Vol.2016-HPC-155 No.4
2016/8/8

モデルの自動再学習機能を備えた 予測モデルに基づくイベント処理ミドルウェア

竹内 幹雄[1,a)]

概要：内外からのメッセージとして受け取るイベントに素早く反応することで，企業は機会を有効活用したりリスクを最小化することができる．素早い反応が重要なため，それを可能にするイベント処理ミドルウェアは多くの企業で利用されるコア技術の一つである．しかしながら実際には，素早く反応しさえすれば収益が上がるわけではなく，収益を最大化する反応には，直前のイベントだけではなく過去のイベントなどより多くの情報からなる文脈が必要になることも多い．文脈の分析は通常一回のイベント処理に許される時間内には終わらないため，文脈から事前に非同期に学習した予測モデルを用いて，素早くイベントを評価する方法が用いられる．ある時点の文脈から学習したモデルは，文脈の変化に伴い徐々に現実から乖離していくため，評価結果を正確に保つには，最新の文脈を用いた定期的なモデルの再学習が必要である．本発表では我々が設計及び試作した，モデルの自動再学習機能を備えた予測モデルに基づくイベント処理ミドルウェアを紹介する．

キーワード：イベント処理，機械学習，予測モデル，再学習

Analytics Oriented Event Processing Middleware with Automatic Retraining Feature

Mikio Takeuchi[1,a)]

Abstract: Businesses that receive events in the form of messages and react to them quickly can take advantage of opportunities and avoid risks as they occur. Since quick reactions are important, event processing middleware is a core technology in many businesses. However, the need to act quickly must be balanced against the need to act profitably, and the best action often depends on more context than just the latest event. Unfortunately, the context is often too large to analyze in the time allotted to processing an event. Instead, out-of-band analytics can train an analytical model, against which an event can be quickly scored. Furthermore, it is insufficient to train the model once, since it grow stale when new data arrives. Instead, automatic retraining of the model with the up-to-date context can keep the score of an event accurate. In this presentation, we introduce the design and prototype of an analytics oriented event precessing middleware with automatic retraining feature.

Keywords: Event Processing, Machine Learning, Predictive Modeling, Retraining

1. はじめに

予測モデル [1] に基づくイベント処理 [2] の概要を図 1 に示す．モデルの学習はイベント処理とは独立に行われる．学習はデータストアに蓄えられた大量のデータを読み込み，学習済みモデルを生成する．既存のデータを更新したり，新たなデータを生み出すことはない．学習はインクリメンタルにもしくは一定の間隔で繰り返し行われる．学習により得られたモデルは，常に少し古い可能性がある．

学習済みモデルを用いた評価はイベント処理の一部として行われる．イベントプロセッサが呼び出す「エージェン

[1] 日本アイ・ビー・エム (株) 東京基礎研究所
IBM Research–Tokyo
[a)] mtake@jp.ibm.com

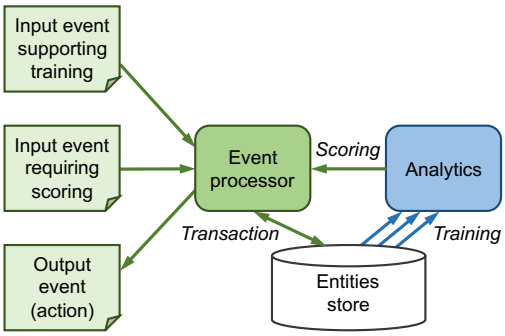

図 1 予測モデルに基づくイベント処理の概要

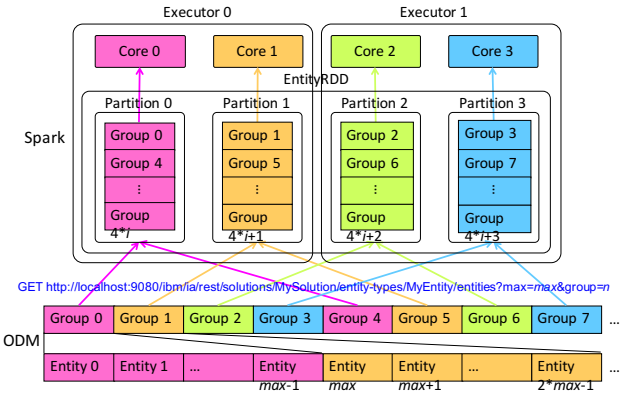

図 2 Entity RDD の概観

ト」と呼ばれるハンドラーがイベントを処理する際に，その時点の最新の学習済みモデルを用いてイベントの内容を評価し，最適な反応（アクション）を導き出し，実行する．

本ミドルウェアでは，モデルの学習には，Apache Spark [3] 上で動作する機械学習ライブラリ MLlib [4] を用いた．学習済みモデルを用いた評価には，REST API が利用可能なオープンソースの評価サーバ Openscoring [5] を用いた．イベントプロセッサとデータ（エンティティ）ストアには，弊社の製品 “Operational Decision Manager: Decision Server Insights” [6]（以下 ODM Insights と表記する）を用いた．学習は一定の間隔で繰り返し行う方法をとった．本ミドルウェアは，アプリケーションに依存しない再利用可能なシステム部と，それを用いて動作するアプリケーションからなるが，それらはいずれも公開 API のみを用いて実装され，ODM Insights 上で動作するソリューション（デプロイメントの単位）の例として一般公開されている．

2. MLlib によるモデルの学習

学習のためのジョブは MLlib を用いて実装され，Spark のクラスタ上で非同期に実行される．本ミドルウェアは，アプリケーションの例として，MLlib が提供する第一世代のアルゴリズムのうち，サポートベクターマシン SVMWithSGD を用いたジョブを含む．第一世代のアルゴリズムは，Resilient Distributed Dataset (RDD) と呼ばれる Spark の API の上に構築されている．RDD はデータの主記憶上でのキャッシングをサポートする，書き換え不可能なデータ構造であり，機械学習のような同じデータを繰り返し読むジョブの実行に適している．また RDD はデータのパーティショニングをサポートしており，ジョブの並列分散実行に適している．MLlib によるモデルの学習には，学習に用いるデータ（トレーニングデータ）を RDD として表現する必要がある．

ODM Insights はデータストアに保存されたデータ（エンティティと呼ぶ）を JSON 形式で読むための REST API を提供している．それを用いると，指定した順番から指定した個数のエンティティを一度に読むことができる（例えば 200 番目のエンティティから 100 個を一度に読むなど）．

我々はこの API の上にエンティティを読むための RDD（Entity RDD と呼ぶ）を Scala 言語 [7] を用いて実装した．図 2 に executor 数が 2 かつ executor あたりの core 数が 2 の場合の概観を示す [*1]．各パーティションはそれを読み込んだ executor によりキャッシュされるため，エンティティのデータストアからの読み込みは，各グループ（読み込み単位）につき一度だけ行われる．

学習が完了したら，学習済みモデルは MLlib の内部形式から Predictive Model Markup Language (PMML) [8] 形式のファイルに書き出される．PMML は予測モデルを表現するための XML ベースの標準フォーマットであり，異なる学習ソフトウェアと評価ソフトウェア間で学習済みモデルを引き渡すのに都合が良い．

3. Openscoring による学習済みモデルを用いた評価

PMML 形式の学習済みモデルを REST API を用いて Openscoring にデプロイすることでそのモデルを用いた評価が可能になる．デプロイ時に一意な model id を指定すると，それがそのモデルを用いた評価に用いる URL の一部になる．評価用の REST API は，テストデータを JSON 形式で送り，評価結果を JSON 形式で受け取る．複数個のテストデータを一度に評価するための REST API も用意されている．

4. ODM Insights によるイベント処理

ODM Insights のプログラミングモデルは，エンティティ，イベント，エージェントの 3 要素からなる．

4.1 エンティティ

エンティティはデータに相当する．本ミドルウェアは，2 種類のシステムエンティティを含む．Job は実行中の長時間走り続けるジョブやサービスの識別子を保持するため

[*1] Spark では，executor はプロセス，core はスレッドに相当し，ジョブは複数の executor と core を用いて並列分散実行される．

のシステムエンティティである．学習時には，Spark のクラスタマネージャーが返す学習ジョブの submission id を記録する．評価時には，Openscoring に学習済みモデルをデプロイした時に用いた model id を記録する．Application はアプリケーション固有の情報（アプリケーション名，エンティティの型名，説明変数と目的変数に用いるエンティティのフィールド名，学習ジョブの jar ファイル名と main クラス名など）を保持するためのもう一つのシステムエンティティである．本ミドルウェアではアプリケーションと実行中のジョブやサービスが一意に識別されているため，同一ミドルウェア上で複数のアプリケーションを動作させたり，一つのアプリケーションに対して，複数の異なる予測モデルを適用しより良い評価を可能にするアンサンブル手法 [9] を用いることも可能である．また再学習により得られた新しい学習済みモデルへの切り替えも，実行中の評価サービスを止めずに行うことができる．

アプリケーションは１つ以上のアプリケーションエンティティをトレーニングデータとして用いる．本ミドルウェアでは，サポートベクターマシンの学習用に SVMGuide1 をいうアプリケーションエンティティを定義している．

4.2 イベントとエージェント

イベントはメッセージ，エージェントはメッセージハンドラーに相当する．

イベントがキューに到着すると，イベントプロセッサがそのイベントに対応する複数のエージェントを指定された優先度で起動する．エージェントはそれを起動したイベントを読むことができ，また起動時に紐づけられたエンティティを読み書きすることができる．またエージェントは任意のキーを用いてデータストアからその他のエンティティを検索して読むことができる．エージェントには，専用のビジネスルール言語で記述された「ルールエージェント」[10] と Java 言語で記述された「Java エージェント」の２種類があり，本ミドルウェアのエージェントは全て Java エージェントである．Java エージェントは任意の Java コードを実行することができ，また新しいイベントをキューに送ったり，自分自身のエージェントを指定した時刻に再実行するようにスケジュールすることもできる．本ミドルウェアは，４つのイベントとそれらに対応する４つの Java エージェントからなる．図 3 にその全体像を示す．

4.2.1 モデルの学習時の挙動

StartTrainingAgent は StartTrainingEvent により起動される．このエージェントは，REST API を用いて Spark に学習ジョブをサブミットし，submission id を Job に保存し，AwaitTrainingEvent を送信する．

AwaitTrainingAgent は AwaitTrainingEvent により起動される．このエージェントは，REST API を用いて，Spark に学習ジョブの状態を問い合わせる．もしジョブがまだ

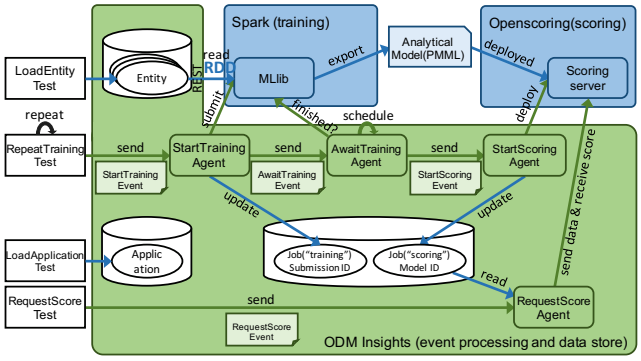

図 3 本ミドルウェアの全体像

実行中なら自分自身を一定時間後に再実行するようスケジュールする．もしジョブがすでに終わっていたら，StartScoringEvent を送信する．

StartScoringAgent は StartScoringEvent により起動される．このエージェントは，REST API を用いて，Openscoring に学習済みモデルをデプロイし，model id を Job に保存する．

4.2.2 学習済みモデルを用いた評価時の挙動

RequestScoreAgent は RequestScoreEvent により起動される．このエージェントは，Job から最新の学習済みモデルの model id を読む．またそれを起動したイベントからテストデータを読み，REST API を用いて，それを Openscoring に送信し評価結果を受け取る．

4.3 外部とのコミュニケーション

ODM Insights は，各種外部ソースからイベントを受け取るためのコネクタを提供している．また外部から ODM Insights にイベントを送信したり，データストア上のエンティティを読み書きするためのテストドライバを提供している．本ミドルウェアは，実際の運用時には必要ないが，テスト及びデモの目的で，アプリケーション情報をデータストアにロードする LoadApplicationTest，トレーニングデータをデータストアにロードする LoadEntityTest，StartTrainingEvent を一定の間隔で繰り返し送信する RepeatTrainingTest，RequestScoreEvent を送信する RequestScoreTest，などのテストドライバを含む．

5. 評価

実験に用いたハードウェアの仕様は以下の通りである．

- CPU: Intel(R) Xeon(R) CPU E5-2690 (8 cores) x 2
- Memory: 128GB
- OS: Red Hat Enterprise Linux Server 6.7 (x86-64)

ソフトウェアの構成等は以下の通りである．

- ODM Insights 8.8 (JVM: Java(TM) SE Runtime Environment (pxa6480sr1fp10-20150711_01 (SR1 FP10)), jvm.options: -Xmx30g)

Entity RDD	spark-csv
81.8	21.5

表 1 学習ジョブの学習部の時間 (秒)

- Spark 1.6.2 (JVM: Java(TM) SE Runtime Environment (pxa6480sr3-20160428_01(SR3)), spark-env.sh: SPARK_WORKER_INSTANCES=9 (1 for driver and 8 for executors), SPARK_WORKER_CORES=1, SPARK_WORKER_MEMORY=8g)
- Openscoring 1.2.15 (JVM: Java(TM) SE Runtime Environment (pxa6480sr3-20160428_01(SR3)))

まずは，別プロセスのデータストアから REST API を使ってトレーニングデータを直接読むことの現実性を確認するため，我々が作成した Entity RDD を使う場合と既存の spark-csv パッケージ [11] を使って同じデータをローカルファイルシステム上の CSV ファイルから読む場合との間で，サポートベクターマシンの学習ジョブの学習部分の実行時間を比較した．トレーニングデータのエンティティは 1000000 個，エンティティが持つ説明変数は 4 個 (4 次元)，学習の繰り返しは 100 回である．Entity RDD が REST API を発行する際の同時読み込みエンティティ数は，実験の結果最も良好な結果が得られた 13000 個を採用した．

結果は表 1 の通りである．Entity RDD 側に約 60 秒のオーバヘッドが確認できるが，これは REST API に固有の，データストア側で Java オブジェクトを JSON にシリアライズするコスト，HTTP での通信コスト，Spark 側で JSON を Java オブジェクトにデシリアライズするコストの合計と考えられる．これらのコストはデータサイズにほぼ比例するためトレーニングデータのサイズには注意する必要がある．オーバヘッドを削減するためのアイデアとしては，データストア側で事前に JSON 形式で持っておくことが考えられるが，今回は公開 API のみを用いて全てを作成する方針だったので採用していない．

また既に述べたように RDD のキャッシュ効果により，一回の学習ジョブの実行中に何回繰り返し学習しても，REST API は各エンティティの読み込み単位につき一度しか呼ばれないのに対し，一回の学習のコストはデータサイズに比例し，繰り返しの回数だけ単純に合計されるため，トレーニングデータが大きくなるとデータの読み込みを除く純粋な学習時間の割合が大きくなる傾向にある．学習時間を短縮するには，Spark の executor 数とそれに応えるための ODM Insights 側のコア数を同時に増やし並列度をあげることが有効と考えられる．また学習ジョブの学習済みモデルを PMML 形式で書き出す部分の時間は，トレーニングデータの読み込み手段とは無関係に約 1.1 秒だった．

次に，別プロセスの評価サーバが提供する REST API を使って評価を行うことの現実性を確認するための実験を行った．非同期に実行されスループットが重要な性能の尺度である学習ジョブと異なり，学習済みモデルを用いた評価はイベント処理から直接呼び出されるため，一回の評価の平均実行時間（レイテンシ）が重要である．

サポートベクターマシンの PMML 形式の学習済みモデルをデプロイ済みの Openscoring サーバに対して，Apache HTTP Server 付属のベンチマークツール ApacheBench [12] を用いて並列度 8 で合計 100000 回の評価リスエストを行った際の一回の評価の平均実行時間は 0.93 ミリ秒だった．ビジネスプロセスにおけるイベント処理は通常数 10 ミリ秒程度に収めることが理想とされているが，この結果は十分それを満たすため，現実性は高いと言える．

6. 関連研究

弊社は予測分析プラットフォーム製品群 SPSS [13] を提供している．ユーザが SPSS Modeler [14] を用いて記述した内部形式の学習ジョブを，SPSS Analytic Server [15] が MapReduce にコンパイルし，それを Apache Hadoop [16] もしくはその主記憶版である Main-memory MapReduce (M3R) [17] 上で高速に実行する．得られた内部形式の学習済みモデルを SPSS Collaboration and Deployment Services [18] にデプロイし，それを用いた評価を行う．SPSS 製品群は各種決定木，クラスタリング，ロジスティック回帰，ニューラルネットワーク，異常検知，ベイジアンネットワーク，サポートベクターマシンなどを含む豊富なアルゴリズムを独自の実装によりサポートする．SPSS Analytic Server は REST API による学習ジョブの実行や状態の問い合わせをサポートする．モデルの再学習に対するサポートは特にない．SPSS Collaboration and Deployment Services は REST API による学習済みモデルを用いた評価をサポートする．これらの SPSS 製品群は，本ミドルウェアの再学習を除く学習部分と評価部分を実現するのにほぼ十分な機能を持つが，SPSS Analytic Server がユーザ独自のデータソースを定義する方法を公開していないため ODM Insights のデータストアから直接トレーニングデータを読むことができないという問題がある．

PredictionIO [19] はモデルの学習と学習済みモデルを用いた評価の両方の機能をもつオープンソースの予測分析ソフトウェアである．モデルの学習と学習済みモデルを用いた評価の両方を MLlib を用いて実装している．REST API による操作が可能であるが，モデルの再学習については特に触れられていない．我々の Entity RDD を利用すれば，本ミドルウェアの再学習を除く学習部分と評価部分の両方を PredictionIO で置き換えられる可能性がある．

Oryx 2 [20] は機械学習に特化したオープンソースのラムダアーキテクチャの実装である．モデルの学習を行う Batch Layer, モデルの更新を行う Speed Layer, 学習済みモデルを用いた評価を行う Serving Layer が Data Transport

で結合された形をとっている．モデルの学習には MLlib を用いている．PMML 形式の学習済みモデルを用いた評価には，Openscoring と同じライブラリを用いている．モデルの再学習は，インクリメンタルなモデルの更新を前提としたアーキテクチャーになっている．本ミドルウェアが，REST API を提供するマイクロサービスを ODM Insights のエージェントによるイベントドリブンなプログラミングモデルで結合したものであるのに対して，Oryx 2 は Spark Streaming [21] のジョブを Apache Kafka [22] による Publish-Subscribe 型のデータドリブンなプログラミングモデルで結合したものであり，設計思想が大きく異なる．またモデルのインクリメンタルな更新と定期的な再学習のどちらが効率的かは，今後の検討が必要である．

Velox [23] は学習済み予測モデルの低レイテンシかつスケーラブルなマネジメントとサービス化に注目したプロジェクトである．学習済みモデルを用いた評価結果の現実への近似度（モデルの質と呼ぶ）を常に監視し，それが低下したことを検知して，一つ前の学習済みモデルに戻ったり，モデルの再学習を開始することが可能である．これを本ミドルウェアを組み合わせることで，より正確な評価を低コストで実現できる可能性がある．

まもなくリリースされる予定の Spark 2.0 では，MLlib で学習したモデルを，Spark 外でネイティブに評価する仕組みが用意される予定である．それを用いれば，本ミドルウェアの学習済みモデルを用いた評価を，ODM Insights が直接実行できるようになる可能性がある．

7. まとめ

Apache Spark 上で動作する機械学習ライブラリ MLlib と，REST API が利用可能なオープンソースの評価サーバ Openscoring を，弊社のイベント処理エンジン ODM Insights と組み合わせて，モデルの自動再学習機能を備えた予測モデルに基づくイベント処理ミドルウェアを設計及び試作した．

サポートベクターマシンをアプリケーションの例として，学習と学習済みモデルを用いた評価の両方を，イベントプロセッサとは別プロセスで行うことの現実性を確認した．前者は REST API を用いてトレーニングデータを読むことにまつわる比較的大きなオーバヘッドが観測されたが，トレーニングデータが 1000000 個の時は学習ジョブの実行時間が約 80 秒と極めて短いため問題にならず，また一定の間隔でのモデルの再学習が妥当である．データサイズが増えるとデータの読み込みのオーバヘッドが目立つようになるが，データの読み込みを除く純粋な学習時間の割合が高くなるため，Spark, ODM Insights 双方のコア数を増やし並列度を高めることが性能向上の鍵と考えられる．後者は現状でも十分な性能が得られることが確認できた．

本ミドルウェアは，ODM Insights のサンプルソリューションとして実装され，オープンソース [24] で一般公開されている．既に ODM Insights を利用されているユーザは README に従えばすぐに試すことができ，それを改造して自分のソリューションを開発することも可能である．また ODM Insights の IBM Bluemix 上での SaaS としての提供が計画されており，簡単に試用ができるようになる予定である．

本発表は，IBM リサーチで継続中のプロジェクト Middleware for Events, Transactions, and Analytics (META) に基づくものである．META プロジェクトでは，予測モデルに基づくイベント処理ミドルウェアにおけるモデルの再学習の他にも，並列分散プログラミング言語 X10 [25][26] の Java 言語へのコンパイル [27] や最適化 [28]，Java 言語との相互運用性 [29]，耐障害性のある実装 Resilient X10 [30]，X10 による分散アナリティックスライブラリの実装 [31]，ビジネスルール言語の一般化 [32] などの研究も行っている．それらの詳細については [33] を参照されたい．

参考文献

[1] F. Provost, and T. Fawcett. *Data Science for Business*. O'Reilly Media, July 2013. http://shop.oreilly.com/product/0636920028918.do
[2] K. Chandy, and W. Schulte. *Event Processing: Designing IT Systems for Agile Companies*. McGraw-Hill Education, October 2009. http://www.mhprofessional.com/product.php?isbn=0071633502
[3] Apache Spark. http://spark.apache.org/
[4] MLlib. http://spark.apache.org/mllib/
[5] Openscoring. http://openscoring.io/
[6] IBM Corporation. Operational Decision Manager: Decision Server Insights. http://www.ibm.com/support/knowledgecenter/SSQP76_8.8.0/com.ibm.odm.itoa/topics/odm_itoa.html
[7] The Scala Programming Language. http://scala-lang.org
[8] The Data Mining Group. Predictive Model Markup Language. http://dmg.org/pmml/v4-2-1/GeneralStructure.html
[9] Z.-H. Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC Press, June 2012. http://www.crcpress.com/Ensemble-Methods-Foundations-and-Algorithms/Zhou/p/book/9781439830031
[10] IBM Corporation. ODM Insights: Creating rule agents. http://www.ibm.com/support/knowledgecenter/en/SSQP76_8.8.0/com.ibm.odm.itoa.develop/topics/tsk_create_rule_agent_intro.html
[11] Spark SQL CSV data source. http://spark-packages.org/package/databricks/spark-csv
[12] Apache HTTP server benchmarking tool. http://httpd.apache.org/docs/2.4/programs/ab.html
[13] IBM Corporation. SPSS Software. http://www.ibm.com/analytics/us/en/technology/spss/
[14] IBM Corporation. SPSS Modeler. http://www.ibm.com/marketplace/cloud/spss-modeler/
[15] IBM Corporation. SPSS Analytic Server. http://www.ibm.com/software/products/spss-analytic-server/

[16] Apache Hadoop. http://hadoop.apache.org/
[17] A. Shinnar, D. Cunningham, V. Saraswat, and B. Herta. M3R: Increased Performance for In-memory Hadoop Jobs. *Proceedings of the VLDB Endowment*, Vol. 5, No. 12, pp. 1736-1747, August 2012. http://dx.doi.org/10.14778/2367502.2367513
[18] IBM Corporation. SPSS Collaboration and Deployment Services. http://www.ibm.com/software/products/spss-collaboration/
[19] PredictionIO. http://prediction.io/
[20] Oryx. http://oryx.io/
[21] Spark Streaming. http://spark.apache.org/streaming/
[22] Apache Kafka. http://kafka.apache.org/
[23] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The Missing Piece in Complex Analytics: Low Latency, Scalable Model Management and Serving with Velox. *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR '15)*, 2015. http://arxiv.org/abs/1409.3809v2
[24] An ODM Insights v8.8 project for machine learning with Spark and PMML based scoring. http://hub.jazz.net/project/mtake/odm-dsi-spark_ml/
[25] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. Praun, V. Sarkar. X10: An Object-oriented Approach to Non-uniform Cluster Computing. *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA '05)*, pp. 519-538, 2005. http://doi.acm.org/10.1145/1094811.1094852
[26] The X10 Programming Language. http://x10-lang.org/
[27] M. Takeuchi, Y. Makino, K. Kawachiya, H. Horii, T. Suzumura, T. Suganuma, and T. Onodera. Compiling X10 to Java. *Proceedings of the 2011 ACM SIGPLAN X10 Workshop (X10 '11)*, pp. 3:1–3:10, 2011. http://doi.acm.org/10.1145/2212736.2212739
[28] M. Takeuchi, S. Zakirov, K. Kawachiya, T. Onodera. Fast Method Dispatch and Effective Use of Primitives for Reified Generics in Managed X10. *Proceedings of the 2012 ACM SIGPLAN X10 Workshop (X10 '12)*, pp. 4:1-4:7, 2012. http://doi.acm.org/10.1145/2246056.2246060
[29] M. Takeuchi, D. Cunningham, D. Grove, and V. Saraswat. Java Interoperability in Managed X10. *Proceedings of the Third ACM SIGPLAN X10 Workshop (X10 '13)*, pp. 39-46, 2013. http://doi.acm.org/10.1145/2481268.2481278
[30] D. Cunningham, D. Grove, B. Herta, A. Iyengar, K. Kawachiya, H. Murata, V. Saraswat, M. Takeuchi, and O. Tardieu. Resilient X10: Efficient Failure-aware Programming. *Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '14)*, pp. 67-80, 2014. http://doi.acm.org/10.1145/2555243.2555248
[31] 千葉立寛，竹内幹雄，戸澤晶彦．分散プログラミング言語 X10 を用いたアナリティクスライブラリの実装と評価．情報処理学会第 145 回ハイパフォーマンスコンピューティング研究発表会研究報告 (IPSJ SIGHPC 145, SWoPP '14)，pp. 1-9, 2014. http://ci.nii.ac.jp/naid/110009808133
[32] A. Shinnar, J. Siméon, and M. Hirzel. A Pattern Calculus for Rule Languages: Expressiveness, Compilation, and Mechanization. *Proceedings of the 29th European Conference on Object-Oriented Programming (ECOOP '15)*, pp. 542-567, 2015. http://dx.doi.org/10.4230/LIPIcs.ECOOP.2015.542
[33] M. Arnold, D. Grove, B. Herta, M. Hind, M. Hirzel, A. Iyengar, L. Mandel, V. A. Saraswat, A. Shinnar, J. Siméon, M. Takeuchi, O. Tardieu, and W. Zhang. META: Middleware for Events, Transactions, and Analytics. *IBM Journal of Research and Development*, Vol. 60, No. 2-3, pp. 15:1-15:10, March-May 2016. http://dx.doi.org/10.1147/JRD.2016.2527419