Vol.2019-SLDM-188 No.1
2019/5/15

実行時間がばらつくタスクに対する動的スケジューリング手法

土橋　亮太[1,a)]　髙島　康裕[1,b)]

概要：本稿では，実行時間がばらつくタスクに対する動的スケジューリング手法を提案する．このような状況では，タスク割り当てとスケジューリング (TAS) の構成と，実際のシステム実行時でタスクの終了に変化が生じる可能性があり，その結果，実行性能の劣化が生じていた．そこで，従来研究では，構成時の状況を保持するため，仮想的な依存関係を付加する手法を提案した．その結果，構成で想定された性能は保証される反面，不必要な制約が存在するため，悲観的な結果となった．そこで，本稿では，EVaDys と呼ぶ動的なスケジューリングを考慮し統計処理に基づく評価を利用する手法により，より実行時の性能向上を実現した．また，その結果を実験により確認した．

キーワード：EVaDys，TAS, 残実行時間，開始時刻予測

1. はじめに

近年，プロセッサコア単体の性能向上は，ほぼ限界に達しているという報告がなされている．その為，プロセッサの性能向上には，1 チップ中に複数のプロセッサコアを実装するマルチ/メニーコアシステムが広く利用されている [1,2] また，様々な計算用途に対応するため，ヘテロジーニアスコアが有望視されている．ここで，ヘテロジーニアスコアでは，対象とする計算によって，利用するコアの種類が異なるため，搭載されているコアを全て同時に利用可能とは限らない．特に，組み込み用途では，その傾向が強い．そのため，タスクのコア割り当てとスケジューリング (TAS) が非常に重要である [3–6]．また，各タスクの実行時間は，入力等によりばらつきがあることが知られており，その変動を考慮することが最終的な性能に対し重要である [7–9]．すなわち，実行時間が変動するタスクに対応した TAS は，ばらつきに対しロバストかつ高性能なシステム構築に必要不可欠である．

この問題に対し，[9] は，TAS の構成とシミュレーションの繰り返し実行によるシステム構築手法を提案した．これは，シミュレーションに基づいて最適化を実現するため，ばらつきモデルはシミュレーションが対応可能であれば，特に問題無い反面，一個の TAS に対し，シミュレーションでの評価が必要となるため，システム構成に多大な時間を要する．それに対し，[10] では，実行時間を正規分布として，統計的処理に基いた優先度に従って，TAS を構成する手法を提案した．また，[11] では，この枠組みをヘテロジーニアスコアに適用し，タスクのコアタイプ割り当てをシミュレーティッドアニーリングで最適化し，更に TAS 構成時の状況を保持するため，*Waiting Dependency* と呼ぶ仮想的な依存関係を付加する手法を提案した．その結果，実際のシステム実行時にも，最適化で得られた状況を保持するため，高精度かつ高性能な TAS 構成を実現した．しかし，この手法では，*Waiting Dependency* の必要性は考慮せずに，条件に合致したタスク間には全てに依存関係を付加していた．そのため，本来不要な「待ち」が発生し，性能劣化が生じる可能性があった．

そこで，本稿では，仮想的な *Waiting Dependency* を付加しない EVaDys 法を提案する．EVaDys では，システム実行中に，ある実行開始可能なタスクに対し，1) そのタスクの実行中に，より優先度の高いタスクが実行開始可能になるかを統計的な処理を用いて予測, 2) そのタスクの実行が，より高い優先度のタスク実行を妨げるかを確認，という手順を経て，そのタスクの実行の開始を決定する手法である．そして，この手法の有効性を確認するため，[10] と [11] の手法と比較し，TAS 構成の性能が最良であることを確認した．

[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
[a)] ryota.tsuchihashi@is.env.kitakyu-u.ac.jp
[b)] takasima@kitakyu-u.ac.jp

2. 準備

2.1 問題定義

まず，本稿で扱う「実行時間がばらつくタスクに対するTAS 問題」について定義を行なう．

定義 1 (実行時間がばらつくタスクに対する TAS 問題)
入力： データフローグラフ *(DFG)*，各タスクの各コアタイプでの実行時間
出力： タスク割り当て，及び，スケジューリング *(TAS)*
目的： 全タスク終了時刻の最小化
制約： 先行制約，コア使用制約，重複制約

ここで，ヘテロジーニアスコアを想定しているため，各タスクの実行時間は各コアタイプ毎に異なるとする．また，この実行時間は，正規分布で表現される．そして，コア使用制約は，コアタイプ毎の個数の組として定義される．出力は，先行制約，同時利用可能なコア数制約と重複制約を満たす TAS とし，目的は全タスクの終了時刻の最小化である．

2.2 実装フロー

本稿でのコアタイプの選択は，Simulated Annealing を用いて最適化を行なった．すなわち，各タスクにおけるコアタイプの選択を符号とし，その符号を摂動により変化させ，[11] と同様，1) 本来の DFG で残実行時間を優先度としたリストスケジューリング，2) 1) の結果より，入力 DFG に Implicit Dependency [10] を付加した DFG でのリストスケジューリング，3) 2) の結果より，2) の DFG に Waiting Dependency [10] を付加した DFG でのリストスケジューリング，を順に計算し，結果の TAS での全タスクの終了時刻を評価として，最適化を行なう．

なお，今回の実装では，各タスクの実行は一度開始したら，終了するまで中断しない．また，システム実行では，上記の実装時に決定したコアで，各タスクを実行し，その変更は行なわないとした．

3. 提案手法

本稿では，Execution-time Variation-aware Dynamical scheduling Method (EvaDys) を提案する．EvaDys では，予めタスクのコア割り当てが決まっている状況で，

step 1: 現在の開始可能タスクで最も優先度が高いタスク t を選出．そのようなタスクが存在しない場合は処理を終了．
step 2: タスク t の実行を開始したと仮定し，その実行中により優先度の高いタスクが開始可能となるか？
step 2.1: もし存在しないなら，t の TAS を決定．
step 2.2: 存在するならば，タスク t の実行により，そのタスクの実行が阻害されるかを判定．
step 2.2.1: されないのであれば，タスク t の TAS を決定．
step 2.2.2: されるのであれば，タスク t を一時的に開始可能タスクから外す．
step 3: タスク t の TAS が決定した場合
step 3.1: それまでに一時退避していたタスクを開始可能に復元．
step 3.2: タスク t の後続タスクの状況を更新．
step 4: Step1 に戻る．

なお，初期状態としては，先行タスクが無いタスクを全て開始可能タスクとする．また，各タスクの優先度は，残実行時間 [10] を利用する．これは，各タスクとその後続タスクに対し，与えられたコア使用制約の下で，全タスクの処理が終了するまでに必要な時間の最小値である．

ここで，Step2 では，新たに開始可能となるタスクを見積もることが必要となる．この見積もりには先行タスクの終了時刻を見積もることが必要となる．そして，先行タスクの終了時刻の見積もりには，既に実行を開始しているタスクだけを対象とすれば良いので，終了時刻のばらつきは，実行時間のばらつきと等価である．また，複数の先行タスクがある場合，その終了時刻の最大値を評価することとなる．以上より，この処理は，SSTA の手法を適用可能であることがわかる [12,13]．

例えば，図 1 を例とする．ここで，各タスク i の実行時間 e_i の分布が $N(\mu_i, \sigma_i^2)$ の正規分布に従うとする．また，タスク 0 の開始時刻 $s_0 = 0$ とする．そして，コア数は十分存在するとする．すると，タスク 1 及び 2 の開始時刻は，タスク 0 の終了時刻で求められるので，$s_1 = s_2 \sim N(\mu_0, \sigma_0^2)$ となる．そして，タスク 3 は，タスク 1 及び 2 の終了時刻の大きい方で計算されるので，$s_3 \sim \max\{s_1 + e_1, s_2 + e_2\}$ である．

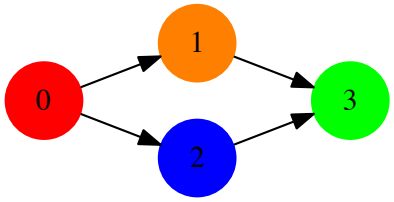

図 1 データフローグラフの例

4. 実験

提案手法の有効性を確認するため，計算機実装による実験を行なった．まず，簡単な例として，タスク数 9 のデータフローグラフに対しての実験を行なった．

ここで，コアタイプは 2 種類とした．各タスクにおける

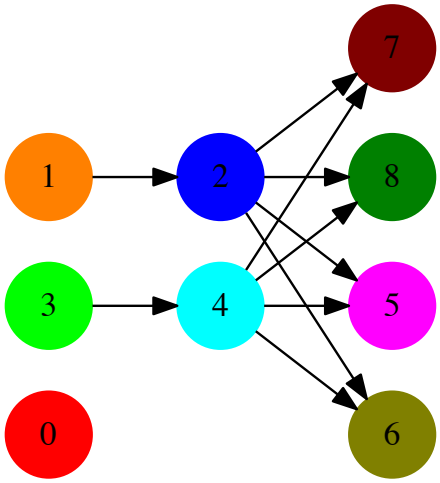

図 2 タスク数 9 のデータフローグラフの例

各コアタイプでの実行時間分布を表 1 に示し，また，コア使用制約を表 2 に示す．

表 1 各タスクの実行時間分布 (μ : 平均, σ : 標準偏差)

Task	TYPE0	TYPE1
0	N(12,0.5)	N(5,0.1)
1	N(5,0.2)	N(3,0.3)
2	N(20,0.5)	N(18,0.02)
3	N(8,1.5)	N(4,0.3)
4	N(12,0.5)	N(9,0.1)
5	N(15,0.3)	N(12,0.7)
6	N(16,1.6)	N(6,0.3)
7	N(29,0.07)	N(15,0.2)
8	N(21,1.05)	N(13,0.8)

表 2 コア使用制約

	TYPE0	TYPE1
set1	3	0
set2	2	1
set3	0	2

そして，TAS 最適化の結果を図 3 に示す．

この結果に対し，1000 回のモンテカルロシミュレーションによって，実行時の性能を比較した．結果を図 4 に示す．ここで，横軸は全タスクの終了時刻，縦軸は頻度となっている．また，EVaDys は本稿の提案手法，Previous が [11]，PEVaS が [10] の手法の結果となっている．結果を見ると，PEVaS は他の 2 手法と比較し，全体的に分布が大きい．また，Previous と EVaDys はほとんど差が無いが，若干，EVaDys の方が小さい傾向にある．これは，平均を見ても，EVaDys が 43.755 で終了しているのに対し，PEVaS が 44.014，Previous が 44.082 と EVaDys が小さくなっていることからもわかる．

次に，[14] で公開されている DFG から，50 タスクのデータに対し，提案手法の枠組みに従って実験を行なった．入力のデータフローグラフを図 5 に示す．なお，[14] では，各タスクの実行時間は定数となっているため，この実験では，与えられた値を平均値とし，標準偏差は適宜決定したものを用いた正規分布とした．また，コア使用制約は，表 3 に示す．

表 3 コア使用制約

	TYPE0	TYPE1
set1	4	0
set2	2	1
set3	0	2

結果を図 6 に示す．また，各手法での平均値は，提案手法である EVaDys で 79.502，Previous で 80.106，PEVaS で 80.606 となっている．図 6 によると，この場合でも，EVaDys の結果が，他の手法の結果よりも，小さい終了時刻を実現できている．

以上より，

- PEVaS は分布が大きい．すなわち，早く終了する場合もあるが，一方で悪化するときもある．
- Previous は平均は PEVaS とほとんど同じであるが，分布の範囲が狭くなっている．これは，Waiting Dependency により，最適化の状況を保持しており，待たなければならない実行は待っていることで，最悪は改善している．しかし，一方で，この仮想的な依存関係は，悲観的な予測に基づく手法となっており，本来不要であるような状況でも，待たなければならない．その結果，実行時間が短かい結果もほとんど無い状況となる．
- EVaDys は，実行時に，待つかどうかの判定を統計的な処理で行なうため，従来の手法よりも適切な待ちとなっている．結果，終了時刻が早くなっている．

よって，提案手法の有効性が確認できた．

5. おわりに

本稿では，実行時間がばらつくタスクに対する動的スケジューリング手法として，EVaDys を提案した．この手法は，開始可能なタスクに対し，そのタスクの実行がより優先度の高いタスクの実行を妨げるかどうかを統計的な処理に基づいて判断し，妨げると判断した場合は，その実行開始を延期することにより，必要なときに優先度の高いタスクの実行を行なえる仕組みを実現した．実験により，提案手法の有効性を確認することができた．

今後の課題としては，タスクの実行時間モデルの正当性の評価が必要であると考えている．現在は，各タスクの実行時間が独立な正規分布であるという仮定を考えている．これに関しては，ある程度の処理が合わされれば，中心極限定理により，正規分布として表現できると予想している．また，相関がある場合は，分散共分散行列のコレスキー分

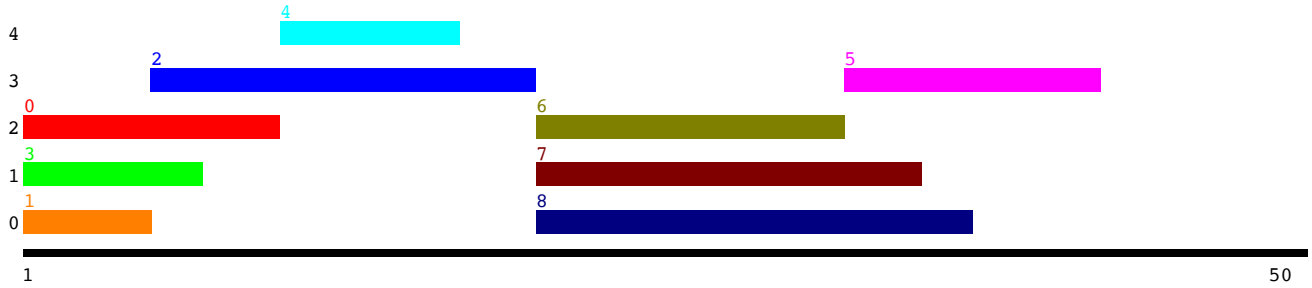

図 3　構成された TAS

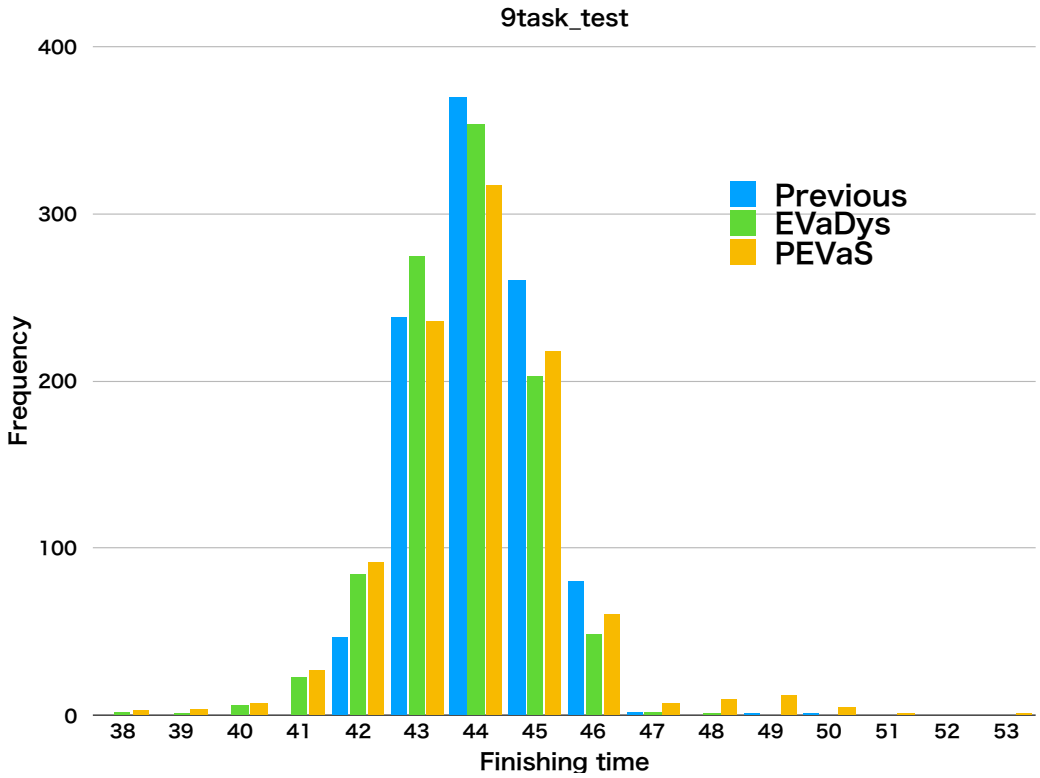

図 4　実行結果

解を利用することにより，独立な乱数の線形和として表現できることも知られている．この事実を利用することにより，本稿の手法の拡張ができると考えている．

参考文献

[1] Y. Chen, W. Chang, C. Liu, B. Chen, and M. Tsai: *Processor and Memory Co-Allocation for MPSoCs with Single-ISA Heterogeneous Multi-Core Architecture*, Synthesis And System Integration of Mixed Information technologies (SASIMI) 2018, 216-221, 2018.

[2] M. Becker, S.Mubeen, D. Dasari, M. Behnam, T. Nolte: *Scheduling Multi-Rate Real-Time Applications on Clustered Many-Core Architectures with Memory Constraints*, Asia and South Pacific Design Automation Conference (ASP-DAC) 2018, 560-567, 2018.

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

[4] Y. Liu, Y. Li, Y. Zhao, and X. Chen: *A scheduling algorithm in the randomly heterogeneous multi-core processor*, ICNC-FSKD, 2140-2146, 2016.

[5] C. Hao, N. Wang, and T. Yoshimura: *A Unified Scheduling Approach for Power and Resource Optimization with Multiple V-dd or/and V-th in High Level Synthesis*, Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD), 2017.

[6] S. Hao, Q. Liu, L. Zhang, and J. Wang: *Processes Scheduling on Heterogeneous Multi-core Architecture with Hardware Support*, Sixth International Conference on Networking, Architecture, and Storage, 236-241, 2011.

[7] F. Wang, C. Nicopoulos, X. Wu, Y. Xie, and N. Vijaykrishnan: *Variation-Aware Task Allocation and Scheduling for MPSoC*. Proc. of ICCAD, 598-603, 2007.

[8] T. Nakada, T. Hatanaka, H. Nakamura, H. Ueki, M. Hayashikoshi, and T. Shimizu: *An adaptive energy-efficient task scheduling under execution time variation based on statistical analysis*, VLSI-SoC, 1-7, 2016.

[9] M. Chen, D. Yue, X. Qin, X. Fu, and P. 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).

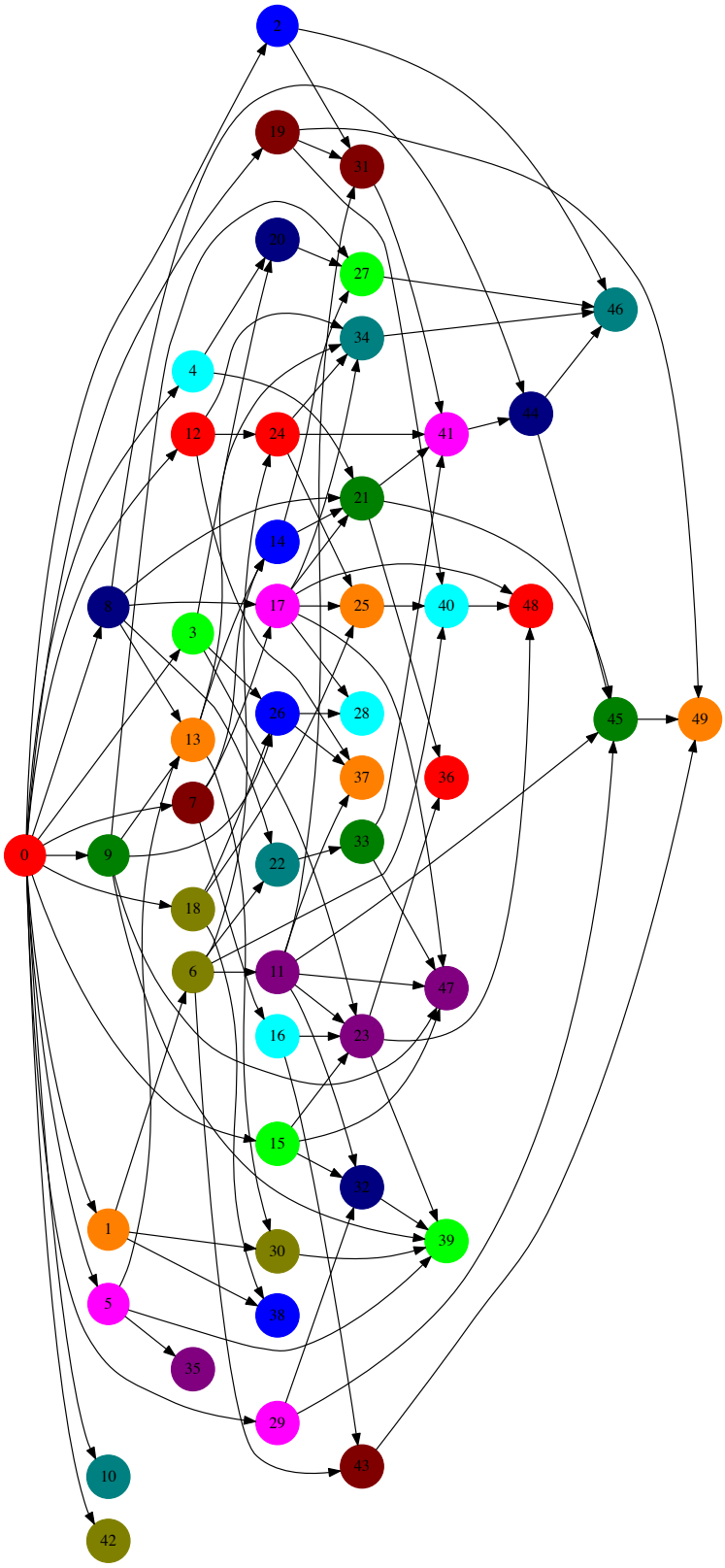

図 5 50 タスクのデータフローグラフ

[10] K. Nomura, Y. Takashima, and Y. Nakamura: *PEVaS: Power and Execution-time Variation-aware Scheduling for MPSoC*, IEEE New Circuits and Systems Conference (NEWCAS) 2016, 2016.

[11] R. Tsuchihashi, K. Nomura, Y. Takashima, and Y.Nakamura: *Task allocation and scheduling optimiza-*

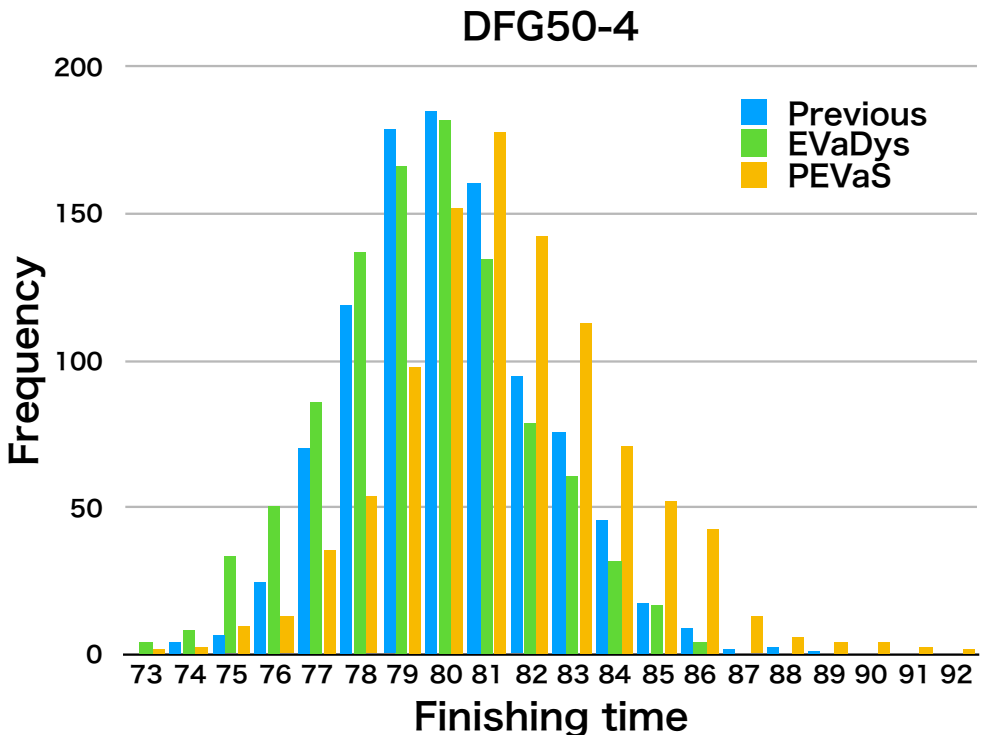

図 6 50 タスクに対する結果

tion in the heterogeneous core system, 2017 International Conference On Computer Aided Design (ICCAD), workshops, 10th IEEE/ACM Workshop on Variability Modeling and Characterization(VMC)(2017).

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

[13] W. Shimoyama, S. Tsukiyama, and Y. Takagi: *An Implementation of a Statistical Static Timing Analyzer Considering Path-Delay Correlation and Evaluation*, *IEICE Trans. on Fundamentals, Vol.J90-A*, NO.11, pp.826–838, 2007.

[14] Kasahara Laboratory: *Standard Task Graph Set*, `http://www.kasahara.cs.waseda.ac.jp/schedule/index.html`.