分散データベースのための Web サービスフレームワークの開発

DEVELOPMENT OF WEB SERVICES FRAMEWORK FOR DISTRIBUTED DATABASE

丸山 俊之#, A), 福井 達 B), 籠 正裕 C), 石井 美保 C), 吉岡 正倫 D), 大島 隆 B)
Toshiyuki Maruyama#, A), Toru Fukui B), Masahiro Kago C), Miho Ishii C), Masamichi Yoshioka D), Takashi Ohshima B)
A) Nippon Gijutsu Center Co.,Ltd.
B) RIKEN Harima Institute
C) Japan Synchrotron Radiation Research Institute
D) SPring-8 Service Co., Ltd.

Abstract

We developed a data acquisition system of abnormal RF waveform at SACLA. When a waveform of such as klystron forward signal deviates from a reference waveform, the abnormal waveform is stored in a database. The stored waveform can be easily browsed through a Web browser without platform dependence. We adopted Cassandra as the database because it is a distributed database management system with high read / write performance and providing high availability without single point of failure (SPOF). Cassandra is possible to elicit high performance by parallel programming. However, the parallel programming requires many steps in case of some programming languages. We developed a Web Service framework for reading waveform data by the parallel programming from Cassandra. When the Web Service executes a request of data acquisition from the HTTP client, it generates multiple request processes automatically in accordance with the number of CPU in the server. As a result, it is easy to upgrade the system performance by increase the number of servers. The paper reports the outline of prototype Web Service and the test results.

1. はじめに

SACLA の加速器制御システムでは、各センサーから得られたデータはデータベースに保存され、そのセンサーデータを運転用 GUI や Web ブラウザで閲覧することができる[1]。データ収集用のデータベースにはリレーショナルデータベース(RDB)を採用している。2011 年のコミッショニング以降、現時点でデータベースに登録されている機器点数は約 5 万点にもなり、1 日に約 4.5GB のデータがアーカイブデータベースに蓄積され続けている。今後は SACLA のビームライン増加に伴い、さらにアーカイブされるデータ量は増える。

LLRF データ収集においても収集した位相と振幅データは最大 60Hz のビームショット毎[2]に、I/Q 波形データは 10 分毎にデータ収集用データベースに保存していた。しかし、レーザー特性の不安定要因となるクライストロン管での RF 波形の変動が不定期に発生した場合、10 分間隔で収集した波形データでは原因を特定することが困難な場合がある。そこで、クライストロン管内で発生した RF 波形と基準波形を比較して閾値から外れる波形を捕捉し、異常波形としてデータベースに保存する異常波形データ収集システムを開発した[3][4]。大量の波形データを保存するとなると膨大なデータ量となり、書き込み速度や長期間データの読み込み応答速度など RDB では限界がある。そこで、データベースには No SQL(Not only SQL)データベースと呼ばれる Cassandra を採用した。

Cassandra は単一障害点(SPOF)のない高可用性とスケーラビリティを持ったオープンソースの分散データベース管理システムである[5]。Cassandra はクラスタ構成において高い読み込みと書き込み性能を持っており、マルチコア CPU と並列プログラミングによって更に高い読み書き性能を引き出すことができる。しかし並列プログラミングは、C 言語などプログラム言語によっては多くの手順を必要とするため、プログラム生産性やメンテナンス性が低下する場合がある。また、Cassandra のデータ問い合わせ言語である Cassandra Query Language (CQL)は、RDB の SQL のように取得データの結合や集計、グループ化、あいまい検索など複雑なデータ処理が行えない。そのため、それらの処理はアプリケーションプログラム側で実装する必要があり、プログラム言語によってはプログラミングの負担が大きくなる。

そこで異常波形データ収集システムでは、Cassandra から比較的簡単に並列処理でデータ取得が行え、そのデータ取得処理を複数の異なるプログラム言語から共有利用できる Web サービスフレームワークをプロトタイプとして開発した。Figure1 に異常波形データ収集システムの構成を示す。

本論文では Web サービスフレームワークの概要と、異常波形データ収集システムへの実装およびテスト結果について報告する。

toshiyuki.maruyama@nichigicenter.co.jp

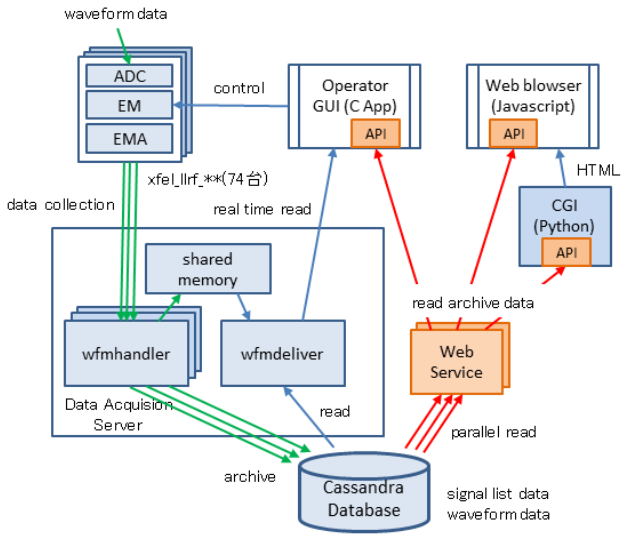


Figure 1: Configuration of an abnormal waveform acquisition system.

2. Web サービスフレームワーク概要

2.1 Web サービスの構成

Web サービスのシステム構成を Figure 2 に示す。Web サービスで使用される開発環境は入手性と安定性、汎用性の高いオープンソースのものを選択した。

Web サービスの開発言語は Python を採用した。Python は比較的簡単に並列処理のプログラムコードを記述することや、正規表現などのライブラリや辞書型・リスト型などのデータ構造を使うことができる。それにより Cassandra での複雑なデータ取得処理は C 言語などに比べて生産性の高いプログラム開発ができる。

Web サービスとクライアント側の API ライブラリとの通信プロトコルは HTTP、データ形式は JSON(JavaScript Object Notation)[6]を採用した。これらは幅広いプログラム言語でサポートされており、Web 技術との親和性の高いシステムの構築も行いやすくなる。HTTP プロトコルの実装には Tornado を採用した。Tornado は Python で記述されたノンブロッキング I/O・非同期通信型 Web フレームワークで、非常にシンプルなコードで比較的簡単に軽量 HTTP サーバーの機能を実装することができる[7]。

2.2 Map/Reduce フレームワーク

Web サービスの Cassandra からのデータ取得処理において、並列処理は簡易的な Map/Reduce フレームワーク構造[8]とした。Map/Reduce とは分散処理を行うためのプログラミングモデルで、データ処理を Map 処理と Reduce 処理の 2 段階に分けて行う。

Cassandra から並列処理でデータ取得を行うプログラムは、まず Map 処理には取得データの抽出処理やデータ加工処理を Map 定義として記述し、Reduce 処理には Map 処理で抽出したデータセットに集計処理などを Reduce 定義として記述する。次にデータ取得の条件パラメータとともに Map 定義と Reduce 定義を Map/Reduce フレームワークに引き渡すことで、あとはフレームワークが自動的に CPU の数に応じてデータ取得の条件パラメータを分割し、Map および Reduce 処理を並列に実行して取得した各データを結合して要求元に返す。Figure3 に Map/Reduce 処理のフローチャートを示す。

これにより、データ取得プログラムは並列処理をフレームワークに任せて Cassandra からデータを取得するためのソースコードのみを記述することとなる。

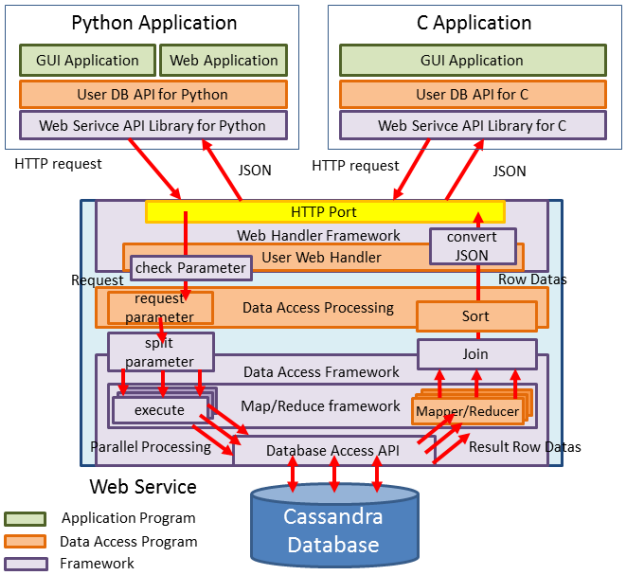


Figure 2: Structure of Web Service.

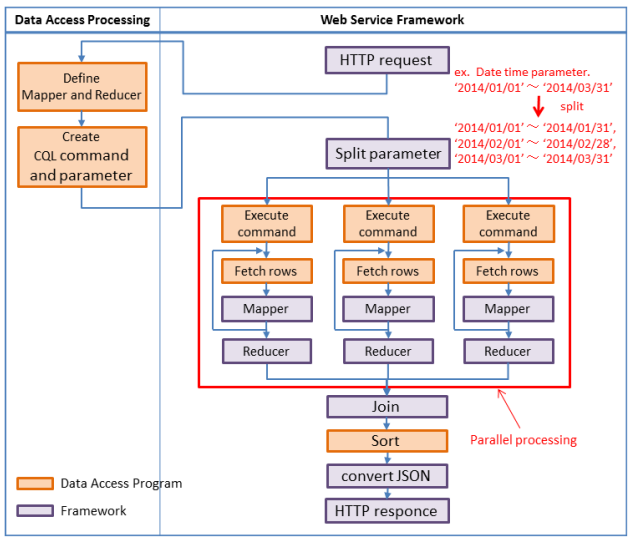


Figure 3: Map/Reduce flow chart.

2.3 Web サービスの追加と分散処理

Python の実行環境と Web サービス本体のプログラムをファイルサーバーで共有しておくことで、新たなサーバー追加の際には関連モジュールのインストールを行う必要がなく、ファイルサーバをマウントするだけで比較的簡単に Web サービスの実行環境を構築することができる。クライアント側 API ライブラリから Web サービスへのデータ要求はラウンドロビンによって負荷分散するようにした。また、Web サービスの追加や障害発生の検知もクライアント API ライブラリが自動的に行う。これにより比較的簡単に Web サービスの拡張によるスケールアウトを行うことができる。Figure 4 に分散起動した Web サービスの構成を示す。

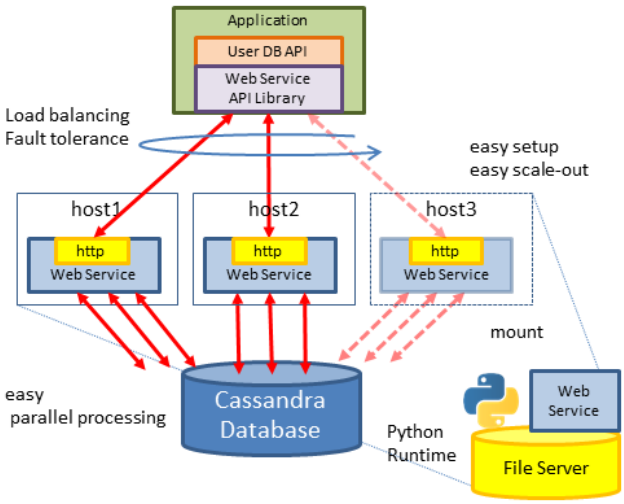


Figure 4: Distributed processing of Web Service.

3. 異常波形データ収集システムへの実装

3.1 異常波形データ収集システム Web 画面

異常波形データ収集システムでは、Web ブラウザで波形データの検索やグラフ表示を行う Web ページを CGI で作成した。Figure 5 に異常波形データ収集システムの Web 画面を示す。ユーザーの操作性向上を目的として Web サービスには主に以下のような実装を行った。

- 期間指定による波形データ検索機能において、Map/Reduce フレームワークで実装し、並列処理による応答時間の短縮を行った。
- 信号名での波形データ検索機能において、Cassandra の CQL だけでは実装できないワイルドカード(%,*)の利用が可能なあいまい検索機能を、Python の正規表現ライブラリを使用して実装した。
- 信号名を入力する際に信号名の頭の数文字を入力すると、Cassandra から取得した信号名をグルーピングして候補信号名として表示する機能を実装した。

これらの Web サービスは CGI 以外にも C 言語のライブラリなどからも呼び出すことができる。

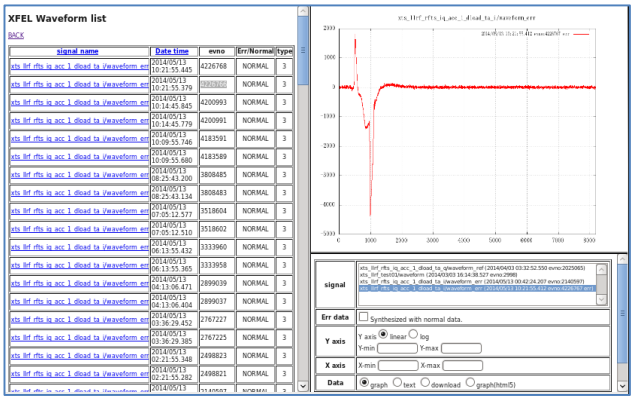


Figure 5: Web page of abnormal waveform acquisition system.

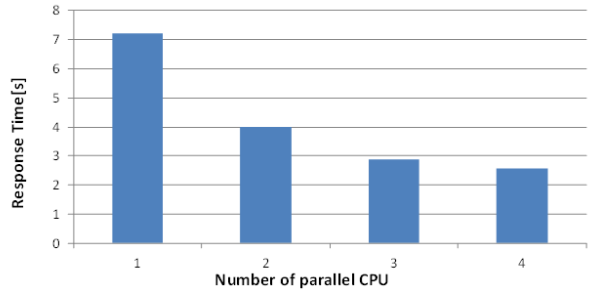
3.2 Map/Reduce フレームワークのパフォーマンス測定

テストスタンドに Table 1 に示す構成でテスト環境を構築し、Cassandra データベースに約 10 万件の擬似波形データを保存した。擬似波形は 1 波形あたり 16bit×8192 点のデータサイズとなる。そのテスト環境で、テスト用クライアントプログラムを実行して Web サービスの Map/Reduce フレームワークの応答時間を測定した(Figure 6)。

テスト a)では、1 個の Web サービスに対し、2000 件の異常波形データのリストを取得するテストプログラムを実行し、Web サービスの並列数を 1 から 4 に変化させて応答時間を確認した。並列数を上げると並列処理によりデータ取得の応答時間が短縮される。ただし、OS など他のプログラムも CPU を使用しているため、CPU のコア数に比例して応答時間が短縮されるわけではない。

テスト b)では、100 件の異常波形データのリストを抽出するテストプログラムを 10 個から 50 個起動してデータ要求処理の数を増加させて、Web サービス 1 個から 3 個を複数のサーバーに分散起動した際の応答時間を確認した。データ取得要求の数が増えるにつれ、1 つの Web サービスでは応答時間の悪化が大きいが、複数の Web サービスを起動することで自動的にクライアントからのアクセスが分散されて、応答時間の悪化が減少している。

a)



b)

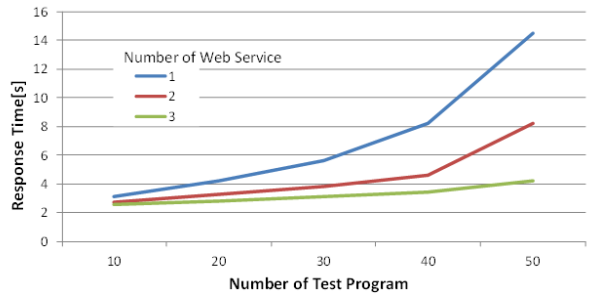


Figure 6: Performance test of Web Service. a) Parallel processing test. b) Distributed processing test.

4. まとめ

異常波形データ収集システムでは、Cassandra からのデータ取得処理は Map/Reduce フレームワーク構成の Web サービス化を行った。これによりデータ取得処理は並列処理により比較的簡単に応答時間の短縮が行える。また、簡単な手順で Web サービスの追加を行うこともでき、クライアントアプリケーションや Web サービスのソースコードを一切変更することなく、比較的簡単に分散処理によるスケールアウトを行うことができる。そして、Web サービスで実装したデータ取得処理は複数の異なる言語から共有利用することもでき、クライアントアプリケーションの開発生産性を向上することができる。

Web サービスを実装した異常波形データ収集システムは 2014 年の夏期停止期間に実機に導入する予定である。

Table 1: Test Environment Information

Database Server	CPU: Intel Xeon (R) CPU X3470 2.93GHz x4core RAM: 8GB HDD: 1.6TB x2 OS: Cent OS 6.2(x64)
Cassandra Database	Cassandra version: 1.2.1 Java: JDK 1.6.0_45 Java Heap: 4GB Node: 3 Replication: 2
Web Service Server and Test Client	CPU: Intel Xeon (R) CPU X3470 2.93GHz x4core RAM: 4GB HDD: 1.6TB x2 OS: Cent OS 6.2(x64)

参考文献

[1] R.Tanaka, et al., “INAUGURATION OF THE XFEL FACILITY, SACLA, IN SPRING-8”, Proc. of ICALEPCS2011.
[2] M. Yamaga, et al., “Event-Synchronized Data-Acquisition System for SPring-8 XFEL”, Proc. of ICALEPCS2009.
[3] M. Yoshioka, et. al., “SACLA の RF 異常波形データ収集フレームワーク”, 第 11 回加速器学会年会 2014.
[4] T. Ohshima, et al., “SACLA における RF 異常波形データの捕捉”, 第 11 回加速器学会年会 2014.
[5] http:// http://cassandra.apache.org
[6] http://ja.wikipedia.org/wiki/JavaScript_Object_Notation
[7] http://www.tornadoweb.org/en/stable/
[8] http://e-words.jp/w/MapReduce.html