HF ドップラー観測データの利活用を目的とする Web アプリケーションのフロントエンド設計

5421 中嶋 柊（指導教員：才田 聡子）

HF Doppler Observation Data Utilization Front-End Design of Web Application

概要： ここに概要を書く. □□□

キーワード Web アプリケーション フロントエンド ページレンダリング TypeScript Next.js

1. はじめに

電離圏研究を目的とする短波ドップラー (HFD) の観測データは, 自然科学分野の研究のためオープンデータとして公開され自由な使用が許可されている.
現状, 当データの効率的な再利用を推進するためデータのグラフィカルな表示を行う数件の web アプリケーションが開発されているが, その多くが開発運用やユーザーエクスペリエンス (以下 UX) 等の観点での問題点を抱えている.
本研究では既存のアプリケーションからの新規アプリケーション作成に伴うデータ取得とページのルーティングを通した UX 向上を行う.

2. 使用技術

2.1 Next.js について

Next.js とは Vercel 社が提供する Web フレームワークである. JavaScript の UI ライブラリの一つである React.js がベースとなっており, その最大の特徴の一つが多様なページレンダリング手法のサポートである. このフレームワークは, 開発者がアプリケーションの特定のニーズに応じて最適なレンダリング戦略を選択できるように設計されている.
Next.js のレンダリングオプションには, 一般的なクライアントサイドレンダリング (CSR) の他に, サーバサイドレンダリング（SSR）, スタティックジェネレーション（SG)、およびインクリメンタル静的再生成（ISR）が含まれる.
2024 年現在, バージョン 12 系以前で用いられていたクライアントサイドでのレンダリングをデフォルトとした Page Router とバージョン 13 系で追加されたサーバーサイドでのレンダリングをデフォルトとした App Router の 2 種類の Routing 手法が存在する. 本研究では Page Router を用いることとした.

2.2 ページレンダリング手法

2.2.1 ページレンダリングとは

Web アプリケーションにおいてページレンダリングとは, ウェブサーバーまたはブラウザが HTML/CSS/JavaScript などのコードを解釈し, ユーザーのデバイス上で視覚的なページとして表示するプロセスのことである. プロセス内の各ステップを実行するタイミング等により分類されており, 適切な手法を選定することによりサイトの高速化やユーザ体験の改良, SEO の改善等が期待できる. 次に代表的なレンダリング手法とそのメリット・デメリットについて記す.

2.2.2 SSR

SSR（サーバーサイドレンダリング）はウェブページのレンダリングをサーバー側で行う手法である. この手法ではユーザーのリクエストに応じてサーバーが HTML を生成し, 完成したページがクライアントに送信される.
SSR はリクエストが起きたタイミングでクライアントサイドと比較して高速なサーバーサイドで HTML が生成されるため, 初回ページロードの速度を向上させ SEO 対策に有効である. 他にも完成された HTML が送信されるため SEO に強い点や, クライアントサイドの JavaScript 実行環境に依存しにくい点がメリットである. しかし, サーバー負荷が大きい点やページ間移動にリロードが伴う点, 後述する SG や ISR と比較するとコンテンツの表示に時間が掛かる点等がデメリットとして存在する.

2.2.3 SG

SG（スタティックジェネレーション）は, ビルド時に静的な HTML ファイルを生成しておく手法である. このアプローチではビルド時にすべてのページが静的な HTML として生成され, リクエストに応じてその HTML が提供される.
ビルド時に HTML が生成されレスポンスの際にはそれを提供するだけのため非常に高速であることがメリットとしてあげられるが, ビルド後の変更が困難なため動的なページ生成には向かないというデメリットが存在する.

2.2.4 ISR

ISR（インクリメンタルスタティックリジェネレーション）は, 静的サイトの生成とサーバーサイドレンダリングを組み合わせた手法である. ISR では, ビルド時に一部のページのみを静的に生成し残りのページはユーザーのリクエストに応じて生成されキャッシュされる. これにより, サイトのビルド時間を短縮しつつ動的なコンテンツの提供が可能となる.

静的生成と動的生成のバランスが取れパフォーマンスとコンテンツの更新の柔軟性が両立されているメリットがあるが, 再生成後初回リクエスト時のオーバーヘッドが大きく読み込みに時間がかかってしまう場合があるというデメリットが存在する.

2.2.5 Next.js におけるページレンダリング

Next.js のページレンダリングは主にページコンポーネントのライフサイクルとデータ取得方法に依存する特徴があり, それらのレンダリング手法をページごとに異なる方法で適用できる柔軟性にも優れている.
例として一部のページは SSR を使用して動的にデータを取得を行い、他のページは SG を使用して静的に生成することも可能である. このような柔軟性により開発者はパフォーマンス, SEO, ユーザーエクスペリエンスのバランスを取りつつ, アプリケーションの特定の要件に合わせて最適なレンダリング戦略を選択できる. これにより, 開発者はより高速で対話的なウェブアプリケーションを構築することができ, 最終的にはユーザーに優れた体験を提供することが可能となる.

2.3 スクレイピングについて

このアプリケーションは, 各観測所にて観測されたデータを素のディレクトリ構造のまま公開されている既存のアプリケーションからデータをスクレイピングしてきて再描画を行うものである. スクレイピングとは web アプリケーションから HTML データをクローラによって取得し、必要とされる情報を抽出するする技術のことである.
スクレイピングの際には取得元のサイトの規約に乗っ取る義務がある. 今回の場合は取得先のデータが OSS 化されていることと学術目的として取得先の団体からの許可を得ていることから, 過剰な負荷がかからない範囲内での利用が可能であると判断した.

3. アプリケーション設計と実装方法

3.1 実装の要件

本研究ではアプリケーション実装のうち外部データの取得とページの描画を主体としている. そのため UX 向上を目的とした, データフェッチからの画面描画速度の高速化を目標に設定した.
今回作成するアプリケーションのうち, web スクレイピングを伴うページは大きく分けて以下の 2 つである.

- 波形データの検索ページ
- 波形の描画ページ

これらのページの実装を行い、ビルド時間とスクレイピングのリクエスト数の 2 軸から最適化を行うものとした.

3.2 スクレイピングに用いた関数

今回実装したスクレイピング関数は大きく 2 つあり, 1 つは指定した日時・観測所の周波数の CSV データを JSON 化して取得するもの, もう 1 つは存在するデータの一覧を取得するものである. スクレイピング先のアプリケーションでは観測所・年度・日時の順にネストされたパスから周波数のデータを取得可能である. 1 つめの関数では URL で具体的なデータの所在ページを指定した上でスクレイピングした. 2 つめの関数では存在するページの一覧化のため観測所のひとつ上の階層から子要素となるページを再帰的に全探索する形でスクレイピングし整形を行った.

3.3 レンダリング設計

次にレンダリング設計について述べる. 以下の 2 パターンの設計で実装を行い、ビルド時間を比較し考察することとした.

1. ISR を用いる方法
2. SG と SSR を用いる方法

3.3.1 ISR を用いる方法

1 つ目の手法は ISR を用いた実装である. この手法はビルドの際に一括でスクレイピング処理を完結させて静的なサイトを生成するものである.
実装方法は getStaticPaths と呼ばれる関数内でディレクトリ一覧をスクレイピングし静的にビルドするページを確定させた上で, getStaticProps という関数でそれら一つ一つのページの波形データを取得してビルドを行う.
また ISR 技術を用いて再ビルドのタイミングを制御することにより, 一定期間ごとに新しく追加されたデータも取得できるようにさせることができるようにする.

3.3.2 SG と SSR を用いる方法

2 つ目の手法は SG と SSR を併用させる方法である. この手法ではディレクトリの取得と波形データの取得をそれぞれ SG・SSR をもちいて分離させる方法である.
実装方法は getStaticProps を用いて検索ページに存在する URL を静的に受け渡し, getServerSideProps を用いて各ページでリクエスト時に波形データを返すというものである.

3.4 実行環境

今回の実行環境とした PC を表　に示す.
また使用したパッケージのバージョンを表　に示す.

表 1　実行環境

名称	MacBookPro
CPU	Apple M1 Max
メモリ	32GB
コマンドシェル	zsh (v1.85.1)

表 2　パッケージのバージョン

名称	バージョン
Node.js	18.12.0
Next.js	13.0.6
TypeScript	4.9.4

4. 実装結果

4.1 スクレイピング関数の実行時間

4.2 静的ビルドの実行時間の比較

5. 考察