APIレスポンスのお手軽キャッシュ化 with Recoil

2022.12.14 UIT Meetup

tomoaki.hirabayashi

自己紹介

Tomoaki Hirabayashi

日本大学工学部 2年 ←福島にある

LINE株式会社

- LINE証券 Frontend

  アルバイト（インターンから継続）

ラーメンが好きです
（写真は大崎の六厘舎）

今回の内容

- キャッシュ化する目的
- Recoilについて
  - Recoilって何
  - 他の選択肢と比較してみる
  - 実際にRecoilで既存の実装をキャッシュ化する
- 実際にプロダクトで使ってみた
  - LINE証券のパフォーマンスランキングをキャッシュ化した
  - 実際に使って思ったこと

APIレスポンスのキャッシュ化

SPAは基本的にコンポーネントがマウントされるたびにFetchが走ってしまう

今回の目的

キャッシュを効かせて
できるだけお手軽にFirstViewを改善したい！

Recoilって何

A state management library for React

- Reactを作っているMeta自身が開発
- シンプルな記法でグローバルな状態管理ができる
- 既存のuseStateもめっちゃ簡単に共通State化できる
- 非同期処理との親和性も◎

他の選択肢と比較してみる

Promise処理の伴う状態管理をしたいときのその他の選択肢

- SWR
- TanStack Query(旧ReactQuery)
- etc…

SWR

```
const fetcher = (...args) => fetch(...args).then(res =>
res.json())

const { data, error } = useSWR('/api/user/123',
fetcher)
```

↑実装例
（swr.vercel.app/ja/examples/basic より）

☆お手軽ポイント☆

- キャッシュの効いたフェッチ処理がたった２行で実現可能←お手軽
- オプション豊富
  - キャッシュ期限、更新間隔 etc..
  - 無限Fetchも標準対応
- キャッシュの挙動を柔軟に拡張可
  - Ex) お手軽に永続キャッシュ化

Recoil

```
const nowTimeApiState = selector({
  key: 'nowTimeApi',
  get: async ({ get }) => {
    const result = await (await fetch(get(reqUrlState))).json()
    return result
  },
})

// apiResult.state = "hasValue" | "loading" | "hasError"
const apiResult = useRecoilValueLoadable(nowTimeApiState)
```

↑Selectorを使えばPromise処理の状態管理もできる

- SWRはデータのPromise処理に割と特化している
- Recoilは汎用的な状態管理ライブラリ
  - →Promise処理がお手軽になる機能はそこまで備えていない

Recoilのお手軽ポイントは？

Recoil

```
const nowTimeApiState = selector({
  key: 'nowTimeApi',
  get: async ({ get }) => {
    const result = await (await fetch(get(reqUrlState))).json()
    return result
  },
})

// apiResult.state = "hasValue" | "loading" | "hasError"
const apiResult = useRecoilValueLoadable(nowTimeApiState)
```

↑Selectorを使えばPromise処理の状態管理もできる

☆お手軽ポイント☆

- 既存のロジックを書き換えたいなら圧倒的にお手軽
- 学習コストは高いけど低い
- 使い方を覚えておけば将来もお手軽

どtちもお手軽だ

それぞれのライブラリにお手軽ポイントがある

ライブラリ	特にお手軽だと思ったポイント
SWR	機能が豊富で大体のことは短いコードで実現可能
Recoil	既存の実装に合わせられる無限の拡張性

Recoilで既存の実装をキャッシュ化する

よくあるfetchするコード

```
const [apiResult, setApiResult] = useState({
    status: 'Loading',
  })
  useEffect(() => {
    fetch('/api/nowtime')
      .then((res) => res.json())
      .then((res) => setApiResult({ status: 'Succeed', data: res }))
      .catch(() => setApiResult({ status: 'Failed' }))
  }, [])

  if (apiResult.status === 'Succeed') {
    return <p>{apiResult.data.now}</p>
  }
  return <p>{apiResult.status}</p>
```

このStateを保持すれば
キャッシュ化できる！

Recoilで既存の実装をキャッシュ化する

Stateの保持にはAtomを使う

```
const nowTimeApiState = atom({
  key: 'nowTimeApi',
  default: {
    status: 'Loading',
  },
})
```

Recoilで既存の実装をキャッシュ化する

useStateをuseRecoilState(atom)に置き換えて

データがない時だけfetchする分岐を書く

```
const [apiResult, setApiResult] = useState({status: 'Loading'})
useEffect(() => {
    fetch('/api/nowtime')
      .then((res) => res.json())
      .then((res) => setApiResult({ status: 'Succeed', data: res }))
      .catch(() => setApiResult({ status: 'Failed' }))
  }, [])
```

Recoilで既存の実装をキャッシュ化する

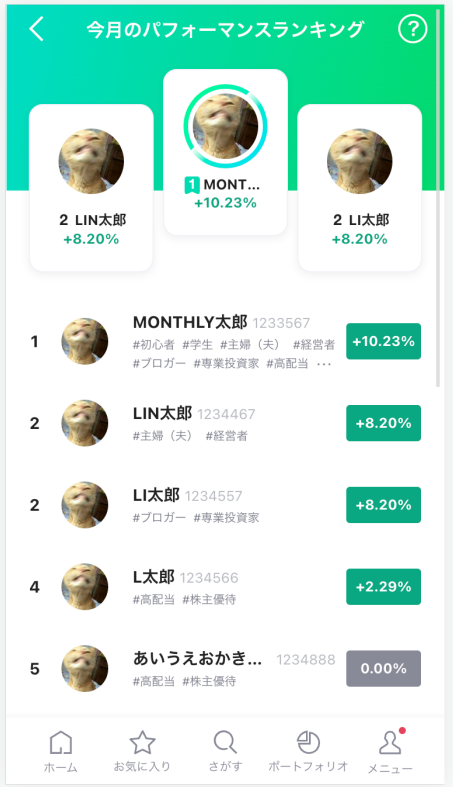
useStateをuseRecoilState(atom)に置き換えて

データがない時だけfetchする分岐を書く

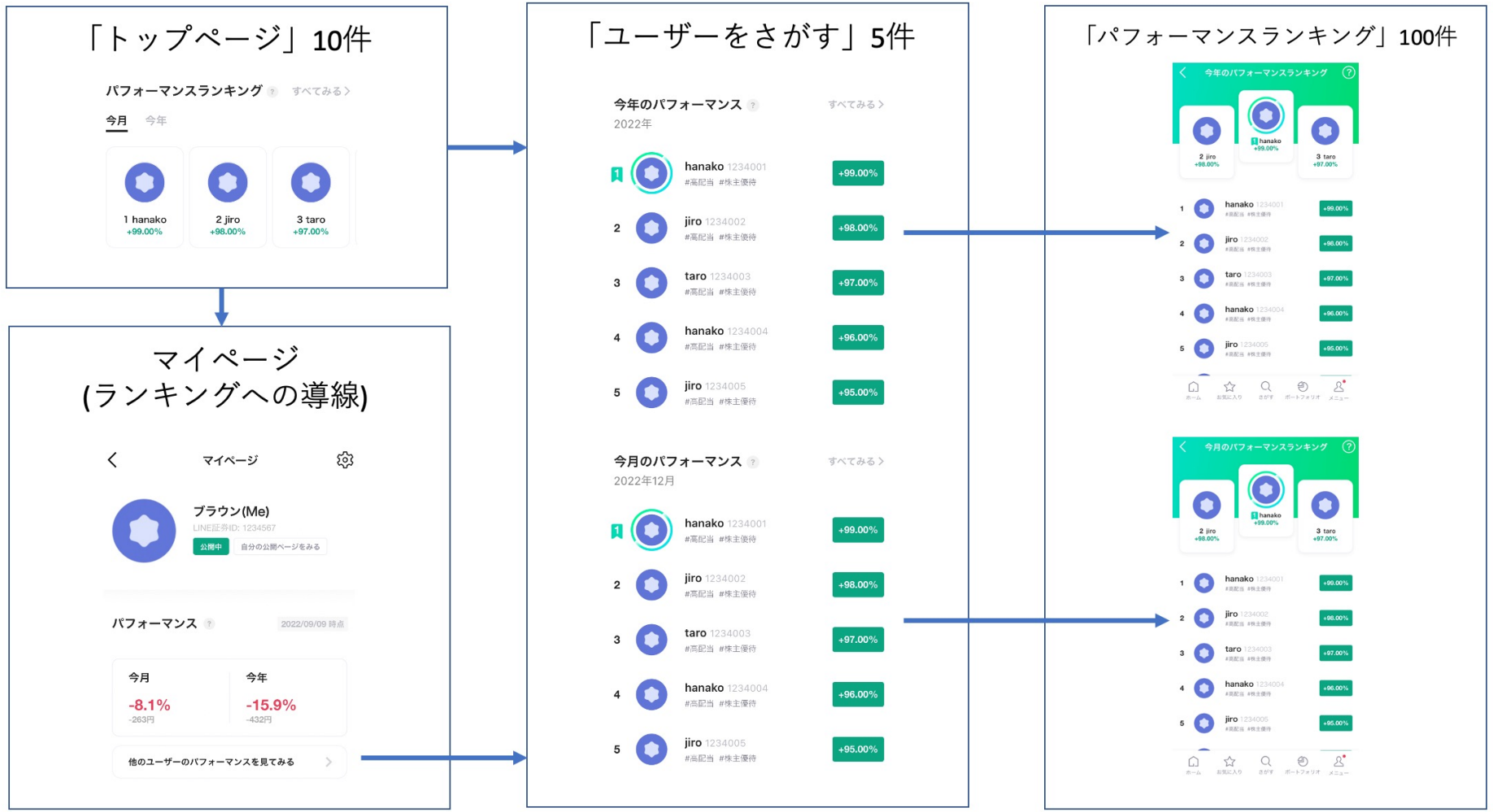
```
const [apiResult, setApiResult] = useRecoilState(nowTimeApiState)
useEffect(() => {
  if (apiResult.status !== 'Succeed') {
    fetch('/api/nowtime')
      .then((res) => res.json())
      .then((res) => setApiResult({ status: 'Succeed', data: res }))
      .catch(() => setApiResult({ status: 'Failed' }))
  }
}, [])
```

実際にプロダクトで使ってみた

LINE証券の
「パフォーマンスランキング」を
キャッシュ化した



実際にプロダクトで使ってみた



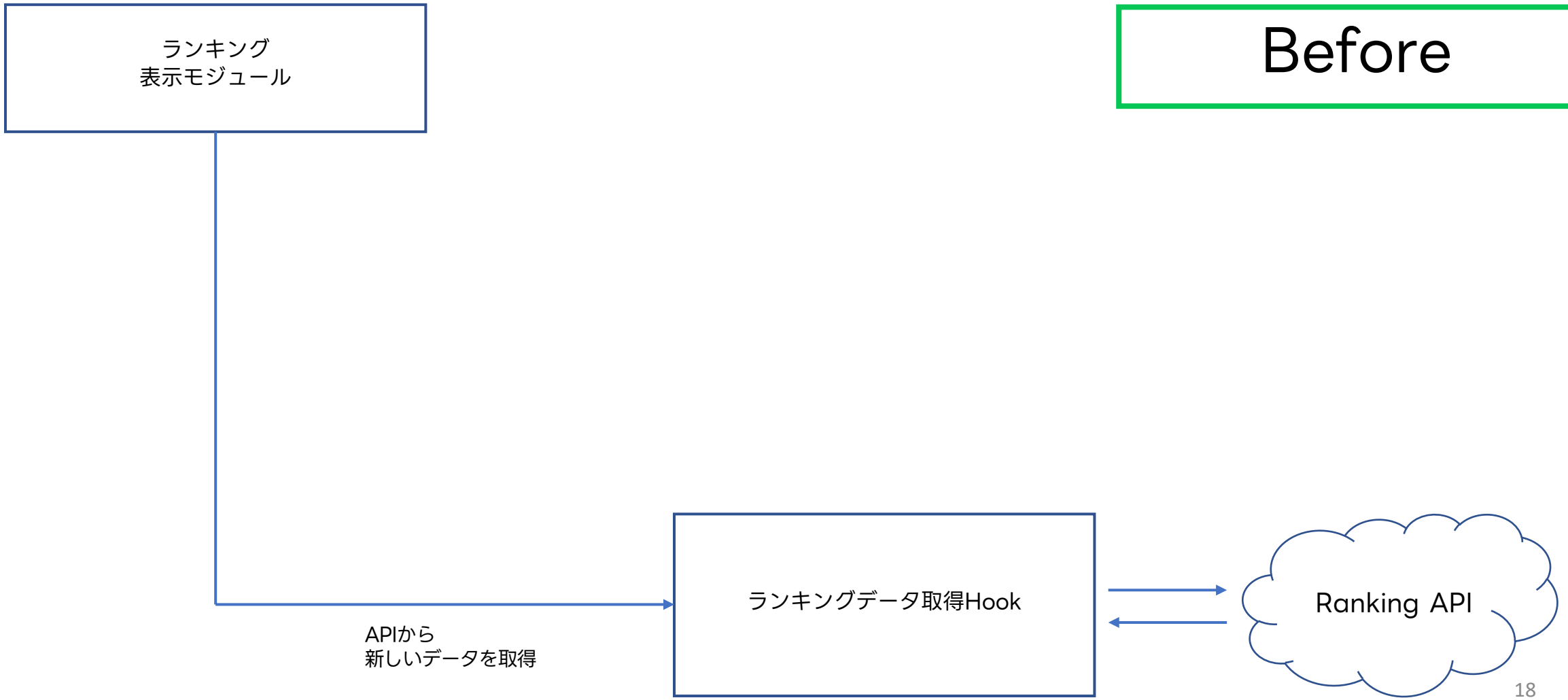
実際にプロダクトで使ってみた

キャッシュ化するために必要だった条件

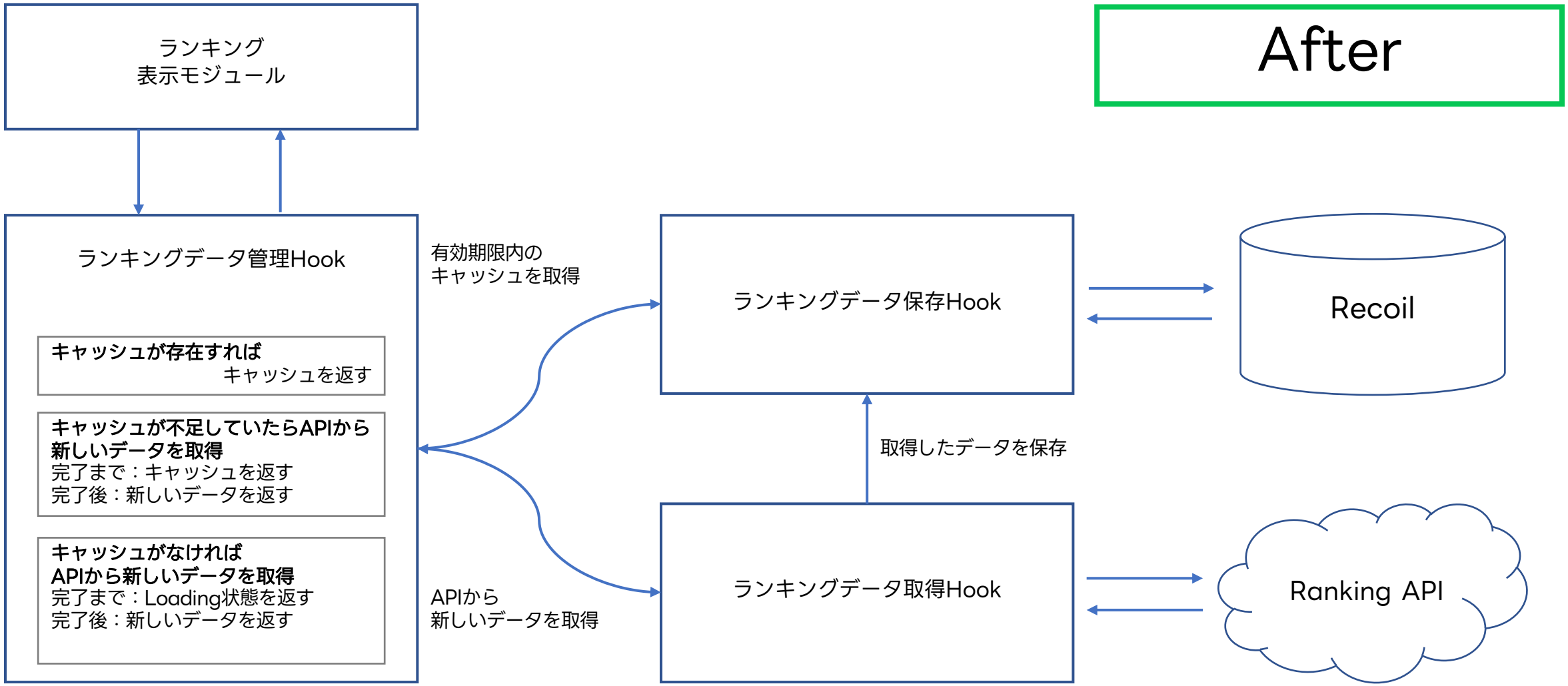
- ページによって必要なデータ件数が違う
- ランキングは更新されるので、キャッシュの賞味期限を設定する必要がある

実際にプロダクトで使ってみた

Before

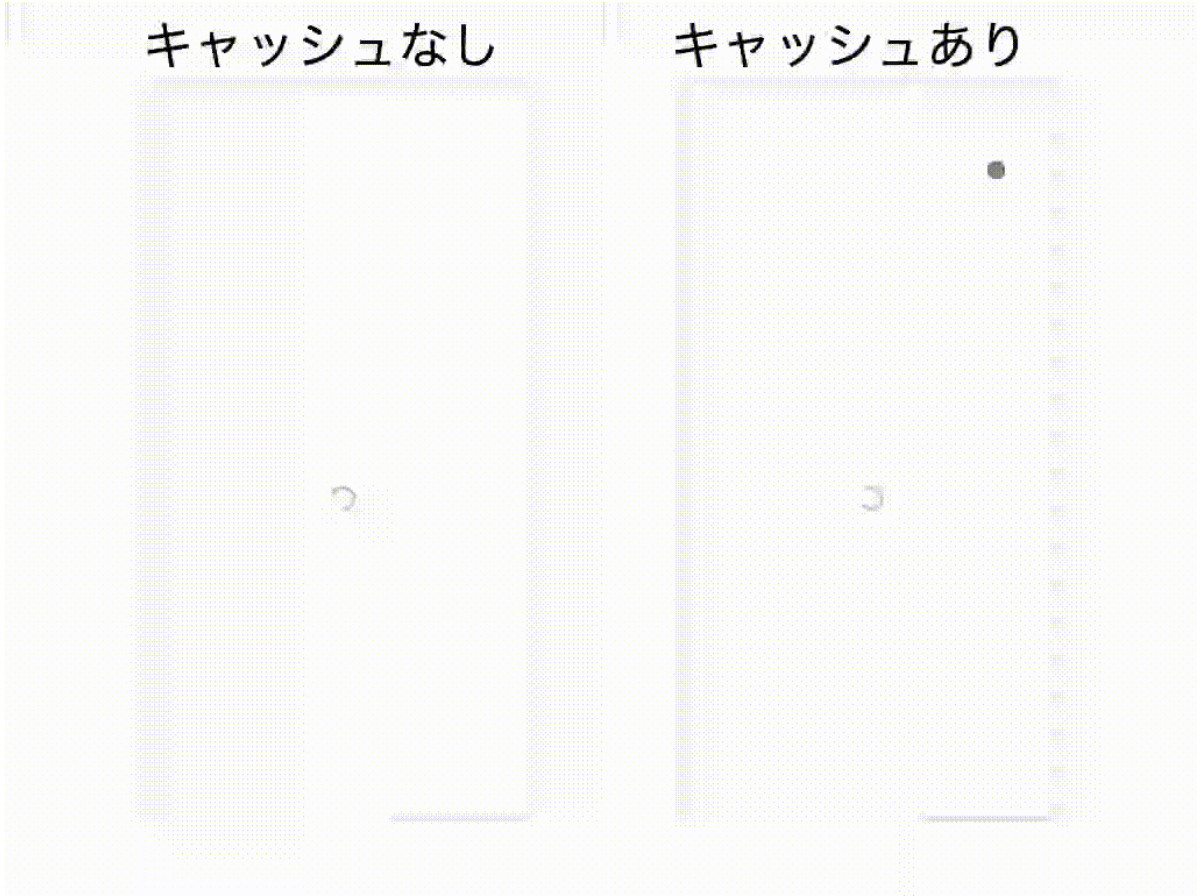


実際にプロダクトで使ってみた



結果

FirstViewの読み込み待ちが大幅に削減された！



6x slowdown, Fast 3G

宣伝

詳しい話はインターンレポートで解説しています

Recoilを使った「パフォーマンスランキング」のキャッシュ化
bit.ly/3BgcQm8

まとめ

- キャッシュの挙動を自分で細かく制御したいならRecoilがお手軽
- ニーズによってはSWRなど他のライブラリが適役な時もある
  - オプションだけで色々実現できるライブラリも多い

ご清聴ありがとうございました