Vol.2017-SLDM-180 No.3
2017/5/10

実行時間が変動するタスクに対するヘテロジーニアスコアでのTAS最適化手法

土橋 亮太[1,a)]　野村 孔命[1,b)]　高島 康裕[1,c)]　中村 祐一[2,d)]

概要：本論文では実行時間が変動するタスクに対するヘテロジーニアスコアでのタスク割り当てとスケジューリング (TAS) 最適化手法について提案する．コアごとで実行時間が変化するヘテロジーニアスコアでは，タスクのコア割り当てがシステムの実行性能に対して重要である．また，コア割り当てが定まった後に実行時間の変動を考慮した TAS が必要である．本論文ではコア割り当てに対しては SA 法を用いた最適化を提案する．TAS に対しては既存手法である PEVaS に資源制約から生じた直前終了タスクからの依存関係の考慮を追加したスケジューリング手法を提案する．実験により，提案手法がスケジューリング性能と歩留まりを向上させることを確認した．

1. 序論

近年，1 個のプロセッサコアの性能向上は限界となっている．そのため，1 チップ上に複数のコアを実装するマルチコア/メニーコアが広く利用されるようになっている．更には，チップの性能の最適化を図るため，実装されているコアが同一なホモジーニアスコアだけでなく，様々な性能のコアが実装されるヘテロジーニアスコアが利用されるようになっている [1, 2]．このヘテロジーニアスであるようなシステムにおいては，1) どの種類のどのコアで実行するか，2) いつ実行するのか，といったタスク割り当てとスケジューリング (TAS) が非常に重要となる．[3] では，各タスクの実行時間と割り当てられたコアの消費電力が変動するモデルにおける TAS 最適化手法を提案した．これは，TAS を決定した後，モンテカルロシミュレーションで歩留まりを測定し，評価することを繰り返し実行する手法である．この手法は，任意の変動モデルに対し柔軟に適用可能である反面，生成する TAS は必ずしも歩留まりを考慮したものではないため，収束が遅いという問題が存在した．そこで [4] では，PEVaS と呼ぶ手法を提案した．この手法では，モンテカルロシミュレーションを行う前に，同一コアへの割り当てで生じる潜在的依存関係を入力データフローグラフに追加する．この追加により，最終的なスケジューリング性能が向上することが確認された．しかし，この手法では，本来先に行うべき優先度の高いタスクの実行タイミングが遅くなってしまう欠点が存在する．その結果，最適なコアの割り当てを得ることができず，終了時刻の遅延につながってしまう．

そこで，本論文では PEVaS のフレームワークを元に実行時間が変動するタスクに対するヘテロジーニアスコアでの TAS 最適化手法を提案する．全体の開始タスクの残実行時間の最悪値を評価値とし，SA 法を用いてコア割り当ての最適化を行い，最適化されたコア割り当てでモンテカルロシュミレーションを行い，歩留まりを計算する．本論文では，SA 法の評価を 1) 依存関係の追加を行わない場合，2) 同一コアへの割り当てで生じる潜在的依存関係のみを考慮する場合，3) 潜在的依存関係を追加した後，各タスクの開始直前に終了したタスクを「直前終了タスク」として，直前終了タスクからの依存関係も追加する場合，で行い，歩留まりを計算する．そして，実験の結果から直前終了タスクからの依存関係を追加することでスケジューリング性能の向上が確認された．

本論文は以降，以下のように構成される．第 2 節では問題定義，正規分布に対する基本演算と残実行時間の計算方法，PEVaS で用いられた同一コアへの割り当てで生じる依存関係について説明する．第 3 節では直前終了タスクからの依存関係，SA 法を用いたコア割り当てのフレームワークについて説明する．第 4 節では実験の結果と考察を述べる．最後に，第 5 節で結論を述べる．

[1] 北九州市立大学 国際環境工学部　〒 808-0135 福岡県北九州市若松区ひびきの 1-1
[2] NEC システムプラットフォーム研究所 〒 211-8666 神奈川県川崎市中原区下沼部 1753
[a)] ryota.tsuchihashi@is.env.kitakyu-u.ac.jp
[b)] w5mcb006@eng.kitakyu-u.ac.jp
[c)] takasima@kitakyu-u.ac.jp
[d)] yuichi@az.jp.nec.com

2. 準備

2.1 問題定義

まず最初に本論文で考える問題を定義する．

定義 1(ヘテロジーニアスコアで実行時間が変動するタスクに対する TAS 問題)

入力 データフローグラフ (DFG)，各タスクの各コアでの実行時間，同時に使用できるコアの組み合わせ

出力 TAS

目的 全タスクの終了時刻最小化

制約 コア使用制約，先行制約，利用コアに重なりがない

2.2 正規分布に対する基本演算

本論文ではタスクの実行時間は正規分布で変動すると仮定する．そのため，正規分布を考慮した演算が必要である．以下では 2 数 $x_1 \sim N(\mu_1, \sigma_1^2)$ と $x_2 \sim N(\mu_2, \sigma_2^2)$ (μ : 平均, σ : 標準偏差) とした時の演算について紹介する．まず，2 数の和 x_1+x_2 の分布は $x_1+x_2 \sim N(\mu_1+\mu_2, \sigma_1^2+\sigma_2^2)$ である．次に，2 数の最大値 $\max\{x_1, x_2\}$ については PEVaS と同様，Clark の手法 [5] を利用する．$x = max\{x_1, x_2\} \sim N(\mu, \sigma^2)$ は式 (1) のように近似することができる．

$$\begin{aligned}\mu &= \mu_1\Phi(\alpha) + \mu_2\Phi(-\alpha) + \beta\phi(\alpha)\\ \sigma^2 &= (\mu_1^2+\sigma_1^2)\Phi(\alpha) + (\mu_2^2+\sigma_2^2)\Phi(-\alpha) + (\mu_1+\mu_2)\beta\Phi(\alpha) - \mu^2\\ &\text{ただし}, \beta = \sqrt{\sigma_1^2+\sigma_2^2-2\sigma_1\sigma_2c_{12}}, \alpha = \frac{\mu_1-\mu_2}{\beta}\end{aligned} \tag{1}$$

ここで，$\phi()$ は標準正規分布の確率密度関数を，$\Phi()$ は標準正規分布の累積密度関数を表す．また x_1，x_2 との共分散 c_{1y}，c_{2y} を持つ正規分布 y と $x = max\{x_1, x_2\}$ との共分散 c_{xy} は式 (2) で表される．

$$c_{xy} = \frac{\sigma_1c_{1y}\Phi(\alpha) + \sigma_2c_{2y}\Phi(-\alpha)}{\sigma^2} \tag{2}$$

2.3 残実行時間

本論文では，PEVaS と同様，あるタスクの実行開始からそのタスクの全ての後続タスクの実行が完了するまでの時間を残実行時間と定義する．その上で，残実行時間を優先度とした統計的静的リストスケジューリングにより，スケジューリングを決定する．各タスクの残実行時間は以下の手法により，実行時間から求めることができる．

1) DFG の中から直接後続タスクがない，もしくは直接後続タスクの残実行時間の計算が全て完了しているタスク x を取り出す．

2) x の残実行時間を x の実行時間と直接後続タスクの残実行時間から計算する．

a) **直接後続タスクがない場合**
x の実行時間がそのまま x の残実行時間となる．

b) **直接後続タスクが 1 つの場合**
x の実行時間に直接後続タスクの残実行時間を加える．

c) **直接後続タスクが 2 つ以上の場合**
x の直接後続タスクを x_1，x_2 とすると，x_1 と x_2 の残実行時間の最大値に x の実行時間を加える．$\max\{x_1, x_2\}$ を $N(\mu, \sigma^2)$ の正規分布とすると，μ, σ^2 は Clark の手法を用いて近似することができる．近似された最大値に x の実行時間を加えることで x の残実行時間を求めることができる．

以上の計算の流れを図 1 を例に説明する．ここで，各タスクの実行時間を表 1 に示し，タスク x の残実行時間を正規分布 $N(\mu_x^r, \sigma_x^{r2})$ と表し，タスク間の相関はないものとする．

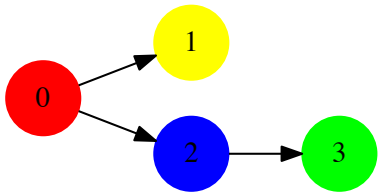

図 1 DFG の例

表 1 各タスクの実行時間

タスク	分布
0	$N(3, 0.2^2)$
1	$N(5, 0.4^2)$
2	$N(4, 0.3^2)$
3	$N(2, 0.2^2)$

最初に，直接後続タスクがない，もしくは直接後続タスクの残実行時間の計算が全て完了しているタスクを取り出す．図 1 の初期状態では，タスク 1 とタスク 3 が該当する．タスク 1，タスク 3 は直接後続タスクは存在しないのでそれぞれの実行時間がそのままそれぞれの残実行時間となる．よって，$N(\mu_1^r, \sigma_1^{r2}) = N(5, 0.4^2)$，$N(\mu_3^r, \sigma_3^{r2}) = N(2, 0.2^2)$ となる．タスク 2 の残実行時間の計算はタスク 3 が完了したことによりが可能となるが，タスク 0 はタスク 2 が完了していないためまだ計算することはできない．タスク 2 は直接後続タスクとしてタスク 3 だけが存在する．正規分布の和の演算より，タスク 2 の残実行時間はタスク 3 の残実行時間にタスク 2 の実行時間を加える

ことで計算することができる．よって，タスク 2 の残実行時間 $N(\mu_2^r, \sigma_2^{r2}) = N(4+2, 0.3^2+0.2^2) = N(6, 0.13)$ となる．タスク 2 が完了したことでタスク 0 の直接後続タスクの残実行時間の計算が全て完了となるのでタスク 0 の残実行時間の計算が可能となる．タスク 0 は直接後続タスクにタスク 1 とタスク 2 が存在するため，Clark の手法をタスク 1 とタスク 2 の残実行時間分布に適用し，最大値計算が可能である．そして，計算結果である最大値にタスク 0 の実行時間を加えることでタスク 0 の残実行時間を求めることができる．2 数の最大値を $x \sim N(\mu, \sigma^2) = max\{x_1, x_2\}$ とおく．タスク 1 の残実行時間 $x_1 \sim N(\mu_1^r, \sigma_1^{r2})$, タスク 2 の残実行時間 $x_2 \sim N(\mu_2^r, \sigma_2^{r2})$ とし，式 (1) に当てはめると，$x = N(6.00665, 0.352496^2)$ となる．この結果に，タスク 0 自身の実行時間を加えるので，$N(\mu_0^r, \sigma_0^{r2}) = N(6.00665+3, \sqrt{0.352496^2+0.2^2}) = N(9.00665, 0.405282^2)$ となる．以上より，全タスクの残実行時間が求めることができる．

以上の計算より求まった残実行時間から，各タスクの優先度をそのタスクの残実行時間の最悪値 $\mu_x^r + 3\sigma_x^r$ とする．この優先度を元に統計的静的リストスケジューリングを行いスケジューリングを決定する．

2.4 潜在的依存関係

PEVaS では TAS を生成し，モンテカルロシミュレーションを行う前に，同一コアでの割り当てで生じる依存関係の追加を行い残実行時間の再計算を行う．この依存関係を「潜在的依存関係」と呼ぶ．PEVaS の TAS の生成からモンテカルロシミュレーションまでのフレームワークを図 2 に示す．

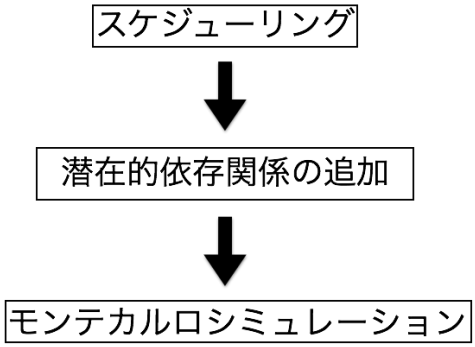

図 2 PEVaS のフレームワーク

潜在的依存関係について図 3 を例に示す．最適化の結果，図 4 のような TAS が得られたとすると，このコアではタスク 0 →タスク 1 →タスク 2 の実行順番を保持するようにタスク 0 →タスク 1，タスク 1 →タスク 2 への依存関係を追加する．ここで，タスク 0 →タスク 1 への依存関係を元々存在する依存関係なので，これは無視する．タスク 1 →タスク 2 への依存関係は存在しないので追加を行う．依存関係の追加により，各タスクの残実行時間の関係は「タスク 0 ＞ タスク 1 ＝ タスク 2」 から 「タスク 0 ＞ タスク 1 ＞ タスク 2 」となりタスク 1 がタスク 2 より優先度が高くなるためタスク 1 が割り当て通り先に実行されるようになる．

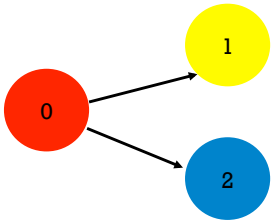

図 3 入力 DFG

	0	1	2
PE	0	1	2

図 4 生成された TAS

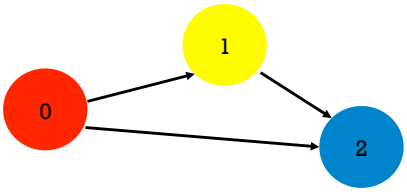

図 5 変更後の DFG

3. 提案手法

3.1 直前終了タスクからの依存関係

本論文では，より最適なコア割り当てを得るために新たな依存関係の追加を行う．図 6 の DFG を例に示す．

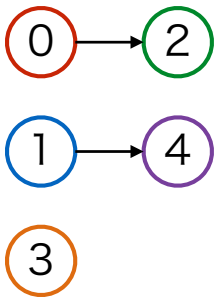

図 6 入力 DFG の例

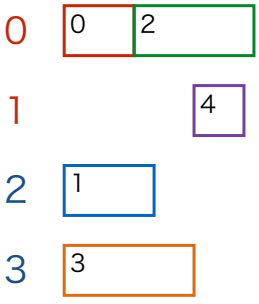

図 7 コアタイプ割り当て最適化時

図 7 はコア割り当て最適化により得られた TAS であり，コアタイプ 0 としてコア 0 とコア 1，コアタイプ 1 としてコア 2 とコア 3 が存在し，コア使用制約は全てのコアを同時に使用できないとする．タスク 4 はタスク 1 からの依存関係があるため，タスク 1 が完了してから実行されようと

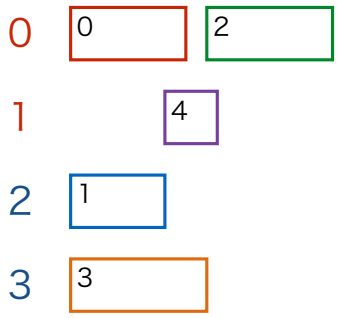

図 8　モンテカルロシミュレーションの一例

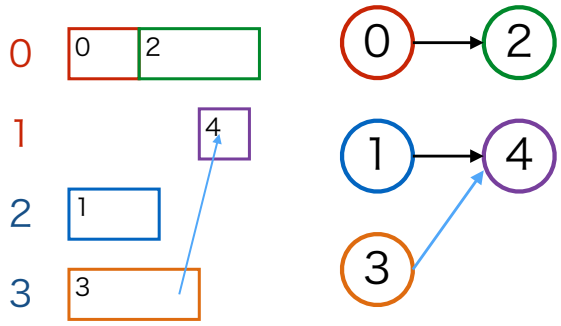

図 9　直前終了タスクからの依存関係の追加

図 10　直前終了タスクからの依存関係を考慮した DFG

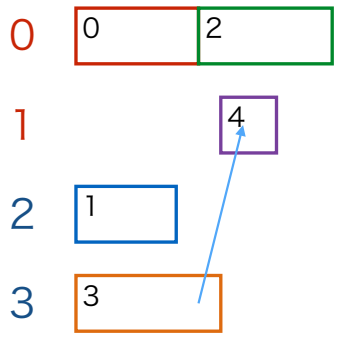

図 11　直前終了タスクからの依存関係を考慮した TAS でのモンテカルロシミュレーションの一例

するが，コアの使用制約により，タスク 3 の実行完了後に割り当てがされている．ここで，実行時間の変動によりモンテカルロシミュレーションの際に，タスク 0 の実行時間が増大したとする．これにより，タスク 1 が終了と同時にタスク 4 の実行が可能となるため，タスク 4 がタスク 1 の完了直後に実行を開始している．そのため，本来先に実行されるべきタスク 2 がコアの使用制限により，実行開始が遅くなってしまい，結果として全体の終了時刻が遅くなってしまっている (図 8)．すなわち，優先度 (残実行時間の最悪値) の大きいタスクの処理を後に回してしまうと，全体の終了時刻が遅くなってしまう傾向がある．そこで，「コア割り当て最適化時に直前に終了したタスク i をタスク j が待った」という情報を与えるために新たな依存関係を追加する．この依存関係を「直前終了タスクからの依存関係」と呼ぶ．図 7 の結果に対して，タスク 4 はタスク 3 の完了を待ってから実行開始しているので，タスク 3 からタスク 4 へ依存関係の追加を行う (図 9)．依存関係を付加した DFG が図 10 となる．この依存関係を考慮したコア割り当て最適化を行なうと，図 11 となった．タスク 0 の実行時間が増加しているものの，タスク 2 の実行開始が遅くなることはなく，図 8 と比べて全体の終了時刻も短くなっている．

3.2　実行時間が変動するタスクに対する SA 法を用いたコア割り当て

本論文では，コア割り当てを SA 法を用いて最適化を行う．以下に SA 法の最適化アルゴリズムを示す．

1)　各タスクのコアの初期タイプをランダムに決定．

2)　以下 i〜vi を評価値（入力 DFG 全体の開始タスクの残実行時間）が最小となるように SA 法で最適化を行う．

i)　入力 DFG における各タスクの残実行時間を計算する．
ii)　優先度とリソース制限の下にコアの割り当てとスケジューリングを行う (図 12(A))．
iii)　入力 DFG に潜在的依存関係を追加後 (図 12(B))，残実行時間を計算し再度スケジューリングを行う (図 12(C))．
iv)　ii) の TAS で生じた直前終了タスクからの依存関係を DFG に追加する (図 12(D))．
v)　この時点での DFG で評価値の計算を行い新しいコア割り当てを受け入れるかどうか選択 (図 12(E))．
vi)　終了条件に達していなければタスクの中から一つだけコアのタイプを変更し i に戻る．

3)　評価値が最も最小となった時のコア割り当てを決定する．

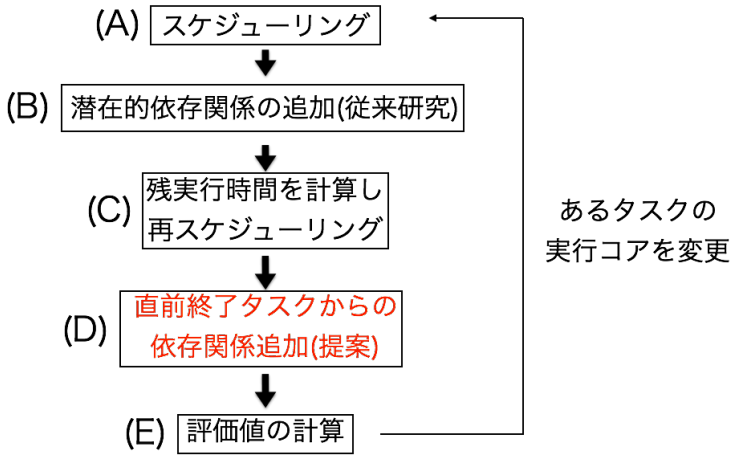

図 12　提案手法のフレームワーク

4. 実験

SA 法によって最適な TAS を求めた後，1000 回のモンテカルロシミュレーションを行い歩留まりを計算する．提案手法の有効性を確認するため，以下の 3 種類の評価方法で比較実験を行った．

方法 1 依存関係の追加なし

方法 2 潜在的依存関係を追加 (従来手法 [4])

方法 3 潜在的依存関係を追加後，再スケジューリングを行い，更に直前終了タスクからの依存関係も追加 (提案手法)

コア割り当て最適化時の開始タスクの残実行時間を評価値とし，最小となる時の割り当てを使用する．実験で用いた SA 法での各パラメータは以下の通りである．

- 初期温度：100
- 温度低下率：0.99
- 各温度での試行回数：1000
- 終了温度：1

図 13 を入力 DFG とし，表 2 に各タスクのコアタイプ 0 とコアタイプ 1 のそれぞれの実行時間分布，表 3 にコア使用制約を示す．

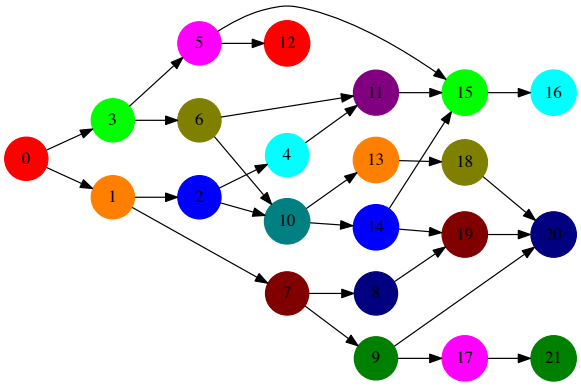

図 13 入力データフローグラフ

4.1 実験結果

表 4 では，図 13 の DFG に対する実験結果を示す．ここで，終了時刻平均の行は 1000 回のモンテカルロシミュレーションにおける全タスク実行の終了時刻の平均，次の 2 行は SA 法で行なった最適割り当ての TAS での開始タスクの残実行時間の平均と標準偏差，そして歩留まりの行は

表 2 各タスクの実行時間分布

ID	TYPE:0	TYPE:1	ID	TYPE:0	TYPE:1
0	N(10,0.5)	N(5,0.7)	11	N(25,0.07)	N(18,0.8)
1	N(23,2.3)	N(12,1.3)	12	N(2.7,0.07)	N(1.2,0.1)
2	N(12,0.6)	N(4,0.02)	13	N(26,0.07)	N(10.5,1.4)
3	N(30,0.07)	N(13,2.1)	14	N(22,2.2)	N(9,0.3)
4	N(17,0.85)	N(10,1.1)	15	N(18,0.9)	N(10,1.4)
5	N(14,0.7)	N(3,0.05)	16	N(30,0.07)	N(20,0.6)
6	N(16,1.6)	N(11,2.3)	17	N(22,2.2)	N(16,1.5)
7	N(29,0.07)	N(12,0.9)	18	N(24,2.4)	N(13,1.06)
8	N(21,1.05)	N(15,0.99)	19	N(19,0.95)	N(10,0.8)
9	N(23,2.3)	N(8,0.4)	20	N(25,2.5)	N(8,1.3)
10	N(11,0.55)	N(4,0.21)	21	N(28,0.07)	N(15,0.8)

表 3 リソース制約 (同時に使用できるコア数)

コア 0	コア 1
4	0
2	1
0	2

実行時間制約を 125 としたときの歩留まりを示している．表 4 において，方法 1 と方法 2 は終了時刻平均が TAS の段階での残実行時間の平均を大幅に上回っており，TAS が過小評価となってしまっている．その結果，方法 1 や方法 2 を用いた SA 法で最適とされたコア割り当てでは，過小評価されている割り当てが選択される．しかし，シミュレーションでは終了時刻平均は非常に大きくなる．比べて，方法 3 は TAS の段階での残実行時間の平均は，終了時刻平均より僅かに過大評価であるが，終了時刻平均は 3 つの方法の中で最小となった．一方で，方法 3 は，終了時刻平均と TAS の段階での残実行時間の平均との誤差は非常に小さく精度の良い TAS を得ることができた．

提案手法でも述べたように，残実行時間 (優先度) が大きいタスクは処理は早めに行うべきである．直前終了タスクからの依存関係を追加することで，優先度の低いタスクを実行してしまうことによって，終了時刻に影響を及ぼしてしまう可能性のある優先度の高いタスクの実行が遅くなってしまうような場合には，あえて優先度の低いタスクの割り当てを行わず待つことで最終的な終了時刻が早くなることを示している．結果，実行時間が変動するタスクに対して最適な TAS を得ることが可能である．

表 4 実験結果

		方法 1	方法 2	方法 3
終了時刻平均		125.233	125.057	115.321
残実行時間 (TAS)	平均：μ	86.7884	100.631	119.044
	標準偏差：σ	2.27881	2.29944	1.99979
歩留まり		56.4%	54.2%	100%

5. 結論

本論文では，実行時間が変動するタスクに対するヘテロジーニアスコアでの TAS 最適化手法を提案した．実行時間が変動するタスクに対して新たな依存関係として「直前終了タスクからの依存関係」を考慮した TAS の評価を行った．この評価を踏まえた上で，使用するコアで実行時間が変動するヘテロジーニアスコアでのコア割り当ての最適化を行った．提案手法の有効性を確認するために，3 種類の TAS の評価方法で比較実験を行なった．その結果，今回提案した直前終了タスクからの依存関係を考慮した TAS のシミュレーションでの終了時刻平均は他の 2 つの方法に比べて非常に小さくなる結果となり，提案手法でのコア割り当て最適化が有効であることを確認した．

今回の提案手法では，直前終了タスクからの依存関係は全てに対して付加している．これは，待つべきタスクに対しては効果的であるが，依存関係を追加する必要がないタスクに対しても追加を行ってしまっているため，非常に悲観的な予測となっている．今後の課題としては，これらのタスクの判別方法の検討が必要である．

参考文献

[1] 古山祐樹, 島岡護, 見神広紀, 林明宏, 木村啓二, 笠原博徳：低消費電力マルチコア RP-X を用いた 1 ワット Web サービスの実現，情報処理学会 第 193 回計算機アーキテ クチャ研究会 (SWoPP2012)，2012.

[2] AMD Accelerated Processing Unit：`https://en.wikipedia.org/wiki/AMD_Accelerated_Processing_Unit`.

[3] Mingsong Chen., Daian Yue., Xiaoke Qin., Xin Fu and Prabhat Mishra.: *Variation-Aware Evaluation of MPSoC Task Allocation and Scheduling using Statistical Model Checking Design*, Automation & Test in Europe Conference & Exhibition (DATE)，2015(9-13 March 2015).

[4] Komei Nomura., Yasuhiro Takashima. and Yuichi Nakamura.: *PEVaS: Power and Exeution-time Variation-aware Scheduling for MPSoC*, IEEE New Circuits and Systems Conference (NEWCAS) 2016, 2016.

[5] Charles E. Clark.: *The Greatest of a Finite Set of Random Variables*, Operations Resarch, Vol.9, No.2(Mar. - Apr.，1961), 145-162.

[6] Daniel Gajski., Nikil Dutt., Allen Wu., Steve Lin.: *High-Level Synthesis Introduction to Chip and System Design*, Fouth Printing(1997), 233-235.