



Buffalo: Open Source Project for Recommender System

김광섭 lucas.kim

카카오 추천팀

01 안녕하세요 버팔로

- └ 특징
- └ 추천시스템과 버팔로
- └ 버팔로 개발 배경

02 버팔로 들여다보기

- └ 알고리즘
- └ 연구/개발 편의 기능
- └ 높은 생산성과 성능

03 마무리

01 안녕하세요 버팔로

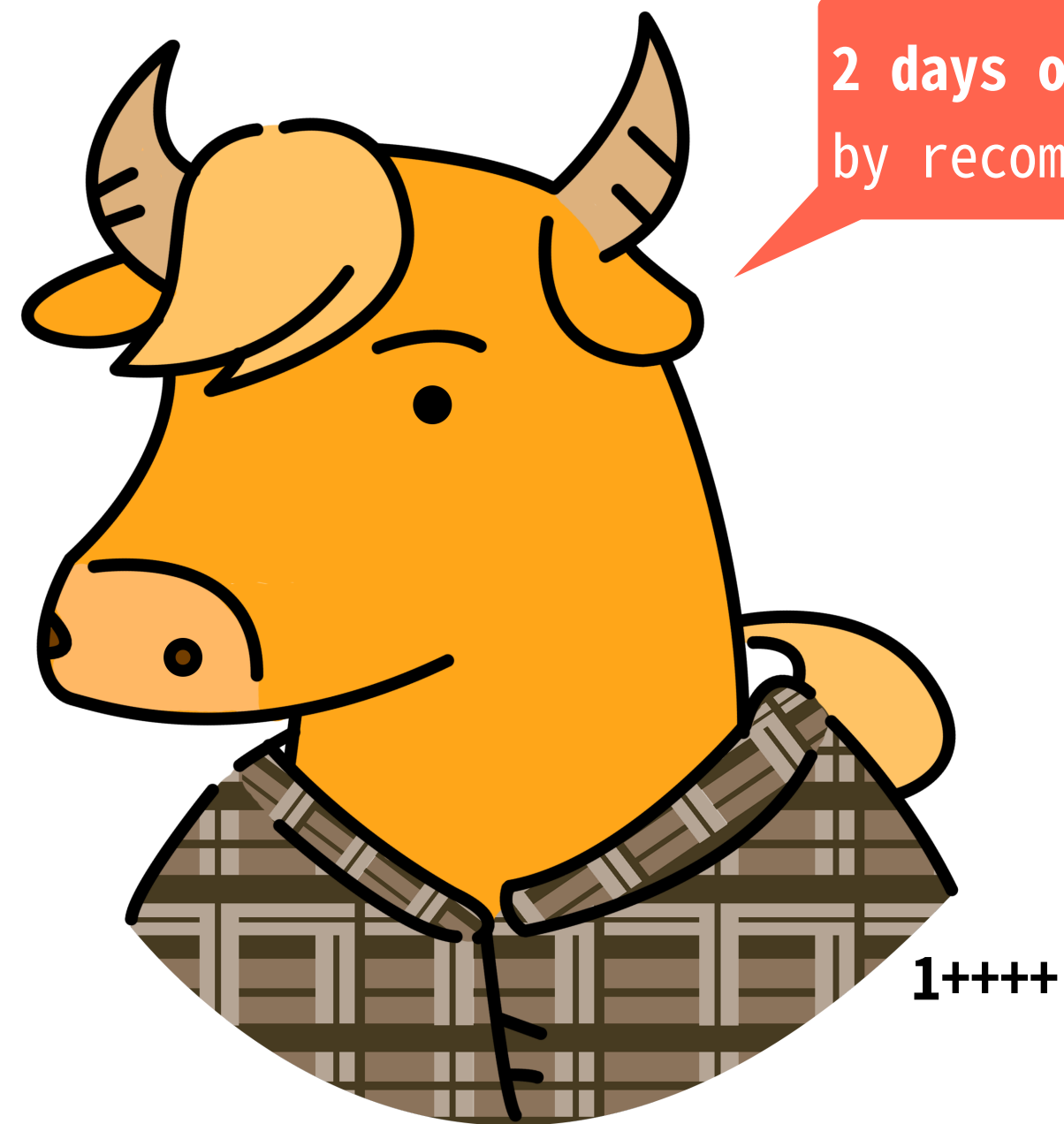


if (kakao) dev 2019

- └ 추천시스템과 버팔로
- └ 버팔로 개발 배경

안녕하세요 버팔로

if (kakao) dev 2019



<https://github.com/kakao/buffalo>

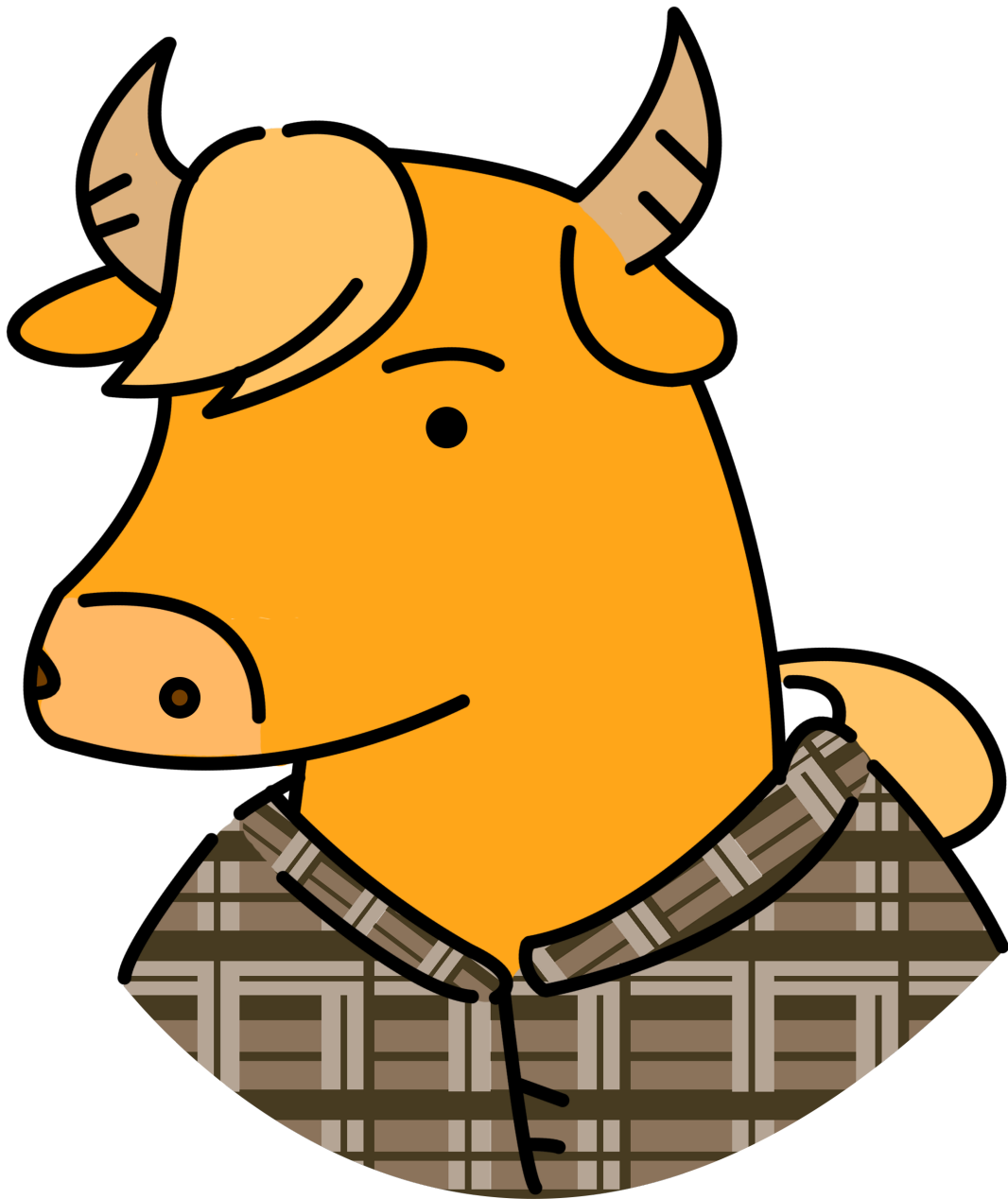
artist: {anes, hawl}.kim@kakaocorp.com

안녕하세요 버팔로

if (kakao) dev 2019

버팔로의 특징

지원 알고리즘	<ul style="list-style-type: none">- Matrix Factorizations
뛰어난 생산성과 성능	<ul style="list-style-type: none">- Memory semi-free for the database- Highly optimized Python/C++ codebase for CPU(Up to 8.5 times faster than SOTA libraries)- Parallel Processing for recommendation tasks- GPU support
편의 기능	<ul style="list-style-type: none">- Evaluable: Auto validation, Built-in Metrics, ...- Hyper-parameter Optimization- Extendable Architecture

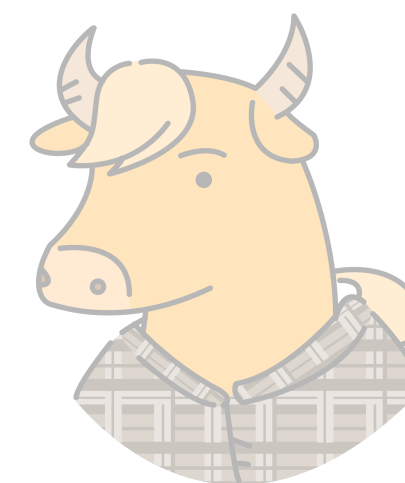


<https://github.com/kakao/buffalo>
artist: {anes, hawl}.kim@kakaocorp.com

안녕하세요 버팔로

if (kakao) dev 2019


버팔로의 속살은 잠시 후에...




추천 시스템과 버팔로


4.2 추천 시스템 아키텍처

오픈소스와 네이버 솔루션의 오케스트레이션





추천 시스템





추천 시스템은 정보 필터링 기술의 일종으로, 특정 사용자가 관심을 가질만한 정보를 추천하는 것이다. 추천 시스템에는 협업 필터링 기법을 주로 사용한다. 소셜 북마크 사이트에서 링크를 사람들에게 추천하고 무비렌즈 데이터 세트에서 영화를 추천하는 방법등이 이에 속한다. [위키백과](#)

추천 시스템과 버팔로


4.2 추천 시스템 아키텍처

오픈소스와 네이비 솔루션의 오케스트레이션





추천 시스템



추천 시스템은 정보 필터링 기술의 일종으로, 특정 사용자가 관심을 가질만한 정보를 추천하는 것이다. 추천 시스템에는 협업 필터링 기법을 주로 사용한다. 소셜 북마크 사이트에서 링크를 사람들에게 추천하고 무비렌즈 데이터 세트에서 영화를 추천하는 방법등이 이에 속한다. [위키백과](#)

추천 시스템에 대해 말할 때, 많은 경우 여기에 대해서 얘기합니다.

추천 시스템과 버팔로


4.2 추천 시스템 아키텍처

오픈소스와 네이비 솔루션의 오케스트레이션





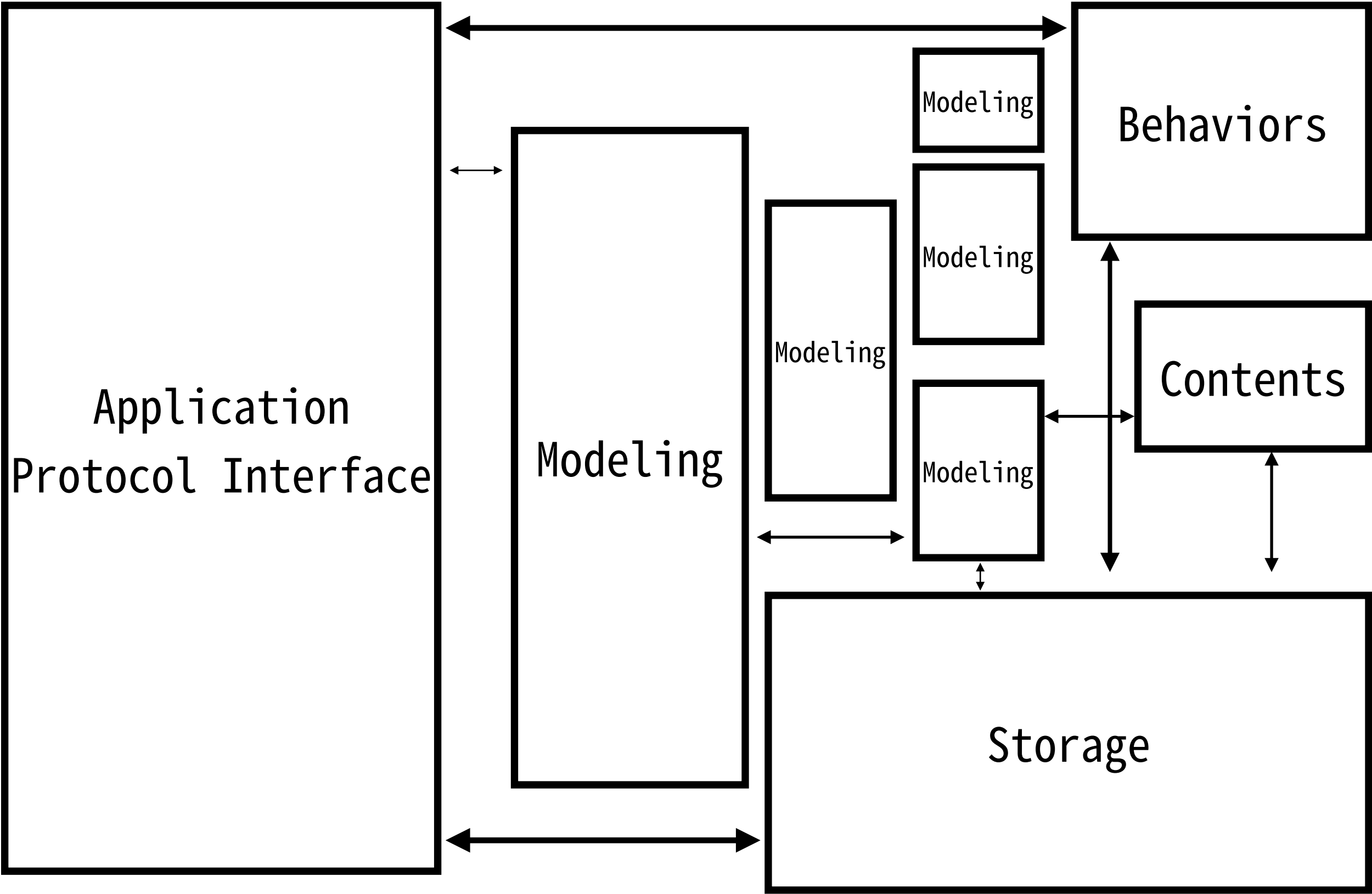
추천 시스템



추천 시스템은 정보 필터링 기술의 일종으로, 특정 사용자가 관심을 가질만한 정보를 추천하는 것이다. 추천 시스템에는 협업 필터링 기법을 주로 사용한다. 소셜 북마크 사이트에서 링크를 사람들에게 추천하고 무비렌즈 데이터 세트에서 영화를 추천하는 방법등이 이에 속한다. [위키백과](#)

추천 시스템은 협업 필터링이나 무비 추천에 초점이 맞춰진 것이 아니라, **사용자에게 콘텐츠 혹은 정보를 소비하는 경험을 제공하는 기술**입니다. 데이터 분석, 모델링에 이어 최종적인 정보 전달을 하기 위해 폭넓은 분야를 다룹니다.

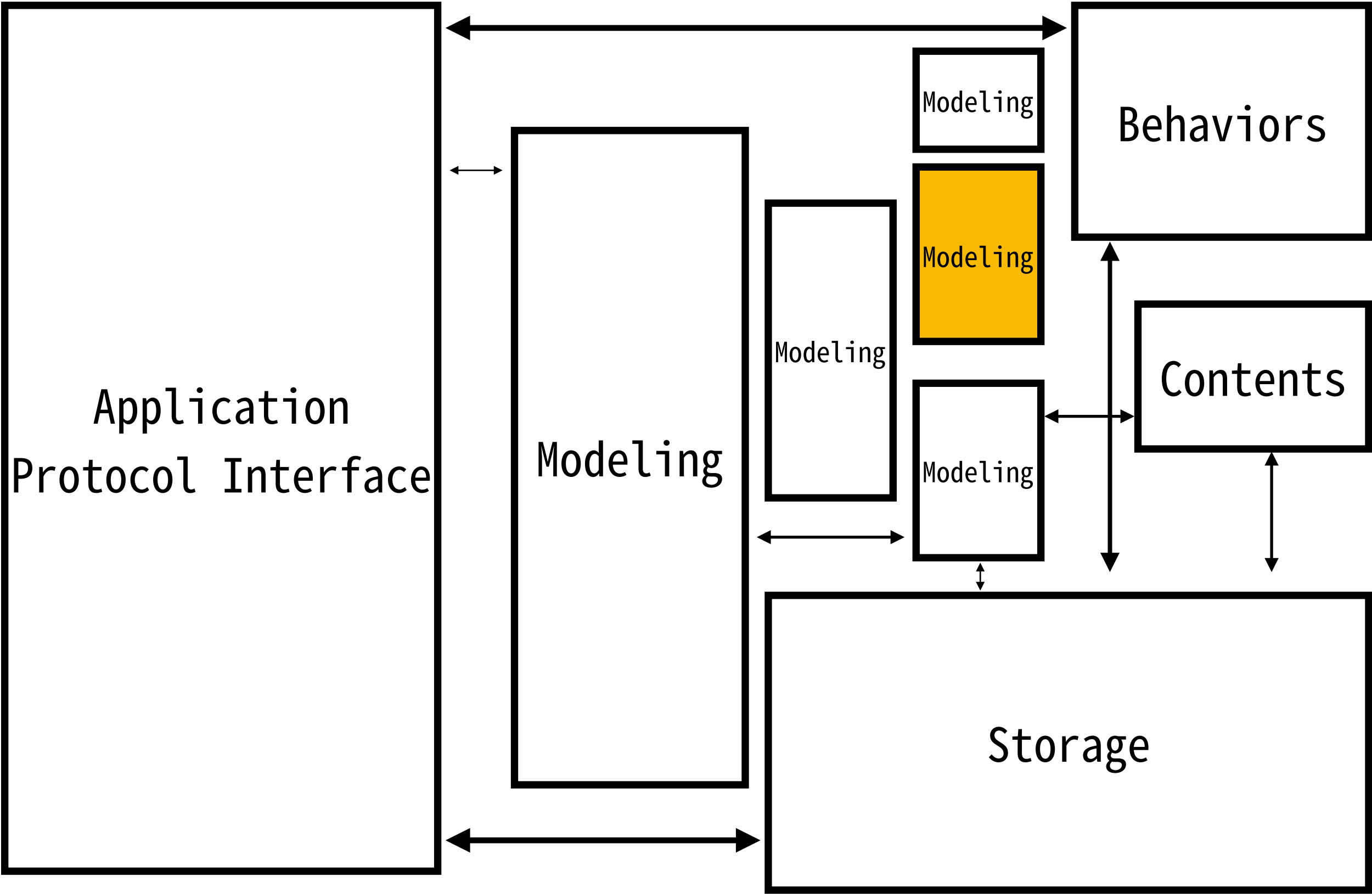
추천 시스템과 버팔로



카카오 토로스 추천 시스템의 개략적인 구조
입니다. 다양한 데이터와 모듈로 엮이기 때문
에 복잡성이 높습니다.

TOROS: System Architecture of Recommender System

추천 시스템과 버팔로



오늘 소개할 버팔로는 토로스 추천 시스템의 일부로서 추천 시스템에 필요한 모델링 기능을 제공하고 있습니다.

TOROS: System Architecture of Recommender System

안녕하세요 버팔로

if (kakao) dev 2019

버팔로는 추천 시스템을 구성하는 하나의 모델링 모듈

안녕하세요 버팔로

if (kakao) dev 2019

“그런데 또 다른 오픈 소스가 필요한가요?”

안녕하세요 버팔로

if (kakao) dev 2019

버팔로 개발 배경

추천 시스템을 개발 하던 2010년 초반에는 지금보다도 관련된 오픈 소스 프로젝트 적었습니다.

특히, 우리가 원하는 크기의 데이터를 적절한 시간에 효과적인 자원으로 분석할 수 있는 프로젝트가 없었습니다.

그렇게 시작해 지금까지 추천 시스템의 대부분 기술 스택을 자체 개발하게 되었습니다.

안녕하세요 버팔로

if (kakao) dev 2019

버팔로 개발 배경

초기엔 범용적인 프레임워크를 염두에 뒀던 것은 아니지만 개발 기간이 오래되고
리팩토링이 반복되면서 자연스럽게 세부 라이브러리들로 나뉘었고
그중 Matrix Factorization 모듈을 엮은 라이브러리가 **Buffalo**의 시작이 되었습니다.

안녕하세요 버팔로

if (kakao) dev 2019

버팔로 개발 배경

첫 개발로부터 많은 시간이 지났지만
지금도 오픈 소스로서 경쟁력이 있다고 판단해 공개하기로 했습니다.

추천 시스템을 연구하는 분들과 기업에서 활용하고자 하는 분들에게

- 쓰기 편하고
- 실용적이며
- 스케일러블하고
- 만족스러운 품질의 결과를

줄 것으로 기대합니다.

안녕하세요 버팔로

if (kakao) dev 2019

버팔로 개발 배경

첫 개발로부터 많은 시간이 지났지만
지금도 오픈 소스로서 경쟁력이 있다고 판단해 공개하기로 했습니다.

추천 시스템을 연구하는 분들과 기업에서 활용하고자 하는 분들에게

- 쓰기 편하고
- 실용적이며
- 스케일러블하고
- 만족스러운 품질의 결과를

줄 것으로 기대합니다.



<https://github.com/kakao/buffalo>

artist: {anes, hawl}.kim@kakaocorp.com

02 버팔로 들여다보기

if (kakao) dev 2019

- └ 알고리즘
- └ 연구/개발 편의 기능
- └ 높은 생산성과 성능

버팔로 들여다보기

버팔로의 특징

지원 알고리즘	<ul style="list-style-type: none">- Matrix Factorizations
높은 생산성과 성능	<ul style="list-style-type: none">- Memory semi-free for the database- Highly optimized Python/C++ codebase for CPU(Up to 8.5 times faster than SOTA libraries)- Parallel Processing for recommendation tasks- GPU support
편의 기능	<ul style="list-style-type: none">- Evaluable: Auto validation, Built-in Metrics, ...- Hyper-parameter optimization- Extendable architecture



<https://github.com/kakao/buffalo>
artist: {anes, hawl}.kim@kakaocorp.com

버팔로 들여다보기

if (kakao) dev 2019

알고리즘

버팔로 들여다보기

if (kakao) dev 2019

- [1] Alternating Least Square, 2008
- [2] Bayesian Personalized Ranking Matrix Factorization, 2009
- [3] Word2Vec, 2013
- [4] CoFactors, 2016

버팔로 들여다보기

if (kakao) dev 2019

- [1] Alternating Least Square, 2008
- [2] Bayesian Personalized Ranking Matrix Factorization, 2009
- [3] Word2Vec, 2013
- [4] CoFactors, 2016

Matrix Factorizations

버팔로 들여다보기

if (kakao) dev 2019

- [1] Alternating Least Square, 2008
- [2] Bayesian Personalized Ranking Matrix Factorization, 2009
- [3] Word2Vec, 2013
- [4] CoFactors, 2016

Matrix Factorizations?

Matrix Factorization은 행렬로 표현된 데이터를 더 작은 차원의 행렬로 분해하는 방법입니다.

장점

- 데이터의 압축
- 숨겨진 특성의 활용

버팔로 들여다보기

if (kakao) dev 2019

R = 멜론 사용자의 청취 이력 (u=사용자, i=음악)

Diagram illustrating a matrix R with dimensions $R = \text{수천만} \times \text{수천만}$.

The matrix is divided into four quadrants:

- Top-left quadrant: Rows u_1, u_2 and columns i_1, i_2 . The value 2 is shown in the cell (u_2, i_2) .
- Top-right quadrant: Columns i_1, i_n . The value 232 is shown in the cell (u_2, i_n) .
- Bottom-left quadrant: Rows u_1, u_n and columns i_1, i_2 . The value 2 is shown in the cell (u_n, i_2) .
- Bottom-right quadrant: Columns i_1, i_n . The value 1 is shown in the cell (u_n, i_n) .

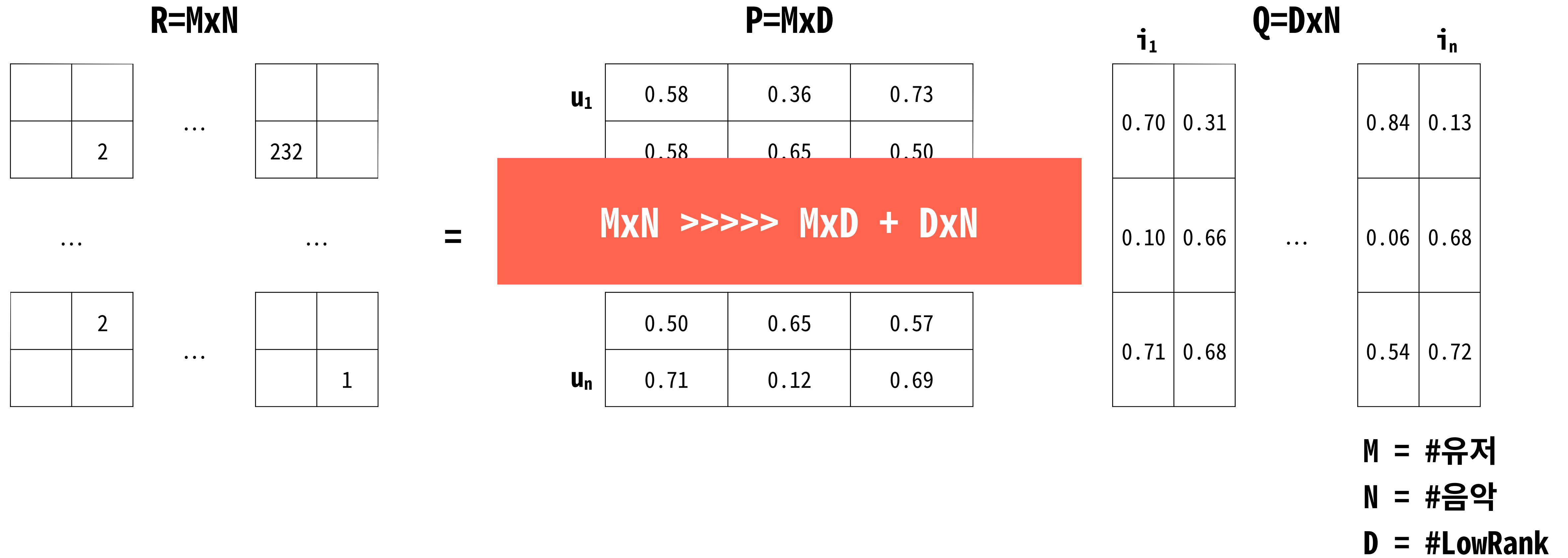
Ellipses (...) indicate continuation of rows and columns.

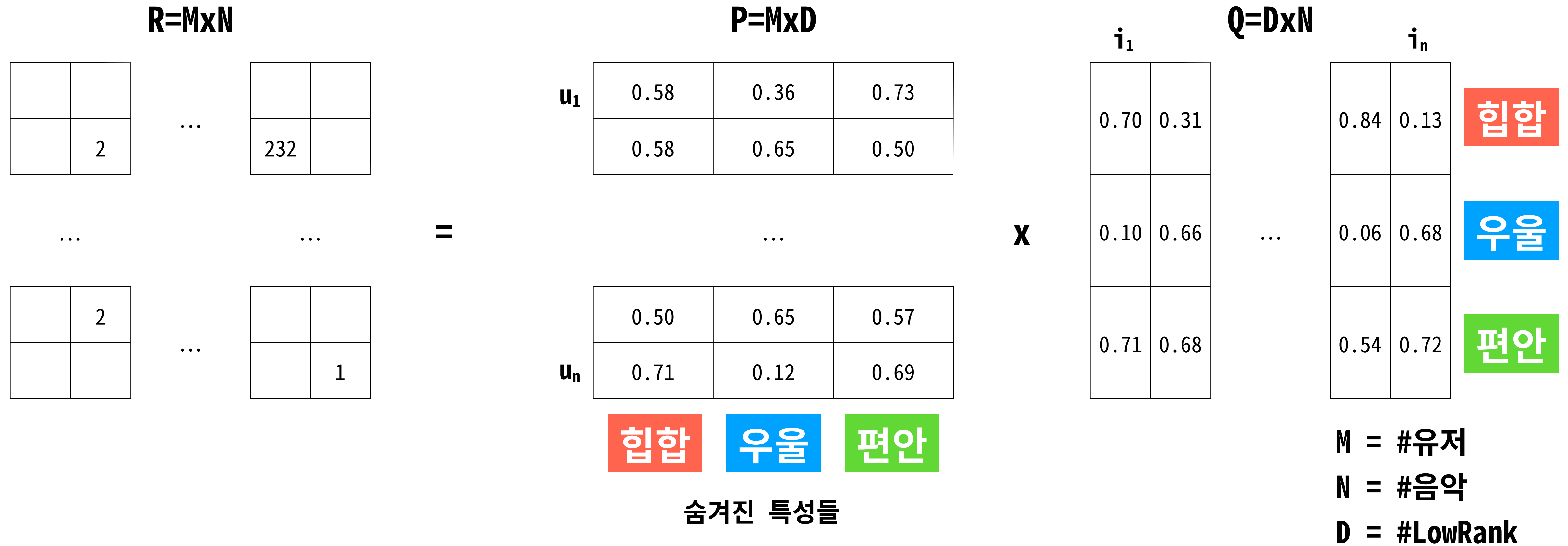
버팔로 들여다보기

R = 멜론 사용자의 청취 이력 (u=사용자, i=음악)

	i_1	i_2			i_n
u_1					
u_2		2	...		232
<div>너무 크다</div>					
	...	R U L L R U L L			...
u_n		2			
			...		1







if (kakao) dev 2019



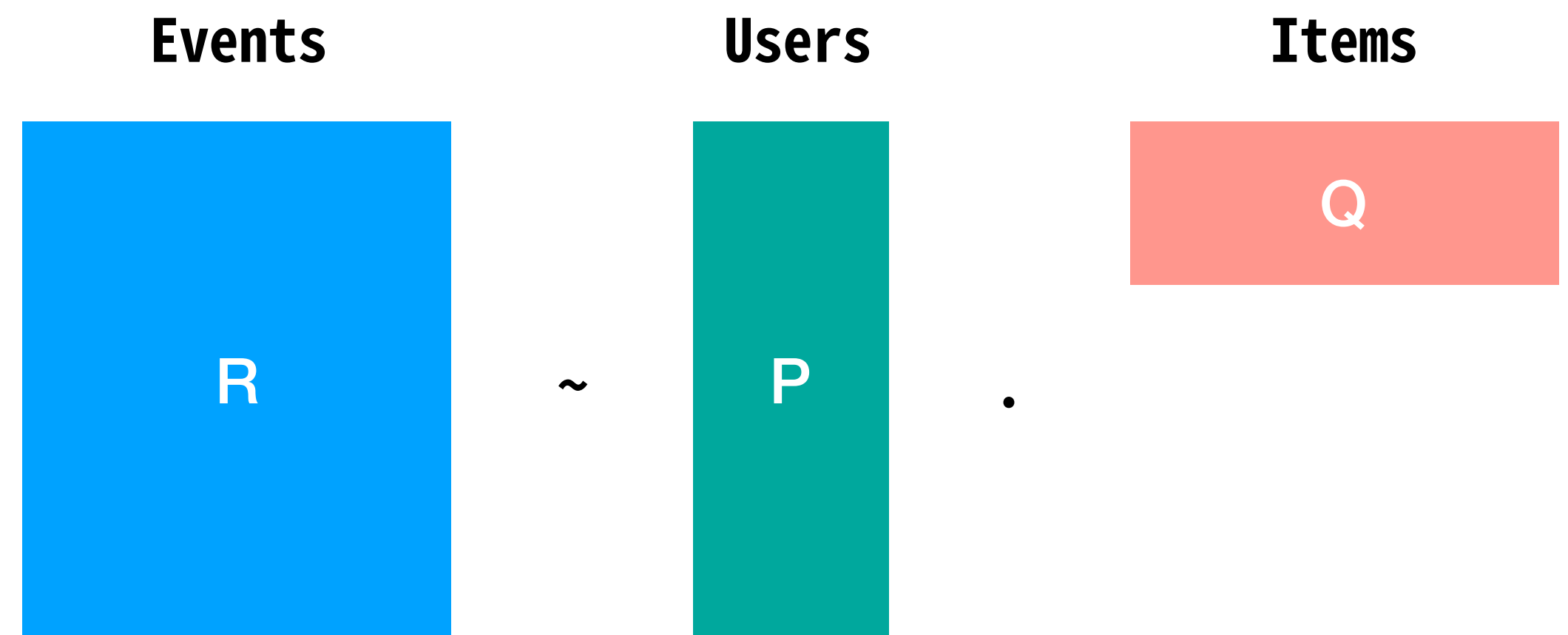
버팔로 들여다보기

행렬 분해를 수행할 때, 목적 함수의 정의 그리고 데이터의 구성과 변형에 따라 서로 다른 성질의 은닉 벡터를 구할 수 있게 됩니다.

여러 행렬 분해 알고리즘을 사용함으로써 데이터에서 다양한 특질을 확보할 수 있게 됩니다.

- Alternating Least Square
- Bayesian Personalized Ranking Matrix Factorization
- Word2Vec
- CoFactors

$$\min_{x_*, y_*} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$



버팔로 들여다보기

if (kakao) dev 2019

버팔로에서 제공하는 알고리즘 외에도 많은 행렬 분해 알고리즘들이 있습니다.

버팔로에서는

우리가 **원하는 크기의 데이터를**

적절한 시간에

효과적인 자원으로

수행할 수 있으며

예측, 유사 사용자/아이템 계산 등 추천 시스템에 필요한 좋은 품질의 결과를 보여주는 알고리즘을 제공합니다.

버팔로 들여다보기

if (kakao) dev 2019

scalability > quality 가 항상 성립하는 것은 아닙니다.

그러나 우리가 겪은 많은 문제의 경우

Scalability는 타협 가능하지 않고

Quality는 타협 가능합니다.

Scalability is matter

well known large database for recommender systems

	# USERS	# ITEMS	# NNZ
MovieLens20M	138,000	27,000	20M
EchoNest	1,019,318	384,546	48M
Amazon reviews	6,643,669	2,441,053	34M
KakaoBrunch12M	306,291	505,926	12M
Yahoo! Music Ratings	1,800,000	136,000	717M
KakaoReco730M	21,940,315	1,467,298	730M

Scalability is matter

well known large database for recommender systems

	# USERS	# ITEMS	# NNZ
MovieLens20M	138,000	27,000	20M
EchoNest	1,019,318	384,546	48M
Amazon reviews	6,643,669	2,441,053	34M
KakaoBrunch12M	306,291	505,926	12M
Yahoo! Music Ratings	1,800,000	136,000	717M
KakaoReco730M	21,940,315	1,467,298	730M

약 16년 동안 수집

약 10일 미만 수집

* 최근(2018년)까지도 학회에 공유되는 연구 결과들은 대부분 MovieLens 20M 보다 작은 데이터베이스에서 진행됨

Scalability is matter

well known large database for recommender systems

	# USERS	# ITEMS	# NNZ
MovieLens20M	138,000	27,000	20M
Echo3M			3M
Amazon4M			4M
KakaoBrunch12M	306,291	505,926	12M
Yahoo! Music Ratings	1,800,000	136,000	717M
KakaoReco730M	21,940,315	1,467,298	730M

Scalability를 잃으면 기회를 잃습니다

약 16년 동안 수집

약 10일 미만 수집

* 최근(2018년)까지도 학회에 공유되는 연구 결과들은 대부분 MovieLens 20M 보다 작은 데이터베이스에서 진행됨

버팔로 들여다보기

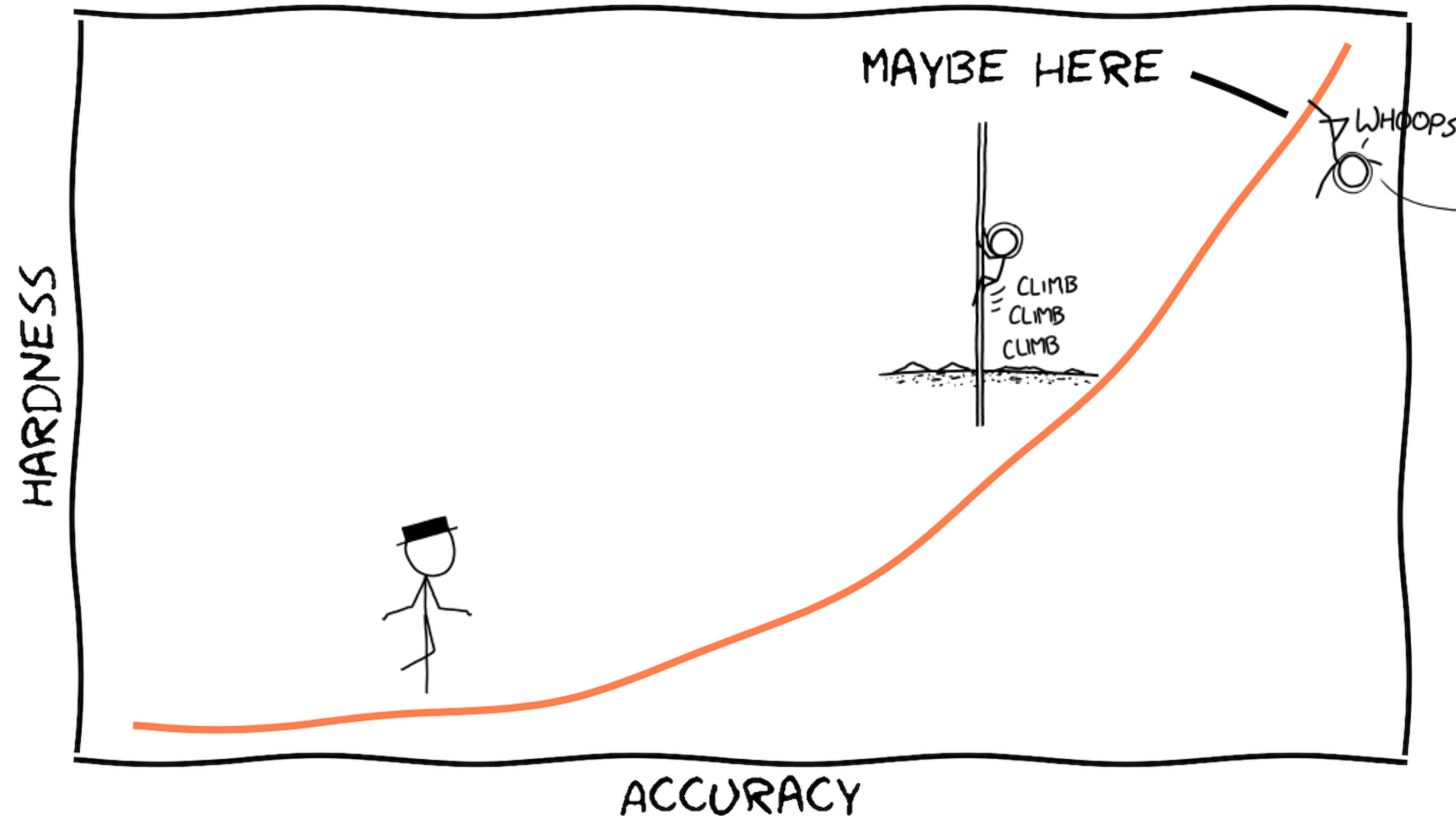
품질은 어디까지 타협해야할까?

다양한 알고리즘을 카카오 데이터에 실험해본 경험에 비춰보면 버팔라고 제공하는 알고리즘들에 비해 체감상 차이가 미미하고, 전체 시스템의 성능에 미치는 영향도 역시 차이가 크지 않습니다.

투자 대비 효율로의 전환

- 긴 파이프라인으로 구성된 시스템
- Ceiling analysis

상향 평준화일까?



버팔로 들여다보기

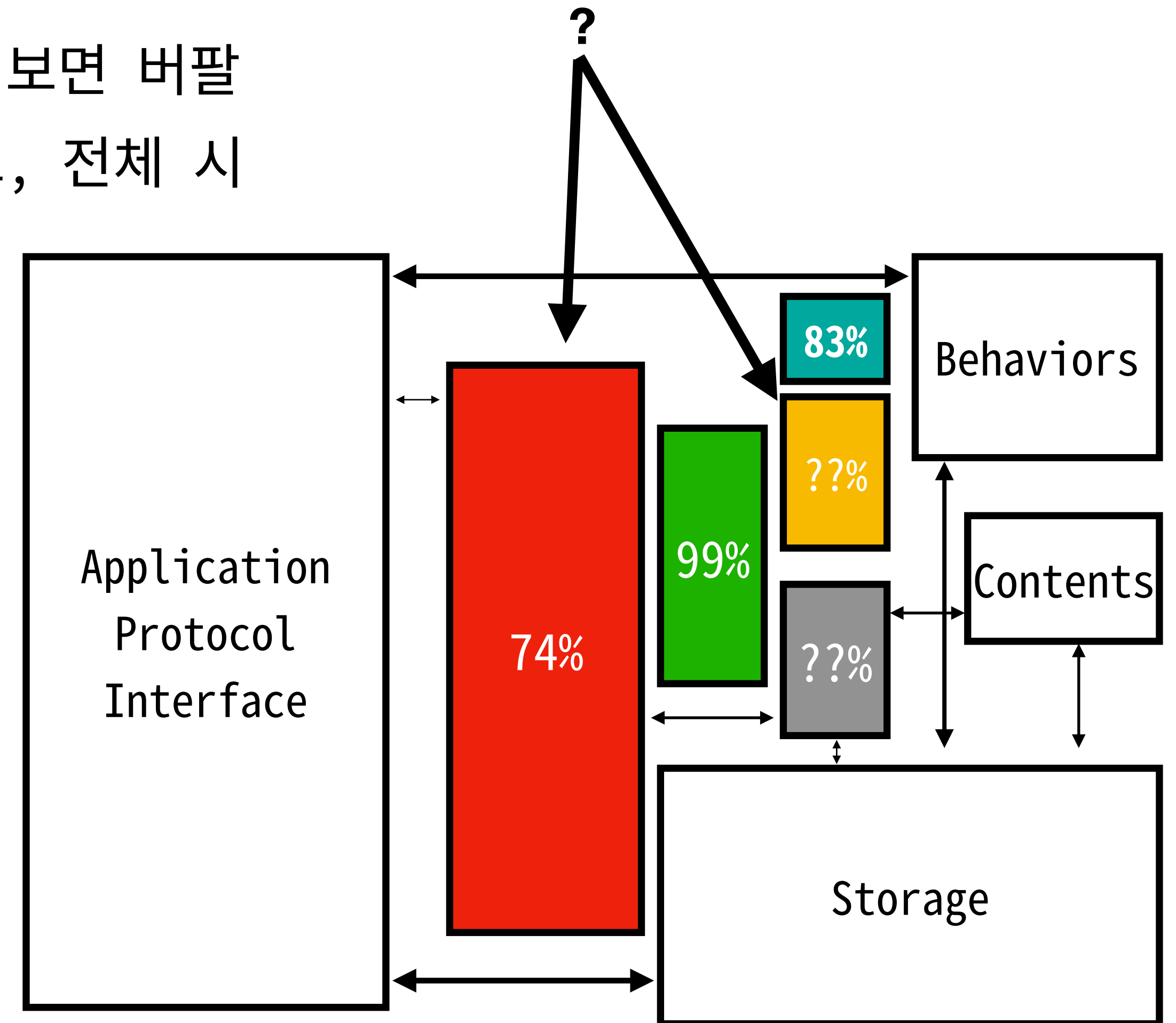
if (kakao) dev 2019

품질은 어디까지 타협해야할까?

다양한 알고리즘을 카카오 데이터에 실험해본 경험에 비춰보면 버팔라고 제공하는 알고리즘들에 비해 체감상 차이가 미미하고, 전체 시스템의 성능에 미치는 영향도 역시 차이가 크지 않습니다.

투자 대비 효율로의 전환

- 긴 파이프라인으로 구성된 시스템
- Ceiling analysis



버팔로 들여다보기

if (kakao) dev 2019

높은 생산성과 성능

버팔로 들여다보기

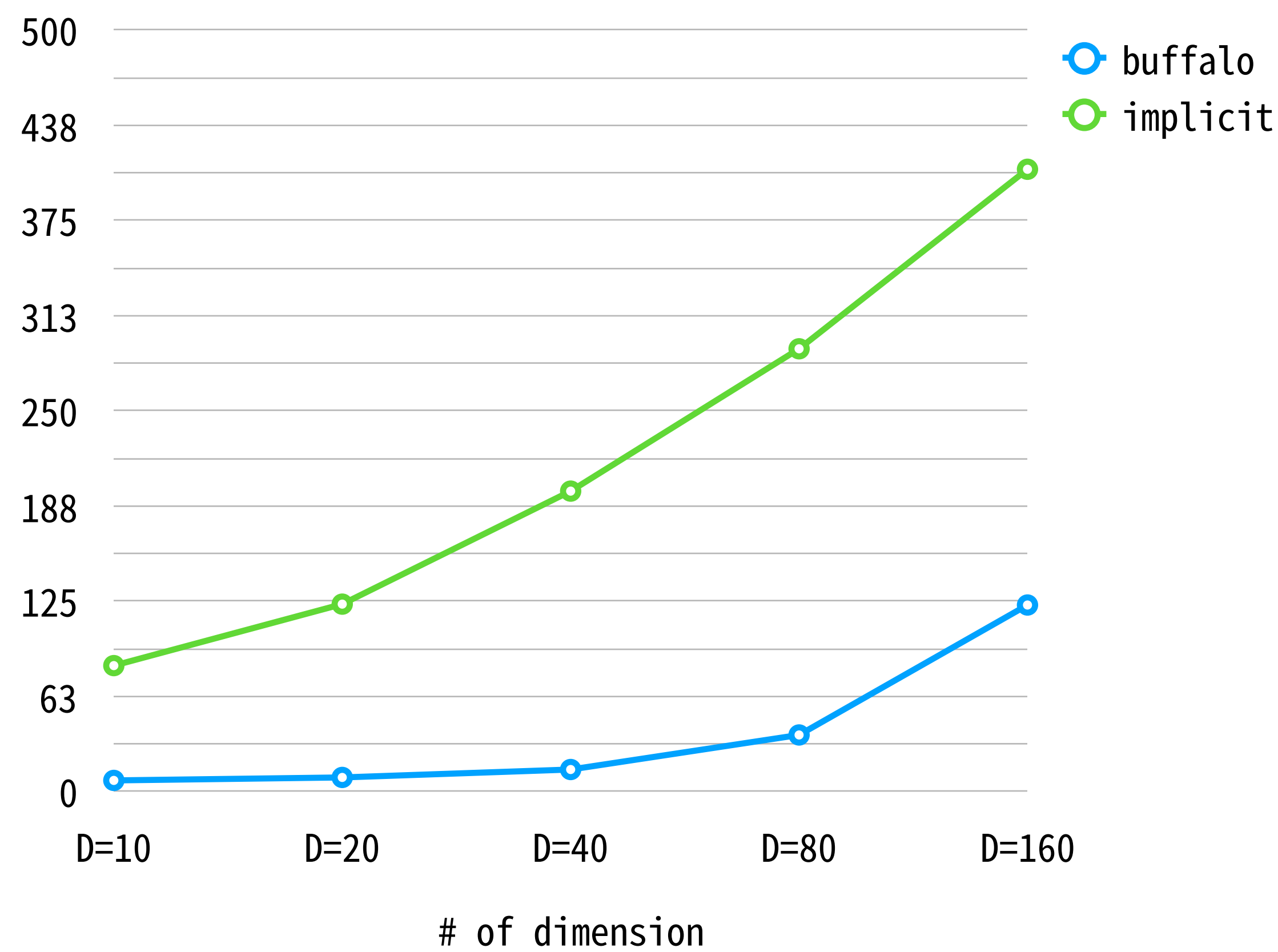
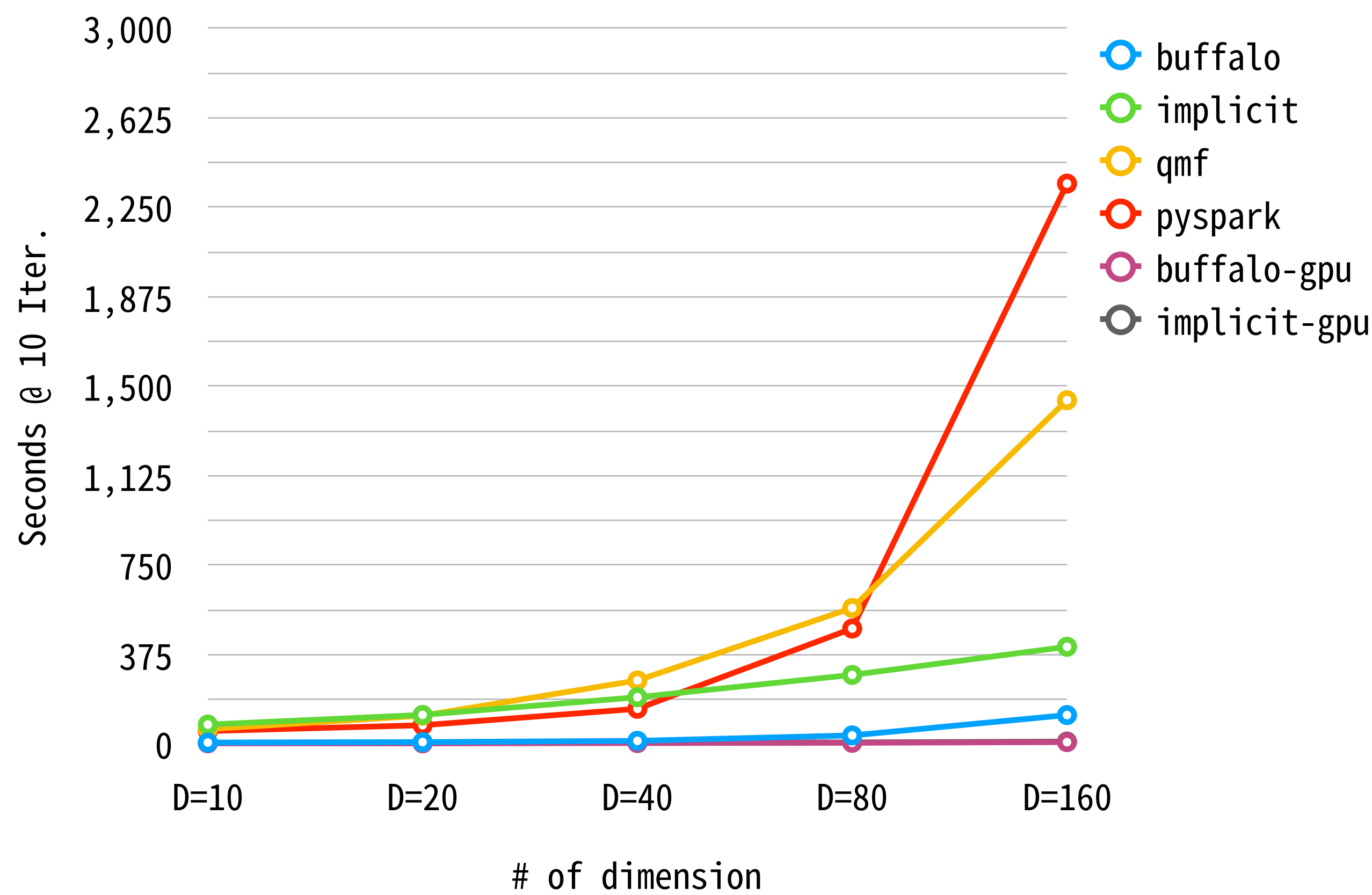
if (kakao) dev 2019

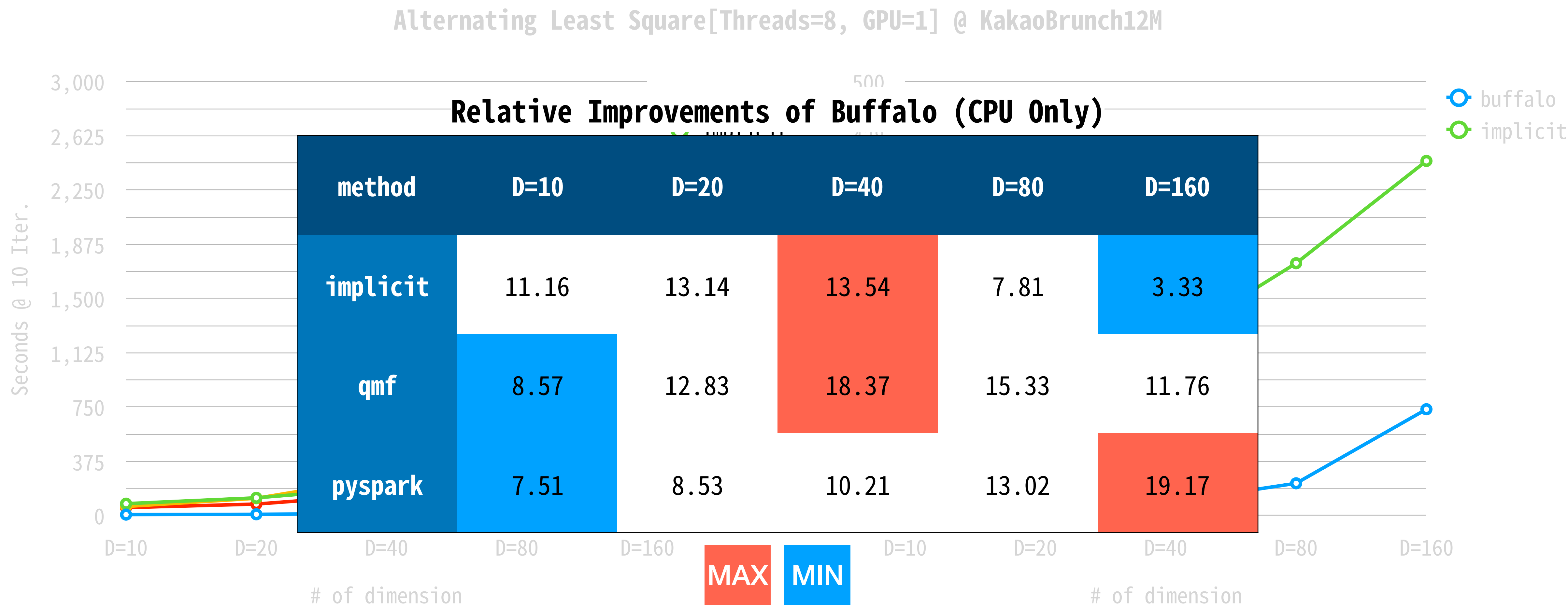
높은 생산성과 성능

정량적인 비교를 위해 현재 공개된 라이브러리들과 벤치마크를 수행했고, 라이브러리는 사용자 수와 성능을 고려해 선택했습니다.

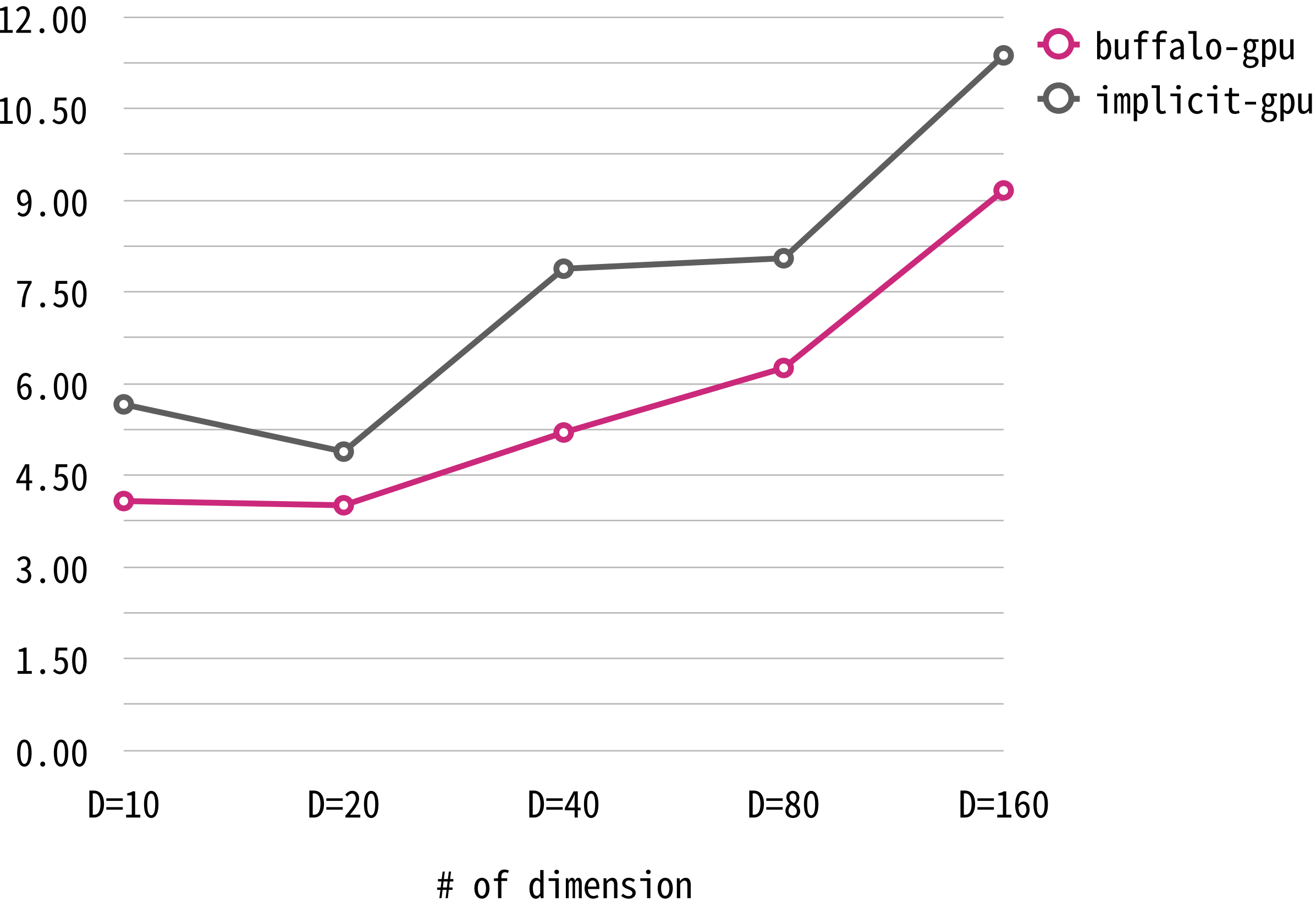
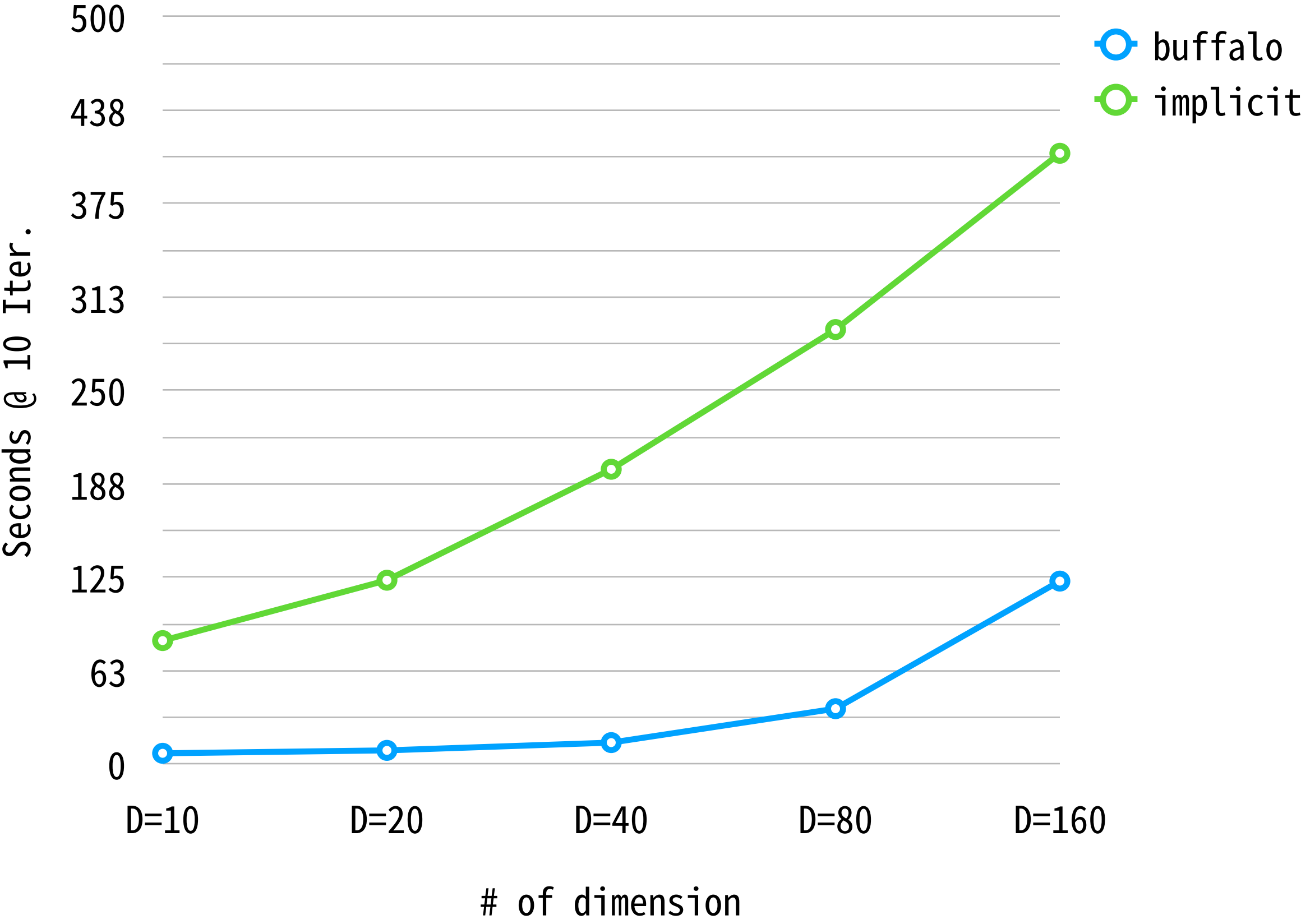
- Apache Spark Scala
- Quora QMF C++
- Implicit C++/Python
- lyst LightFM Python

Alternating Least Square[Threads=8, GPU=1] @ KakaoBrunch12M

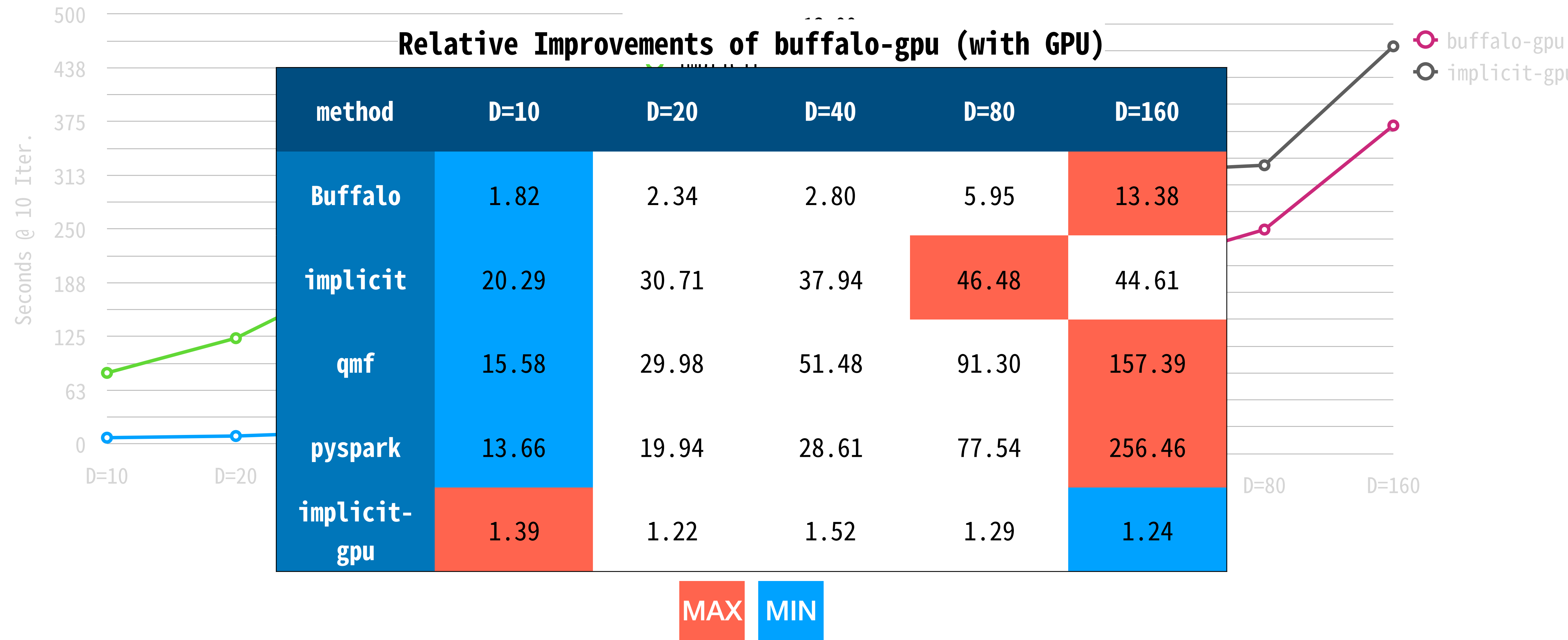




Alternating Least Square[Threads=8, GPU=1] @ KakaoBrunch12M



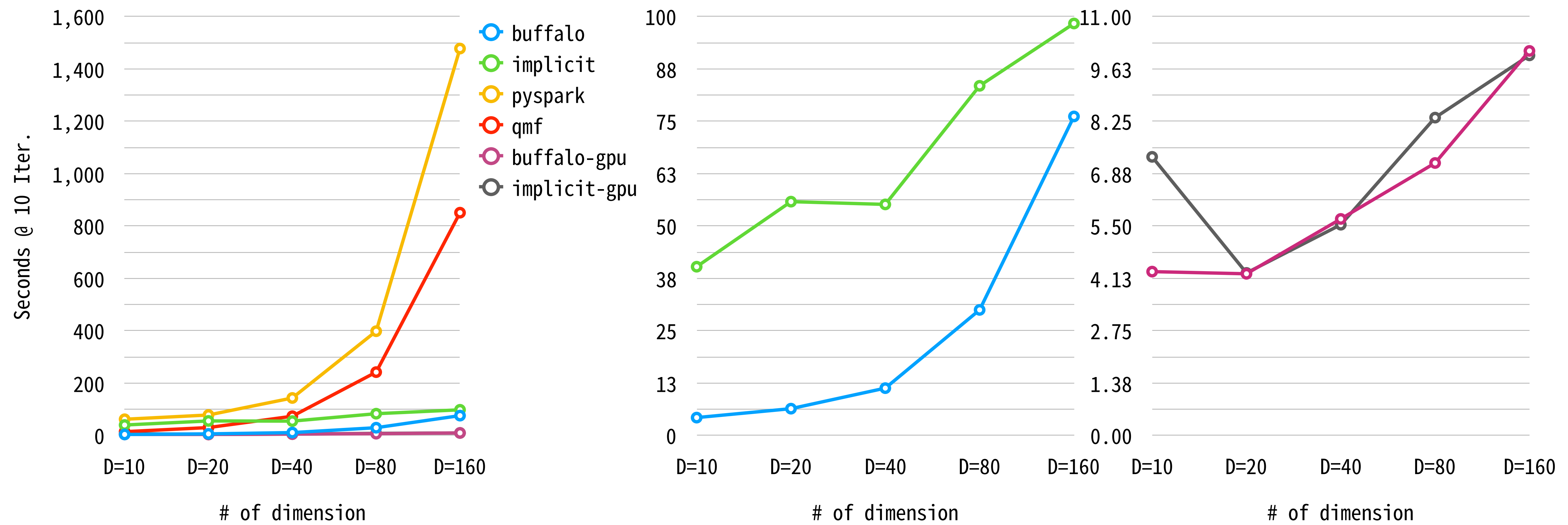
Alternating Least Square[Threads=8, GPU=1] @ KakaoBrunch12M



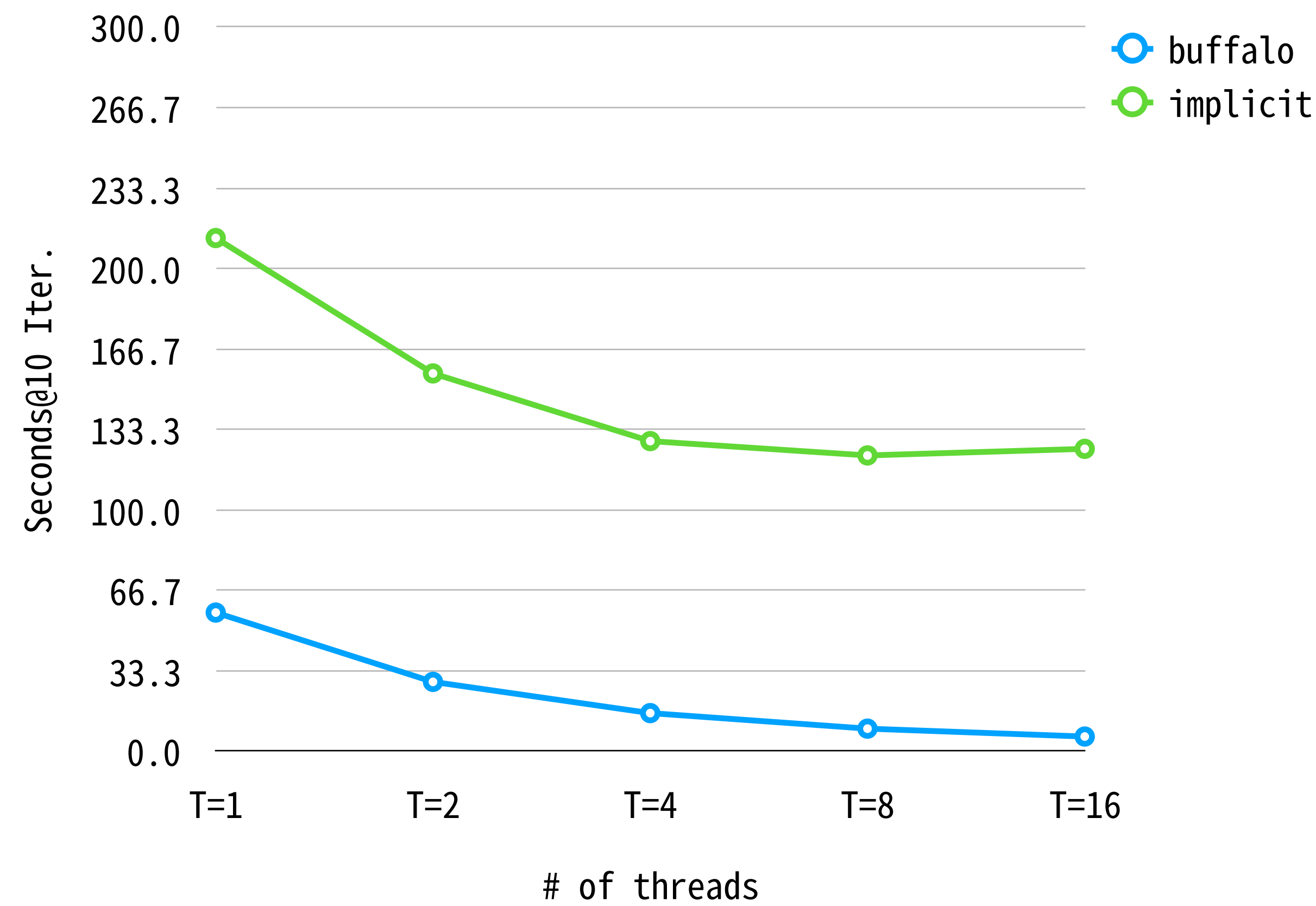
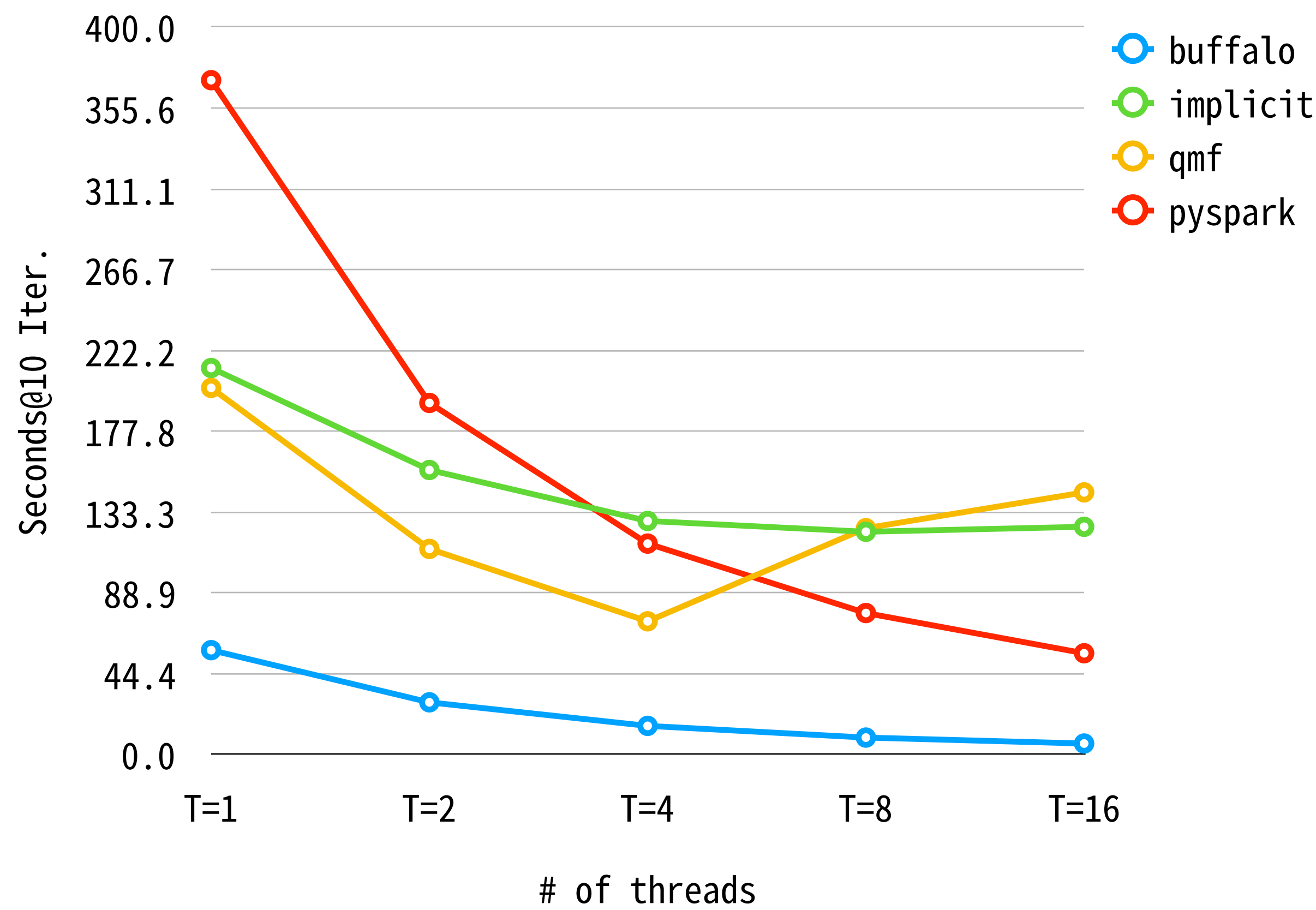
버팔로 들여다보기

if (kakao) dev 2019

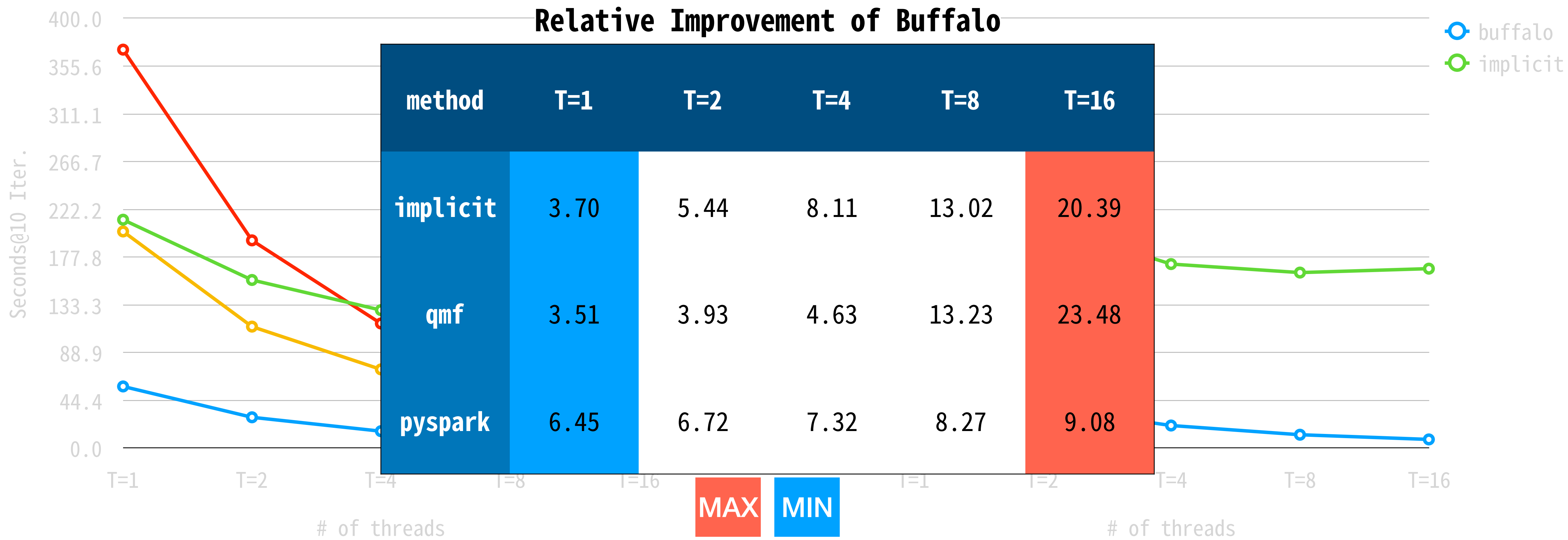
Alternating Least Square[Threads=8, GPU=1] @ MovieLens20M



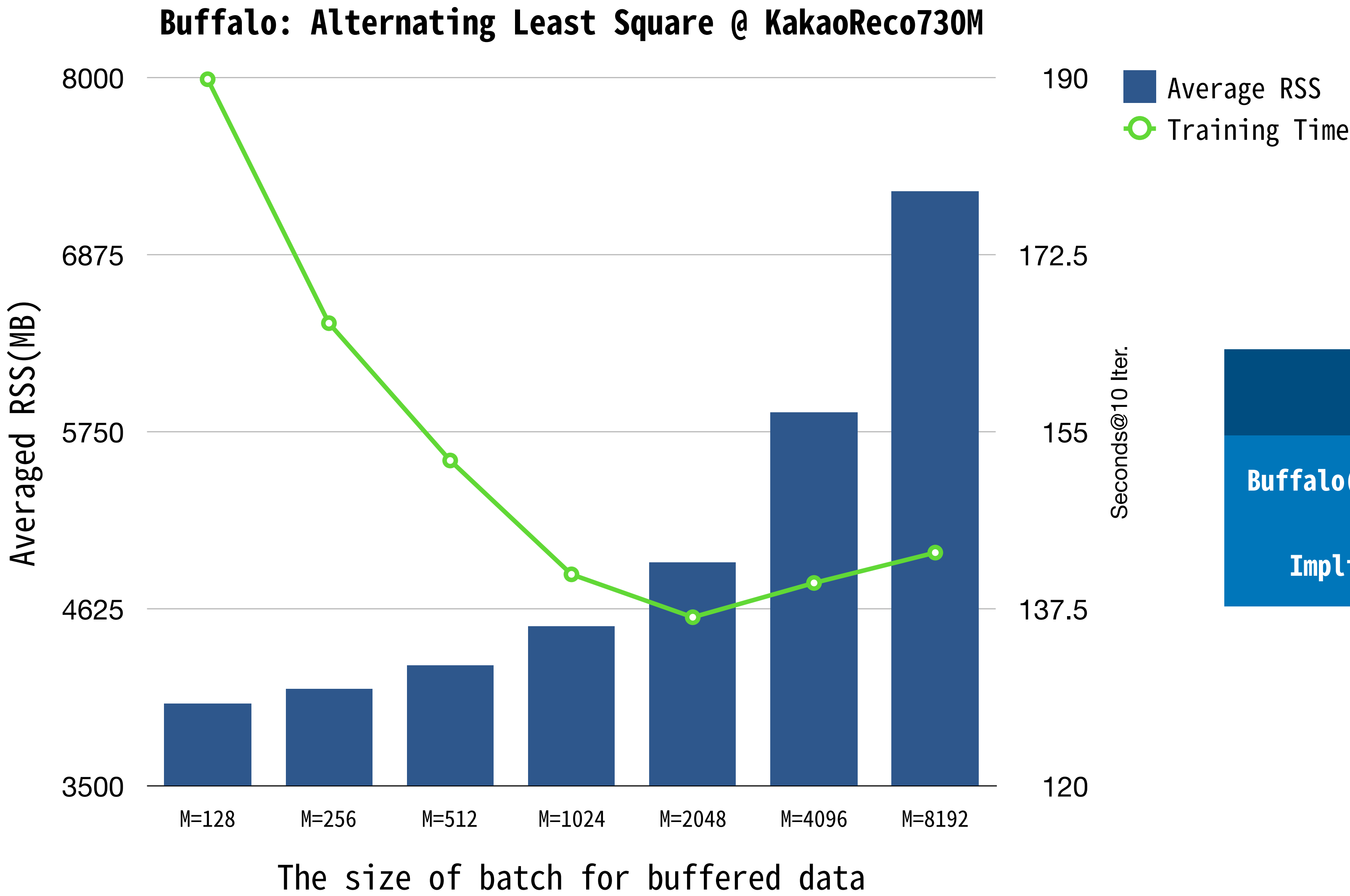
Alternating Least Square[D=20] @ KakaoBrunch12M



Alternating Least Square[D=32] @ KakaoBrunch12M



버팔로 들여다보기



KakaoReco730M		
	Average RSS	Elapsed
Buffalo(M=512)	4,269MB	152 secs
Implicit	24,592MB	684 secs

*17%

*22%

버팔로 들여다보기

높은 생산성과 성능

추천 시스템의 기본적인 유사 콘텐츠 추천과 개인화 추천에 대한 병렬 처리 기능을 제공합니다.

```
from buffalo.algo.als import ALS
from buffalo.parallel.base import ParALS
als = ALS.new('./ml20m.bin')
```

Base ALS

```
als.most_similar('Starwars')
```

Parallel ALS

```
par = ParALS(als)
```

```
par.num_worker = 8
```

```
par.most_similar(['Starwars', 'Startrek'])
```

```
from buffalo.algo.als import ALS
```

```
from buffalo.parallel.base import ParALS
```

```
als = ALS.new('./ml20m.bin')
```

```
par = ParALS(als) # Parallel ALS
```

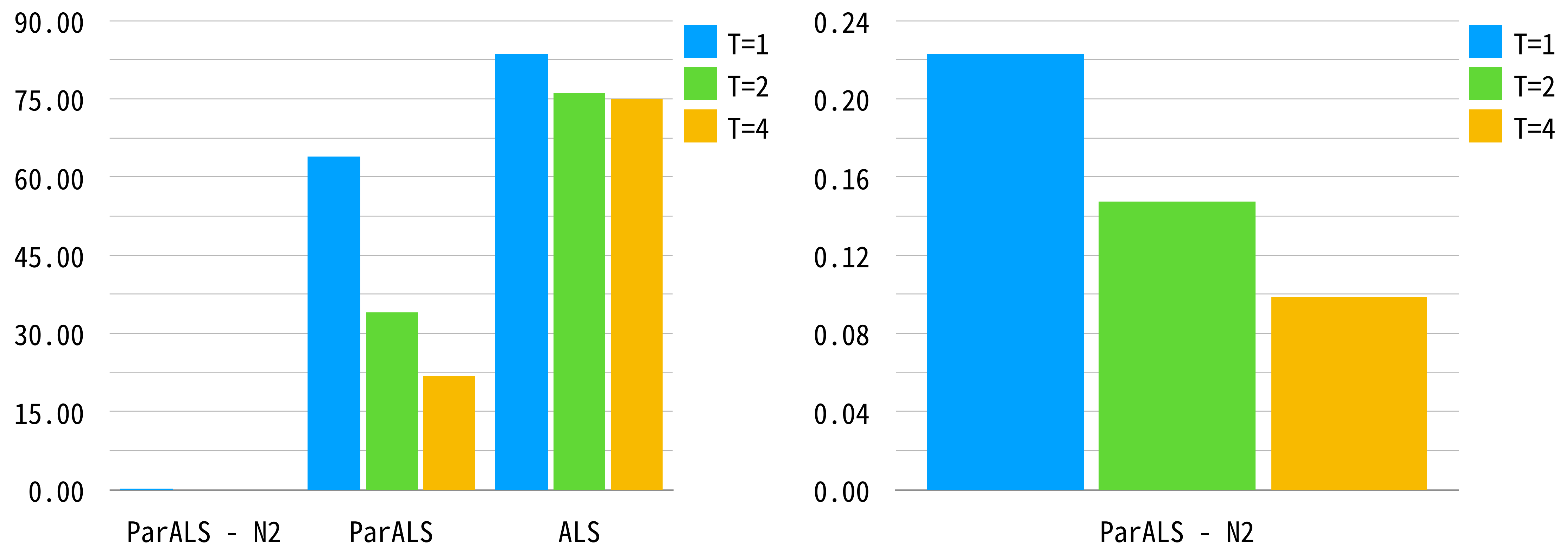
```
par.num_worker = 8
```

Parallel ALS with N2

```
par.set_hnsw_index('toros.n2.ml20m.bin')
```

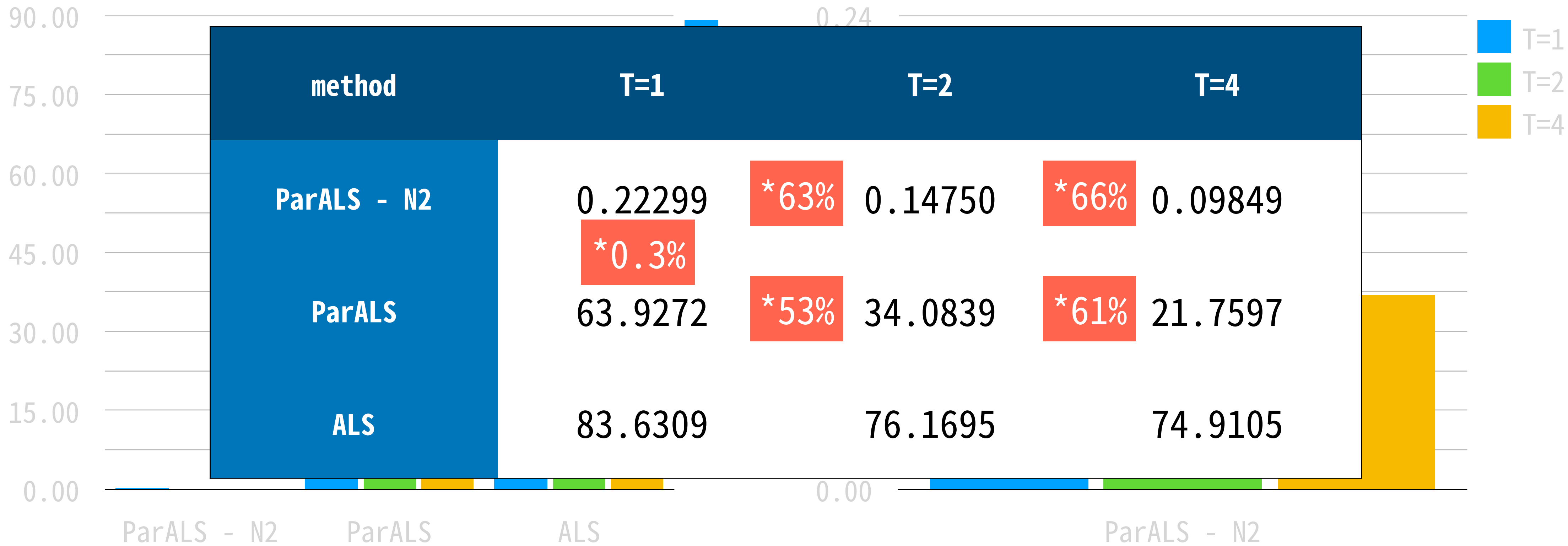
```
par.most_similar(['Starwars', 'Startrek'])
```

10K x Top10-MostSimilar Queries @ KakaoBrunch12M



버팔로 들여다보기

10K x Top10-MostSimilar Queries @ KakaoBrunch12M



버팔로 들여다보기

높은 생산성과 성능

버팔로는 현재 공개된 라이브러리들 보다 수배에서 수십배 빠르면서 동시에 뛰어난 메모리 관리 능력을 보여줍니다.

- 최적화된 Python/C++ 코드 베이스와 병렬처리를 효과적인 구현

T1			T2			T3		
1	2	3	4	5	6	7	8	9
T1		T2			T3			better

- 청크 구조의 데이터 배치 스트림을 통한 메모리 관리

버팔로 들여다보기

if (kakao) dev 2019

편의 기능

버팔로 들여다보기

if (kakao) dev 2019

Validation

버팔로는 두 개의 데이터 포맷을 지원합니다.

Stream 데이터에서 Matrix Market 데이터로 변환 기능을 제공합니다.

Matrix Market(희소 행렬)

#users #items #NNZ

1 2 3

3 2 1

...

$$R(i, j) = v$$

Stream

Starwars EmpireStrikesBack ToyStory

LegendsOfTheFall InTheMoodForLove

...

데이터의 흐름 =>

Validation

```
class MatrixMarketOptions(DataOption):
    def get_default_option(self) -> aux.Option:
        opt = {
            ...
            'data': {
                'internal_data_type': 'matrix',
                'validation': {
                    'name': 'sample',
                    'p': 0.01,
                    'max_samples': 500
                },
            },
            ...
        }
    return aux.Option(opt)
```

```
class StreamOptions(DataOption):
    def get_default_option(self) -> aux.Option:
        opt = {
            ...
            'data': {
                'internal_data_type': 'stream',
                'validation': {
                    'name': 'newest',
                    'n': 1,
                    'max_samples': 500
                },
            },
            ...
        }
    return aux.Option(opt)
```

Evaluation

```
def example1():  
    als_option = ALSOption().get_default_option()  
    als_option.validation = aux.Option({'topk': 10})  
    data_option = MatrixMarketOptions().get_default_option()  
    data_option.input.main = '../tests/ext/ml-100k/main'  
    data_option.input.iid = '../tests/ext/ml-100k/iid'  
  
    als = ALS(als_option, data_opt=data_option)  
    als.initialize()  
    als.train()
```


버팔로 들여다보기

if (kakao) dev 2019

Evaluation

```
def example1():  
    als_option = A  
    als_option.val  
    data_option =  
    data_option.in  
    data_option.in  
  
    als = ALS(als_  
    als.initialize  
    als.train()
```

```
MatrixMarket Header(943, 1682, 79500) Validation(500 samples)  
[data.py:71] Set data buffer size as 67108864(minimum required batch size is 461).  
[ Validation: ndcg:0.04599 map:0.02595 accuracy:0.08792 rmse:2.99161 error:2.78435  
[ Iteration 1: RMSE 0.436 Elapsed 0.025 secs  
[ Validation: ndcg:0.07738 map:0.05011 accuracy:0.13880 rmse:2.94502 error:2.73830  
[ Iteration 2: RMSE 0.348 Elapsed 0.022 secs  
[ Validation: ndcg:0.07139 map:0.04544 accuracy:0.13016 rmse:2.93846 error:2.73221  
[ Iteration 3: RMSE 0.312 Elapsed 0.022 secs
```

```
MovieLens 100k metrics for validations  
{  
    "ndcg": 0.07489254232592911,  
    "map": 0.04879477838494232,  
    "accuracy": 0.13879781420765025,  
    "rmse": 2.9268705104775945,  
    "error": 2.720654326558113  
}
```

버팔로 들여다보기

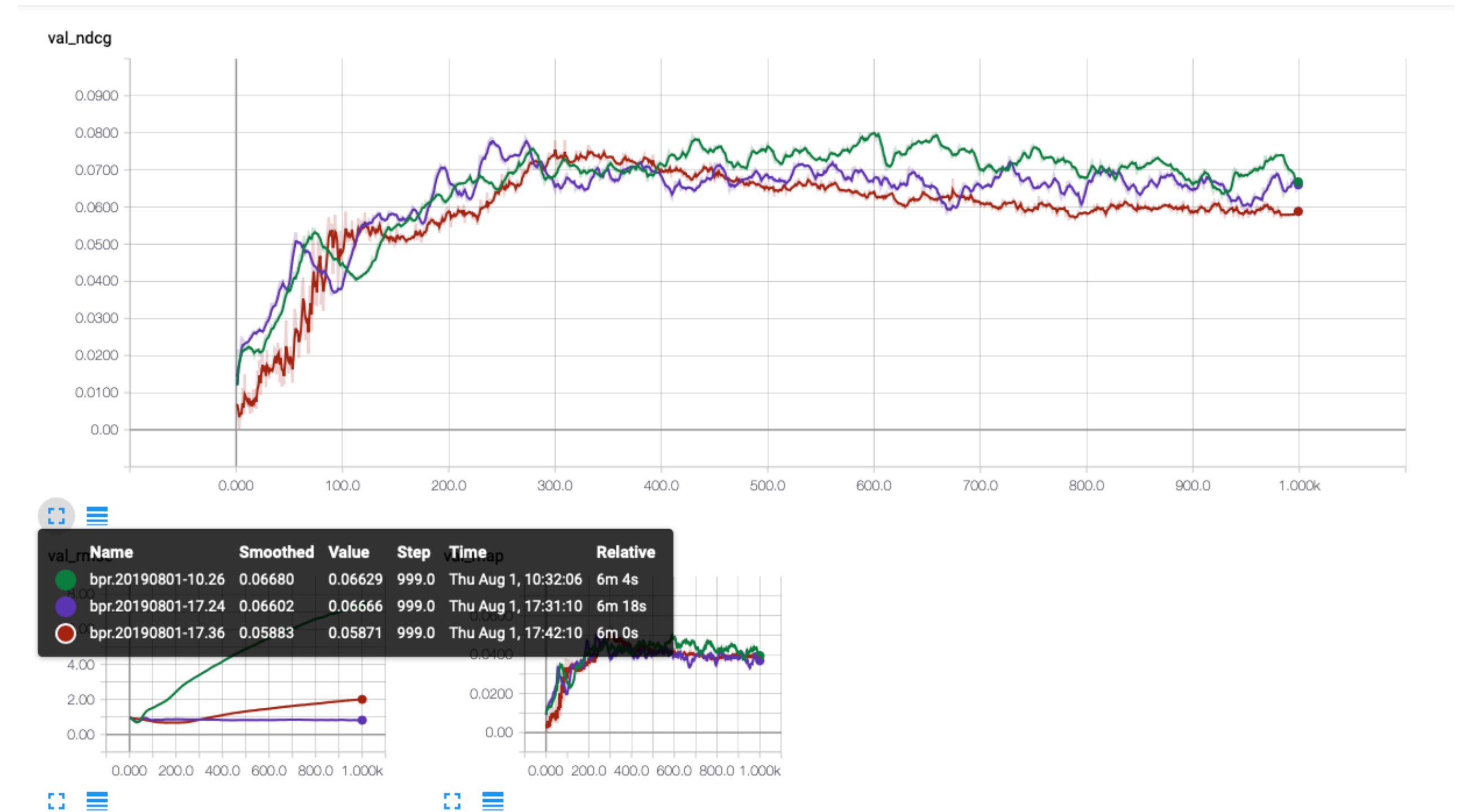
if (kakao) dev 2019

Tensorboard

```
from buffalo.misc import aux
from buffalo.algo.bpr import BPRMF
from buffalo.algo.options import BPRMF0ption
```

```
opt = BPRMF0ption().get_default_option()
opt.tensorboard = aux.Option({'root': './tb',  
                             'name': 'bpr'})
```

```
bpr = BPRMF(opt)
bpr.initialize()
bpr.train()
```



버팔로 들여다보기

if (kakao) dev 2019

Hyper-parameter optimization

```
opt.update({
    'loss': 'val_ndcg',
    'max_trials': 100,
    'min_trials': 0,
    'deployment': True,
    'start_with_default_parameters': True,
    'space': {
        'adaptive_reg': ['choice', ['adaptive_reg',
                                     [0, 1]]],
        'd': ['randint', ['d', 10, 30]],
        'reg_u': ['uniform', ['reg_u', 0.1, 1]],
        'reg_i': ['uniform', ['reg_i', 0.1, 1]],
        'alpha': ['randint', ['alpha', 1, 32]]
    }
})
```

```
[INFO    ] 2019-08-xx xx:xx:xx [optimize.py:
44] Starting with default parameter result:
{'train_loss': 0.24002212208588575,
'val_ndcg': 0.08186079712106625, ...
```

```
[INFO    ] 2019-08-xx xx:xx:xx [optimize.py:
67] Found new best parameters: {'train_loss':
0.16249954664375768, 'val_ndcg':
0.10879374067776124, ... 'status': 'ok'} @
iter 1
```

```
[INFO    ] 2019-08-xx xx:xx:xx [optimize.py:
72] Saving model... to ./
example1.ml100k.als.optimize.bin
```

03 마무리

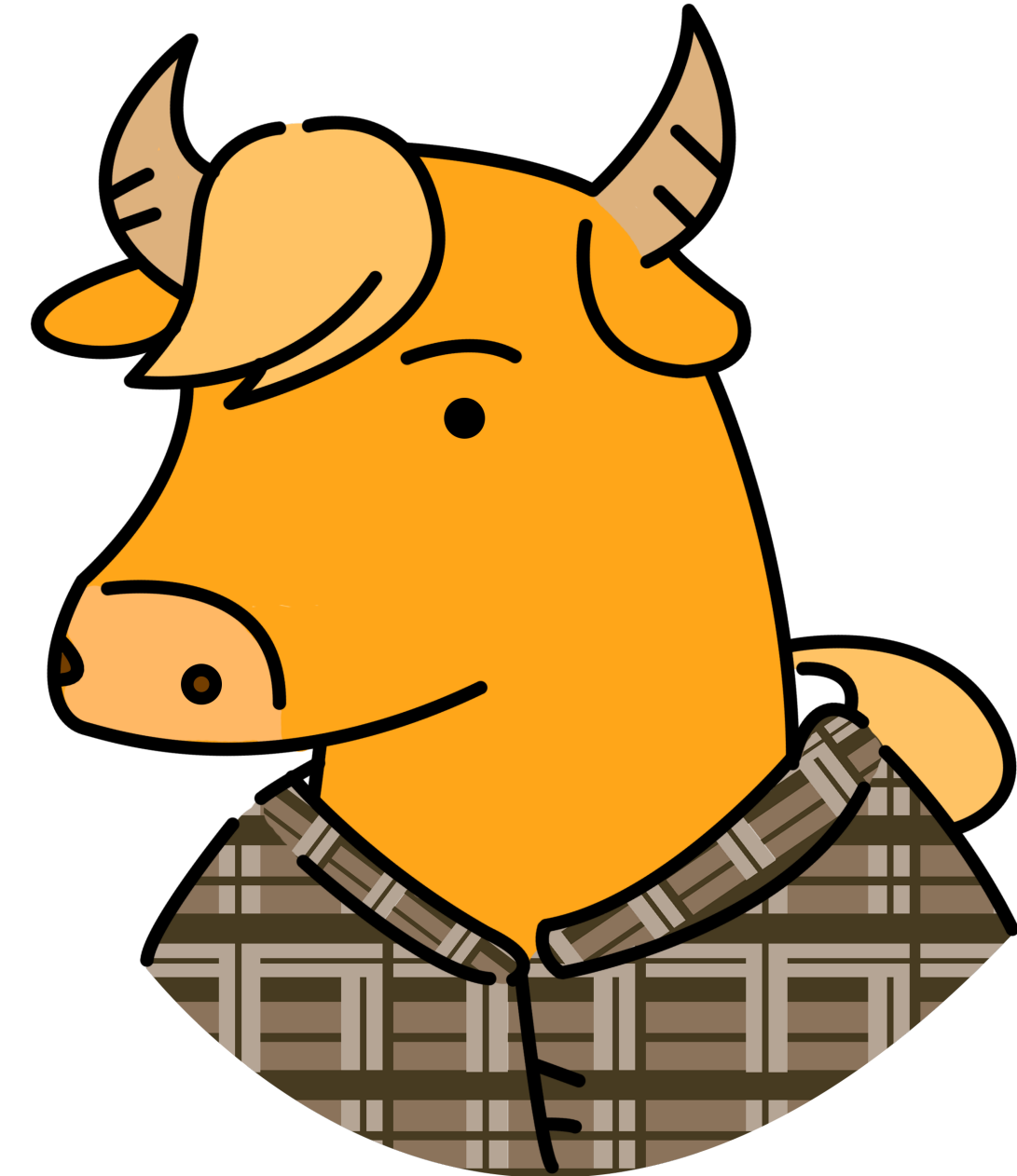
if (kakao) dev 2019

03 마무리

if (kakao) dev 2019

Buffalo

- C++/Python
- CPU 최적화, GPU 지원
- 효과적인 시스템 자원 활용
- 연구/개발 목적의 편의 기능 제공



<https://github.com/kakao/buffalo>

PULL REQUEST WELCOME

03 마무리

if (kakao) dev 2019

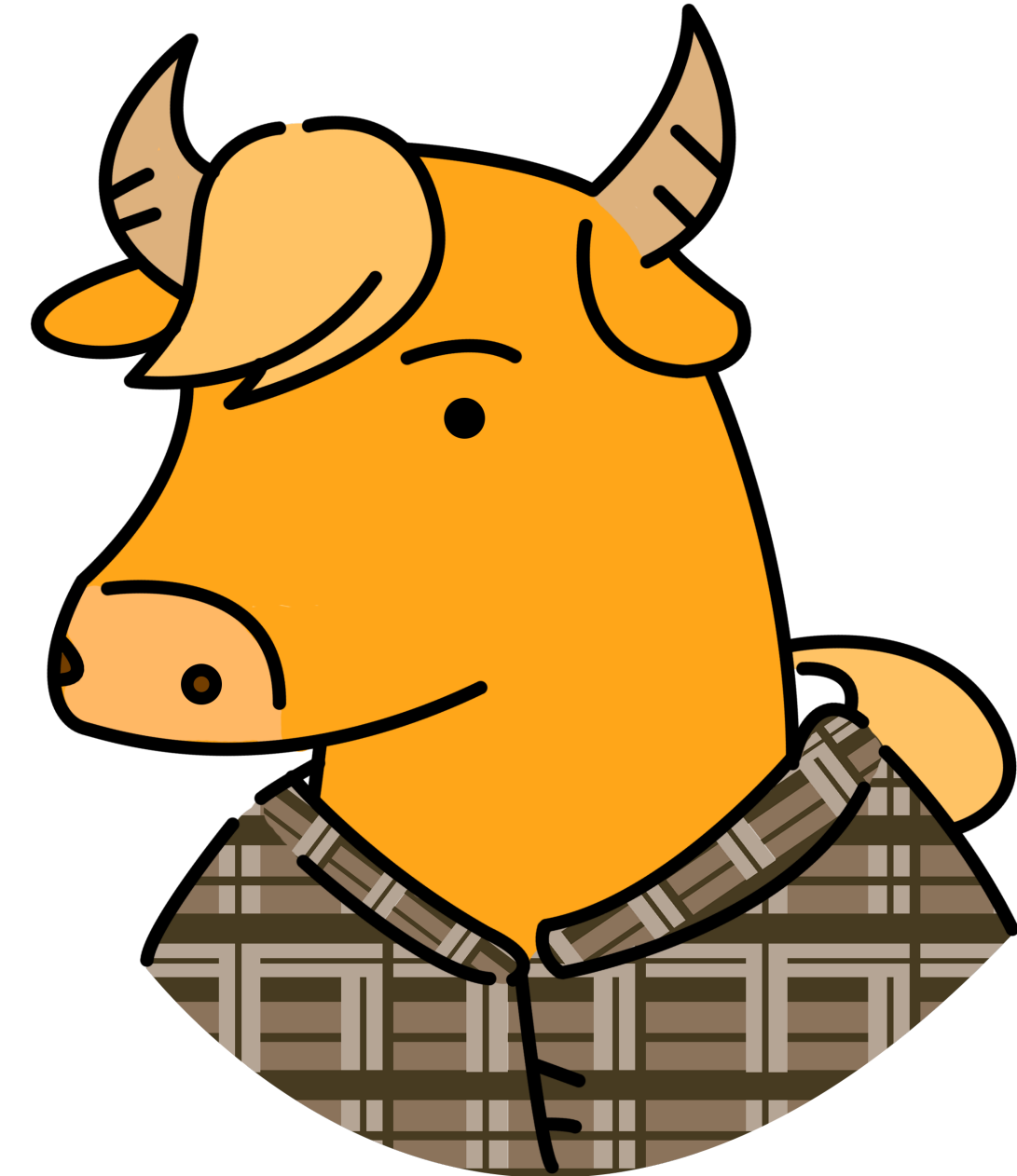
향후 계획

벤치마크 표준화

- 예측, 유사도
- 맥락 고려: 시간, UI/UX 상황, ...
- 다양한 데이터: 블로그, 뉴스, 음악, ...
- 양질의 데이터: 편향성 억제, 전문가 레이블 ...

지속적인 오픈 소스 활동

- 신규 작업: 실시간 추천 어플리케이션 서버, ...
- 기존 성과물의 개선 작업: n2, buffalo



<https://github.com/kakao/buffalo>
PULL REQUEST WELCOME

03 마무리

if (kakao) dev 2019

감사합니다.

03 마무리

if (kakao) dev 2019

QnA

References

if (kakao) dev 2019

- [1] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." 2008 Eighth IEEE International Conference on Data Mining. Ieee, 2008
- [2] Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 2009.
- [3] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
- [4] Liang, Dawen, et al. "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence." Proceedings of the 10th ACM conference on recommender systems. ACM, 2016.