一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CS2019-47(2019-09)

[依頼講演] ファンクション集約によるサービスファンクションのスケジューリング

金光　永煥[†,††]　金井　謙治[††]　甲藤　二郎[†††]　中里　秀則[†††]

† 東京工科大学コンピュータサイエンス学部
†† 早稲田大学理工学術院総合研究所
††† 早稲田大学理工学術院基幹理工学部情報通信学科

あらまし　本稿では，サービスファンクションの集約に基づくファンクションの計算資源への割り当て，及びスケジューリング手法を提案する．従来のファンクション配備・スケジューリング手法では，残余処理能力，及び空き時間スロットを持つ計算資源へファンクションを割り当てている．その結果，従来手法では多くの計算資源へファンクションが割り当てられ，計算資源が枯渇する問題がある．そこで提案手法ではファンクション集約によって実行粒度を上げ，かつファンクション同士を共有することによってファンクション及び計算資源を有効利用する手法を提案する．実験の結果，応答時間，使用資源数，ファンクション共有数の向上が認められた．

キーワード　NFV, Function Chaining，クラスタリング，スケジューリング，ワークフロー．

Clustering-based Service Function Scheduling Algorithm for Service Function Chaining

Hidehiro KANEMITSU[†,††], Kenji KANAI[††], Jiro KATTO[†††], and Hidenori NAKAZATO[†††]

† School of Compute Science, Tokyo University of Technology.
†† Waseda Research Institute for Science and Engineering, Waseda University.
††† Department of Communications and Computer Engineering, Waseda University.

Abstract　Virtualized service and network functions are deployed on virtual machines (VMs) to realize essential processing to realize service function chaining (SFC). Issues on SFC is SF allocation to a VM and to minimize the response time and number of function instances. In this paper, we propose an SF clustering-based scheduling algorithm, called “SF-clustering for utilizing virtual CPUs” (SF-CUV), to solve the SF allocation and SF selection problems simultaneously. Experimental results show that SF-CUV can utilize vCPUs to minimize the response time.

Key words　NFV, Function Chaining, Clustering, Scheduling, Workflow.

1. はじめに

近年の情報通信ネットワークでは，ルータ，ファイアウォール，ロードバランサ等のネットワーク処理装置の仮想化（NFV: Network Function Virtualization）により，ネットワーク機能の可搬性の実現されている [1], [2]．NFV によって物理装置が不要となる一方，ネットワーク管理者は，仮想化されたネットワーク機能（VNF: Virtualized Network Function）の配備を考慮する必要性が生じる．また，ネットワーク経由で処理される処理モジュールをサービスとして考えた場合，これらはサービスファンクション（SF）と呼ばれている．SF は，一般に入力と出力情報を備えた仮想化ファンクションであり，SF どうしで入出力データの通信を行うものはサービスファンクションチェイン（SFC）と呼ばれる [3], [4]．SFC には，SF の計算資源への割り当てが必要になるが，いくつかの課題が存在する．例えば SF 間の通信時間の最小化，使用される仮想マシン（VM）またはコンテナのインスタンス数の最小化，応答時間及び遅延の最小化が挙げられる．これらの課題に対する従来手法では，SF の割り当て，VM インスタンス数，SF のルーティング経路に関して，探索アルゴリズム，線形計画法等に基づく手法が提案されている．[5]〜[19].

SFC において高スループット処理を実現するためには，少ない計算資源数で 1 つあたりのチェインの処理の所要時間を最小化する必要がある．例えばクラウド間で SFC を行う場合，VM

上に配備するファンクションを複数の SFC 間で共有することによってファンクションのインスタンス起動に要する負荷や時間を最小限に抑える必要がある．さらに，複数ファンクション同士をできるだけ同時に処理できるような SF 処理の並列度を保つ必要がある．すなわち，起動させるファンクション数，及び割り当てられる VM 数双方を最小に抑えるような基準が必要となる．しかしながら，現状で現実的な計算量で解を導出するためのアルゴリズムは存在しない．

本稿では，SF 数及び計算資源数の双方の最小化，及び応答時間（スケジュール長）の最小化を狙った SF スケジューリングアルゴリズム（SF-CUV: SF-Clustering for Utilizing vCPUs）を提案する．SF-CUV では，2 フェーズの処理から構成される．すなわち (i) SF クラスタリング及び予備的 vCPU 割り当てフェーズ，及び (ii) SF の再割り当て及びスケジューリングである．特に (ii) では，仮想 CPU（vCPU）の負荷制約条件を満たしつつ，vCPU のアイドル時間スロットに対して SF を割り当てることにより，スケジュール長の最小化を実現する．実験の結果，SF-CUV は SF インスタンス数，及び少ない vCPU 数でスケジュール長が従来手法よりも短くなることが確認された．

2. SF-CUV アルゴリズム

2.1 概　　要

SF-CUV は，(i) SF クラスタリング，及び (ii) SF スケジューリングの 2 つのフェーズから構成される．(i) では，各 SF は「単一の SF クラスタ」，すなわち唯一つの SF を保持するクラスタに属する状態と想定する．2 つの SF クラスタどうしが集約されて一つの SF クラスタとなり，これによって SF 間の通信が局所化される．また，SFC は，各 SF が互いにデータ依存関係にあるようなワークフロー構成を想定する．したがって，本稿におけるスケジュール長とは，開始となる SF（START SF）の開始時刻から，最後に処理される SF（END SF）の完了時刻までの期間である．SF-CUV の目的はスケジュール長の最小化であるが，これと同時に，処理に要する SF インスタンス数及び vCPU 数（VM に割り当てられる仮想 CPU）の最小化を満たすことである．

次に, SF の集約条件について述べる．スケジュール長は END SF の完了時刻であるから，各 SF が vCPU へ割り当てられ，かつ実行順が定まって初めて導出される．したがって，(i) において SF を集約する時点では正確なスケジュール長は決まらない．そこで SF-CUV では，WSL（Worst Schedule Length）[20] という指標を用いて SF 集約を行う．WSL とは，各 SF ができるだけ後に実行される場合において，外部ホストからのデータ到着を待つことなく開始できる場合におけるスケジュール長の上限値である．すなわち自身とは依存関係のない SF が先にスケジュールされた後に実行開始できる場合のスケジュール長である．また，文献 [20] より，WSL を小さくするようなタスク割り当ては，スケジュール後のスケジュール長の上限及び下限値も小さくなることが証明されている．SF-CUV の (i) では，この WSL を小さくなるように SF 集約を行う．

SF クラスタは，1 つ以上の SF を含むものであり，i 番目の

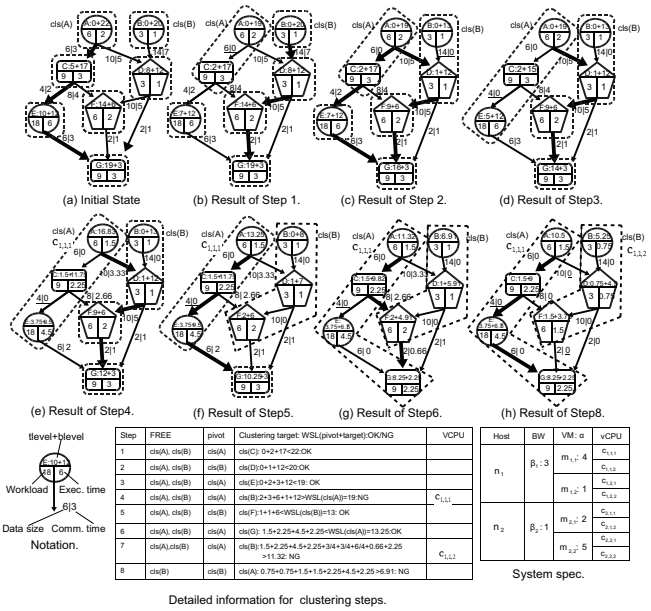


図 1　SF-CUV における (i) の例.

SF クラスタを $cls(i)$ と定義する．ここで，初期の SF クラスタは $cls(i) = \{v_i\}$ である．本稿における SF 集約（クラスタリング）を，$cls(i) \leftarrow cls(i) \cup cls(j)$ と定義する．

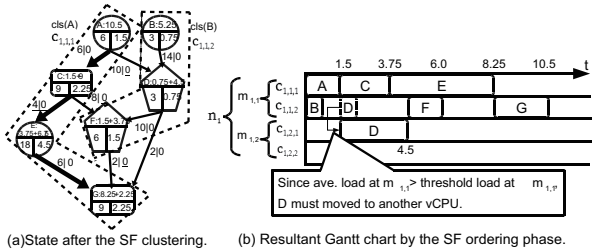
2.2 SF クラスタリング及び予備的 vCPU 割り当て

最初に，1 つの SF のみ保持する各 SF クラスタは，UEX という，未走査集合に属するものとする．もし SF クラスタがクラスタリングによって走査されると UEX から除外される．そして，$UEX = \emptyset$ となれば (i) は完了する．まず，クラスタリングの起点となる SF クラスタを “pivot"として，$FREE$ という集合から選択する．$FREE$ とは，その先行 SF がすべて走査済みであるような SF クラスタの集合である．pivot は，$FREE$ の中で WSL を支配しているものである．そして，pivot から辿って得られる WSL 値の実行経路上にある後続 SF が属する SF クラスタを “target"として選択し，pivot と target をクラスタリングする．

［例 1］ 図 1 に，SF クラスタリング及び予備的 vCPU 割り当ての例を示す．この例では，太矢印は WSL を決める実行経路を示す．初期状態（a）から，各コスト値は System spec. で定義されている性能値から，平均処理速度 ((4+1+2+5)/4=3) 及び平均通信帯域幅 ((3+1)/2=2) に従って割り当てられているものとする．$cls(A)$ はクラスタリングの起点，すなわち pivot として選択される．なぜなら，$FREE$ において Fig. 1 の表のうち, $WSL(cls(A)) = 22 > WSL(cls(B)) = 20$ であるからである．一方，$cls(C)$ は $cls(A)$ から辿る WSL を支配する後続の SF クラスタであるから，target として $cls(C)$ が選択される．(d) において，pivot として $cls(A)$，target として $cls(B)$ がそれぞれ選択されてクラスタリングされる．(h) において $FREE = \{cls(B)\}$ であるから，pivot として $cls(B)$，target として $cls(A)$ がそれぞれ選択され，クラスタリングされる．□

2.3 SF の再割り当て及びスケジューリング

各 SF は，(i) によって vCPU へ割り当てられているため，このフェーズでは実際のスケジューリングを行う．このフェーズ



(a)State after the SF clustering. (b) Resultant Gantt chart by the SF ordering phase.

Host	BW	VM: a	vCPU
n_1	$\beta_1:3$	$m_{1,1}:4$	$c_{1,1,1}$
			$c_{1,1,2}$
		$m_{1,2}:1$	$c_{1,2,1}$
			$c_{1,2,2}$
n_2	$\beta_2:1$	$m_{2,1}:2$	$c_{2,1,1}$
			$c_{2,1,2}$
		$m_{2,2}:5$	$c_{2,2,1}$
			$c_{2,2,2}$

(c) System spec.

Step	fList	Selected SF	Allcation target
1	A, B	A	$c_{1,1,1}$
2	B, C	C	$c_{1,1,1}$
3	B, E	E	$c_{1,1,1}$
4	B	B	$c_{1,1,2}$
5	D	D	$\frac{U(C,c_{1,1,1})+U(D,c_{1,1,2})}{2} > U_{th}(m_{1,1})$ then move D from $c_{1,1}$ to $c_{1,2,1}$
6	F	F	$c_{1,1,2}$
7	G	G	$c_{1,1,2}$

(d) Detailed information for SF ordering steps.

図 2 Example of SF Ordering and Actual vCPU Allocation.

では，下記の条件を満たしつつ各 SF を vCPU のアイドル時間スロットへ割り当てる．(i) で SF が vCPU へ割り当てられることにより，このフェーズにおけるスケジュール優先度を実際の割り当てパターンに従って算出することができる．

$$\frac{U_{th}(c_{k,l})}{\frac{1}{|c_{k,l}|}\sum_{m=1}^{|c_{k,l}|} U(c_{k,l,m}, v_i)} \geq 1, \quad (1)$$

ここで $U_{th}(c_{k,l})$ は CPU コア $c_{k,l}$ において定められた，単位時間あたりの最大占有率である．

SF v_i が，その完了時刻を最小にするような vCPU，すなわち $c_{k,l,m}$ のアイドル時間スロットへ割り当てられる時，もし (1) を満足すればこの割り当ては行われ，そうでなければ，ホスト数の増加を抑えるために「同一ホスト n_k 内の別の CPU コア内の vCPU」への割り当てを試みる．それでも条件を満たさなければ，SF-CUV は他ホスト内の vCPU への割り当てとなる．

［例 2］ 図 2 に，(ii) のフェーズの例を示す．(a) は図 1 の (h) に対応する．この状態から，(d) の Step 1 において $fList$ から A を選択する．ここで $fList$ とは，その先行 SF がすべてスケジュール済みであるような SF の集合である．$A(A) = c_{1,1,1}$ であるから，実際の割り当て先となる vCPU は n_1 から選択される．この場合，A の完了時刻が最小となるような vCPU は $c_{1,1,1}$ である．そして，$fList$ から A が除外され，Step2 として C が $fList$ から選択される．特に Step 5 において D が選択され，SF-CUV は D を $c_{1,1,2}$ のアイドル時間スロットへの割り当てを試みる．ここで $U(C, c_{1,1,1}) = 80$ であると仮定すると，$U(D, c_{1,1,2}) = 60$，$U_{th}(m_{1,1}) = 60$ である．(1) において $\frac{80+60}{2} = 70 > U_{th}(m_{1,1}) = 60$ であるから，D は n_1 内の他の vCPU へ割り当てられる．代替の vCPU のうちで，D の完了時刻が最小となるのは $c_{1,2,1}$ である．そのため，D は $c_{1,2,1}$ のアイドル時間スロットへ割り当てられる．このように，全ての SF が各 vCPU へ再割り当てされる．□

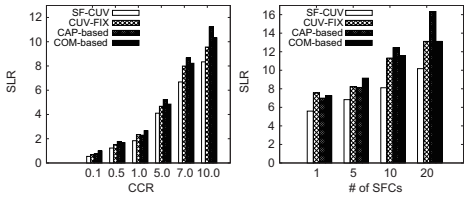
3. 予備的実験

3.1 実験環境

シミュレーションによって，従来の SFC のアルゴリズムと比較を行った．本実験では，1 つ以上の SFC が複数のクラウドで同時に処理され，各クラウドでは複数の物理ホストが配備されているものとする．複数の SFC のスケジューリングにおいて，i 番目の SFC を $S_i = (V_i, E_i)$ とし，全体の SFC を $S = (V, E) = (V_1 \cup V_2 \cup \ldots V_n, E_1 \cup E_2 \cup \ldots E_n)$ とする．すなわち S は，シミュレーションの入力となる．

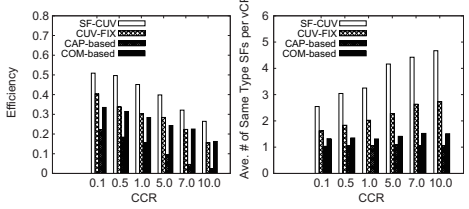
本シミュレータを，jdk1.8.0_191, and the CPU is Intel(R) Xeon(R) E-2176M 2.70 GHz with 32 GB RAM 上で実行した．この状況下で，シミュレーションではスケジュール長，vCPU がどの程度有効利用されたか，及びどの程度 SF が共有されたかを下記の手法と比較した．

- SF-CUV において SF の再割り当てを行わないアルゴリズム (CUV-FIX):
- 容量制約に基づく SF 割り当て (CAP-based) [6], [9]:
- 通信の局所性に着目した SF 割り当て (COM-based) [5], [10]:



(a) With varying CCR (b) With varying ♯ of SFCs

図 3 スケジュール長に関する比較結果



(a) Efficiency (b) Degree of SF Sharing

図 4 資源の有効利用に関する比較結果

3.2 スケジュール長の比較

この実験では，SFC において communication-to-computation ratios (CCR) [20] を変動させてスケジュール長を比較した．ここで $0 < CCR \leq 10$ とし，計算集約型からデータ集約型まで，様々な特徴を持った SFC を入力とした．スケジュール長の指標として，Schedule Length Ratio (SLR) [20] を用いた．図 3 に，スケジュール長の比較結果を示す．(a) は，CCR を変動させた場合，(b) は SFC 数を変動させた場合の比較結果である．3 (a) より，SF-CUV は他手法よりも小さな SLR 値を達成していることが分かる．一方，3 (b) より，SFC の数が増えるほど SLR は大きくなるが，SF-CUV は最も小さな SLR 値を達成していることが分かる．以上より，SF-CUV は，スケジュール長に関していずれの CCR，及び SFC 数において従来よりも良い結果となった．

3.3 資源の有効利用の比較

この実験では，資源，すなわち計算資源（vCPU）及び SF のインスタンス数に関する比較を行った．そのため，2 つの指標，すなわち (i) Efficiency [20]，及び (ii)vCPU に割り当てられる SF 数の平均値に関して比較を行った．特に Efficiency は vCPU あたりの Speed Up 率であり，どの程度 vCPU が有効利用されているかを示す値である．図 4 に，比較結果を示す．図中，(a) は Efficiency の結果，(b) は SF の共有の度合いの結果を示す．図 4 (a) では，SF-CUV は最も高い Efficiency を示しており，CAP-based の手法では最も低い Efficiency であった．CAP-based では最も容量の空きが最も大きな vCPU へ SF を割り当てているため，必然的に未割り当ての vCPU が優先的に SF の割り当て先となるためである．CUV-FIX については，割り当てられた vCPU 数は，クラスタリングフェーズにおいては SF-CUV と同一であるが，SF-CUV は再割り当てフェーズがある．そのため，そのような再割当ては Efficiency に貢献している，という結果であることが分かる．

図 4 (b) より，SF-CUV は全ての CCR において最も高い SF 共有度を達成していることが分かる．一方，CAP-based が最も低く，このことから容量制約に基づく SF 割り当ては資源の有効利用にはつながらないという結果が得られた．

4. まとめと今後の課題

本稿では，SF の集約（クラスタリング）に基づく SF の vCPU への割り当て，及びスケジューリングアルゴリズム SF-CUV を提案した．SF-CUV は，スケジュール長，vCPU 数，そして起動する SF 数の最小化を特徴とする．シミュレーションによる予備的実験の結果，SF-CUV が従来手法よりもスケジュール長が小さく，かつ資源の有効利用を達成できることが分かった．

今後の課題としては，実環境への実装と実用性の検証が挙げられる．

謝辞 本研究成果は，戦略的情報通信研究開発推進事業 (国際標準獲得型)「スマートシティアプリケーションに拡張性と相互運用性をもたらす仮想 IoT-クラウド連携基盤の研究開発 (Fed4IoT)」の支援を受けている．また，総務省「IoT 機器増大に対応した有無線最適制御型電波有効利用基盤技術の研究開発：技術課題ア「有無線ネットワーク仮想化の自動制御技術」」，及び科学技術研究費（基盤研究（C））:「クラウド間連携と仮想化ファンクション集約による計算資源の有効利用に関する研究」による一部支援を受けている．

文　献

[1] R. Mijumbi, J. Serrat, J-L. Gorricho et al., “Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Commun. Surv. & Tutorials*, vol. 18, no. 1, pp. 236–262, Sep., 2016.

[2] A. Belbekkouche, M.M. Hasan, and A. Karmouch, “Resource Discovery and Allocation in Network Virtualization," *IEEE Commun. Surv. & Tutorials*, vol. 14, no. 4, pp. 1114–1128, Feb., 2012.

[3] D. Bhamare, R. Jain, M. Samaka, et al., “A survey on service function chaining," *J. Netw. Comput. Appli.*, Vol. 75, pp. 138–155, Sep., 2016.

[4] P. Quinn and J. Guichard, “Service Function Chaining: Creating a Service Plane via Network Service Headers," *Computer*, vol. 47, no. 11, pp. 38–44, Nov. 2014.

[5] L. Wang, Z. Lu, X. Wen, et al., “Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization," *IEEE Access*, vol. 4, pp. 8084–8094, Nov., 2016.

[6] M. C. Luizelli, W. L. C. Cordeiro, L. S. Buriol, et al., “A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining," *Comput. Commun.*, vol. 102, 67–77, Nov., 2016.

[7] T-W. Kuo, B-H. Liou, K. C-J Lin, et al., “Deploying Chains of Virtual Network Functions: On the Relation Between Link and Server Usage," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1562–1576, Aug., 2018.

[8] M. Ghaznavi, N. Shahriar, S. Kamali, et al., “Distributed Service Function Chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11. pp. 2479–2489, Nov., 2017.

[9] D. Bhamare, M. Samaka, A. Erbad, et al., “Multi-Objective Scheduling of Micro-Services for Optimal Service Function Chains," *in Proc. IEEE ICC 2017 SAC Symp. Cloud Commun. Netw. Track*, pp. 1–6, May, 2017.

[10] M. T. Beck and J. F. Botero, “Scalable and coordinated allocation of service function chains," *Comput. Commun.*, vol. 102, pp. 72–88, Oct. 2016.

[11] Y. Sang, B. Ji, G. R. Gupta, et al., “Provably Efficient Algorithms for Joint Placement and Allocation of Virtual Network Functions," *in Proc. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM 2017)*, pp. 1–9, May, 2017.

[12] S. Sahhaf, W. Tavernier, M. Rost, et al., “Network service chaining with optimized network function embedding supporting service decompositions," *Comput. Netw.*, vol. 93, pp. 492–505, Oct., 2015.

[13] H. A. Alameddine, S. Sebbah, and C. Assi, “On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach," *IEEE Trans. Netw. Serv. Managem.*, vol. 14, no. 4, pp. 860–874, Dec., 2017.

[14] Z. Li and Y. Yang, “Placement of Virtual Network Functions in Hybrid Data Center Networks," *in Proc. IEEE Int. Symp. High-Performance Interconnects*, pp. 73–79, Aug., 2017.

[15] S. Khebbache, M. Hadji, and D. Zeghlache, “Virtualized network functions chaining and routing algorithms," *Comput. Netw.*, vol. 114, pp. 95–110, Jan., 2017.

[16] H. A Alameddine, S. Sebbah, and C. Assi, “On the Interplay Between Network Function Mapping and Scheduling in VNF-Based Networks: A Column Generation Approach," *IEEE Trans. Netw. Serv. Managem.*, Vol. 14, No. 4, pp. 860–874, Dec., 2017.

[17] V. Eramo, E. Miucci, M. Ammar et al., “An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," *IEEE/ACM Trans. Netw.*, Vol. 25, No. 4, pp. 2008–2025, Aug., 2017.

[18] O. Soualah, M. Mechtri, C. Ghribi et al., “An Effcient Algorithm for Virtual Network Function Placement and Chaining," *in Proc. IEEE Annu. Consumer Communications & Networking Conference (CCNC)*, pp. 647–652, Jan., 2017.

[19] X. Lin, D. Guo, Y. Shen et al., “DAG-SFC: Minimize the Embedding Cost of SFC with Parallel VNFs," *in Proc. Int. Conf. Parallel Processing (ICPP 2018)*, 10 pages, Aug. 2018.

[20] H. Kanemitsu, M. Hanada, and H. Nakazato, “Clustering-based Task Scheduling in a Large Number of Heterogeneous Processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3144–3157, Nov., 2016.