タスクスケジューリングのためのコンテキスト指向ソフトウェアアーキテクチャに関する研究

M2023SE005 村上友太
指導教員：野呂昌満

1 はじめに

オペレーティングシステムにおいては，通常，プロセスの種類に応じて異なるスケジューリングポリシを適応する．スケジューリングポリシの適応方法は静的に定義されている．これらは本来，資源の占有状況に応じて動的に最適な適応を行うべきである．

この問題を解決するために，スケジューリング目的の変化に対応した動的スケジューリングに関する研究が行われている．スケジューリング目的の変化への対応とは，スケジューリングポリシの選択基準を動的に変更することを指す．動的スケジューリングはスケジューリングポリシの適応方法を動的に決定する．これには機械学習，とりわけ強化学習が用いられる [1]．機械学習では，入力データの分布が変化することでコンセプトドリフトが生じる．これは，Q-Learning[2] において Q-value が変化することに相当する．コンセプトドリフトに対応するために，強化学習スケジューリングではオンライン学習が用いられている．タスクの状態を逐次入力することにより，スケジューリング目的の変化に対応し続けることができる．一方，コンセプトドリフトへの適応には時間がかかる．

本研究の目的は，スケジューリングポリシの適応方法を動的に変更することを可能にするためのアーキテクチャを提案し，その有効性を検証することである．適応方法の動的な変更によって，従来のスケジューリングよりもオーバーヘッドは大きくなるので，全体のスループットは低下する．一方，資源の占有状況に対して優先されるべき処理のスループットは向上することが期待できる．

この研究目的を具体化すると，研究課題は以下の通りに定義できる．

課題 1 適応方法を動的に変更することで最適なスケジューラを選択するアーキテクチャの定義

課題 2 スケジューリング目的の変化に対応するシステムプロトタイプの作成

課題 3 アーキテクチャの定性的な評価

最適なスケジューラの選択をするスケジューリングシステムを実現するために，複数のスケジューラと複数の適応方法の協調を行うアーキテクチャを定義する. 協調はコンテキスト指向 [3] に基づいて行う．スケジューリング目的の変化に対応するコンポーネントとして，オンライン強化学習を用いたスケジューラを使用する．

2 関連研究

2.1 Q-learning

Q-learning[2] は強化学習の一種であり，行動価値を表す Q-value を (1) 式で求める．α は学習率，γ は割引率を表してる．学習が進むにつれ，Q-value の値を正確に求められるようになる．学習では，現在の報酬と将来得られる報酬を総合的に考えるようになっているので，現在の最適解を求め続けるだけでは，結果的に最適にならない問題に適している．

$$\boldsymbol{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \boldsymbol{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha(r_t + \gamma \underset{\boldsymbol{a}'}{max} \boldsymbol{Q}(\boldsymbol{s}_{t+1}, \boldsymbol{a}') - \boldsymbol{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t)) \quad (1)$$

Thomas ら [4] はスケジューラとして Deep Q-Network (DQN) を用い，ジョブショップスケジューリングを行った．Thomas らは，DQN によって，従来のスケジューラよりも最適なスケジューリングが可能であることを示した．ただし，その結果が得られるには多くの学習時間が必要とされている．

2.2 水谷らのアーキテクチャ

水谷ら [5] は複数の方法で偽造された画像を検出するために，個々の偽造方法に特化して設計されたニューラルネットワーク（NN）を協調させるコンクリートアーキテクチャを定義した. 提案されたアーキテクチャはコンテキスト指向に基づいて定義されており，偽造画像検知に用いる NN を動的に選択する構造となっている．

水谷らのアーキテクチャは，コンテキストと NN 協調論理に関わるメタレベルと，偽造画像検知に関わるベースレベルを明確に分離した構造として定義されている．これにより，互いの依存関係が疎となるので，変更容易性を保証している．

3 課題解決へのアプローチ

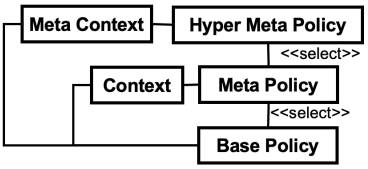

図 1 参照アーキテクチャ

スケジューリングポリシの適応方法を動的に変更するシステムの参照アーキテクチャを図 1 に示す．目的の変化に対応するための動的スケジューリングを機械学習を用いて行うためには，メタコンテキストについて考慮する必要がある．コンテキストはタスクの状態であり，メタコンテキストは Q-value である．タスクの状態集合の取り扱いは学習の状態に支配され，タスクの状態集合がスケジューリングポリシを決定するようにアーキテクチャを定義する．このアーキテクチャを定義するためには，メタコンテキストがコンテキストを変化させる必要がある．

したがって，スケジューリングポリシの切り替えを行うメタレベルよりも上位であるハイパーメタレベルが必要になる．ハイパーメタレベルによって，再構成論理が分離され変更が容易になる．ハイパーメタレベルの選択ポリシは以下の通りである．

- 学習および適応が完了した時，その学習器をスケジューラとして選択するポリシを選択する
- 学習および適応が完了していない時，伝統的なスケジューラだけ選択するポリシを選択する

動的知的スケジューリングの本質的な要件はタスク状態に応じてポリシを変えることにある．コンテキスト指向 [3] をアーキテクチャスタイルとして強化学習の構造を再定義すると，図 2 のメタコンテキストアーキテクチャとなる．Q-learning において，Meta Context は Q-value となる．Context をタスクの状態とすることでスケジューリングにも対応可能である．強化学習を用いたスケジューラは，DQN を使用する．報酬はスループットの向上と公平性を保証するために，平均待ち時間を最小化し，かつクオータを最大化させるように設定する．これらの具体例を反映したコンクリートアーキテクチャの定義にあたり，水谷らのコンクリートアーキテクチャに Q-learning の構造を取り入れる．

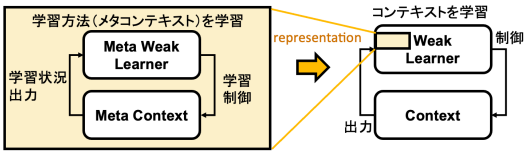

図 2　強化学習の構造

上記アーキテクチャを変更容易性の観点から評価を行い，スケジューリング目的の変化への対応が可能かどうかを検証するためにプロトタイプを作成する．提案するアーキテクチャでは，コンテキストやメタコンテキストに基づく動的再構成を行う．変更容易性の評価では，これらのコンテキストが変更される要因が何かを考えて，変更シナリオを決定する．本研究では，スケジューリング目的の変化に対応する動的知的スケジューラとして強化学習を用いる．プロトタイプを用いて強化学習器がスケジューリング目的の変化に対応できるかを検証する予定である．本研究では，学習を完了させることができなかったので，目的変化に対応可能な強化学習スケジューラを作成するための学習方法を考察する．以上により，以下の成果が得られる．

- 目的に適したコンポーネントを動的に選択するためのアーキテクチャ
- そのアーキテクチャが実装可能であること

4　アーキテクチャの定義

提案する概念アーキテクチャを図 3 に示す．このアーキテクチャは，スケジューリングポリシの適応方法を動的に変更するためのオペレーティングシステムのアーキテクチャである．メタコンテキストに応じて切り替え対象となるリソースマネージャを変更し，それらをコンテキストに応じて切り替えることで最適なマネージャを選択する構造を定義している．加えて，再構成論理を実現するためのメタレベルやハイパーメタレベルのコンポーネントと，再構成の対象となるベースレベルのコンポーネントとが明確に分離された構造として定義している．

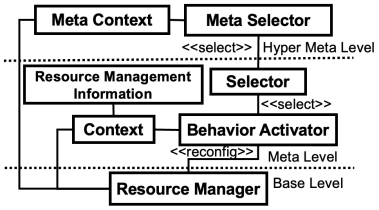

図 3　提案する概念アーキテクチャ

提案する具象アーキテクチャを図 4 に示す．機械学習スケジューラとして強化学習で用い，伝統的なスケジューラとして Linux で使われているスケジューラを用いた際の，メタコンテキストやコンテキストを導入することで具体化した．強化学習を用いるので，メタコンテキスト学習とコンテキスト学習は，図 2 に示したリフレクティブな構造として定義できる．Linux では，タスクの優先度によって，どの伝統的なスケジューラを選択するか決まるので，Scheduler Context としてタスクの priority を持つように定義した．

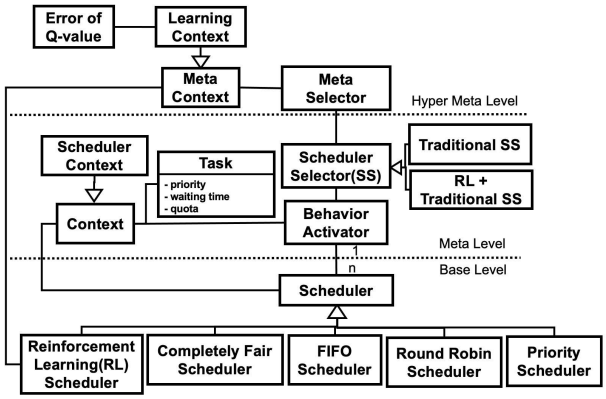

図 4　提案する具象アーキテクチャ

強化学習を用いたスケジューラの学習が進み，性能が向上したとき，それを一般優先度のタスクに対するスケジューラとして扱うように，Scheduler Selector を変更する．具体的には，一般優先度のタスクに対して，CFS を選択させていた Traditional SS を，RL Scheduler を選択させるようにする RL + Traditional SS に変更し，スケジューラの選択を行うようにする．学習の進捗度は，報酬から求められる Q-value の更新に対する差分で判断できるので，Learning Context が Error of Qvalue と関連付くように定義した．

図 4 の具象アーキテクチャの動的側面を図 5 に示す．Meta Selector は Meta Context から Q-value の情報を受け取り，それを基に Scheduler Selector を再構成する．

Behavior Activator は Context からタスクの状態を受け取り，それを基に再構成した Scheduler Selector を使って，最適なスケジューラの選択を行う．選択されたスケジューラがタスクを受け取り，ディスパッチを行うことで，タスクの状態と Q-value が変化するので，それらを Context と Meta Context に反映させる．

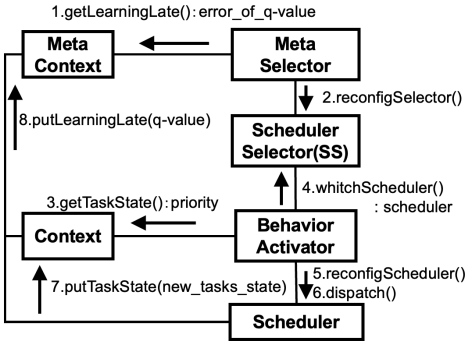

図 5　具象アーキテクチャの動的側面

5　プロトタイプを用いた実験

前章で説明した具象アーキテクチャ（図 4）に基づきプロトタイプを設計，実装する．設計したプロトタイプを図 6 に示す．スケジューリングの目的変化へ強化学習器が対応するかどうかを確かめるためにプロトタイプを作成するので，切り替え機構は作成しないこととした．タスクの状態を蓄えておくデータ構造として，Process Control Block(PCB) を利用する．

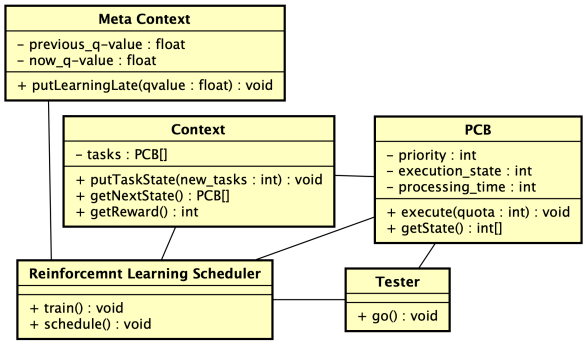

図 6　設計したプロトタイプ

表 1　強化学習器の構造とハイパーパラメータ

(a) 強化学習器の構造

層の数	隠れ層のユニット数	中間層の活性化関数	出力層の活性化関数
3	30	softmax	ReLU

(b) ハイパーパラメータ

学習率	割引率	ϵ	最適化手法
0.0001	0.90	0.30	Adam

図 6 の強化学習器の構造とハイパーパラメータは表 1 の通りであり，学習データはランダムに生成したタスクの状態とした．スケジューリングに Convolutional Neural Network を使用し，パターン認識の問題として解決を試みた研究は数少ない．スケジューリングは処理速度が重要視されるので，本研究では Fully Connected Network を利用する．報酬関数も学習の制御に関わるが，その具体的な計算手順が記載されている論文が数少なく，どの程度報酬を与えるかは主観で決定した．プロトタイプを用いた実験で得られた学習過程を図 7 に示す．

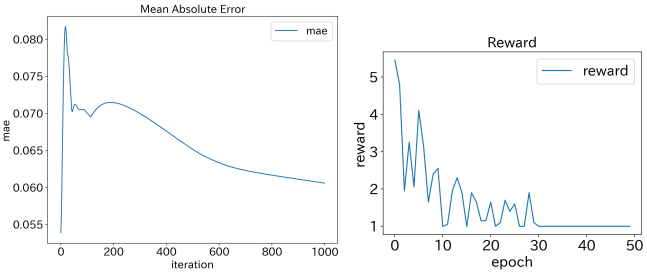

図 7　学習過程

6　考察

6.1　変更容易性

提案するアーキテクチャが支援する変更容易性について，変更例に基づいて議論する．スケジューリングシステムにおいてコンテキストやメタコンテキストが変更される要因を考え次の変更例を設定した．

変更例 1. 単一あるいは複数のスケジューラの追加

変更例 2. 強化学習スケジューラの学習に利用する報酬関数の変更

変更例 3. 学習の進捗度を測る指標の変更

変更例 4. ページングにおいてページ置換方式の適応方法を動的に変化させるように変更

変更例 1 で変更するスケジューラが，伝統的なスケジューラや強化学習を用いたスケジューラであれば，コンポーネントの追加とコンテキストの変更で対応可能である．例えば，処理時間順に基づくスケジューラを追加したい場合，処理時間をコンテキストが持つ情報として追加することで対応できる．一方，強化学習以外を用いた機械学習スケジューラに変更する場合は，学習の進捗度を表す指標が変わるので，メタコンテキストにも変更が生じると考える．いずれの変更にしても，新たなスケジューラが選択させるように Scheduler Selector の内容を変更する必要がある．

変更例 2 では，報酬が得られる事象の変更と Scheduler Cotext にある報酬関数を変更することで対応可能である．デッドラインを超えずにタスクが完了したとき報酬を与えるようにしたい場合，コンテキストが持つ情報として，デッドラインを追加し，報酬を決定する論理を変更することで対応できる．この変更はメタレベルだけで行われ，ベースレベルのスケジューラには影響は与えない．

変更例 3 では，タスクの状態の解釈を学習の状態によって決定する際，その指標となる要素を変更するので，メタ

コンテキストを変更することで対応する．学習が進み，スケジューラとして使えることがわかる指標としてスループットを追加する場合，それをメタコンテキストとして追加し，メタコンテキストの層を変更することで対応できる．この変更はハイパーメタレベルだけで行われる．

変更例 4 では，図 3 のアーキテクチャで示した複数のポリシから最適なものを選択する機構を，ページングに応用する．ページ置換方式の選択を考えた場合のアーキテクチャを図 8 に示す．コンテキストをページアクセス順序とすれば，メタコンテキストである学習の状態によってそれらの取り扱いを決定し，コンテキストにより最適なページ置換方式を選択することができると考える．同じページへのアクセスが頻繁に来ない場合，オーバーヘッドの少ない FIFO がよく，同じページへのアクセスが頻繁に来る場合，頻繁にアクセスされるページを残す NRU がよい．規則性がない場合，ランダムに選択することが考えられるが，機械学習によって，より最適な置換を行えるようになれば，その学習器を利用するようにハイパーメタレベルで制御を行うようにすれば，全体として目的に応じた最適ページ置換を行うことができる．

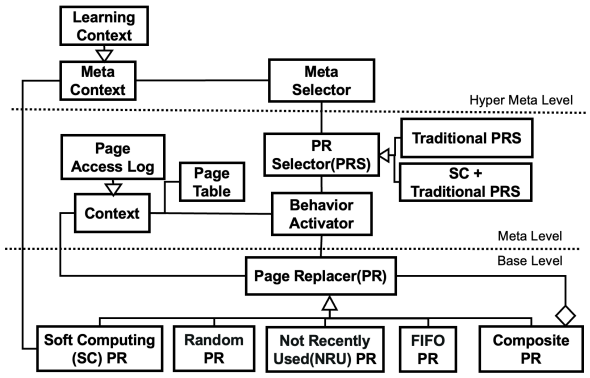

図 8 ページ置換方式を切り替える具象アーキテクチャ

以上の議論をまとめると，変更例 1,4 には多相性とコンテキスト指向で対応でき，変更例 2,3 にはコンテキスト指向で対応できることがわかる. すなわち，提案するアーキテクチャは多相性とコンテキスト指向に基づく動的な再構成により，変更容易性を保証しているといえる．

6.2 学習結果に対する考察

図 7 より，本研究で作成した強化学習器は未学習が起きていると考える．Mean Absolute Error の変化を見ると，誤差が収束に向かっていることがわかるが，その値は学習の初期のときよりも高い値となっているので未学習が起きている．この未学習の原因は，報酬関数にあると考える．本研究では，平均待ち時間が高くなるほど報酬を減らし，クオータの値が高くなるほど報酬を増やしている．このとき，平均待ち時間による報酬の減少を大きくしてしまうと，どんなクオータを割り当てても報酬が 0 となってしまい，学習が進まない．クオータの値による報酬の増加を大きくしてしまうと，報酬の範囲が大きくなり，学習に必要なデータ量が多くなると考える．したがって，付与する報酬が報酬関数にどのような影響を与えているか分析し，数理モデルを明確にする必要がある．

学習が完了したとしてもそれを評価する指標がなければ必要以上に学習データが入力され，過学習を起こしてしまう．誤差を収束させるために追加で学習をした場合，ランダムに処理時間を生成していても同じデータを繰り返し学習してしまう可能性が高くなる．過学習を起こす前に，学習が完了したことを適切に評価する指標が必要である．本研究で作成した強化学習器では，学習の完了を Q-value によって評価していた．Q-value の誤差が小さくなると学習が完了したと言えるが，どの程度小さくなればスケジューラとして使うことができるようになるかわからない．得られた報酬で学習の完了を評価する場合も同様である．これらの値は強化学習スケジューラとして評価する指標としては不適切だと考える．スループットは最適なスケジューリングができているかを確かめる指標となると考える．

7 おわりに

本研究では，スケジューリング目的の変化に対応するために，コンテキスト指向を適用することで最適なスケジューラの選択を行うアーキテクチャを定義した．アーキテクチャは多相性とコンテキスト指向により，変更が容易であることを示した．加えて，目的に適したポリシを動的に選択するためのアーキテクチャとしてページングにも応用できることを示した．学習結果に対する考察では，報酬関数と性能評価方法に原因があると考えた．これを基に強化学習器を再学習させることで，プロトタイプを用いた定量的な評価を行うことができると考える．

今後は以下を進める予定である．課題 2，3 は最終的な目標であり，現状はシミュレーションによる検証を行う予定である．

課題 1. 強化学習を用いたスケジューラの実現

課題 2. 上記スケジューラを Linux のスケジューラとして追加し，そのカーネルをビルド

課題 3. プロトタイプの出力結果を用いて提案するアーキテクチャがスループットの向上に寄与したか検証

参考文献

[1] Shyalika Chathurangi, et al. “Reinforcement Learning in Dynamic Task Scheduling: A Review,” *SN Computer Science*, 2020.

[2] C. Watkins, et al. “Q-learning,” *Machine learning*, 1992.

[3] 紙名哲生. “文脈指向プログラミングの要素技術と展望,” JSSST, 2014.

[4] Thomas Gabel, et al. “Adaptive reactive job-shop scheduling with reinforcement learning agents,” *IJITIC*, 2008.

[5] Akira Mizutani, et al. “Design of Software Architecture for Neural Network Cooperation: Case of Forgery Detection,” *2021 28th APSEC*, 2021.