5A-03

異種データストア向けデータストア移行技術の提案と評価

大越　淳平†　　西川　記史†
(株)日立製作所　研究開発グループ†

1 序論

近年，クラウドファースト[1]や DevOps[2]の潮流に伴い，オンプレミス・クラウド間や開発・本番環境間において，データ構造の定義であるスキーマや格納されたデータをデータストア間で移行するデータストア移行のニーズが高まっている。一方で，多様なアプリケーションへの対応や，DevOps サイクルの迅速化を目的に，伝統的な RDB（Relational Database）に加え，NoSQL（Not only SQL）と呼ばれるデータストアを混在させるケースが増加している。

これらの社会・技術潮流を受け，データストアの移行においても，従来の RDB のみならず，NoSQL も含めた異種データストアに対応したデータストア移行技術が求められている[1-3]。しかし，異種データストアの移行においては，（1）API（Application Programming Interface）や（2）データモデルがデータストアごとに異なり，データストアの組み合わせで移行プログラムの開発が必要となることから，移行工数が増大するという問題が生じる（図 1）。

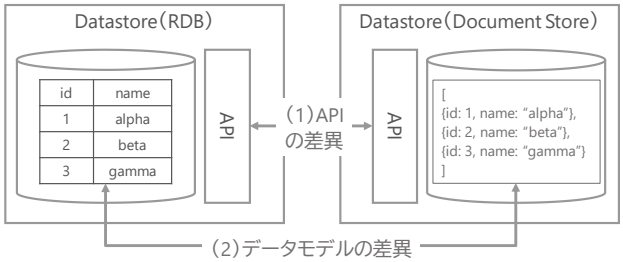

図 1 異種データストアの移行における問題

本研究の目的は，以上の状況を鑑み，異種データストアの移行において生じる移行工数の増大を解決する，異種データストア移行技術を提案することにある。

2 提案システム

移行工数の増大はデータストアごとの API・データモデルの差異に起因している。本研究では，これらの問題に対し，共通 API と共通データモデルにより，データストアの統一的な操作体系を提供することで解決を図る（図 2）。本システムの特徴を以下に順に述べる。

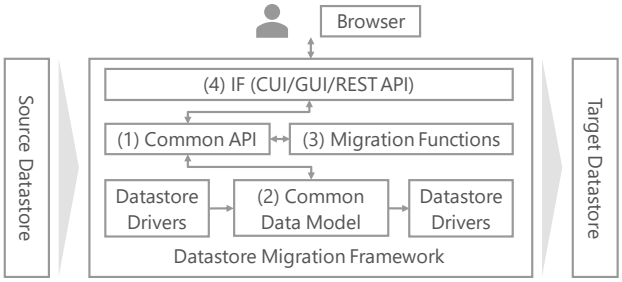

図 2 提案システム

(1) 共通 API：各移行機能や共通データモデルに対する共通 API を提供する（表 1）。共通 API は，各データストアに対するプリミティブな操作を提供する低レベル共通 API（#1-4）と低レベル API を用いて実装する各移行機能に対応した高レベル共通 API（#5）からなる。

表 1 共通 API（主要部分の一部）

#	API	概要
1	DataStoreContext	データストアに対するエントリポイントの提供
2	DataStoreSchema	スキーマに対する IF の提供
3	DataStoreTable	テーブルに対する IF の提供
4	DataStoreColumn	カラムに対する IF の提供
5	DataStoreMigration	データストアの移行に関する IF の提供

(2) 共通データモデル：各データストアに格納されたデータを，データストアが想定するデータモデルを意識せずに操作可能とする共通データモデルを提供する。

(3) 移行機能群：アセスメント，データ移行，検証など，データストアの移行に必要な各移行機能を提供する。

(4) IF：CUI/GUI（Character/Graphical User Interface）/REST API 等による IF（Interface）を提供する。

なお，低レベル共通 API・共通データモデルを提供するコンポーネントとして，OSS（Open Source Software）である Apache MetaModel[4]（以下，MetaModel）を採用した。MetaModel は異種データストアに対する統一的な操作を共通 IF として提供しており，当該 OSS を用いることで関係モデルに類似したデータモデルにて異種データストアを扱うことが可能となる。

一方で，MetaModel ではデータ移行に必要な各種機能（アセスメント，データ移行等）が提供

Proposal and Evaluation of Datastore Migration Technology for Heterogeneous Datastores
Jumpei Okoshi, Nishikawa Norifumi,
Hitachi, Ltd. Research & Development Group

[1] 企業が情報システムの設計・移行に際してクラウドサービスの採用を第一にする方針
[2] ソフトウェアの開発・運用を密連携させることでビジネス価値を向上させる開発手法

表 2 CUI 実装によるフィージビリティ検証（太字が入力）

```
L1: scala> val postgres = new DataStoreContext(platform = "postgresql", …)
L2: scala> postgres.getTableNames
L3: res: List[String] = List(customer, lineitem, nation, orders, part, partsupp, …)
L4: scala> postgres.getTableByName("customer").getColumnByName("c_name")
L5: res: dmfw.DataStoreColumn = Column[name=c_name,columnNumber=1,type=VARCHAR, …]
L6: scala> val mongodb = new DataStoreContext(platform = "mongodb", …)
L7: scala> mongodb.getTableNames
L8: res: List[String] = List(user)
```

されておらず，特に，ネスト構造の取り扱い[3]など異種データストアの移行で生じる問題に対応できない。本研究では，当該 OSS をベースとしつつ，（1）の高レベル共通 API，（3）の移行機能群を構築することで，異種データストア間のデータ移行を可能とした。

３フィージビリティ検証

本研究では，CUI を備えた提案システムをプロト実装し，PostgreSQL・MongoDB[4]を対象に API の動作を確認することで，異種データストアの移行における基本機能のフィージビリティを検証した。表 2 に実行結果の一部を示す。提供された共通 API（表 1-#1）を用い DataStoreContext を定義することで，異種データストアに対する統一的な操作が可能となっていることが分かる。具体的には，PostgreSQL や MongoDB に対し，DataStoreContext（表 1-#1）を定義（L1，L6）することで，getTableNames によるテーブル一覧の取得（L2，L7）や，メソッドチェインを用いた getTableByName による DataStoreTable（表 1-#3）の取得，さらに getColumnByName による DataStoreColumn（表 1-#4）の取得（L4）が実行されている。本 API の他に，データ取得 API 等を組み合わせることにより，異種データストア間のデータストア移行が可能となる。

４評価結果

本研究では，工場向け IoT システムを想定したデータストア移行のユースケースにおいて，工数の削減効果を評価した。以下に詳細を述べる。

ユースケース：工場に設置された IoT 機器から収集されたセンサデータの可視化・分析を行う IoT システムの開発環境と本番環境のデータストアを移行する。

移行対象：データストア，ETL（Extract/Transform/Load）フロー，アプリケーションの 3 種を移行する。

移行工数：ユースケースにおける移行プロセスと文献[5]より各プロセスの工数を算出した。

削減効果：ETL アプリケーションの再利用における社内事例から本フレームワークの適用による削減効果をフルスクラッチ開発比で 50%とした。

評価結果：

- 計画（Planning）：本フレームワークの適用によりデータストア部分（全体の 3 分の 1）の工数の 50%が削減される。
- 移行（Migration）：本フレームワークの適用によりデータストア部分の 50%が削減される。
- 移行後（Post migration）：削減効果なし。

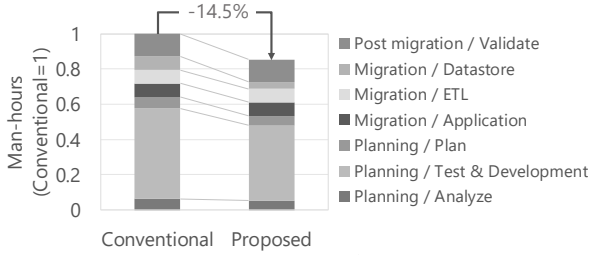

図 3 工数の削減効果

以上の結果により，総計の工数のうち，14.5%が削減されることがわかった。

５結言

本研究では，異種データストアの移行で生じる移行工数の増大に対し，共通 API と共通データモデルを備えたデータストア移行技術を提案した。また，CUI を備えたプロト実装により基本機能のフィージビリティを検証するとともに，工場向け IoT システムを想定したデータストア移行のユースケースにおいて工数の 14.5%を削減可能であることを示した。

今後の課題として，実環境における検証，および移行工数の削減効果の評価が挙げられる。

参考文献

[1] M. Scavuzzo et al. Interoperable Data Migration between NoSQL Columnar Databases. EDOCW, 2014.
[2] L. Rocha et al. A Framework for Migrating Relational Datasets to NoSQL. ICCS, 2015.
[3] A. Bansel et al. Cloud-Based NoSQL Data Migration. PDP, 2016.
[4] https://metamodel.apache.org
[5] https://www.scnsoft.com/blog/salesforce-data-migration

[3] ネスト構造を許すデータモデルからネスト構造を許さないデータモデルへの変換等
[4] RDB・NoSQL それぞれにおいて代表的なデータストア