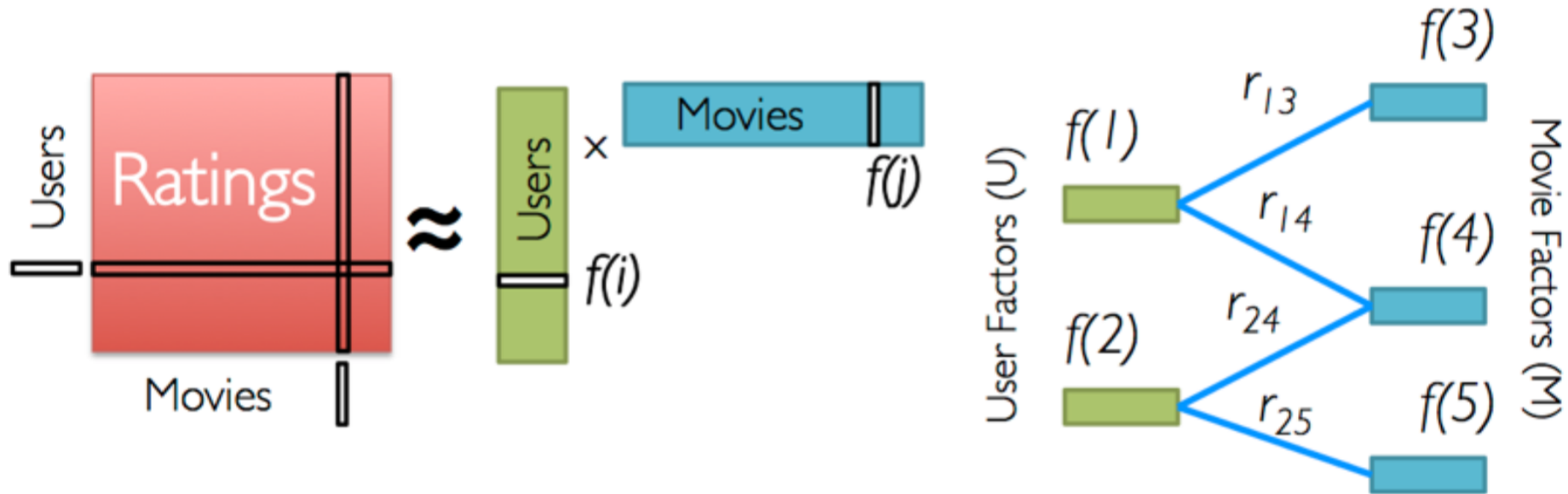
# Content-based: похожесть текстов

<b>TFIDF Normed Vectors</b>	a	Accelerating	and	applications	art	behavior	Building	Consumer	CRM	customer	data	for	Handbook	Introduction	Knowledge	Management	Marketing	Mastering	mining	of	relationship	Research	science	technology
Building data mining applications for CRM				0.502			0.502		0.344		0.251	0.502							0.251					
Accelerating customer relationships: using CRM and relationship technologies		0.432	0.296						0.296	0.216											0.468			0.432
Mastering Data Mining: the art and science of Customer Relationship Management			0.256		0.374					0.187	0.187					0.256		0.374	0.187	0.374	0.256		0.374	
Data Mining your website											0.316								0.316					
Introduction to Marketing														0.636			0.436							

# Content-based: похожесть текстов

- Похожесть как косинус между векторами TF-IDF
- Используем инвертированный индекс

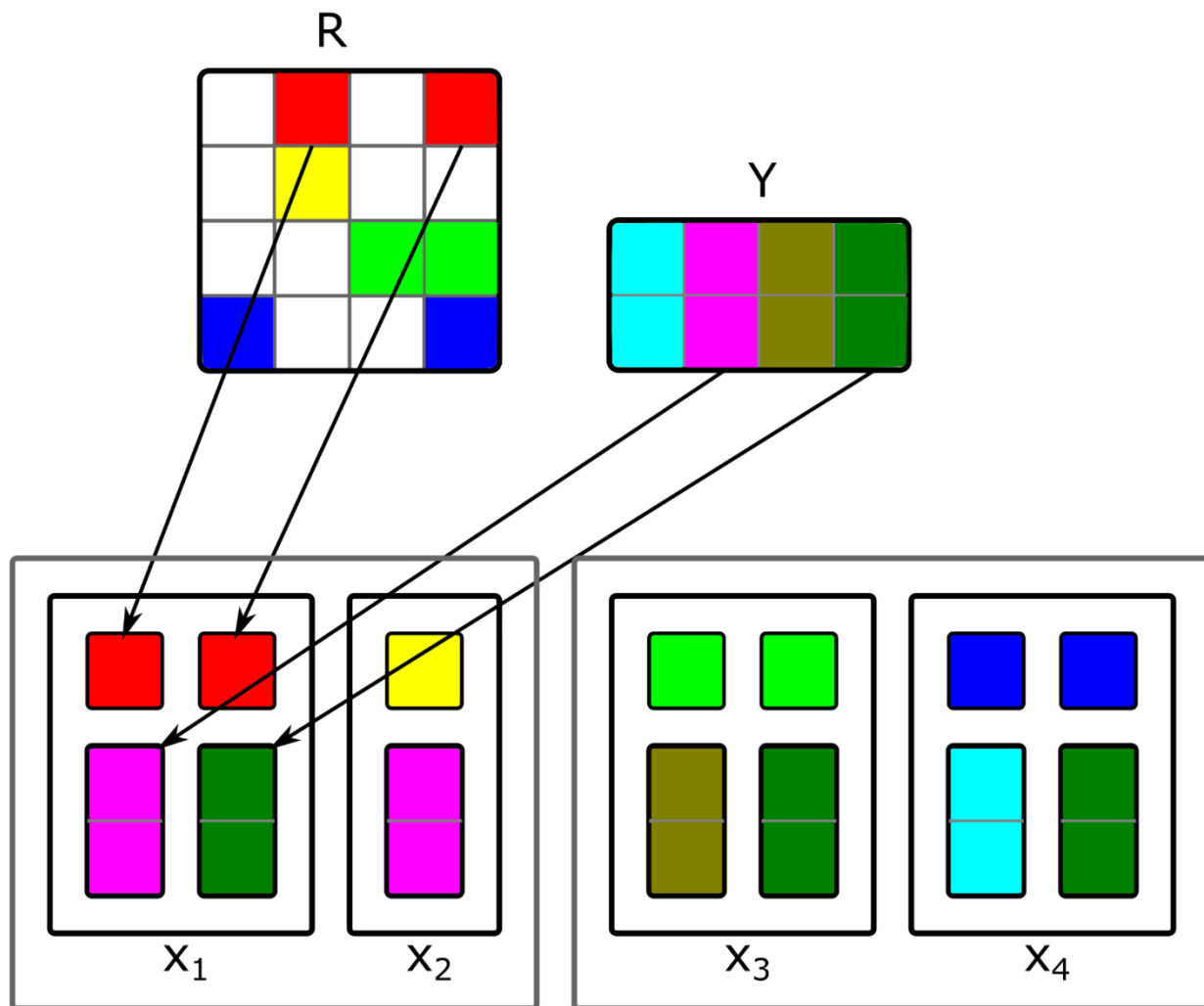
# ALS: матричные факторизации



Iterate:

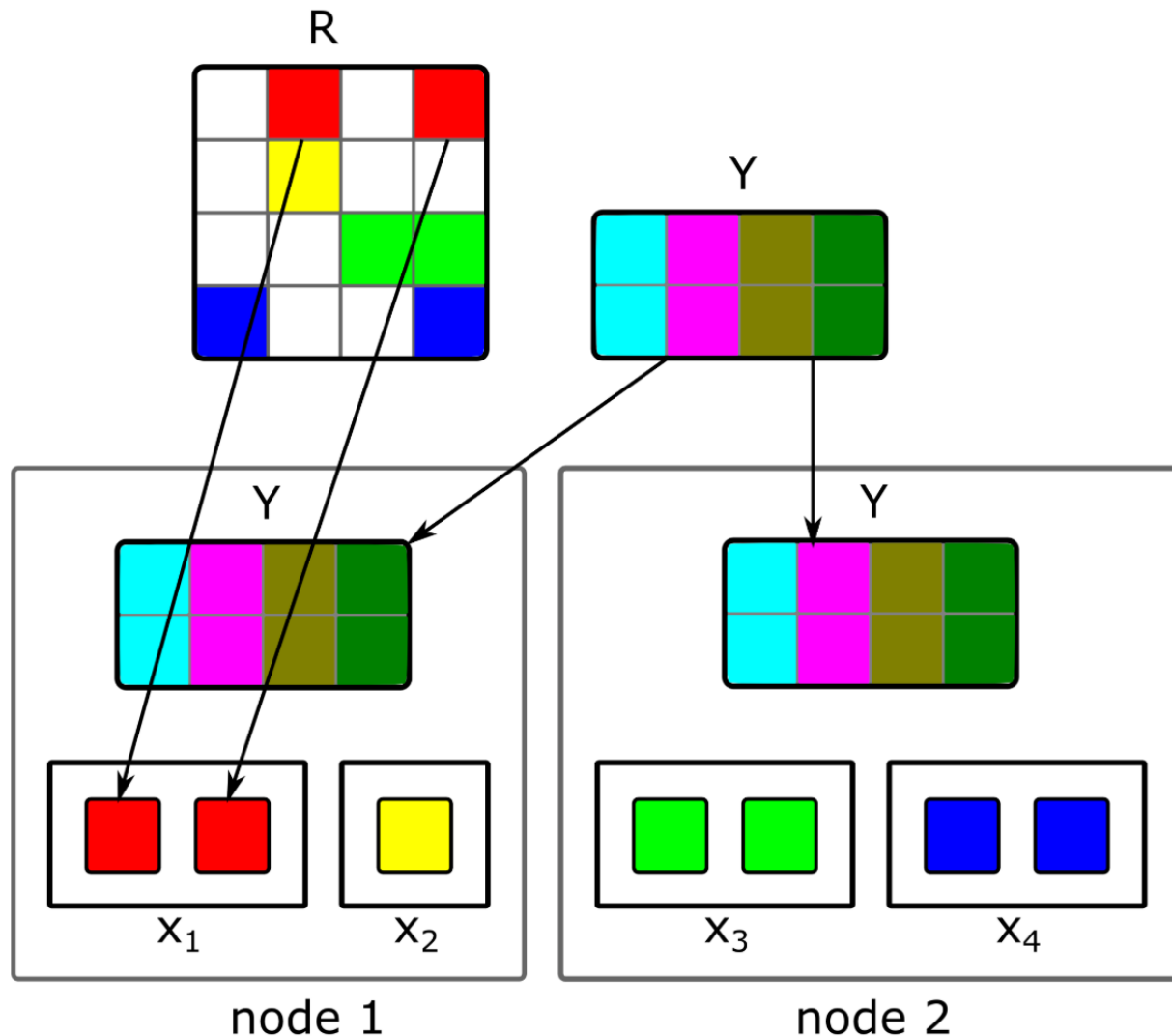
$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

# Наивный параллельный ALS

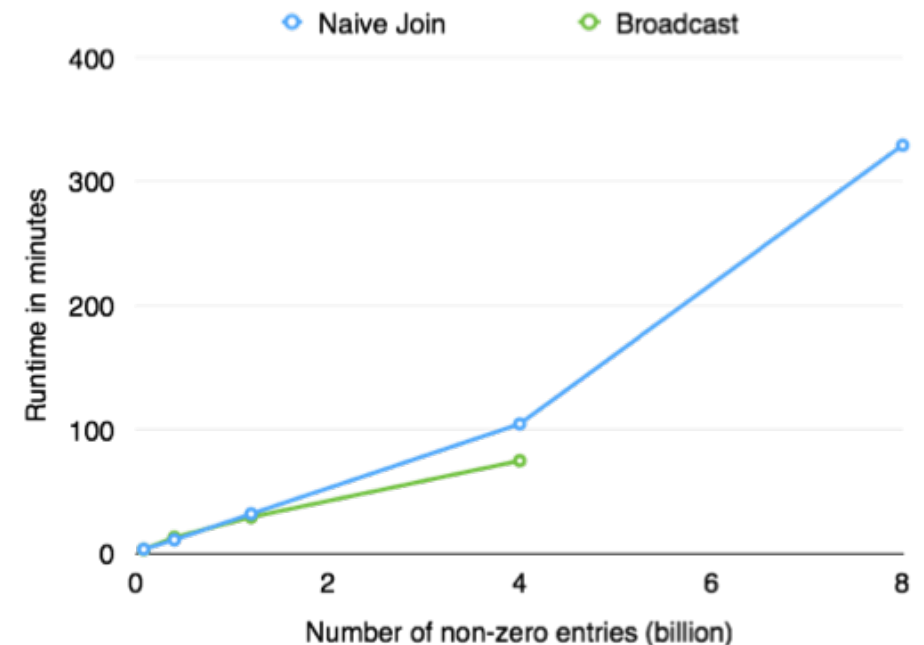


- $R = XY$ , обновляем  $X$
- Join  $R$  и  $Y$  по товарам  $\rightarrow (u, r_{ui}, y_i)$
- Группируем по  $u$
- На каждой машине пересчитываем  $x_u$
- Один и тот же вектор  $y_i$  может понадобится нескольким юзерам на машине, и будет отправлен несколько раз!

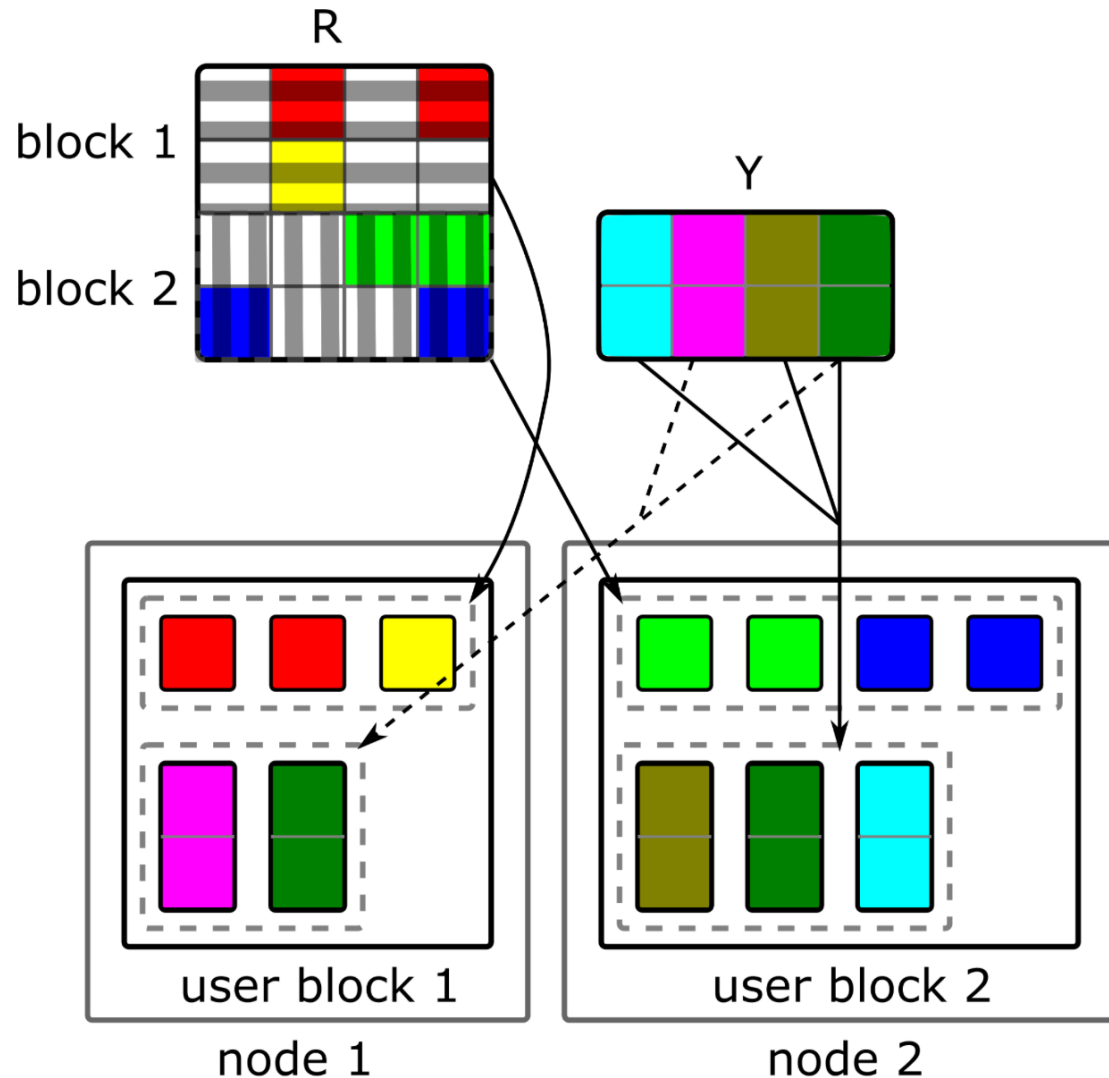
# Может быть спасет broadcast Y?



- Не сильно быстрее
- Имеет предел масштабирования (Y должен влезать в память)

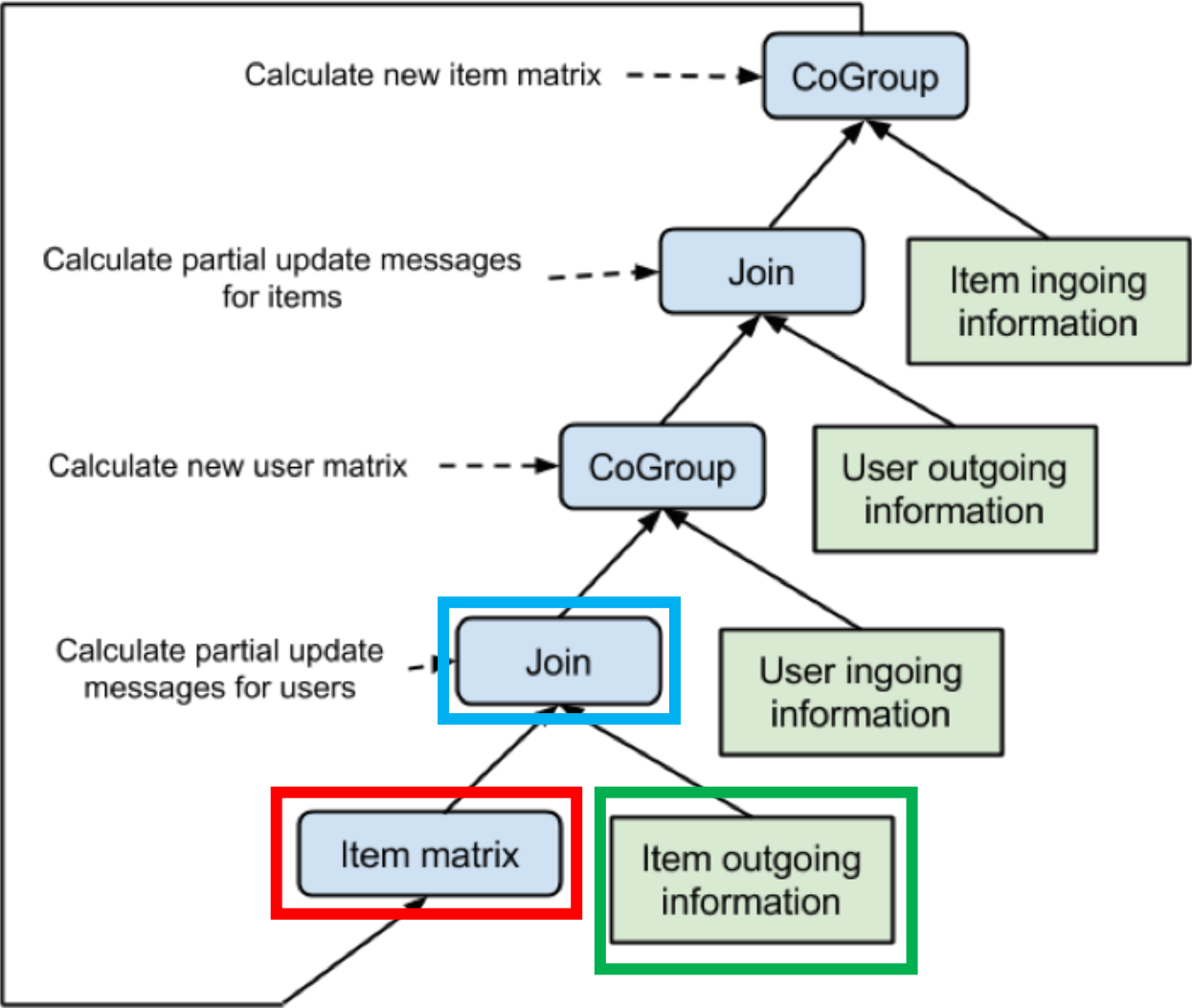
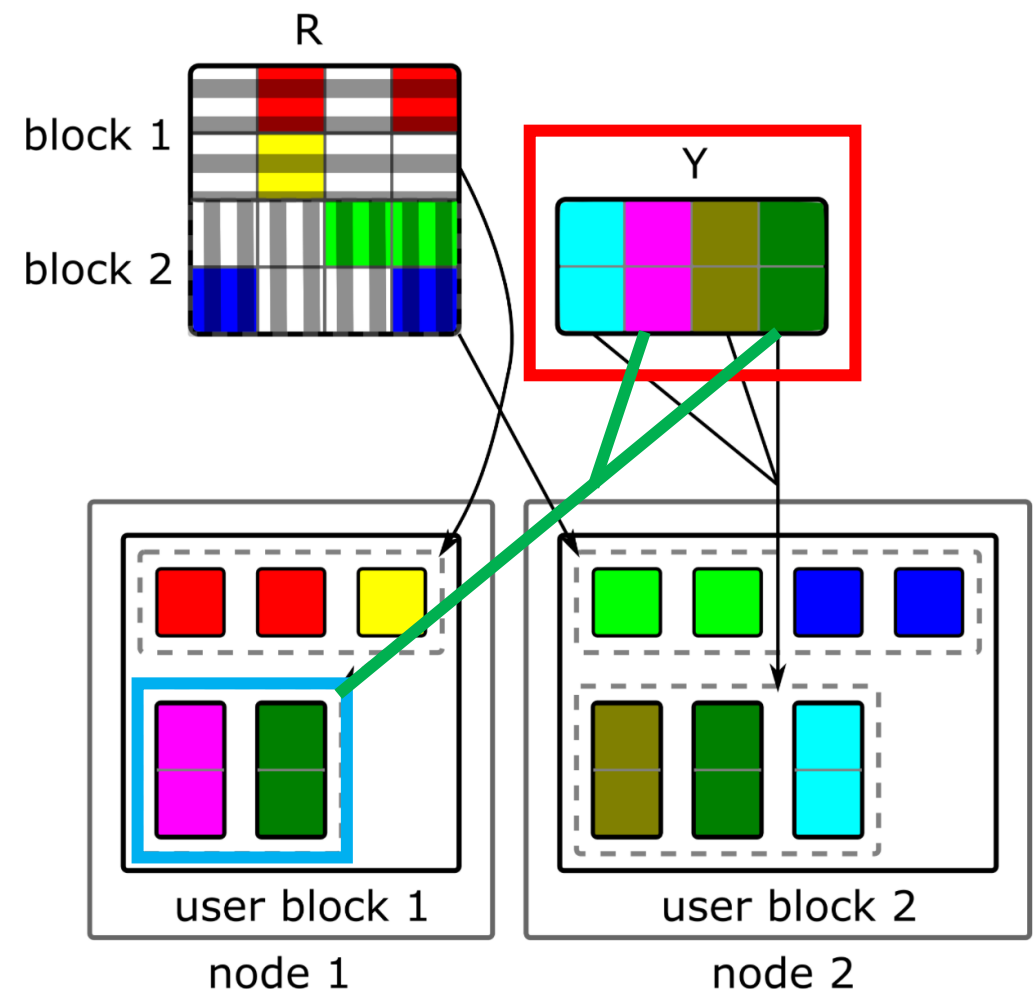


# Блочный ALS



- Пользователи и товары бьются на группы, каждая группа обрабатывается на своей машине
- Перед началом итераций вычисляем на какие машины отправлять новые  $y_i$  и  $x_u$  (хранится в памяти, линейно относительно измерений)
- Пересылаются только нужные части матриц

# Блочный ALS





# ALS B MLlib

```
from pyspark.mllib.recommendation import ALS, Rating
sc.setCheckpointDir("/checkpoints/")
model = ALS.train(train_scores_for_als, 100, 30, seed=42)
```

```
import math
RMSE = math.sqrt(
    model
    .predictAll(test_scores_for_als
                .map(lambda x: (x.user, x.product)))
    .map(lambda x: ((x.user, x.product), x.rating))
    .join(test_scores_for_als
          .map(lambda x: ((x.user, x.product), x.rating)))
    .map(lambda x: (x[1][0] - x[1][1]) ** 2)
    .mean()
)
print RMSE
```

# ДЗ 1: Outbrain Click Prediction

- <https://www.kaggle.com/c/outbrain-click-prediction>
- Персональные рекомендации в Интернете
- **100 ГБ** несжатых данных



# ДЗ 1: Кластер в Azure

- Как создать:

[https://github.com/ZEMUSHKA/lsml\\_hse/blob/master/azure/docs/CREATE\\_CLUSTER.md](https://github.com/ZEMUSHKA/lsml_hse/blob/master/azure/docs/CREATE_CLUSTER.md)

- 3 машины в кластере, на каждой:
  - 4 ядра
  - 28 ГБ оперативки
  - 511 ГБ диска
- Денег хватит на **две недели** непрерывного счета

# Ссылки

- <https://stanford.edu/~rezab/sparkworkshop/slides/xiangrui.pdf>
- ДЗ 1 Baseline (можно сделать такие же запросы в Spark SQL):  
<https://www.kaggle.com/tkm2261/outbrain-click-prediction/bigquery-is-cool-lb-0-63692>
- <http://cs229.stanford.edu/proj2016/report/JaiswalGopinathLimaye-OutbrainClickPrediction-report.pdf>
- <https://data-artisans.com/how-to-factorize-a-700-gb-matrix-with-apache-flink/>