令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」

ビッグデータ技術応用教材資料 2

本ビッグデータ技術応用教材資料は、文部科学省の教育政策推進事業委託費による委託事業として、一般社団法人全国専門学校情報教育協会が実施した令和 7 年度「地方やデジタル分野における専修学校理系転換等推進事業」の成果物です。

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

目　次

ビッグデータ技術応用 3
はじめに 3

ビッグデータ技術応用__実習１ 9
導入編 9
Part1 15
Part2 24
Part3 34
Part4 43

ビッグデータ技術応用_実習２ 51
導入編 51
Part1 56
Part2 66
Part3 75
Part4 85

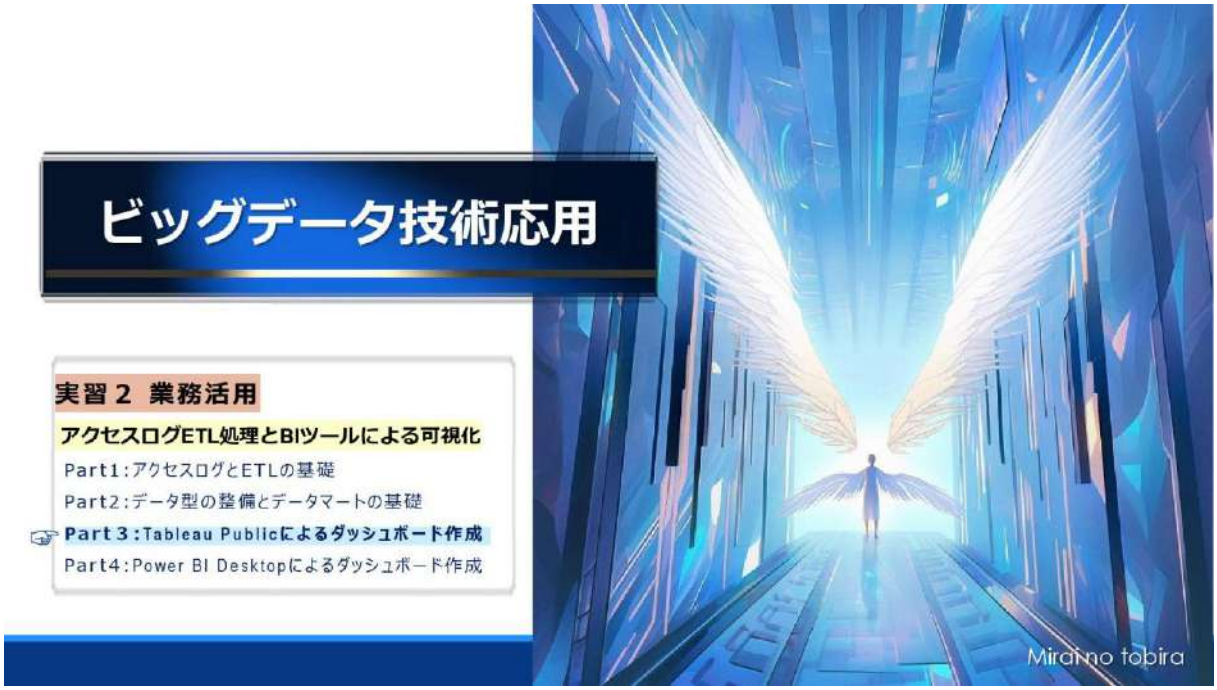
ビッグデータ技術応用_実習３ 94
導入編 94
Part1 99
Part2 107
Part3 116
Part4 124

ビッグデータ技術応用_実習まとめ編 132

ビッグデータ技術応用_実習２

Part3


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１　分析実践
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
実習３　大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎
Part２：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part４：Power BI Desktop によるダッシュボード作成



Part３：Tableau Public によるダッシュボード作成

実習２ 業務活用

データを『見える化』する力を身につける

Part 3 :Tableau Public によるダッシュボード作成 の達成目標

実習 2 業務活用

- ✓ Tableau Publicの基本操作（データ接続、画面構成）を理解する
- ✓ CSVデータをTableauに接続し、データ型を設定できる
- ✓ ディメンションとメジャーを使ってグラフを作成できる
- ✓ 複数のグラフを組み合わせてダッシュボードを構築できる
- ✓ ダッシュボードにインタラクション（フィルター）を設定できる
- ✓ 作成した成果をWebに公開できる

Tableau Publicとは？

実習 2 業務活用 Part 3

BIツール：Tableau Public

- ▶ 世界中で広く使われているBI (Business Intelligence) ツールの一つ
- ▶ ドラッグ＆ドロップで簡単に高機能なグラフやダッシュボードが作成可能
- ▶ **Public版は無料**で利用できるが、作成物はWebに公開される
- ▶ データからインサイトを得るための強力な武器

なぜBIツール？
Python (matplotlib/seaborn) でもグラフは作れますが、BIツールはより
手軽に（コード不要）／**インタラクティブに**（操作できる）／**美しく** データを可視化できます

準備：ツールとデータ

実習2 業務活用 Part3

タスク1：準備するもの

① **Tableau Public Desktop:**

- ▶未インストールの場合は公式サイトからダウンロード＆インストール
- ▶Tableau Publicアカウントでログイン

② **データファイル:**

- ▶Part 2で作成した access_log_mart.csv

データ接続と型設定

実習2 業務活用 Part3

タスク2：データをTableauに接続

手順：

1. Tableau Public 起動 →「接続」→「テキストファイル」
2. access_log_mart.csv を選択
3. データソース画面で列とデータ型を確認
4. **重要：**
   ip 列のデータ型アイコンをクリック →「地理的役割」→「IPアドレス」を設定

ワークシート画面

実習２ 業務活用 Part３

グラフ作成の舞台：ワークシート

データ接続後、「シート1」をクリックしてワークシート画面へ

グラフ作成① 折れ線

実習２ 業務活用 Part３

タスク3-2：時間帯別アクセス（折れ線グラフ）

手順：

1. **メジャー「レコード数」**を**「行」**シェルフへドラッグ
2. **ディメンション「Hour」**を**「列」**シェルフへドラッグ
3. → 自動で折れ線グラフが完成

グラフ作成② 棒グラフ

実習2 業務活用 Part3

タスク3-3：曜日別アクセス（棒グラフ）

手順：

1. 新しいシートを作成
2. **メジャー「レコード数」**を**「行」**シェルフへ
3. **ディメンション「Dayname」**を**「列」**シェルフへ
4. → 自動で棒グラフが完成！
5. （オプション）「Dayname」の並び順を調整

グラフ作成③ 地図

実習2 業務活用 Part3

タスク3-4：アクセス元マップ

手順：

1. 新しいシートを作成
2. **ディメンション「Ip」（地球儀アイコン）**をダブルクリック
3. → 自動で地図が表示
4. **メジャー「レコード数」**をマークカードの**「サイズ」**へドラッグ
5. → アクセス数に応じて点のサイズが変わる

ダッシュボード作成

実習２ 業務活用 Part３

タスク4：ダッシュボードで統合

作成した３つのグラフ（シート）を1画面にまとめます

手順：

1. 新しいダッシュボードを作成（田アイコン）
2. 左側のシート一覧から、各シートを右側の領域へドラッグ＆ドロップして配置
3. サイズやレイアウトを調整

インタラクション設定

実習２ 業務活用 Part３

タスク4-3：ダッシュボードを『動かす』

グラフ同士を連動させます

手順：

1. ダッシュボード上の「アクセス元マップ」シートを選択
2. 右上に表示される「フィルターとして使用」（漏斗マーク）をクリック
3. → これで地図をクリックすると他のグラフが連動する

🎯 インタラクションの価値
見る人が自由にデータを深掘りできるようになり、分析の幅が広がります

保存と公開

実習２ 業務活用 Part３

タスク5：成果を世界に公開

Tableau Public では、作成したワークブックは
サーバーに保存・公開します

手順：

1. 「ファイル」→「Tableau Public に保存」
2. ログイン → 名前を付けて「保存」
3. → 自動的にWebブラウザで公開ページが開く

⚠ 注意
公開されたURLは誰でもアクセス可能です
個人情報や機密情報が含まれていないことを確認してください

実習２ Part３のまとめ

実習２ 業務活用 Part３

BIツールの代表格、Tableau Public の基本操作をマスターしました

- ✅ データ接続とデータ型の重要性を理解した
- ✅ ドラッグ＆ドロップで基本的なグラフを作成できた
- ✅ 複数のグラフを組み合わせてダッシュボードを構築できた
- ✅ フィルター機能でインタラティブな分析を体験した
- ✅ 作成した成果をWebで共有する方法を学んだ

NEXT

Part４：Power BI Desktop によるダッシュボード作成

実習２ 業務活用

実習２Part４では、
Part 2で作成した access_log_mart.csv を使って
BIPower BI Desktop でダッシュボードを作成します

NEXT



ビッグデータ技術応用

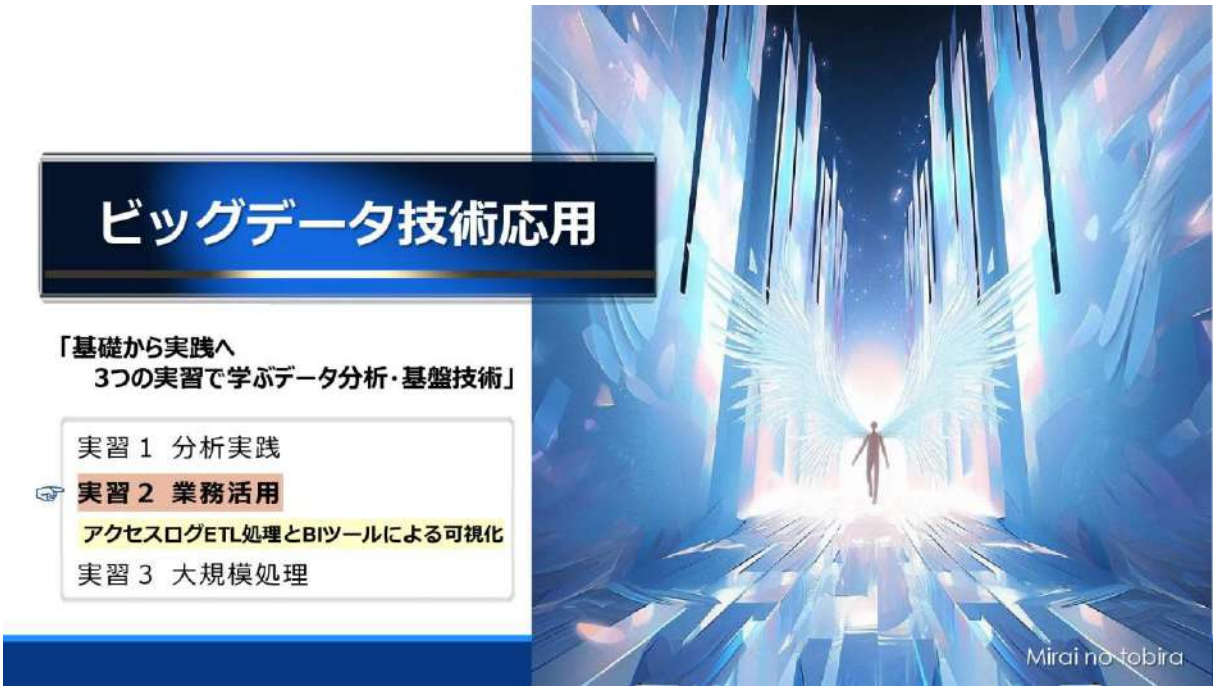
実習２ 業務活用

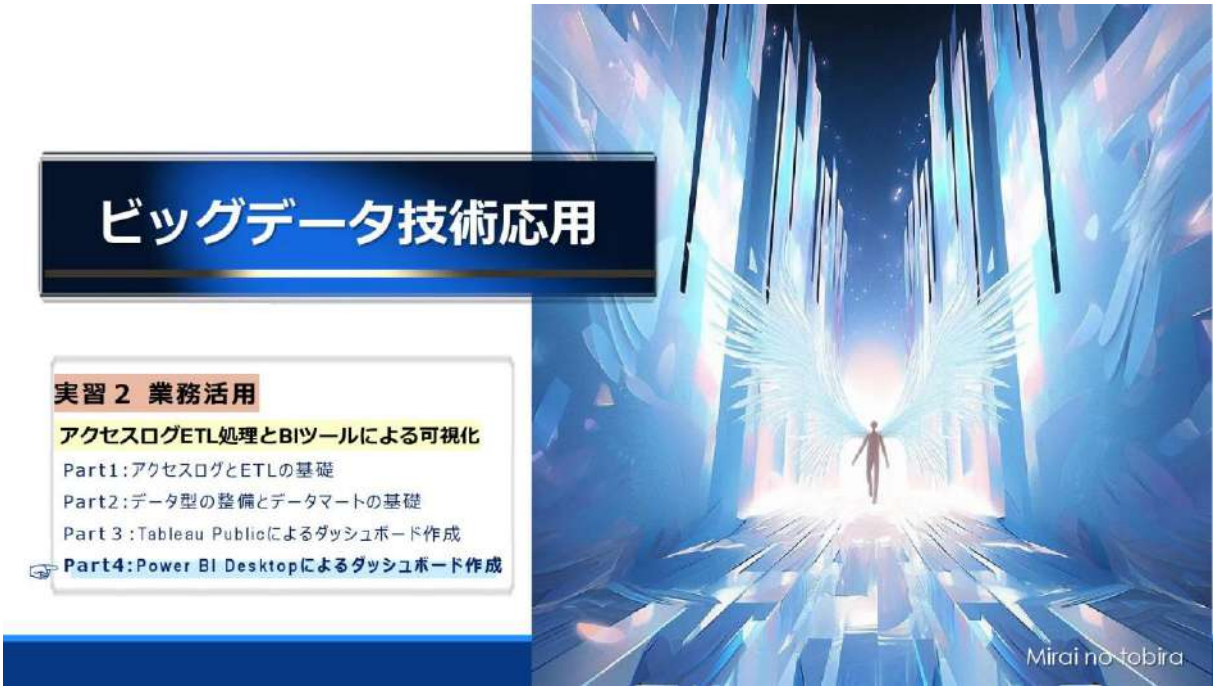
アクセスログETL処理とBIツールによる可視化

Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成

ビッグデータ技術応用_実習２

Part4


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１　分析実践
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
実習３　大規模処理
Mirai no tobira


ビッグデータ技術応用
実習２　業務活用
アクセスログETL処理とBIツールによる可視化
Part1：アクセスログとETLの基礎
Part2：データ型の整備とデータマートの基礎
Part３：Tableau Publicによるダッシュボード作成
Part4：Power BI Desktopによるダッシュボード作成
Mirai no tobira

実習２ 業務活用

ビッグデータ技術応用

実習１ 分析実践 → **実習２ 業務活用**（アクセスログETL処理とBIツールによる可視化） → 実習３ 大規模処理

Part１：アクセスログとETLの基礎
Part２：データ型の整備とデータマートの基礎
Part３：Tableau Public によるダッシュボード作成
Part４：Power BI Desktop によるダッシュボード作成



Part４：Power BI Desktop によるダッシュボード作成

実習２ 業務活用

もう一つのBIツール：Tableauとの違いは？



Part4:Power BI Desktopによるダッシュボード作成 の達成目標

実習２ 業務活用

- ✓ Power BI Desktopの基本操作（データ取得、画面構成）を理解する
- ✓ CSVデータをPower BIに取得し、Power Queryでデータ型を設定できる
- ✓ レポートビューで基本的な視覚化（グラフ）を作成できる
- ✓ 複数のグラフを配置してレポート（ダッシュボード）を構築できる
- ✓ グラフ間のインタラクションを確認し、レポートを保存できる

Power BI Desktopとは？

実習２ 業務活用 Part４

Microsoft の BIツール : Power BI Desktop

- ▶ Microsoftが提供する無料のBIツール（Windows専用）
- ▶ Excelライクな操作感＋高機能なデータ加工（Power Query）
- ▶ 豊富なグラフ種類と、柔軟なレポートデザインが可能
- ▶ Office 365 との連携が強力
- ▶ Web公開には Power BI Service（有償の場合あり）が必要

Tableau Publicとの主な違い

- ▶ **OS** ：Power BI DesktopはWindowsのみ、Tableau PublicはWin/Mac対応
- ▶ **公開** ：Power BIはローカル保存可、Tableau PublicはWeb公開必須
- ▶ **操作感** ：Power BIはExcel寄り、Tableauは独自のデザイン思想

準備：ツールとデータ

実習2 業務活用 Part4

タスク1：準備するもの

① **Power BI Desktop:**

▶未インストールの場合は Microsoft Store等からインストール

▶Windows PCが必要

② **データファイル:**

▶Part 2で作成した access_log_mart.csv

データ取得と型設定

実習2 業務活用 Part4

タスク2：データを Power BI に取得

手順：

1. Power BI起動 →「データを取得」→「テキスト/CSV」
2. access_log_mart.csv を選択 →「読み込み」
3. 右側「フィールド」ペインに列が表示される
4. 「ホーム」タブ →「データの変換」で Power Query エディターを開く
5. 各列のデータ型を確認・修正（例: timestampを日付/時刻へ）
6. 「閉じて適用」

レポートビュー画面

実習２ 業務活用 Part４

グラフ作成の舞台：レポートビュー

Power Query を閉じると、レポートビューに戻ります

視覚化の作成

実習２ 業務活用 Part４

タスク3：グラフ（視覚化）を作成する

基本手順：

1.「視覚化」ペインでグラフ種類を選択

2.「フィールド」ペインから項目を、視覚化ペインの「軸」「値」等へドラッグ

3.（必要なら）「値」の集計方法を「カウント」等に変更

作成するグラフ：

▶ **時間帯別アクセス：** 折れ線グラフ（X軸: hour, Y軸: timestampのカウント）

▶ **曜日別アクセス：** 集合縦棒グラフ（X軸: dayname, Y軸: timestampのカウント）

→ X軸の並び順を「dayofweek」基準で並べ替え

・アクセス元マップ： マップ（場所: ip, バブルサイズ: timestampのカウント）

→ データビューでip列のデータカテゴリを「IPアドレス」に設定

レポートの作成

実習２ 業務活用 Part４

タスク4：レポート（ダッシュボード）を構築

作成した３つのグラフをレポートキャンバス上に配置し
レイアウトを整えます

インタラクションの確認

実習２ 業務活用 Part４

グラフ同士の連携を確認

Power BI では、デフォルトで同じページ上のグラフが連動します

試してみよう：

- **曜日グラフの「Monday」をクリック**
  → 他のグラフが月曜データに絞り込まれる
- **地図上の大きな円をクリック**
  → 他のグラフがその地域のデータに絞り込まれる

Tableauとの比較
Tableauでは明示的に「フィルターとして使用」を設定しましたが
Power BIはデフォルトで連動するのが特徴です

ファイルの保存

実習２ 業務活用 Part４

タスク5：レポートを保存する

作成したレポートは .pbix ファイルとしてローカルに保存します

手順：

1. 「ファイル」→「名前を付けて保存」
2. ファイル名（例: AccessLogReport.pbix）を入力して「保存」

成果物
この .pbix ファイルが Part４の成果物となります

実習２のまとめ

実習２ 業務活用

BIツールの代表格2つのBIツールを体験しました
Part４では Power BI Desktop の基本操作を学び、
実習２全体を通して以下のスキルを習得しました

- ✅ **ETLプロセス（ログ解析→データマート作成）の実装**
- ✅ **BIツール（Tableau & Power BI）でのデータ接続・可視化**
- ✅ **インタラクティブなダッシュボード（レポート）の構築**

NEXT

実習３：Sparkによる大規模データ処理

ビッグデータ技術応用

実習３では
実習１,２では扱いきれなかった
PCのメモリを超えるような「巨大データ」
を扱う技術に挑戦します

NEXT

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析･基盤技術」

実習１ 分析実践
実習２ 業務活用
☞ 実習３ 大規模処理
Sparkによる大規模データ処理



ビッグデータ技術応用_実習 3

導入編


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira

実習構成

ビッグデータ技術応用

実習１ 分析実践
Python による
顧客データ分析と
レポーティング

実習２ 業務活用
アクセスログETL処理と
BIツールによる可視化

実習３ 大規模処理
Spark による
大規模データ処理

ビッグデータ技術基礎		ビッグデータ技術応用		
第1章 ビッグデータとその活用	→	実習１		
第2章 データ分析基礎	→	実習１		
第3章 スクリプト言語による分析	→	実習１	実習２	
第4章 ツールとモニタリング	→		実習２	
第5章 列指向ストレージによる高速化	→			**実習３**
第6章 データマートの基本構造	→		実習２	
第7章 大規模分散処理	→			**実習３**

実習の背景

ビッグデータ技術応用

pandas ：大きな弱点 → データをすべてPCのメモリに読み込んで処理する

→ メモリ不足でエラー、処理が異常に遅くなる

数GBを超えるデータ → pandasだけで扱うのは、現実的に非常に困難

Apache Spark： 分散処理のためのフレームワーク

→ たくさんのコンピュータ（ノード）で手分けして同時に処理

複数のマシン（or1台のマシンの複数のCPUコア）を使って高速に処理可能

実習３ 大規模処理

ビッグデータ技術応用

実習１ 分析実践 → 実習２ 業務活用 → **実習３ 大規模処理** Spark による 大規模データ処理



Part１：分散処理の必要性と Spark 環境
Part２：Spark DataFrame の基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）

実習ステップ

ビッグデータ技術応用

〔実習３の４つの Part〕

Part１： pandas の限界を体感し、Spark 環境（Google Colaboratory）を準備
Part２： Spark を使って大規模な CSV や Parquetファイルを読み込み、DataFrameの基本的な操作を学ぶ
Part３： 同じ集計処理を pandas と Spark で実行し、処理時間を比較する
Part４： Spark でより実践的な集計を行い、結果を pandas DataFrame に変換して可視化する

ゴール：Spark の基本的な使い方をマスターし、その効果を実感すること
実行環境：実習１,２と同様 Google Colaboratory を推奨
データ：Part１のワークブック内で生成する数GB規模の擬似センサーデータ

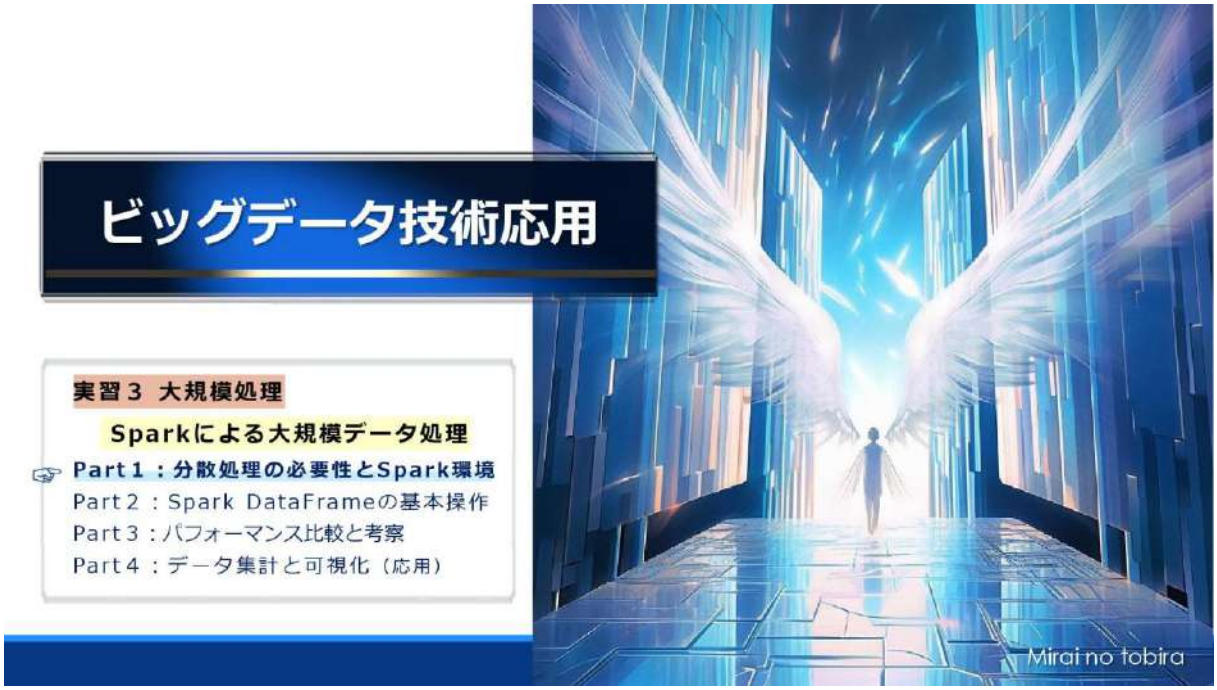

ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Mirai no tobira


NEXT
ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira

ビッグデータ技術応用_実習３

Part1


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part１：分散処理の必要性とSpark環境

実習３ 大規模処理

なぜ Spark が必要？ Pandas の限界を知る

Part１:分散処理の必要性とSpark環境 の達成目標

実習３ 大規模処理 Part１

- ✓ Pandas が大規模データ処理で限界に達する理由を体感する
- ✓ 分散処理の基本的な考え方を理解する
- ✓ Apache Spark の役割を知る
- ✓ Google Colab上で PySpark 環境をセットアップできる
- ✓ Spark アプリケーションの入口 (SparkSession) を作成できる
- ✓ 分析用の大規模データを生成できる

Pandas の限界

実習 3 大規模処理 Part 1

タスク1：Pandas の限界を体験

Pandas は便利ですが、データを「全量メモリに読み込む」のが基本です

もし データサイズ > PCのメモリ容量 なら？

`MemoryError!` または、処理が異常に遅くなります（スワップ発生）

解決策：分散処理

実習 3 大規模処理 Part 1

解決策：データを『分けて』処理する

メモリに乗り切らないなら、データを小さく分割し、
複数のCPUコアやマシンで手分けして処理すれば良い！

→ これが**分散処理**の考え方です

メリット
- メモリ制限を超えるデータを扱える
- 並列処理により高速化が期待できる

Apache Spark とは？

実習３ 大規模処理 Part１

分散処理の実行エンジン：Apache Spark

- 大規模データ向けの高速な**分散処理フレームワーク**
- メモリ上での処理が得意で、Hadoop MapReduceより高速
- Python (PySpark), Scala, Java, R など多様な言語に対応
- DataFrame API により、pandasに似た感覚で分散データを操作可能

Spark 環境セットアップ (Colab)

実習３ 大規模処理 Part１

タスク3：ColabでSparkを使う準備

Google Colab上で PySpark を使うための準備は簡単です

ステップ1：インストール

```
!pip install pyspark -q
```

ステップ2：SparkSession作成

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("MyApp").getOrCreate()
```

SparkSession
これがSparkを使うための「入口」 ノートブックの最初に必ず実行します

SparkSession の確認

実習３ 大規模処理 Part１

Sparkの準備完了

SparkSessionが作成されると、spark という変数が使えるようになります

```
spark # 変数の中身を表示
```

出力例：

```
<pyspark.sql.session.SparkSession object at 0x...>
SparkSession - in-memory
...
Spark UI
...
```

このような情報が表示されれば、Sparkを使う準備はOKです

大規模データの生成

実習３ 大規模処理 Part１

タスク4：『メモリ不足』を起こすデータを作る

Sparkの効果を実感するために、
pandasでは扱いにくいサイズのデータ（数GB）を生成します

```
def generate_sensor_data(num_rows, file_path):
# ... （データ生成ロジック） ...
df.to_csv(file_path, index=False)

# 例: 5000万行 （約2GB） のデータを生成
NUM_ROWS = 50_000_000
CSV_PATH = 'large_sensor_data.csv'
generate_sensor_data(NUM_ROWS, CSV_PATH)
```

⚠ 注意
- データ生成には時間がかかります（数分）
- Colabのディスク容量に注意してください

生成データの確認

実習３ 大規模処理 Part１

巨大ファイルの誕生

データ生成後、ファイルサイズを確認します

```
!ls -lh large_sensor_data.csv
```

出力例：

```
-rw-r--r-- 1 root root 2.1G Oct 20 19:30 large_sensor_data.csv
```

→ 数GBのCSVファイルができました。これを pandasで読み込むのは大変そうです

実習３ Part１のまとめ

実習３ 大規模処理 Part１

大規模データ処理への第一歩を踏み出しました

- ✅ Pandasのメモリ限界を体験した
- ✅ 分散処理と Spark の必要性を理解した
- ✅ Colab上 に PySpark 環境を準備できた
- ✅ 分析対象となる大規模CSVデータを生成した

Part１：分散処理の必要性とSpark環境　ビッグデータ技術応用 実習３

NEXT

Part２：Spark DataFrame の基本操作

実習３ 大規模処理

実習３ Part２では、
Spark を使って巨大CSVファイルを読み込み
DataFrame として操作する方法を学びます

NEXT

ビッグデータ技術応用

実習３ 大規模処理

Sparkによる大規模データ処理

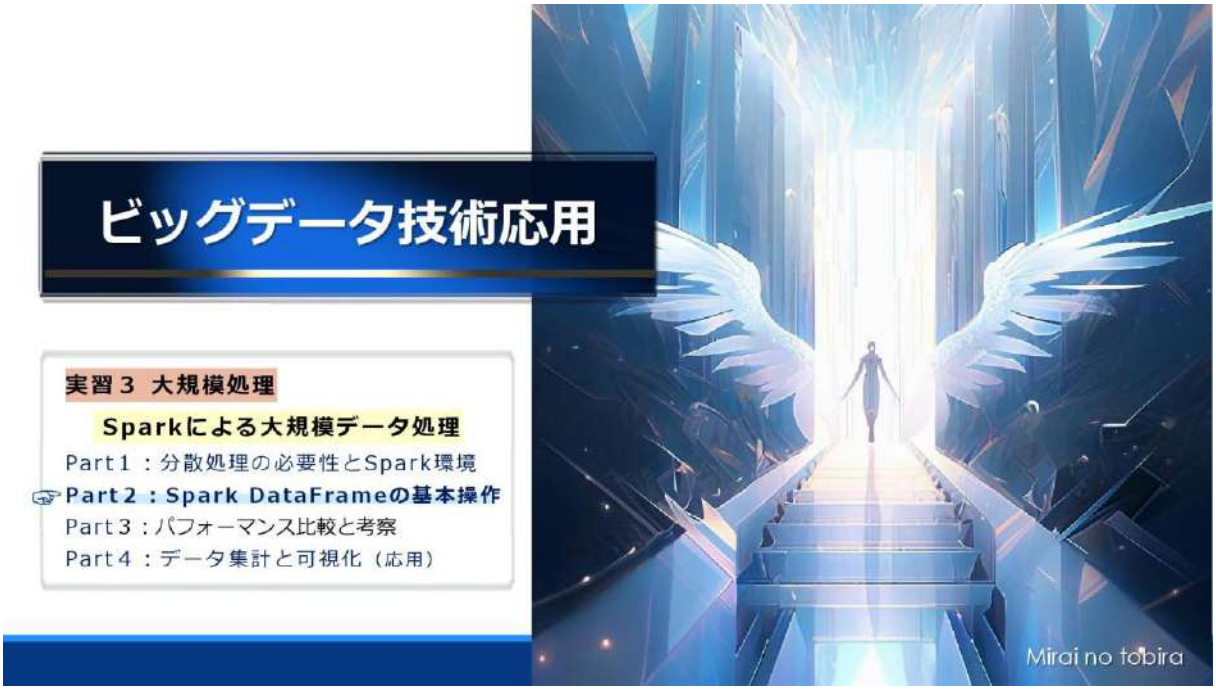
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）



ビッグデータ技術応用_実習３

Part2


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part 2 ：Spark DataFrame の基本操作

実習 3 大規模処理

巨大データを自在に操る第一歩

Part 2 : Spark DataFrame の基本操作 の達成目標

実習 3 大規模処理 Part 2

- ✓ Spark で巨大CSVファイルを DataFrame として読み込む
- ✓ DataFrame のスキーマ（データ型）を確認する
- ✓ 基本的なAction (show, count) と Transformation (select, filter, groupBy) を使う
- ✓ Spark の「遅延評価」とは何かを理解する
- ✓ データを効率的な Parquet 形式で保存・読み込みする

巨大CSVの読み込み

実習３ 大規模処理 Part２

タスク2：Spark で巨大CSVを読み込む

spark.read.csv() を使います

```
sensor_df = spark.read.csv(
'large_sensor_data.csv',
header=True, # ヘッダーあり
inferSchema=True # スキーマ自動推論
)
sensor_df.show(5) # Action実行 （ここで初めて読み込みが走る）
```

⚠ 注意点
- `inferSchema=True` は便利だが、巨大ファイルでは推論に時間がかかる
- `.show()` のようなActionを実行するまで、実際の読み込みは開始されない（**遅延評価**）
- 読み込み（特に初回）には時間がかかる（数分）

スキーマ確認 & カウント

実習３ 大規模処理 Part２

データの中身を確認：スキーマと行数

スキーマ確認 (.printSchema()):

```
sensor_df.printSchema()
```

```
root
|-- sensor_id: string (nullable = true)
|-- timestamp: string (nullable = true)
|-- temperature: double (nullable = true)
|-- humidity: double (nullable = true)
|-- pressure: double (nullable = true)
```

行数カウント (.count() - Action):

```
total_rows = sensor_df.count() # 全データをスキャンするため時間がかかる
```

🎯 ポイント
- `timestamp` が `string` になっている → 自動推論の限界。後で修正が必要
- `.count()` はActionなので、実行するとSparkクラスタ全体で計算が走る

Part２：Spark DataFrame の基本操作　ビッグデータ技術応用 実習３

Transformation vs Action

実習３ 大規模処理 Part２

Sparkの操作：Transformation と Action

Transformation (変換)

- 新しいDataFrameを作る操作
- 例: select, filter, groupBy
- 指示しただけでは**実行されない**
- 「処理計画」を組み立てるイメージ

➡ (Action実行)

Action (実行)

- 結果を返す、または書き出す操作
- 例: show, count, collect, save
- これを呼び出すと初めて計算が走る
- 「実行！」の号令イメージ

遅延評価 (Lazy Evaluation)

Actionが呼ばれるまでTransformationを実行しない仕組み。これによりSparkは処理全体を最適化できます

基本操作① Select & Filter

実習３ 大規模処理 Part２

タスク3：列選択と行フィルタリング

Pandasと似た感覚で操作できます。

列選択 (.select() - Transformation):

```
selected_df = sensor_df.select("sensor_id", "temperature")
```

行フィルタリング (.filter() / .where() - Transformation):

```
# temperature が 30より大きい行を抽出
high_temp_df = sensor_df.filter(sensor_df.temperature > 30)
# または SQLライクな書き方
high_temp_df = sensor_df.where("temperature > 30")
```

結果表示 (.show() - Action):

```
high_temp_df.show(5) # ここで初めて filter が実行される
```

基本操作② GroupBy & Agg

実習３ 大規模処理 Part２

タスク3：グループ集計

groupBy() でグループ化し、agg() で集計します

```
from pyspark.sql.functions import count, avg # 集計関数をインポート
# sensor_id ごとにグループ化 (Transformation)
grouped_df = sensor_df.groupBy("sensor_id")
# 集計 (Transformation)
agg_df = grouped_df.agg(
count("*").alias("record_count"),
avg("temperature").alias("avg_temp")
)
# 結果表示 (Action)
agg_df.show(10)
```

ポイント
- 集計関数 (count, avg など) は pyspark.sql.functions からインポートして使う
- .alias() で結果の列名を指定できる

Parquet 形式の利点

実習３ 大規模処理 Part２

タスク4：なぜParquetが良いのか？

大規模データを扱うなら CSV より Parquet が断然有利

CSV
- 行指向
- 圧縮効率 低
- 全列読み込み
- スキーマ情報 なし

Parquet
- **列指向**
- **圧縮効率 高**
- **必要列のみ読込**
- **スキーマ情報 あり**

結果
ファイルサイズが小さくなり、読み込み速度が劇的に向上
データ型推論 (inferSchema) も不要に

Parquet への変換・保存

実習３ 大規模処理 Part２

CSV → Parquet へ変換・保存

Spark DataFrame を Parquet 形式で保存するのは簡単です

```
parquet_output_path = 'sensor_data.parquet'

# .write.parquet() を使う (Action)
sensor_df.write.mode("overwrite").parquet(parquet_output_path)
```

⚠ 書き込みには時間がかかります

Parquet の読み込み

実習３ 大規模処理 Part２

Parquet を高速読み込み

読み込みは spark.read.parquet() を使います

```
# Parquetディレクトリを指定して読み込む
sensor_df_parquet =
spark.read.parquet(parquet_output_path)

# スキーマ確認 (inferSchema不要！)
sensor_df_parquet.printSchema()

# 表示 (Action) - CSVより速いはず！
sensor_df_parquet.show(5)
```

🎯 比較ポイント
CSV読み込み(スライド3)の `.show(5)` と比べて、実行時間がどれくらい短縮されたか確認しましょう

実習３ Part２のまとめ

実習３ 大規模処理 Part２

Spark DataFrame の基本的な操作をマスターしました

- ✅ **Spark での CSV/Parquet 読み込み**
- ✅ **スキーマ確認、行数カウント**
- ✅ **Transformation** (select, filter, groupBy, agg)
- ✅ **Action** (show, count, write)
- ✅ **遅延評価の概念**
- ✅ **Parquet 形式の利点と使い方**

NEXT

Part３：パフォーマンス比較と考察

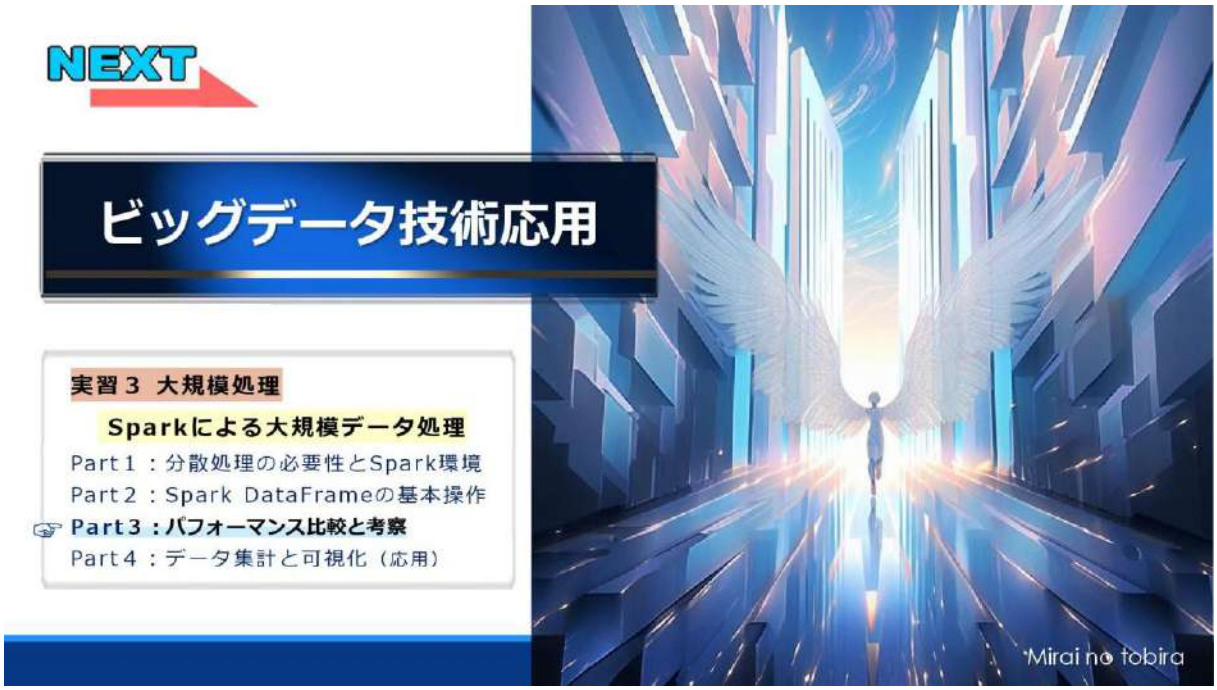
実習３ 大規模処理

実習３ Part３ では、

同じ処理を **pandas と Spark** で実行し

処理速度を比較します

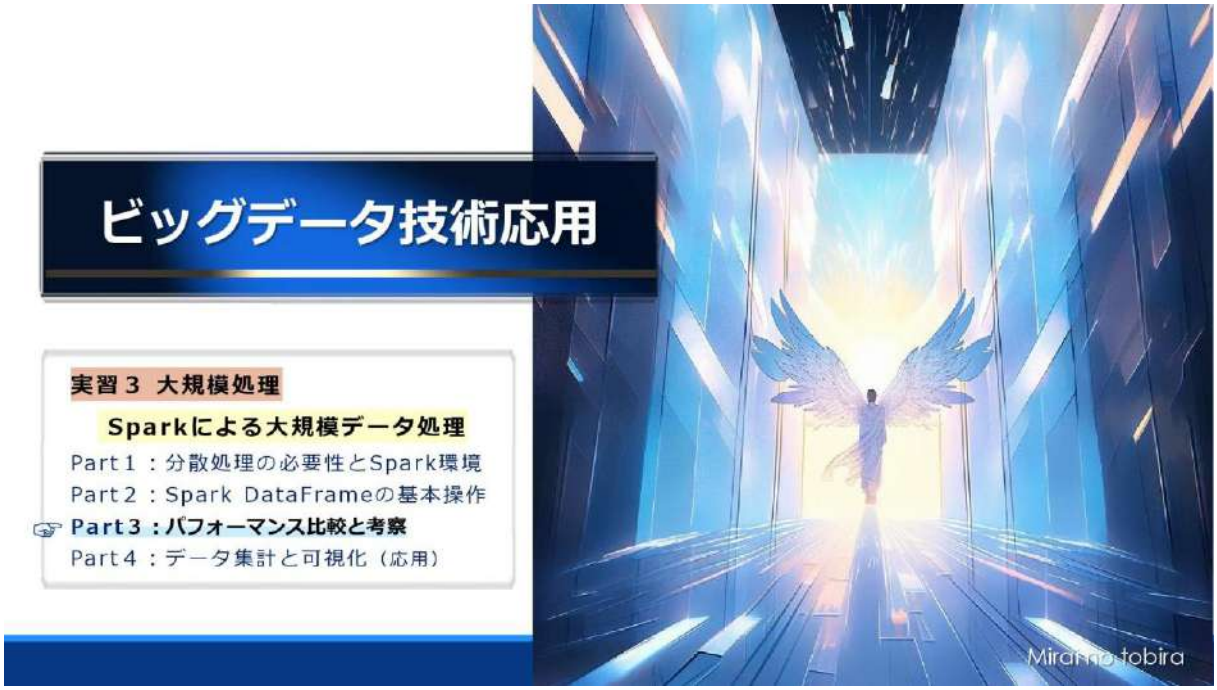
Spark の真価を体感しましょう


NEXT
ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

ビッグデータ技術応用_実習３

Part3


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part３：パフォーマンス比較と考察

実習３ 大規模処理

Pandas vs Spark：速いのはどちら？

Part３：パフォーマンス比較と考察 の達成目標

実習３ 大規模処理 Part３

- ✓ 同じ集計処理を pandas と Spark で実行し、時間を計測する
- ✓ 大規模データに対する両者のパフォーマンス（速度）の違いを比較する
- ✓ なぜ Spark が高速なのか、その理由（分散処理など）を考察する
- ✓ %%time マジックコマンドの使い方をマスターする

比較実験の準備

実習３ 大規模処理 Part３

タスク1：比較実験

比較対象：
- **Pandas:** シングルマシン、メモリ処理
- **Spark:** 分散処理エンジン

比較する処理：
- センサーIDごとの統計量（レコード数、温度の平均/最大/最小）の算出

使用データ：
- Part２で作成した `sensor_data.parquet`（数GB）

計測方法：
- `%%time` マジックコマンド

Pandasでの挑戦

実習３ 大規模処理 Part３

タスク2：Pandasで挑戦

ステップ1：データ読み込み (Parquet)

```
%%time
try:
  pandas_df = pd.read_parquet('sensor_data.parquet',
engine='pyarrow')
except MemoryError:
  print("❌ メモリ不足！")
  pandas_df = None
```

ステップ2：集計処理 (もし読み込めたら)

```
%%time
if pandas_df is not None:
  pandas_agg = pandas_df.groupby('sensor_id').agg(...)
  print(pandas_agg.head())
```

Sparkでの挑戦

実習３ 大規模処理 Part３

タスク3：Sparkで挑戦

ステップ1：データ読み込み定義 (これは一瞬)

```
sensor_df_spark =
spark.read.parquet('sensor_data.parquet')
```

ステップ2：集計処理 (Action実行時に計算開始)

```
%%time
from pyspark.sql.functions import count, avg, max, min
spark_agg = sensor_df_spark.groupBy("sensor_id").agg(
  count("*").alias("count"),
  avg("temperature").alias("avg_temp"),
  # ... max, minも同様 ...
)
spark_agg.show(5) # Action!
```


結果比較

実習３ 大規模処理 Part３

タスク4：結果比較

計測結果を比較してみましょう

処理	Pandas 時間	Spark 時間
センサー別集計	[エラー or XX 秒]	[YY 秒]

なぜ Spark は速い？①

実習３ 大規模処理 Part３

考察：Spark高速化の秘密① 分散処理

Spark が速い最大の理由は**「分散処理」**

- ▶ Spark はデータを複数のパーティションに分割
- ▶ それらを複数の Executor（CPUコアやマシン）で**並列に処理**
- ▶ Pandas は基本的に1コアで順番に処理

→ データが大きければ大きいほど並列処理の効果でSparkが有利

なぜ Spark は速い？②

実習３ 大規模処理 Part３

考察：Spark高速化の秘密② メモリ管理＆遅延評価

メモリ管理：

- ▶ Pandasは全データをメモリに乗せようとする
  → メモリ不足のリスク
- ▶ Sparkは必要なデータだけをExecutorのメモリに乗せて処理
  → メモリ効率が良い

遅延評価：

- ▶ Actionが呼ばれるまで処理を遅らせる
  → Sparkが処理全体を最適化（例：不要なデータの読み込みを省略）可能

実習３ Part３のまとめ

実習３ 大規模処理 Part３

Pandas と Spark の比較実験をして、Spark の威力を体感しました

- ✅ **大規模データでは Pandas は限界があり、Spark は処理可能**
- ✅ **Spark は分散処理により Pandas より高速に処理できる**
- ✅ **その理由は、並列処理、効率的なメモリ管理、遅延評価にある**
- ✅ **%%time で処理時間を計測・比較する方法を学んだ**

→ 結論として、ビッグデータを扱うなら Spark が必須

NEXT

実習３Part４ ：データ集計と可視化（応用）

実習３ 大規模処理

実習３ Part４では、

- ✓ もう少し複雑な集計処理に挑戦
- ✓ Spark の集計結果を **pandas DataFrame** に変換
- ✓ Matplotlib/Seaborn で見慣れたグラフを作成

を行います


NEXT
ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

ビッグデータ技術応用_実習３

Part4


ビッグデータ技術応用
「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」
実習１ 分析実践
実習２ 業務活用
実習３ 大規模処理
Sparkによる大規模データ処理
Mirai no tobira


ビッグデータ技術応用
実習３ 大規模処理
Sparkによる大規模データ処理
Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
Part４：データ集計と可視化（応用）
Mirai no tobira

Part４：データ集計と可視化（応用）

実習３ 大規模処理

Spark で処理し、Pandas で描く：実践的ワークフロー



Part４：データ集計と可視化（応用）の達成目標

実習３ 大規模処理 Part４

- ✓ Spark で応用的な集計（日時変換、ウィンドウ関数、SQL）を行う
- ✓ Spark DataFrame を Pandas DataFrame に変換する (.toPandas())
- ✓ .toPandas() の注意点（メモリ）を理解する
- ✓ 変換後の Pandas DataFrame を使い慣れたツールで可視化する
- ✓ Spark と Pandas/可視化ライブラリを連携させる流れを習得する

応用的な集計① 日時処理

実習３ 大規模処理 Part４

タスク2：応用的な集計① 日時処理

Part２で string のままであった timestamp 列を日時型に変換し
時間帯の平均値を計算します

ステップ1：日時型へ変換

```
from pyspark.sql import functions as F
df_with_ts = sensor_df.withColumn(
  "ts", F.to_timestamp(F.col("timestamp"), 'yyyy-MM-dd HH:mm:ss')
)
```

ステップ2：Hour抽出 & 集計

```
df_with_hour = df_with_ts.withColumn("hour", F.hour(F.col("ts")))
hourly_avg = df_with_hour.groupBy("hour").agg(
  F.avg("temperature").alias("avg_temp"), ...
).orderBy("hour")
hourly_avg.show()
```


応用的な集計② ウィンドウ関数

実習３ 大規模処理 Part４

タスク2：応用的な集計② ウィンドウ関数

センサーごとに時間順に並べ、過去N件の移動平均を計算する例

```
from pyspark.sql.window import Window

# ウィンドウ定義: sensor_idで分割し、tsで並び替え
windowSpec = Window.partitionBy("sensor_id").orderBy("ts")

# 移動平均を計算（現在行と過去2行）
df_with_ma = df_with_ts.withColumn(
  "temp_ma_3", F.avg("temperature").over(windowSpec.rowsBetween(-2, 0))
)
df_with_ma.select("sensor_id", "ts", "temperature",
"temp_ma_3").show(10)
```

応用的な集計③ Spark SQL

実習３ 大規模処理 Part４

タスク2：応用的な集計③ Spark SQL

DataFrame API の代わりに、SQLクエリで集計することも可能

ステップ1：テーブル登録

```
sensor_df.createOrReplaceTempView("sensor_data")
```

ステップ2：SQL実行

```
sql_result = spark.sql("""
  SELECT sensor_id, COUNT(*) as count, AVG(temperature) as avg_temp
  FROM sensor_data
  GROUP BY sensor_id
""")
sql_result.show(10)
```


Spark から Pandas へ

実習３ 大規模処理 Part４

タスク3：Spark → Pandas へデータを受け渡す

Sparkで集計した「結果」（通常はメモリに乗るサイズ）を
使い慣れた pandas DataFrame に変換します

```
# 例：時間帯別平均データをPandasに変換
hourly_avg_pd_df = hourly_avg_df.toPandas() # Action!

# 変換後の確認
print(type(hourly_avg_pd_df))
hourly_avg_pd_df.info()
```

⚠ 重要注意点

`.toPandas()` は、**分散していたデータをすべてDriverのメモリに集めます**
変換対象のデータが巨大な場合、ここで **MemoryError** が発生します
必ず、**集計後**の十分に小さくなったDataFrameに対してのみ使用してください

Pandas で可視化

実習３ 大規模処理 Part４

タスク3：Pandas DataFrameでいつも通り可視化

Pandas DataFrame を使えば、その後は実習１,２の実習と同様

```
import matplotlib.pyplot as plt
import seaborn as sns

# 例：時間帯別 平均温度の折れ線グラフ
plt.figure(figsize=(12, 6))
sns.lineplot(data=hourly_avg_pd_df, x='hour',
y='avg_temperature', marker='o')
plt.title('Average Temperature by Hour')
plt.show()
```


実践的なワークフロー

実習３ 大規模処理 Part４

Spark + Pandas/可視化ライブラリ連携

実務でもよく使われる Spark + Pandas 連携ワークフローワークフローです

1. 読み込み
Sparkで大規模データ(Parquet等)をロード

↓

2. 分散処理
Sparkで重い集計や加工を高速実行

↓

3. 軽量化
結果がメモリに乗るサイズに凝縮

↓

4. 変換
.toPandas()で Pandas DFへ

↓

5. 可視化
Matplotlib等でグラフ化・分析

ポイント
- **重い処理・集計：** Sparkの分散処理能力を活用
- **最終的な可視化・レポート：** 使い慣れたPandas + Matplotlib/Seaborn を活用
- **橋渡し役：** `.toPandas()` （ただしメモリに注意）

Part 4 ／ 実習3 まとめ

実習 3 大規模処理

Part 4 では応用的な集計と可視化連携を学び、
実習 3 全体を通して以下のスキルを習得しました

- ✅ Spark の必要性と基本操作
- ✅ Pandas vs Spark のパフォーマンス比較体験
- ✅ Parquet 形式の利点
- ✅ Spark での集計結果を Pandas で可視化する連携

ビッグデータ技術応用 まとめ

ビッグデータ技術応用

実習 1, 2, 3 を通して
データ分析～ETL～BI～大規模処理まで
データ活用の重要スキルを幅広く学びました

今回学んだ Spark は入口、ビッグデータの世界は広大
さらに深く次のような項目を学んでいくことが望まれます

- Spark Streaming (リアルタイム処理)
- MLlib (Sparkの機械学習ライブラリ)
- Hadoopエコシステム (HDFS, Hive, etc.)
- クラウドプラットフォーム (AWS, GCP, Azure) での Spark
- データパイプライン構築 (Airflowなど)


ビッグデータ技術応用
Mirai no tobira

ビッグデータ技術応用_実習まとめ編

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習１ 分析実践

Python による顧客データ分析とレポーティング

実習２ 業務活用

アクセスログETL処理とBIツールによる可視化

実習３ 大規模処理

Sparkによる大規模データ処理



ビッグデータ技術応用

実習３ 大規模処理

Sparkによる大規模データ処理

Part１：分散処理の必要性とSpark環境
Part２：Spark DataFrameの基本操作
Part３：パフォーマンス比較と考察
☞**Part４：データ集計と可視化（応用）**



実習構成

ビッグデータ技術応用

実習 1 分析実践

Python による
顧客データ分析と
レポーティング

Pythonでの
データ分析プロセス
**Clean →
Analyze →
Visualize →
Report**

実習 2 業務活用

アクセスログETL処理と
BIツールによる可視化

ETL処理 & BIツールでの
ダッシュボード作成
**Python →
Tableau/Power BI**

実習 3 大規模処理

Spark による
大規模データ処理

Sparkによる
分散処理の基礎
**Pandas限界 →
Spark**

ビッグデータ技術応用 習得済スキル

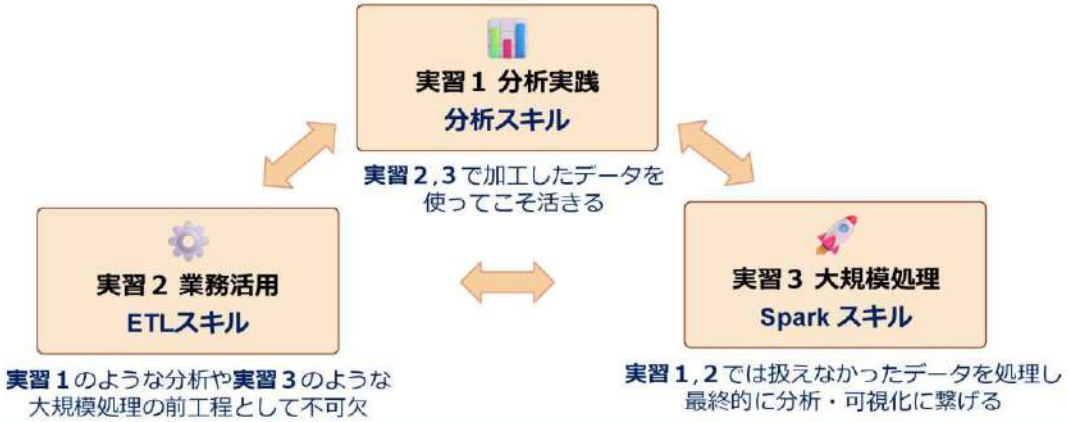
ビッグデータ技術応用

**実習 1,2,3 を通してデータ分析～ETL～BI～大規模処理まで
データ活用の重要スキルを幅広く学びました**

- **Python データ分析スキル**
  Pandas, Matplotlib, Seaborn を使った実践的な分析力
- **ETL基礎スキル**
  生データ (テキスト含む) を分析可能な形に加工する力 (正規表現含む)
- **BIツール活用スキル**
  Tableau, Power BI を使ってデータを分かりやすく見える化する力
- **大規模データ処理入門スキル**
  Sparkの基本を理解し、メモリを超えるデータを扱う考え方
- **問題解決 & 考察力**
  データから課題を発見し、解決策を考え、結果　を伝える力

スキルの繋がり

ビッグデータ技術応用



実習１のような分析や実習３のような
大規模処理の前工程として不可欠

実習１,２では扱えなかったデータを処理し
最終的に分析・可視化に繋げる

学びは続く：Next Step

ビッグデータ技術応用

応用教材で得た基礎を元にさらに専門性を深めましょう

- ✓ **データ分析深化**
  機械学習、統計モデリング、特定分野（マーケティング、金融など）の分析手法
- ✓ **データエンジニアリング深化**
  データパイプライン構築 (Airflow)、クラウド (AWS, GCP, Azure)、
  データベース (SQL, NoSQL)、ストリーミング処理 (Kafka, Spark Streaming)
- ✓ **BI深化**
  高度なダッシュボードデザイン、DAX/LOD 表現、データガバナンス
- ✓ **実践経験**
  Kaggleコンペへの挑戦、インターンシップ、卒業研究でのデータ活用

「ビッグデータ技術応用」のその先

今回学んだ Spark は入口、ビッグデータの世界は広大
さらに深く次のような項目を学んでいくことが望まれます

- Spark Streaming (リアルタイム処理)
- MLlib (Sparkの機械学習ライブラリ)
- Hadoopエコシステム (HDFS, Hive, etc.)
- クラウドプラットフォーム (AWS, GCP, Azure) での Spark
- データパイプライン構築 (Airflowなど)

ビッグデータ技術応用

「基礎から実践へ
3つの実習で学ぶデータ分析・基盤技術」

実習 1 分析実践
実習 2 業務活用
実習 3 大規模処理




ビッグデータ技術応用
Mirai no tobira

令和7年度「地方やデジタル分野における専修学校理系転換等推進事業」
情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

ビッグデータ技術応用教材資料 2

令和 8 年 2 月

一般社団法人全国専門学校情報教育協会
〒164-0003 東京都中野区東中野 1-57-8 辻沢ビル３F
電話：03-5332-5081 FAX.03-5332-5083

●本書の内容を無断で転記、掲載することは禁じます。