情報特別演習最終レポート

筑波大学情報学群情報科学類 (201611350) 江畑 拓哉

January 28, 2018

Contents

1 概要

今情報特別演習において私は、機械学習を初学者が学ぶための Web アプリ開発を行った。当初は大規模データベースを用いた機械学習 API を作るという目標であったが、特に機械学習を学んでいくにあたり、これの中身を理解するための初学者向けの解説の供給が少ないと感じたため、開発目標をやや変更した。

今回学習した内容は、① 代表的な時系列分析の一つである (S)ARIMA モデル ② 大規

模データベースの、特に時系列データベースの比較とその利用法 ③ 更に機械学習等を実装する際に重要となる高速計算を JVM 言語内で行う手法 ④ 開発言語として取り上げた Clojure/ClojureScript の、言語自体・これを用いた Web 開発手法、の 4 分野である。

結果として ARIMA モデルとそれに付随する ADF 検定などを実装・解説を作成し、時系列データベースとして OpenTSDB を採用し設置した。更に JVM 上で高速計算を行うために、*jblas* や *Intel®MATH KERNEL LIBRARY* を用いた GPU 演算 、OpenCL の利用例を調べ比較し、一部を Web アプリの実装に活用した。そして *Clojure* という言語を身に着け、Clojure/ClojureScript を用いて JavaScript のライブラリである *React.js* などを利用する手法についてまとめ、これらを利用して Web アプリの概形を実装した。

2 序論

この情報特別演習の初期テーマの決定はグループメンバーからの提案が元であった。その概要は、大規模データベースである *ApacheHBase*™ (以降 “HBase” と呼称する) を用いて入力されたデータの因果関係を分析、予測する機械学習 API を作成するというものである。ここから因果関係と相関関係の違いについて学習し、機械学習手法について吟味した結果、ニューラルネットを用いた学習手法と、(S)ARIMA モデルを用いた時系列データ予測と Random Forest を用いた欠損値補完を組み合わせたものの 2 つの手法が議題に上がったが、後者を選択することになりこれを研究することになった。その中で機械学習を学ぶ際にその内部を知る必要があり、学んでいく際にその資料の供給が少ないことを感じ、その資料も兼ね備えたいと考え、また機械学習結果を視覚的にわかりやすく伝える目的も合わせて、Web アプリという形で開発を行うことに目標を定めた。

機械学習という名前は世間に広く流布しており、それを用いた API として例えば Microsoft Azure の *ComputerVisionAPI* 等を上げることができるが、この中身が解説されることはその必要性や機密性の問題から非常に少ない。また Python や R といった言語に付属している統計処理、機械学習の関数などもその殆どが簡便化されており、例えば *auto.arima*() 関数などはその中身を踏み込むことなく実行、モデルの比較を行うことができる。これは機械学習の利用者という立場からすれば素晴らしい進歩であると考えられるが、その反面機械学習を学ぶ立場になった場合、その中身を知らない状態で API や関数を利用することができるため、手放しに機械学習を理解できたと誤解してしまう可能性がある。

このため主要な機械学習についてその中身を知ることができ、且つ理解ができるツールの作成は統計や機械学習を学びたいと志している、或いは先述のようなツールの中身について興味を持った者に対して、この Web アプリの開発は一定の需要があるのではないかと考えている。

3 時系列データベースの比較と OpenTSDB の利用法

一般的に時系列データのような、単調増加する行キーを持つデータを通常の大規模データベースに保存することはデータの分散という点から問題が発生する。[2] このため時系列データを扱うためのデータベースを考える必要がある。幸いなことに、初期案にあった HBase に対して時系列データを扱うことができるように拡張した *OpenTSDB* というデータベースがある。OpenTSDB は HTTP API として操作が許されており、また保存されているデータを確認することが容易であるように設計されている。今回は HBase を完全分散モードで利用する、というチームメンバーの目標に沿ってこちらのデータベースを採用した。

他の時系列データベースの提案として考えられるものに、① *InfluxDB* 、② *Graphite* 、 ③ *Datomic* などを挙げることができる。

3.1 InfluxDB

InfluxDB は InfluxData が開発を行っているデータベースであり、高機能なクラウドシステムを有料で使うことができるほか、オープンソースソフトウェアとしての利用も可能である。このデータベースの利点の一つに、利用方法が簡単であることが挙げられる。シングルノードでの利用に関してのみに焦点を絞れば、2017 年 12 月において *Arch Linux* でのインストールは、パッケージのインストールとサービスの起動の 2 つのコマンド で利用可能になる。またデータベースの読み書きに関しては、SQL に近い記法を用いた HTTP API を用いて行うことができ、今回の演習における開発言語である Clojure で必要な部分に関するラッパーを書くことは非常に簡単であった。また TICK stack と呼ばれる InfluxDB を含む時系列データを扱うための環境を追加でセットアップすればデータのモニタリング、収集、リアルタイム処理をより効率的に行うことができる。本演習の初期目標ではデータの入力をユーザが行うことができる設定になっていたため、悪意あるデータを監視することが容易であるという点、データベースの外側の分野までの広いサポート環境があるという点からこのデータベースは非常に魅力的である。

このデータベースが扱うデータモデルの概要を以下に示す。

Table 1: InfluxDB data model

name(required)		
timestamp (required)	fields (required)	tags (optional)
⋮	⋮	⋮

それぞれの用語についてその意味と例を挙げると以下のようになる。

- name データの名前 (ex. 日経平均株価)
  データの名前であり、何に関してのデータであるかを表す。

- timestamp 時刻データ (ex. 2018-01-27T00:00:00Z)
  時刻データであり、いつのデータであるのかを示す。この場合の “いつのデータ” とは、データの登録日時ではなく、そのデータの発生日時である。

- fields 測定値群 (ex. (終値：12000) (始値：11000))
  そのデータが持つ値を示す。いくつかの属性に従って複数の値を格納することができるが、ここに登録されるデータは索引付けされるべきものではないという点でタグ群と意味が異なる。

- tags タグ群 (ex. (記録者：A) (ソース：東京株式市場))
  そのデータの持つ属性や追加情報を示す。ここに登録されるデータは索引付けさており、データの絞り込みを行う目的に用いられる。

3.2 Graphite

Graphite は Python を中心にして書かれた時系列データベースであり、同じく Python の Web フレームワークである Django と組み合わせることが、 Graphite 自身の Web UI コンポーネントが Django であるという点から非常に容易である。同時に Python は機械学習に関する API が豊富に存在しているため、本演習が純粋に “Web API の作成” のみの目標であったならば当然こちらを用いて開発を行っていただろう。またデータベースの導入自体も Python のパッケージ管理システムである pip を用いて行うことができることから、純粋に Python によってすべての問題を解決することができる。更に Graphite のドキュメントは豊富に存在しており、例えば Monitoring with Graphite [1] を挙げることができる。

Graphite の内部について簡単に説明を行うと、主に 4 つのコンポーネント、Carbon、Whisper、Cario、Django を中心に展開する。

- Carbon は、後述するデータベースそのものと言える Whisper に登録する役割を担っており、メトリクス[1]のバッファリングを行ったり他のデータベースにメトリクスをリレーさせたりすることができる。

- Whisper は、入手したデータをファイルシステムに書き込み・読み出しを行う役割を担っており、この部分は Ceres と呼ばれるコンポーネントに置き換えることができる。両者の違いは、Whisper が保存領域を固定サイズとして確保するのに対して、Ceres は任意のサイズの保存領域を確保できるということにある。

[1] metrics: 入手したデータを分析して数値化したもの

- Cario は、Graphite のグラフィックエンジンを担当しており、保存されているデータを視覚化する上で非常に重要な役割を果たしている。

- Django は、Cario によって出力されたデータを表示する役割を担っており、データを扱う開発者はこの部分を見てデータを確認することになる。

このデータベースが扱うデータモデルは階層構造を取っており、一例を紹介すると以下のようになる。

”stock_price.nikkei_index.close_price 12000 1517055464”

上の文字列を送信することによって、stock_price の中の nikkei_index の中にある close_price という階層に 12000 という値を Unix 時間である 1517055464 のデータとして登録している。つまりこのデータは以下のようにな形に保存されたと考える。

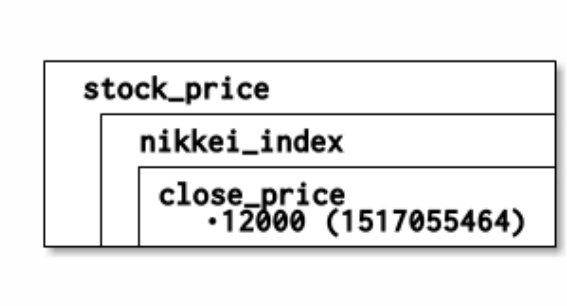

Figure 1: Graphite example

3.3 Datomic

Datomic は他のデータベースとはかけ離れた設計が行われた新しい世代の分散型データベースである。 Clojure の作者である Rich Hickey 氏らが作成し、有料でメンテナンスとアップデートが付属されたクラウドシステムを使うことができる。また一年に限っては無料でこの機能を利用することもできる。これとは別に存在する無料版に関しては分散できるピア数などの制限がかかる。
Datomic には 2 つの目標「情報を時間によって紐付け蓄積する」 「データベースアプリケーションのモデルをリモートアクセスするものからそれぞれのプログラムの中にあるものとする」[1] がある。この考え方によって得られた大きな 2 つの特徴に、① Append-Only ②データベースに独立したクエリーエンジンがある。

Append-Only とはその名の通り、追加のみという意味で言い換えれば変更ができない

[1] http://endot.org/notes/2014-01-10-using-datomic-with-riak/

ということを意味する。これは情報を時間に紐付けることによって最新の情報を見ることができるため、情報を “書き換える” 必要がなくなったためにできたことであり、トランザクション処理などのデータの管理を容易にすることができる。

データベースに独立したクエリーエンジンとは、アプリケーション側でトランザクションやクエリ処理を実行するという意味を示しており、今まではデータベースに HTTP API などを用いてクエリを投げデータベース側がそのクエリを処理して結果を送信していたものをアプリケーション側に移すということになる。その意味で Datomic はアプリケーション側をピア [2] と呼称する。

ピアが扱うデータはデータベースではなくピア側のキャッシュに Read Only な形で LRU [3]形式で保持される。データベースは書き込まれたデータを保存し、更新があればそれぞれのピアが持つデータベースに対して常に開いているノードに告知し、アプリケーション側から要求されるデータ群をそのまま返すことになる。これによってピア側のメモリキャッシュを貪欲に使うことができ、データベースのボトルネックを解消することができるようになっている。更にピア側のキャッシュ上のデータベースは実質ゼロコストで用いることができるため、 LRU が最適であるような目的のアプリケーションにこのデータベースを適用した場合データへのアクセスという点において他のデータベースに性能で劣ることはない。またクエリ処理を分散しているため、多くのクエリ処理をこなさなければならないピアが増えたとしてもキャッシュ上のデータを使っている限りはその処理によってデータベースに負荷がかかることもない。またデータベースの更新をピアに告知しなければならないという点でデータベースへの書き込みがネックになる可能性もあるが、これは論理的に分かれているデータごとにデータベースそのものを分割することで解決することができる。

データベースのアクセス方法は Datalog と呼ばれる Clojure らしいシステムによって扱われるため、SQL に慣れている場合には苦労する可能性があるがアプリケーションに柔軟に組み込むことができる。これはデータがキャッシュ上に Read-Only な形で存在しているという特性と、 Clojrue が関数型言語の側面を持っているという点を考えれば、データベース上のデータを手元にあるデータであるかのように直に利用することができるということを意味している。また保存しているデータは必ず Datom という最小単位に分割されており、これを元にして様々な形にデータを変形することができる。

このデータベースが扱うデータ例を以下に示す。

```
{:nikkei-index/type “close-price”
  :nikkei-index/value 12000
  :nikkei-index/timestamp 1517055464}
```

データは nikkei-index/type に対する値として “close-price” が格納されている。nikkei-index に “close-price” が含まれているわけではない。

[2]peer
[3]Least Reacently Use

h

Table 2: Datomic の特徴

目指すもの	· 情報は時間によって紐付ける · データベースアプリケーションのモデルをそれぞれのプログラム内に移動する
大きな特徴	· Append-Only データベース · データベース側ではなくアプリケーション側にクエリ処理エンジンがある

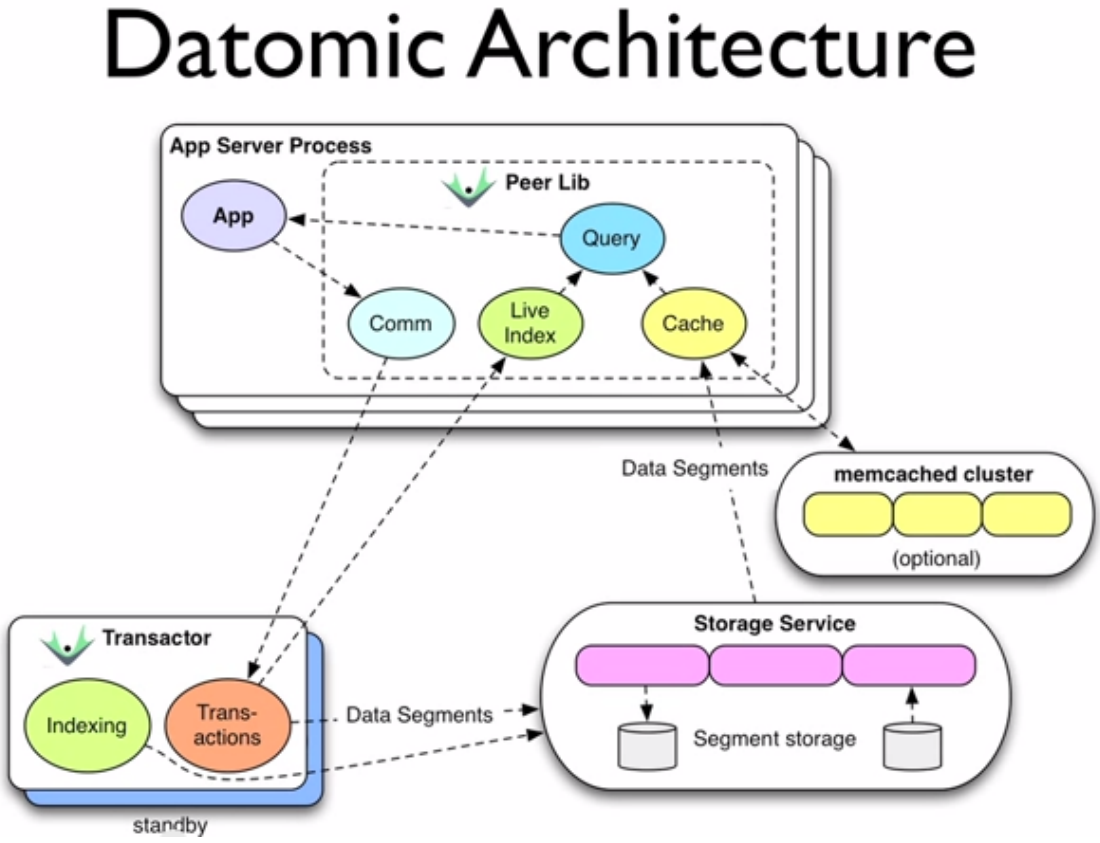

Figure 2: Talk Notes: Using Datomic With Riak より

3.4 OpenTSDB

OpenTSDB の特徴の説明、セットアップや利用方法に関して説明を行う前に、その基盤である HBase とその周辺知識について簡単にまとめる。HBase とは *Apache*™ *Hadoop*® (以降 “Hadoop” と呼称する) と呼ばれる、大規模データの分散処理フレームワークのためのデータベースである。そして Hadoop の分散サービスを形成するために *Apache Zookeeper*™ (以降 “Zookeeper” と呼称する) という管理ツールが使われる。

3.4.1 HBase

HBase は NoSQL の一つである。NoSQL は大別して、①キーバリュー型②ワイドカラム型③ドキュメント型④グラフ型、があり HBase はワイドカラム型[4]に属している。

Table 3: ワイドカラム型の例 (Name 列を取り出すこと等を得意とする)

ID	Name	Email	Birthday	Authorization
001	Bob	bob @ foo.com	1998/01/02	true
002	John	john @ bar.com	1987/02/01	false
⋮	⋮	⋮	⋮	⋮

Hadoop の HDFS (Hadoop Distributed File System) を担っており、複数台のマシンのディスクを一台のディスクであるかのように扱うことができる。全体のデータは Region という単位で分割されており、これをそれぞれのディスクに 1 つ以上割り振っていくことで分散を行う。

続いて HBase の論理データモデルについて説明を行う。最上位概念は Namespace と呼ばれるもので、この中には Table と呼ばれるデータを表形式で保持している概念を 1 個以上含んでいる。 一つ以上の RowKey 、一つ以上の ColumnFamily で構成されている。そして ColumnFamily には一つ以上の ColumnQualifier が存在している。行キーである ColumnQualifier と列キーである RowKey の交差点にはそれぞれ Cell と呼ばれる領域があり、ここにデータが格納されることになる。データは Timestamp とともに保存されており、 Cell にはそのデータが重ねて保存される。つまり Cell には Timestamp に紐付けられたデータが複数存在することになる。また、ワイドカラム型であるという特性上、Table は Rowkey でソートされた状態で保存されることになる。

HBase の物理モデルの Table の構造はキーバリュー形式で保存されている。物理モデルの詳細はデータの分散などの説明も必要となるが、これ以上の内容は本演習で理解することができなかったため説明を省く。

[4]簡単に説明するとデータを行ごとではなく列に対して管理しており特定の列を取り出して処理することに最適化されており、高いパフォーマンスやスケーラビリティを持っている。

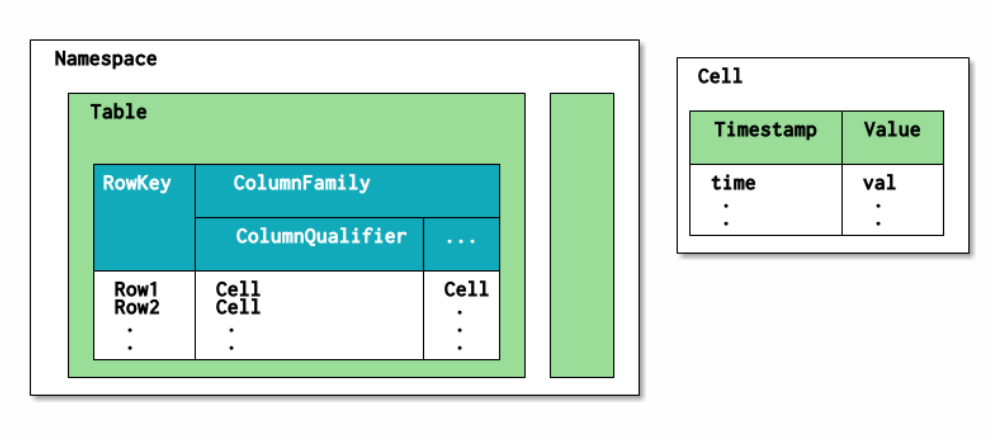

Figure 3: HBase の論理データモデル

3.4.2 Hadoop

Hadoop は大規模データセットの分散処理フレームワークである。

3.4.3 Zookeeper

4 Clojure を用いた JVM における高速計算技法

4.1

4.2

4.3

5 ARIMA モデルによる時系列分析

5.1

5.2

5.3

5.4

6 Clojure/ClojureScript を用いた Web 開発

7 MKKL の開発

8 まとめと今後の課題

References

[1] Jason Dixon. *Monitoring with Graphite. Tracking Dynamic Host and Application Metrics at Scale*. Vol. 290. O'Reilly Media, 2017.

[2] *The Apache HBase^{TM} Reference Guide*. Revision 0.94.27. Copyright © 2012 Apache Software Foundation. https://www.slideshare.net/hamadakoichi/randomforest-web, Dec. 2015.