産業応用工学会論文誌, Vol. 12, No. 2, pp. 203-205 (Sept. 2024)
The Japanese Journal of the Institute of Industrial Applications Engineers
Online edition: ISSN 2187-5146, Print edition: ISSN 2189-373X, DOI: 10.12792/jjiiae.12.2.203

レター

アプリケーション自動分割技術の IoT サービス適用の検討と評価

山登　庸次 [a,*]

Study and Evaluation of the IoT Services Adopting of Application Automatic Division

Yoji Yamato[a,*]

(Received February 23, 2024; revised April 20, 2024; accepted May 8, 2024)

Abstract

We have proposed the concept of environment-adaptive software that automatically converts and operates program code so that it can appropriately utilize the environment in which it is placed. This paper applies an automatic dividing method to IoT services as an element of environment-adaptive software, thereby making it easier for users to customize their own services.

キーワード : 環境適応ソフトウェア，IoT サービス，機能追加，自動分割，動的分析
Keywords : Environment-adaptive software, IoT services, Function addition, Automatic division, Dynamic analysis.

1. はじめに

CPU だけでなく, GPU（Graphics Processing Unit）, FPGA（Field Programmable Gate Array）や IoT（Internet of Things）機器（(1)-(9)等）等ヘテロなハードウェアが，用いられるようになっている。 Microsoft 社は，クラウド技術使い（例えば(10)-(14)），GPU や FPGA 処理を提供，利用している(15)。しかし，ヘテロなハードウェアを利用するためには，大半の技術者にとって壁が高い(16)-(21)。GPU，FPGA ではハードウェアに応じ専門言語や, IoT 機器では IoT PF（Platform）知識が必要となってくることが多い。

通常のソフトウェアを, 分析して, 適用する環境（GPU，FPGA，IoT 機器等）にあわせて，適切に変換，設定を行い，環境に適応した動作をさせることを，自動で行うことが今後求められていくと考える。

そこで，私は，通常 CPU 向けコードを，変換等自動で行い配置先環境で適切に動作させる，環境適応ソフトウェアを提案している。その要素として，GPU，FPGA に自動オフロードする方式を提案している(22)-(26)。IoT 機器向けに，ユーザは基本サービスを選びユーザ独自処理と利用 IoT 機器を指定すれば，IoT サービスを自動構築する方式も提案している。更に，汎用的プログラムでも，呼び出し関係に基づいて, アプリケーションを分割して，サービス機能追加変更の容易化をしている。

これまで自動分割方式検証は，C 言語の計算系のアプリケーションに限られていた。本稿では，IoT サービスに自動分割方式を適用することで，これまで，事前に事業者の基本サービス提供が必要だった IoT サービスでも，ユーザが独自カスタマイズを容易化できるようにする。

* Corresponding author. E-mail: yoji.yamato@ntt.com
a 日本電信電話（株）　ネットワークサービスシステム研究所
〒180-8585　東京都武蔵野市緑町 3-9-11
Network Service Systems Laboratories, NTT Corporation
3-9-11, Midori-cho, Musashino-shi, Tokyo, Japan 180-8585

2. 既存技術と本稿の課題

以前著者は，環境適応ソフトウェア処理として，図 1 の全体像を提案した。処理として，Step1-6 は，コード分析し，配置環境に応じた変換，リソース量調整，配置場所調整，検証の一連を行い，運用を開始する，その流れを，GPU や FPGA 等のアクセラレータ，IoT 機器等の少リソース端末の両方で検証してきた。更に，分析できるアプリケーションを増やすため，ユーザ処理を追加変更する際，アプリケーションを関連処理に基づいて分割して，変更を局所化する，自動分割方式を提案している。

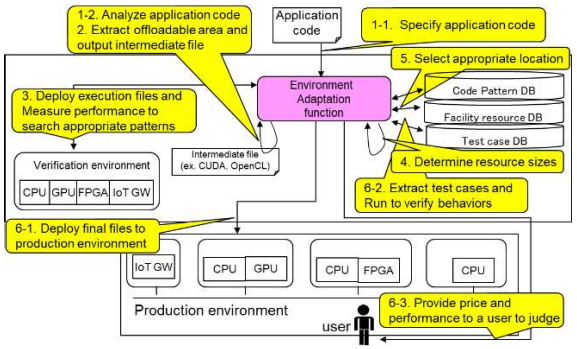

Fig. 1.　Processing flow of environment-adaptive software.

本稿の課題を整理する。著者は環境適応ソフトウェアのコンセプトを以前提案してきた。しかし，汎用プログラムの自動分割はこれまで，C 言語の計算系アプリケーションの検証だけであり，多くのユーザが使うと思われる IoT 等の新たなサービスは検証されていなく，有効性が示されていなかった。そこで，本稿は，IoT サービスでの自動分割を狙いとし，IoT ならではの考慮事項を踏まえて自動方式を微変更して，Azure IoT PF で使われる IoT サービスでも分析できるように実装し，検証する。

3. 汎用的プログラム自動分割の IoT 適用

3.1 IoT 向け自動分割方式

プログラムに，ユーザが行いたい独自処理を追加変更することを考える。行数があるアプリケーションは，変更影響が広範囲に及ぶため，関連する処理で分割して，変更を局所化することで，追加変更を容易化する。

GPU 自動オフロードの際，GPU で計算処理可能かは静的分析で分かるが，GPU 処理した際の性能は実際に測定しないと分からないのが通常であり，静的分析と動的分析を組合わせて，オフロード部を自動探索していた。動的分析は，個々のユーザ毎に異なる対応をするために重要な要素であった。そこで，IoT サービスの自動分割でも，個々のユーザに対応するために，静的分析と動的分析を組合わせた方式をとる。IoT サービスでは，IoT データの集約が必要となるため，特に分析する際に，通信が行われる処理の場合は，必ず別グループに分割する。

提案方式の動作は以下のようになる。

IoT サービスのプログラムコード群を静的分析，動的分析し，呼出関係がある情報をもとに分割を行う。

ファイル A に関数 a1,a2,a3,,,ファイル B に関数 b1,b2,b3,,,が定義されているとする。

静的分析では，Main 関数相当から呼ばれた先の関数 a1 が b2 を呼んでいる場合に，AB が一回と加算する。それを全ファイルに対してカウントする。その結果，AB：20，AC：30，BA：10 等が分析結果になる。

動的分析では，サンプル試験を一定数実施した結果，Main 関数から呼ばれた先の関数 a1 が b2 を 10 回呼んでいる場合に，AB が 10 回と加算する。それをサンプル試験実施分だけカウントする。その結果，AB：10，AC：20 等が分析結果になる。

動的分析で 1 回呼出しがあるファイル群は同グループ。

静的分析で 3 回呼出しがあるファイル群は同グループ。

静的分析で 2 回以下呼出しファイル群は残同グループ。

静的分析で通信が行われるファイル群は別グループ。

動的分析でユーザ指定サンプルテストケースで呼び出し関係があるファイル群は同じグループとなる。また，静的分析で全ファイルを分析した際に一定度の回数である 3 回以上呼び出し関係があるファイル群も同じグループになる。Stand Alone 的で他関数とあまり関わりがない関数ファイルは残りのグループになる。また，IoT データ集約に関わる通信がある場合は別グループになる。ここで，動的分析でユーザが使う指定ある際や，静的分析で呼び出し多い場合は関連強いため同じグループにする。また，IoT 向けとして，IoT GW でセンサデータをクラウド送信等は多いため，データ集約は別グループにする。

3.2 実装

対象は C 言語と Python のアプリケーションとし，C 言語解析には Clang(27)，Python 解析には ast(28)を用いる。アプリケーション解析実装は，Python 3 で行う。実装は，入力として，ソースコードファイル群と，サンプルテストケース情報を受け，出力として，静的分析，動的分析の場合のグループ情報を出力する。グループ情報は，分割されたグループとそれに所属するファイル群からなる。また，サンプルテストケース実行回数は 10 回とした。10 回で動的分析に十分なログを取得できるためである。

4. 評価

4.1 評価対象と評価手法

評価対象は，IoT データを環境センサから集め，IoT GW が一括して IoT PF に送信し，IoT PF で結果を Web 表示する基本的な IoT サービスとする。

センサ：Omron 2JCIE-BU01 から 1 分毎に環境情報の温度，湿度等 10 種類の IoT センシング情報を収集。

IoT GW：Armadillo-IoT G3L(29)。

IoT PF：Azure IoT。データ処理は，Azure IoT Hub REST API を介して行われ，結果等は Azure の VM に渡される。

自動分割された後，ユーザが追加するプログラム：温度，湿度の平均値を計算して出力。

IoT サービス分割状況を確認し，元々のアプリケーションの全行数と，分割された行数をカウントし比較する。

4.2 結果と考察

表 1 は，温度表示アプリケーションを提案方式で分割した際の，分割グループ数と全体行数と新機能をアプリケーションに追加する先のグループの行数を示している。

分割グループでは，Omron センサからデータ収集する部分と，センサデータを蓄積して表示する部分は分割されており，IoT GW へと IoT PF へで，分離して機能配置できる。温度表示アプリケーションは 2 つに分割され，全体 480 行に対して，新機能追加グループは 300 行となっている。自動分割がされ，ユーザが温度や湿度の平均値計算等の独自処理を追加した際の確認する範囲が小さくなり，機能追加変更が容易になっていることがわかる。

著者の環境適応ソフトウェアでのこれまでの IoT 環境適応は，事前に基本となるサービスを事業者が準備していることが前提だった。今回，自動分割技術を IoT サービスに適用できるようにすることで，基本サービス等な

Table 1. Example of division using the proposed method.

Application	Divided group #	total code #	target group code #
Show temperature	2	480	300

い IoT サービスにユーザが行いたい機能追加をできるようにした。自動分割では，IoT GW と IoT PF で適切に分割して配置できるため，適材適所利用もできると考える。

コストを考察する。機能追加することを考えた際に，温度表示アプリケーションを今回サンプルに利用した。機能追加時確認行数が 2/3 になった。影響確認稼働が低くなり，改造コストを抑えることができると考える。今回小さかったが，大きなアプリケーションで削減確認行数が 1,000 行超える際は，人・日単位で稼働削減できる。

今回，ユーザ毎の対応を行うために，従来も良く使われていた静的分析に，動的分析を組合わせる方式を行った。分析自体は，関数呼び出し関係を用いており，それで十分分割できることを確認したが，同データへの書き込み，DB やファイルのアクセスログ，連続実行等，更なる情報により，より詳細な分析が可能となると考える。

5. まとめ

本稿では，IoT サービスにも自動分割方式を適用することで，ユーザが独自変更を容易化できるようにした。

提案方式では，静的状態と動的状態の両分析から，呼び出し関係が無く関連が無いファイル群同士の分割及び通信があるファイル群同士の分割を自動提案する。提案方式により，分割ファイル群の行数が，元のファイル群の行数に比べて小さくなり，ユーザが行いたい独自処理を追加する際の変更影響確認範囲が小さくなる。サンプルアプリケーションとして，温度表示する IoT サービスに提案方式を適用し，IoT GW と IoT PF に分割され適材適所利用ができ，全体 480 行に対して新機能追加グループは 300 行となり，機能追加変更の容易化を確認した。

文　献

(1) P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines," Technical report of GE, 2012.

(2) H. Noguchi, et al., "Autonomous Device Identification Architecture for Internet of Things," IEEE WF-IoT 2018, 2018.

(3) Y. Yamato, et al., "Proposal of Real Time Predictive Maintenance Platform with 3D Printer for Business Vehicles," IJIEE, Vol.6, 2016.

(4) Y. Yamato, et al., "Method of Service Template Generation on a Service Coordination Framework," UCS 2004, 2004.

(5) H. Noguchi, et al., "Device Identification Based on Communication Analysis for the Internet of Things," IEEE Access, 2019.

(6) H. Noguchi, et al., "Distributed Search Architecture for Object Tracking in the Internet of Things," IEEE Access, 2018.

(7) oneM2M, https://onem2m.org/technical/published-specifications/

(8) MQTT, https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

(9) Azure IoT Hub, https://learn.microsoft.com/ja-jp/azure/iot-hub/

(10) O. Sefraoui, et al., "OpenStack: toward an open-source solution for cloud computing," IJCA, Vol.55, 2012.

(11) Y. Yamato, "Proposal of Optimum Application Deployment Technology for Heterogeneous IaaS Cloud," WCSE 2016, 2016.

(12) Y. Yamato, et al., "Fast Restoration Method of Virtual Resources on OpenStack," IEEE CCNC 2015, 2015.

(13) Y. Yamato, "Proposal of Automatic GPU Offloading Method from Various Language Applications," ICIET 2021, 2021.

(14) Y. Yamato, "Automatic Verification for Plural Virtual Machines Patches," ICUFN 2015, 2015.

(15) A. Putnam, et al., "A reconfigurable fabric for accelerating large-scale datacenter services," ISCA'14, 2014.

(16) J. Sanders and E. Kandrot, "CUDA by example : an introduction to general-purpose GPU programming," Addison-Wesley, 2010.

(17) J. E. Stone, et al., "OpenCL: A parallel programming standard for heterogeneous computing systems," Comput Sci Eng., Vol.12, 2010.

(18) J. Fung and M. Steve, "Computer vision signal processing on graphics processing units," ICASSP '04, Vol.5, 2004.

(19) T. Sterling, et al., "High performance computing : modern systems and practices," Cambridge, MA, ISBN 9780124202153, 2018.

(20) S. Wienke, et al., "OpenACC-first experiences with real-world applications," Euro-Par 2012 Parallel Processing, 2012.

(21) M. Wolfe, "Implementing the PGI accelerator model,"GPGPU10, 2010.

(22) Y. Yamato, "Study and Evaluation of Automatic GPU Offloading Method from Various Language Applications," Int J Parallel Emergent Distrib Syst., Taylor and Francis., Taylor and Francis, 2021.

(23) Y. Yamato, "Automatic Offloading Method of Loop Statements of Software to FPGA," Int J Parallel Emergent Distrib Syst., Taylor and Francis, 2021.

(24) Y. Yamato, "Improvement Proposal of Automatic GPU Offloading Technology," ICIET 2020, 2020.

(25) Y. Yamato, "Proposal of Automatic Offloading for Function Blocks of Applications," ICIAE 2020, 2020.

(26) Y. Yamato, "Study and Evaluation of Improved Automatic GPU Offloading Method," Int J Parallel Emergent Distrib Syst., Taylor and Francis, 2021.

(27) Clang website, http://llvm.org/

(28) ast web site, https://docs.python.org/3/library/ast.html

(29) Armadillo web site, https://armadillo.atmark-techno.com/about/iot-gw

山登　庸次

2000 年東京大学理学部卒。2009 年同大大学院総合文化研究科にて博士（学術）。2002 年日本電信電話（株）入社。同社にて，クラウドコンピューティング，IoT，適応ソフトウェアの研究開発に従事。現在，同社研究所特別研究員。IEEE シニア会員。