Linked Data を用いたサービス連携アーキテクチャの提案と評価

M2011MM041　小島　弘誉

指導教員　青山　幹雄

1. はじめに

クラウドなどのサービスをブローカで連携する方法が注目されているが, 連携には大量のデータを扱えるスケーラビリティが求められている. 本稿は Linked Data を用いた連携アーキテクチャを提案する. サービス連携を制御と処理の二層に分け, Linked Data で振舞いを制御することにより, 非集中な連携を可能とする. これにより, スケーラブルなデータ連携を実現する.

2. 研究課題

2.1. スケーラブルなデータ連携が困難

不特定多数のサービス連携では, 連携に参加しているサービスの負荷がブローカに集中する. これにより, スケーラブルなデータ連携が困難になっている.

2.2. クラウドサービスの疎結合な連携が困難

不特定多数のサービス連携では, 連携参加者の時間, 位置, 状態によらずデータを送受信する必要がある.

3. 関連研究

(1) ESB(Enterprise Service Bus): SOA(Service-Oriented Architecture)に基づきエンタープライズ全体のサービス統合を実現する技術である[1].

(2) Linked Dataを用いた技術連携: セマンティクスの違いに対して, Linked Dataを用いて多様な設計データを連携する研究がされている[4].

4. 連携アーキテクチャのアプローチ

連携を制御と処理の二層に分け, Linked Data で振舞いを制御するアーキテクチャにする(図 1). このアプローチをとることでブローカへの負荷の集中を各クラウドサービスに分散させ, 負荷が非集中の連携アーキテクチャを実現する. これを二層 LDA(Linked Data Architecture)と呼ぶ.

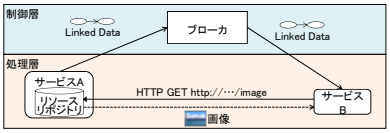

図 1 提案のアプローチ

制御層はPublish/Subscribe Architecture(以降PSA)[2]に基づき連携参加者の時間分離, 空間分離を可能とする. 処理層はリソース指向に基づきデータ交換することで連携参加者の状態分離を可能とし, 疎結合な連携を実現する.

5. 二層LDAの提案

5.1. 二層LDAの構造

二層 LDA の構造を図 2 に示す. メタレイヤで連携の制御, ベースレイヤで連携の処理を行う.

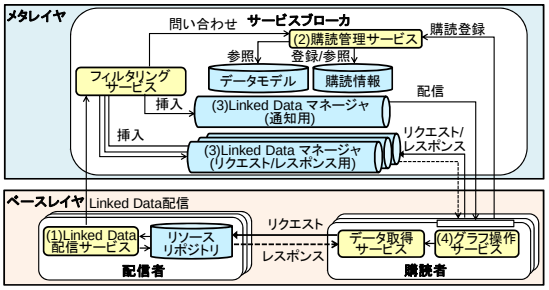

図 2 二層 LDA の構造

(1) Linked Data 配信サービス

リソースを指定する Linked Data を配信する. 配信される Linked Data は次の二種類ある. (i)配信者のリソースリポジトリで保持しているデータを指定する Linked Data と(ii)配信者以外のサービスのリソースリポジトリで保持しているデータを指定する Linked Data である.

(2) 購読管理サービス

購読者の情報を管理する. 購読登録では購読者のエンドポイント, 取得するLinked Dataの条件, 振舞いが登録される.

(3) Linked Data マネージャ

購読者の振舞いに合わせて Linked Data を配信する. 購読者の連携パターンには通知型とリクエスト/レスポンス型がある. 通知型は全ての購読者のLinked Dataを一つのメッセージキューで管理して配信を可能とする. リクエスト/レスポンス型では任意のタイミングでの配信を可能とするため, 配信者毎にメッセージキューを用意し, 管理する.

(4) グラフ操作サービス

配信されたLinked Dataから取得したいデータのURIを抽出する.

5.2. 二層LDAの振舞い

複数の購読者と複数の配信者のデータ交換を例に二層 LDA の振舞いを図 3 に示す. 配信者(サービス X), 購読者(サービス A, サービス B)を記述する. 購読者は異なる振舞いを持つサービスである. サービス A はリクエスト/レスポンス型, サービス B は通知型のサービスである. また, サービ

スYはサービスXが配信したLinked Dataが指すリソースリポジトリを保持しているとする.

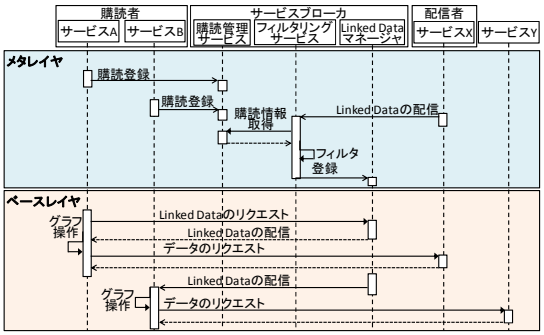

図 3 二層 LDA の振舞い

5.2.1. 全体の振舞い

連携全体の振舞いを説明する.

(1) 購読登録

購読者のサービスA, サービスBが購読登録する.

(2) Linked Dataの配信

配信者のサービスXがサービスブローカにLinked Dataを配信する.

(3) Liked Dataマネージャへの登録

サービスブローカは購読管理サービスから購読情報を要求し, 取得する. 購読情報が記述されている情報に従ってLinked Dataをフィルタリングし, Linked Dataに購読者の振舞い, 配信時間, 購読者の宛先などを指定して, 購読管理サービスに登録する.

(4) Linked Dataの配信

サービスAのようなリクエスト/レスポンス型のサービスの場合は購読者からLinked Dataを要求し, Linked Dataを取得する. そして, グラフ操作サービスを用いてLinked Dataから要求するURIを取得する. そのURIを用いてサービスXのリソースリポジトリにアクセスし, データを取得する. サービスBのような通知型のサービスの場合はサービスブローカから購読情報に記載されたトリガに従ってLinked Dataを配信する.

(5) データの取得

グラフ操作サービスを用いて, Linked Dataから要求するURIを取得する. 取得したURIを用いて, サービスYからデータを取得する. サービスYはLinked Dataを配信していないが, サービスXが配信したLinked Dataがサービス Yのリソースリポジトリを指しているために, サービスYのリソースリポジトリにアクセスし, データを取得する.

5.2.2. 二層分割による効果

連携アーキテクチャを二層に分けることによる効果をメタレイヤ, ベースレイヤに分けて示す.

(1) メタレイヤ

メタレイヤでは連携参加者が疎結合な連携を可能とするために振舞いの制御を行う. 疎結合な連携を可能とする二つの性質を示す.

(a) 空間分離

連携参加者はお互いの位置を知らなくても連携することを可能とする. ブローカが連携参加者のエンドポイントを管理し, 登録された購読要求に従って, Linked Dataを配信するため, 連携途中で配信者や購読者のエンドポイントが変更されても, 連携が可能である.

(b) 時間分離

連携参加者はLinked Dataの配信, 取得を任意のタイミングで行うことを可能とする. ブローカは配信されたLinked Dataをメッセージキューで保持するため, Linked Dataを任意のタイミングで連携することが可能となる.

(2) ベースレイヤ

ベースレイヤでは異なるリソースリポジトリに配置されたデータをリンクさせたLinked Dataを用いてデータを取得する. それにより, データ交換の分散とデータ保持の分散を可能とし, 負荷が非集中の連携を実現する.

(a) データ交換の分散

購読者がLinked Dataの指すリソースリポジトリから直接データ交換する. そのため, 不特定多数の連携でも, ブローカに負荷は集中せず, 購読者とリソースリポジトリを保持するサービスに負荷分散する.

(b) データ保持の分散

二層LDAではブローカがデータを保持しない. データは配信者, または, リソースリポジトリを保持しているサービスに配置されるため, ブローカに集中していたデータ保持の負荷を配信者と他のリソースリポジトリに分散させることが可能となる.

6. プロトタイプによる実証

6.1. プロトタイプの構成と実現

実装したプロトタイプの構成を図4に示す.

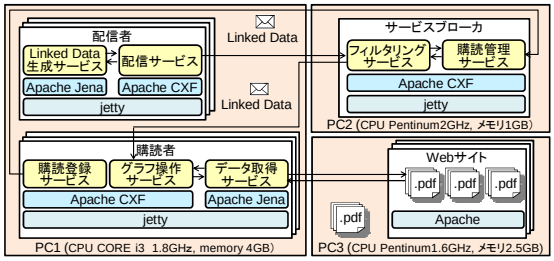

図 4 二層 LDA プロトタイプの構成

連携時間を正確に計測するため, 配信者と購読者は同じPC上に配置した. ネットワーク遅延などで連携時間の計測結果に影響を与えないため, 全てのアクタを同一のネットワ

ーク上に開発した. プロトタイプの実行環境を示す.

(1) サーバソフトウェア

配信者, 購読者, サービスブローカにはサーバソフトウェアに Jetty 2.7.1 を用い, Web サイトには Apache 2.0.64 を用いた. 近年, Web サービスは Apache Maven などのプロジェクト管理ツールを用いて開発される. その際にはサーバソフトウェアに Jetty を用いることが一般的である. また, Web サイトではサーバソフトウェアに Apache を用いることが一般的であるため, Apache を用いた.

(2) Web サービス

Web サービスは Apache CXF 2.7.1 を用いて開発した. Apache CXF には PSA を実現するライブラリが用意されているため, Apache CXF を用いた.

(3) Linked Data 処理機能

Linked Data 処理機能は Apache Jena 2.9.3 を用いて開発した. プロトタイプでは Linked Data を作成する機能と Linked Data から要求する URI を検索する機能を開発する必要があったため, SPARQL のライブラリが用意されている Apache Jena を用いた.

6.2. 適用ユースケース

二層 LDA を定期購読ユースケース(図 5)に適用する. アクタは執筆者, 編集者, e コーマス, 購読者の四つである. 編集者は Linked Data を用いて三人の執筆者の情報や原稿のデータを管理しているとする. 執筆者は画像や PDF ファイルなどサイズの大きいデータを保持し, それらを Web サイトで公開しているとする.

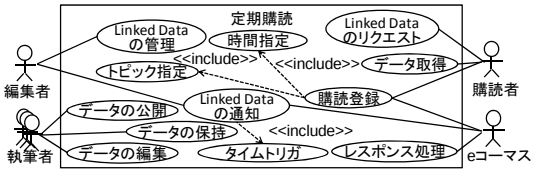

図 5 定期購読ユースケース

6.3. Linked Dataの構造

Linked Dataの構造を図 6 に示す. ユースケースに基づき, 三人の執筆者が共著の本を執筆することを例にLinked Dataを定義した. 構造はLinked Data Platform[3]に基づいている.

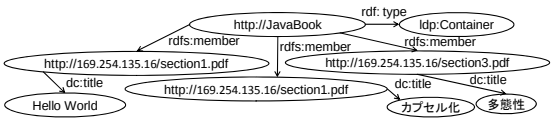

図 6 Linked Data の構造

6.4. プロトタイプの振舞い

プロトタイプの振舞いを図 7 に示す. ユースケースでは Linked Data が三つのファイルを指しているため, Web サイトに三回データをリクエストしている.

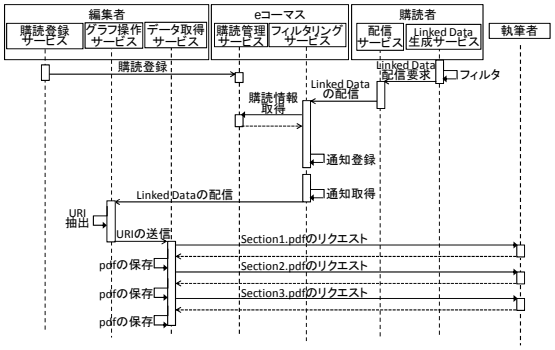

図 7 プロトタイプのシーケンス図

6.5. 連携時間の計測

データサイズを 1KB, 25KB, 50KB, 100KB, 200KB と変更し, 連携時間を計測した. Linked Data を 3,000ms 間隔で 100 回送信して全体の連携時間を計測した. 計測結果を表 1 に示す. 全体の連携時間とは配信者が Linked Data を配信してから, 購読者がデータを取得し, 保存するまでの時間である.

表 1 計測結果

データサイズ(KB)	1	25	50	100	200	500
平均連携時間(ms)	74.61	112.68	183.6	385.88	424	1094.09
標準偏差	8.93	11.26	8.74	19.77	86.21	140.52

7. 評価と考察

7.1. 関連研究との比較によるスケーラビリティの評価

二層 LDA と PSA を比較することでスケーラビリティを評価する. 例として共著の文書や画像アルバムなど複数のデータの配信を想定して振舞いとデータ配置の差異を図 8 に示す.

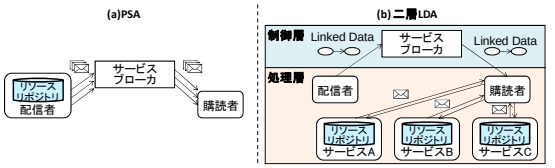

図 8 各アーキテクチャの振舞いの比較

各アーキテクチャにおける連携全体の負荷をデータ交換, データ保持の観点から比較する(表 2).

表 2 各アーキテクチャにおける負荷の比較

アーキテクチャ	振舞い	データ配置
(a) PSA	ブローカに集中	ブローカに集中
(b) 二層 LDA	サービスに分散	サービスに分散

(1) 振舞いの比較

PSA ではブローカを介してデータ交換を行う. そのため, データ交換がブローカに集中する. しかし, 本研究が提案する二層 LDA ではブローカを介さずデータ保持者と直接データ交換する. そのため, データ交換が各サービスに分散する.

(2) データ配置の差異

PSA では連携する全てのデータを一度ブローカに保持する必要がある. そのため, データの配置がブローカに集中する. しかし, 二層 LDA ではブローカがデータを保持する必要はない. また, Linked Data を配信するサービスは必ずしもデータを保持する必要はなく, 配置の異なるソースリポジトリ指す Linked Data を配信することが可能である. そのため, 二層 LDA はデータの配置を分散させることが可能である.

以上から, 二層 LDA はデータ交換の分散とデータ配置の分散を可能にし, 全体として負荷が非集中の連携を実現する. よって, 二層 LDA は他のアーキテクチャよりスケーラビリティが高いと評価できる.

7.2. 二層LDAの妥当性の確認

不特定多数のデータ連携を実現する際に重要な二つの項目を評価することで二層 LDA の妥当性を確認した.

(1) 連携参加者の結合度

不特定多数の連携アーキテクチャを実現するためには連携参加者の状態によらず連携が可能である必要がある. 連携参加者がエンドポイントを変更しても連携を可能としなければならない. 二層 LDA は制御層を PSA に基づくことで, 連携参加者の時間分離, 空間分離を実現した. 処理層はリソース指向に基づいてデータを取得するため, 状態分離を可能とする. 二層 LDA は時間分離, 空間分離, 状態分離を可能とし, 疎結合な連携を実現した. そのため, 結合度は低い.

(2) 連携時間

連携時間をアーキテクチャの振舞いの比較, 計測結果の比較に分けて評価する.

(a) アーキテクチャの振舞いの比較による評価

二層 LDA の Linked Data 配信時間 T1, データ取得時間 T0+T2, PSA のデータ配信時間 T2 と定義する(図 9). 二層 LDA の全体の連携時間は T1*2+T0+T2 となる. PSA はブローカの全体の連携時間は T2(ms)となる. これらから T2*2>T1*2+T2+T0 より, T2/2>T1 が求まる. そのため, データサイズが大きい連携では二層 LDA の連携時間の方が短い.

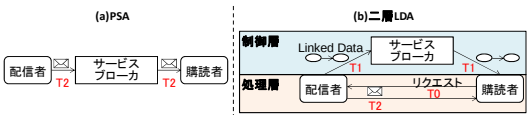

図 9 連携時間の比較

(b) 計測結果の比較による連携時間の評価

提案プロトタイプと PSA プロトタイプの計測結果を用いて, 静的な評価が正しいことを確認する. データサイズを 1KB, 25KB, 50KB, 100KB, 200KB, 500KB と変更し, 連携時間を計測する. Linked Data を 3,000ms 間隔で 100 回送信して平均の連携時間とその近似式を求めた(図 10).

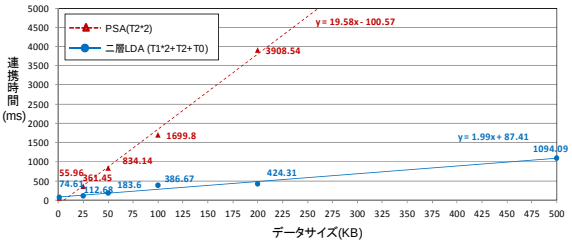

図 10 計測結果の比較

(c) データサイズが 10.68KB 未満では二層 LDA の方が PSA より連携時間が長いが, 10.68KB 以上では二層 LDA の方が PSA より連携時間が短い. 二層 LDA と PSA の傾きが 10 倍異なるため, サイズが100KB のデータでは二層 LDA の連携時間が PSA の約 1/4 倍となる.

以上から, 妥当な連携アーキテクチャと評価できる.

8. 今後の課題

8.1. 複数の連携参加者用いた負荷の計測

多数の実機を用いて, サービス, ブローカにかかる負荷を計測することが必要である.

8.2. Linked Dataの信頼性管理

Linked Data が指すデータがリポジトリに存在していない場合や Linked Data の指し先が異なっていることが考えられるため, Linked Data の信頼性を管理する必要がある.

9. まとめ

本稿は Linked Data を用いたサービス連携アーキテクチャを提案した. 連携を制御と処理の二層に分け, Linked Data で振舞いを制御することにより, 非集中な連携を可能とし, スケーラブルなデータ連携を実現する. また, プロトタイプを開発し, 有効性を示した.

10. 参考文献

[1] D. A. Chappell, Enterprise Service Bus, O'Reilly, 2004.
[2] P. T. Eugster, et al., The Many Faces of Publish/ Subscribe, ACM Computing Survey, Jun. 2003, pp. 114-131.
[3] Linked Data Platform Working Group, http://www.w3.org/2012/ldp/wiki/Main_Page.
[4] M. Graube, et al., Linked Data as Integrating Technology for Industrial Data, Proc. NiBS 2011, Sep. 2011, pp. 162-167.