環境適応における包括的運用中再構成

Comprehensive service reconfiguration during service operation

山登庸次
Yoji Yamato

日本電信電話（株）　ネットワークサービスシステム研究所
Network Service Systems Laboratories, NTT Corporation

１．はじめに

近年，クラウド，IoT 等[1]-[6]様々な領域で，FPGA 等ヘテロハード利用が増えているが，活用の壁は高い．私は，環境に合わせ，コード変換等自動で行い高性能に動作させる，環境適応ソフトウェアを提案し[7]-[9]，取組んできた．

環境適応ソフトウェアでは，変換，配置等の適応処理を行った後，アプリケーションの運用を開始する．更に，運用開始後，定期的に，運用に即したより適切な構成がないか検証し，変更した方が効果が高ければ再構成を行う．

本稿では，今まで，再配置等，個々の要素毎に検討していた運用中再構成処理を包括的に行う整理を行う．

２．運用開始前処理のレビュー

環境適応ソフトウェアでは，運用開始前に，ユーザ指定アプリケーションを，分析し，コード変換，リソース量調整，配置場所調整，検証を行った後，運用開始する．次に，運用開始後に，利用特性等分析して，適切な再構成を行う．

コード変換は，変換先としてマルチコア CPU，GPU，FPGA を対象とする．マルチコア CPU，GPU ではオフロードに適したループ文を，試験環境性能測定を元に高速なパターンを遺伝的アルゴリズムにより探索する．FPGA では，算術強度等を用いて効果が高いと見込まれるループ文を複数オフロードし性能を測定して高速なパターンを探索する．

リソース量では，変換時に測定の CPU とオフロード先の実行時間を元に，処理時間が同オーダーになるリソース比を定め，ユーザ価格条件を満たすリソース量を決定する．

配置場所では，変換アプリケーションのサンプルテストケースでの実行時間と，クラウドやエッジに配置した際のコストをモデル化して，ユーザの応答時間，価格の条件を満たす配置を線形計画手法を用いて計算し，順に配置する．

配置計算後，商用サーバにアプリケーションを配置して，Jenkins で正常動作確認後，ユーザ承認を得て運用開始する．

3．運用開始後の包括的再構成

運用開始前は，ユーザの想定サンプルテストケースで適応処理を行うため，運用開始後，性能が期待通りでないことが想定される．例えば，当初は SQL クエリが多い想定だったが，運用では NoSQL クエリが多く，NoSQL アクセラレート FPGA ロジックの方が適切だったなどである．

実運用に対応するため，1 か月等の定期的に，より適切な構成がないかを検証環境で試行し，効果が高い場合に再構成する，運用中再構成を行う．再構成について，今まで FPGA ロジック再構成，再配置等要素毎に別論文にしてきた．本稿では，それらを統合して包括的再構成を整理する．

再構成対象は，マルチコア CPU，GPU，FPGA へのコード変換，リソース量調整，配置調整である．市中では，クラウドで仮想マシンの数を自動増減することがされるが，リソース量再構成では，数だけでなく，GPU と CPU のメモリ比変更等も行う．全再構成試行中で，運用開始前はユーザ想定サンプルテストケースで適応処理を行っていたが，商用での実運用データを用いて試行することは共通である．

コードの変換では，どのハードの場合でも，一定期間毎に，商用環境のリクエストを集計し，負荷が上位のアプリケーションを把握し，そこで使われる商用最頻データを用いて最適化を検証環境で実施する．GPU では遺伝的アルゴリズムを使う等，最適化試行は運用開始前と同様である．試行した最速パターンについて，再構成前と再構成した場合の性能を測定し，商用環境でのリクエスト頻度の場合に，再構成した性能改善効果が閾値以上であれば再構成を行う．

リソース量では，CPU とオフロード先実行時間を商用最頻データを用いて再測定し，処理時間が同オーダーになるようにリソース比を定め，条件を満たすリソース量にする．

配置場所では，運用開始前は 1 アプリケーション毎に順番に配置すればよかったが，運用開始後の再構成時は，多くのアプリケーションがサーバに配置されているため，ユーザの当初要件は満たしながら，全アプリケーションに対し応答時間と価格の最適解を，線形計画手法で計算する．

運用開始前は全適応処理を行ってから開始するが，再構成時は配置だけ再構成試行する等，独立に行っても良い．

4．まとめ

本稿では，環境適応ソフトウェアの運用中再構成処理を包括的に整理した．現在はこれまで個々に実装評価していた再構成処理を全て行った際の，処理時間等測定している．

参考文献

[1] H. Noguchi, et al., "Distributed Search Architecture for Object Tracking in the Internet of Things," IEEE Access, 2018.
[2] Y. Yamato, et al., "Proposal of Real Time Predictive Maintenance Platform with 3D Printer for Business Vehicles," International Journal of Information and Electronics Engineering, Vol.6, No.5, pp.289-293, 2016.
[3] Y. Yamato, et al., "Fast Restoration Method of Virtual Resources on OpenStack," IEEE CCNC 2015, Jan. 2015.
[4] H. Noguchi, et al., "Device Identification Based on Communication Analysis for the Internet of Things," IEEE Access, DOI: 10.1109/ACCESS.2019.2910848, Apr. 2019.
[5] H. Noguchi, et al., "Autonomous Device Identification Architecture for Internet of Things," IEEE WF-IoT 2018, 2018.
[6] Y. Yamato, "Automatic Verification for Plural Virtual Machines Patches," ICUFN 2015, pp.837-838, 2015.
[7] Y. Yamato, "Improvement Proposal of Automatic GPU Offloading Technology," ICIET 2020, pp.242-246, Mar. 2020.
[8] Y. Yamato, "Automatic Offloading Method of Loop Statements of Software to FPGA," Int. J. Parallel Emergent Distrib. Syst., Taylor & Francis, Apr. 2021.
[9] Y. Yamato, "Proposal of Automatic Offloading for Function Blocks of Applications," ICIAE 2020, pp.4-11, Mar. 2020.