テクニカルレポート
CDSL Technical Report

転送先のリソースに配慮した自動ログ複製による小規模IoT向け分散ログ管理

小山 智之[1,a)]　串田 高幸[1]

概要：Linux OS が搭載された IoT 機器はサーバと同様にログを生成する．こうしたログの管理や検索のために分散管理手法が提案されてきた．しかし，これらは IoT 機器での利用を前提としていないため個々のノードの可用性が考慮されていないためログの同期に課題が生じている．本研究では，分散型アーキテクチャを採用し，個別ノードの可用性が低い IoT 機器に着目した提案を行った．提案の特徴は，ノードのグループ化とログのレプリカ動的配置，PULL 型アーキテクチャである．実験では，ログの複製時間とノードごとのハードウェア資源利用率を測定し評価を行う．

1. はじめに

1.1 背景

IT システムにおいてログは様々なソフトウェアから，様々な用途で出力される．例えば，Linux や UNIX をはじめとする OS では，ハードウェアやネットワーク，カーネルの動作に伴いログが生成される．こうしたログが出力される場面の例には，OS 起動時やユーザログイン時，ハードウェアの接続/切断時がある．Web サーバを提供するミドルウェア Nginx は，受け取った HTTP リクエストをアクセスログやエラーログとして記録する．WordPress をはじめとするアプリケーションは内部でのエラーや機能の動作をログとして記録する．Linux の場合，一連のログは一般にテキストファイルとして出力され，特定のパス /var/log/ へ配置される．こうしたログのユースケースは，次に大別される [1].

(1) **検知と調査の両面を兼ね備えるセキュリティ:** 攻撃やマルウェア感染，データ盗難をはじめとするセキュリティ上の問題を検知して対応すること
(2) **コンプライアンスや規制 (組織外)，規定 (組織内):** 多様な法律や命令，フレームワークや現地の企業政策を満たすこと
(3) **システムとネットワークのトラブルシューティングと通常のオペレーション:** システムの問題を調査したり，システムやアプリケーションの可用性を監視したりすること

一般にソフトウェアから出力されるログは，開発時の動作検証やエラーの原因特定に利用されるため (3) を満たす．(1) および (2) は必要とされる場面や実装されているソフトウェアが限定的である．例えば Nginx や WordPress にある標準機能では，(1) および (2) はサポートされていないが (3) はサポートされている．これらを踏まえ，本研究では一般的なログである (3) を対象とする．

ログはシステムで障害が発生した場面で，システムの動作を追跡するために欠かせない．Web サービスでは，複数のアプリケーションサーバやデータベースサーバ，キャッシュサーバが互いに連携して動作している．これらのうち，1 つのサーバに障害が発生しただけでも，ドミノ倒しのようにシステム全体へ影響が波及する．特に規模と複雑性が増加するにつれトラブルシューティングは困難になる [2]．外部からはこうした状況が，システム全体の障害としか判断できない．一方，ログは外部からの表面的な情報に比べ，より多くの客観的で詳細な情報が得られる．そのため，ログはシステムにおける障害対応に必要不可欠とされている．

ログを生成する IoT 機器: ログを生成する対象には，サーバやネットワーク機器だけでなく IoT 機器がある．IoT の普及にともない，従来はインターネットに接続されなかった機器がインターネットへ接続されている．具体的には，電力プラントの遠隔監視や冷蔵庫をはじめとする家電製品や鍵，電力や水道のメーターがあげられる [3–5]．こうした機器の中には Linux OS を搭載したものがありサーバと同様のログを出力する．Raspberry Pi に代表されるシングルボードコンピュータは，Linux OS が動作するマイクロコンピュータとして知られている．ログは IoT 機器が

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0117123

増えるにつれサーバと同様に統合管理が必要となる．しかし，IoT 機器はサーバとは異なる制約や条件があり，サーバのログ管理手法は適用できない．

ノードの物理配置: サーバと IoT 機器の配置を比較する．サーバは，オンプレミスの場合は図 1 のとおりデータセンタに集約され配置される．また，パブリッククラウドの場合もリージョンとよばれる地理的なまとまりで配置される．サーバの台数が増えるにつれ，データセンタへのサーバの集中配置が促進される．データセンタに配置されたサーバは，常に電源とネットワークが供給され安定した環境で動作する．特にネットワークは，個々のサーバが有線で接続されるため無線の場合に比べ安定している．

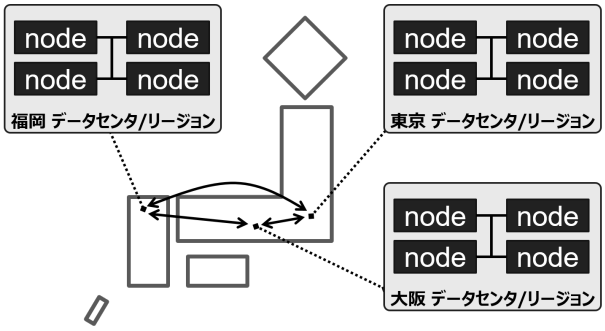

図 1 データセンタにおけるサーバの配置

IoT 機器は，サーバに比べ置く場所を問われないため図 2 のとおり地理的に分散して配置される．IoT 機器は増えるにつれ，個々の機器の分散が促進される．それぞれの機器は，配置場所によって電源やネットワークの品質にばらつきがある．ネットワークでは，有線に加え無線が利用されるため周囲の環境がネットワーク品質に影響を及ぼす．

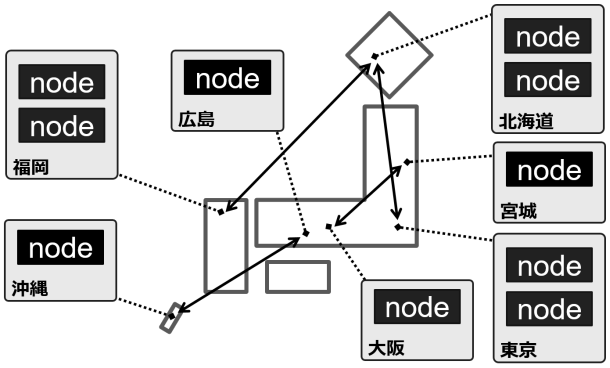

図 2 IoT 機器の配置

1.2 課題

ログの管理手法の 1 つに分散型手法がある．これは，ログを複数のノードに分散して配置し，複数台でシステム全体のログを管理するものである．

これまでの分散型のログ管理手法はサーバを前提として設計されている．一般にサーバはデータセンタに密集して配置され，有線によりネットワークへ接続される (図 1．また，データセンタでは冗長化により停電時も継続的に安定した電源が供給される．したがって，サーバは停止時間なく継続的に動作する．

IoT 機器は，屋内外を問わず 1 台から設置できる (図 2)．また，その特性によりサーバと比較すると 1 拠点あたりの IoT 機器の台数は少なく，それぞれが地理的に離れている．また，設置場所はデータセンタではないため，電源は冗長化されず安定した電源は得られない．したがって，IoT 機器は地理的に分散した配置と不安定な電源やネットワークにより，サーバと同様のログ分散管理を適用できない．地理的に分散した配置は，ノード間におけるネットワーク遅延の原因となる．

MySQL をはじめとするデータベースクラスタでは，レプリケーションの遅延はクラスタに所属するノード間でのデータ不整合を引き起こす．これはログにおいて，ログの欠落や不整合を引き起こす．不安定な電源は，それぞれの IoT 機器におけるログの取得と検索の際に可用性を低下させる．例えばログを検索したい場合，特定の IoT 機器の電源が失われていると検索したい機器のログを取得できない．

IoT 機器におけるネットワークの遅延と不安定な電源に伴う課題は，障害が発生した場合の原因特定を妨げる．そのため，こうした可用性を低下させる課題の解決は IoT 機器のログを分散管理する場合に必要不可欠である．

2. 関連研究

関連研究では，分散したロボットのログを収集するため ROS(Robot Operating System) フレームワーク ROSBAG をカスタマイズすることでシングルノードでは 1 台に集中していたネットワーク読み取りとローカルディスクへの書き込みを 2 台に分散させた [6]．しかし，個別ロボットの可用性が考慮されていないため，個別ノードで障害が発生した場合にそれまでのログを入手できない．

ログ管理サーバ間のログを複製する手法では，データベースのレプリケーションによりログの分散管理を行っている．この取り組みでは，ログ管理サーバが 4 台で固定されているため，ノード数が増加した場合に性能面でオーバーヘッドが発生しスケールできない．

分散したハニーポットを対象とした研究では，分散したハニーポットに集積されたログを管理サーバに集約することで，攻撃の可視化と検出を行った [7]．この取り組みでは，ログを収集するノードが増加した時の管理サーバのオーバヘッドが考慮されておらずボトルネックが発生する．

Hadoop と Spark によるログ分析環境を提案した研究 [8] では，バッチ処理により収集した大規模なログを分散させ処理を行なっている．Hadoop はクラスタを構成する個別ノードに高いハードウェア性能（CPU: 2 コア, RAM: 4GB) を必要とするため，ハードウェア性能の低い IoT 機器では使用できない．

ISP(Internet Service Provider) における利用者のアク

セスログを使った研究では，ログを DFS(Distributed File System) で収集し処理している [9]．この取り組みも研究 [8] と同様に高いハードウェア性能が必要なため IoT には適用できない．

3. 提案

課題である IoT 機器から生成されるログの可用性を解決するため，新たにログ分散管理手法を提案する．提案の特徴には，ログレプリカの動的配置と PULL 型アーキテクチャがある．まず，個別ノードで電源の障害が発生した場合でもログを取り出せるために，ログレプリカの動的配置を提案する．次に，個々のノードへのログ転送で発生するコストを抑えネットワーク遅延を削減するために，PULL 型アーキテクチャを提案する．上記に加えて，実装にあたりノード数の増加に伴うノード把握にコスト増加を低減するため，ノードグループを提案する．

3.1 ログレプリカの動的配置

ログレプリカ (複製) の動的な配置を提案する．個別ノードの可用性 (稼働率) をレプリカ配置に利用する．レプリカを配置しない場合，ノードの電源が失われるとそのログにアクセスができず検索や取得ができない．ノードの稼働率をもとにログを複製することで，仮にノードの電源が失われた場合でもログのレプリカによりログの検索や取得ができる．

ログの複製 (Replica) の配置を図 3 にしめす．このグループは，masterA と node1 から node4 までの合計 5 台で構成されている．ここで，ノードあたりに配置するレプリカ数 R とグループ i におけるノード数 N_i とする．このとき，レプリカの総数は $R \cdot N_i$ であらわされる．各ノード自身のログを含めるとレプリカの総数は $(R+1) * N$ になる．

レプリカの配置はノードごとに割り当てる ID(以下，ノード ID) をもとに決定する．ノード ID は時刻と乱数をもとに生成する一意な識別子である．各ノードは自身のノード ID よりも大きな ID をもつノードのレプリカをもつ．例えば，図 3 では $R = 2$ より，ノード 1 はノード 2 とノード 3 の 2 台分のレプリカを保持する．ノード 4 では，自身のノード ID より大きな ID をもつノードがいないため，グループ内でノード ID が小さなノードから順に複製先を選ぶ．この場合はノード masterA とノード 1 が選ばれる．

3.2 PULL 型ログ転送

ログ収集のアーキテクチャは大別すると PUSH 型と PULL 型がある．PUSH 型を利用しているソフトウェアには Mackerel や Datadog，PULL 型を利用しているソフトウェアには Prometheus や Zabbix がある．本提案では個々の IoT 機器へのログ受信に伴う負荷による可用性低下を防ぐため，PULL 型アーキテクチャを採用する．

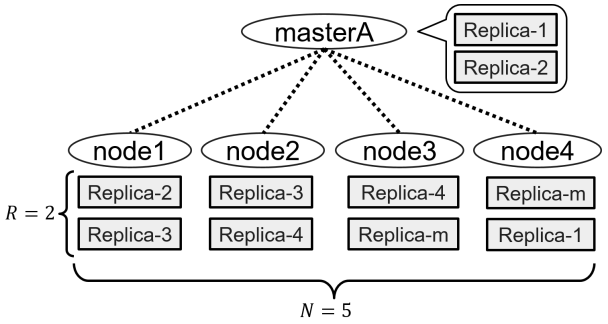

図 3 レプリカの配置

PUSH 型: 図 4 の PUSH 型では，ログサーバ (灰色) はログを送信するノード (白色) からのログを待ち受ける．ログを送信するノードは任意のタイミングでログサーバへログを送信する．このタイミングは，リアルタイムや一定時間ごと，一定件数ごとが使用される．そのため，ログサーバはログの受信タイミングを制御できず，ログを送信するノードが増えるにつれ負荷が上昇する．障害をはじめとするイベント発生時には，ログが通常よりも多く出力され DDoS 攻撃と同様の状況が生じる．

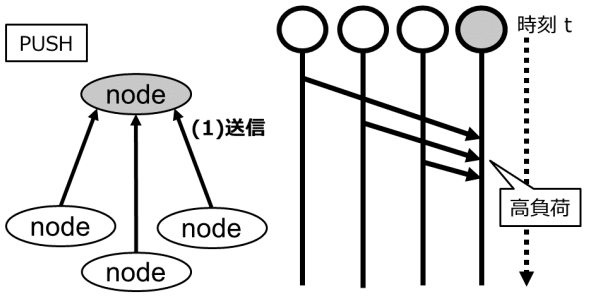

図 4 PUSH 型アーキテクチャ

PULL 型: 図 5 の PULL 型では，ログサーバ (灰色) はログ送信ノード (白色) へログを取得するために (1) 要求を送信する．要求を待ち受けているログ送信ノードは，ログサーバからの要求を受信すると対応する (2) 応答をログサーバへ返す．このアーキテクチャでは，ログサーバがリソースの空き状況に合わせてノードからログを取得でき，PUSH 型に比べログサーバへの負荷が少ない．これにより課題となるネットワーク遅延による可用性の低下を防げる．

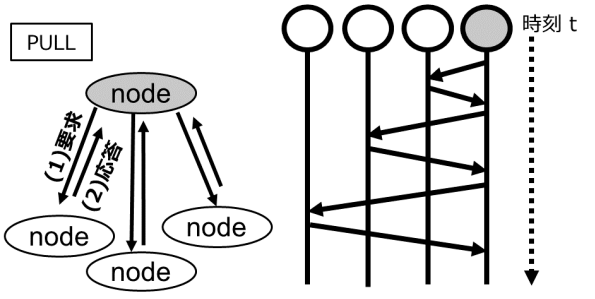

図 5 PULL 型アーキテクチャ

3.3 ノードのグループ化

図 6 は，ノード同士の接続によるグループ作成をあらわす．このグループに属するノード同士でログの複製を行う．図 6 の場合，Group-A には masterA と node1，node2 の 3 台が属している．別のノード Group-B には masterB と node8, node9 の 3 台が属している．

グループ内の 1 ノードにはマスターを割り当て，マスターは IP アドレスを含むグループに属するノードの一覧を保持する．マスターノード (図 6: masterA, masterB) は，ルートノード (図 6: root) に自身の IP アドレスを登録する．ルートノードはマスターノードの IP アドレス一覧を管理する．

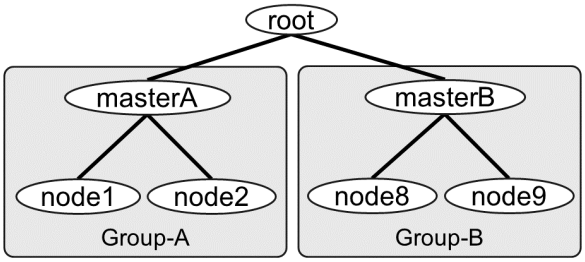

図 6 ノード同士の接続

3.4 既存グループへのノード追加

以下では，構築済みのグループに新たにノードが追加される場合のアルゴリズムを図 7 に示す．

(1) 新たにグループへ参加する node は，root ノードへ要求を送る．
(2) 要求を受け取った root ノードは，ノード数の上限に達していないグループのリストを node へ返す．
(3) node はグループリストをもとに各ノードへ疎通確認を行い，応答の最も早いノード (masterA) へグループ参加要求を送る．
(4) グループ参加要求を受け取ったノード (masterA) は，node を参加させられる場合は許可応答を返し，参加させられない場合は拒否応答を返す．
(5) node は疎通確認の結果から応答の早いノードを順番に手順 3 から手順 4 を許可応答が返されるまで繰り返す．

4. 実装と評価

4.1 実装

DNS による分散ノード管理

ノードグループの管理には DNS を使用する．root ノードおよび master ノードは DNS における権威サーバの役割をもたせる．図 8 は DNS によるノード管理構造を表す．root ノードでは master ノードの一覧情報を管理する

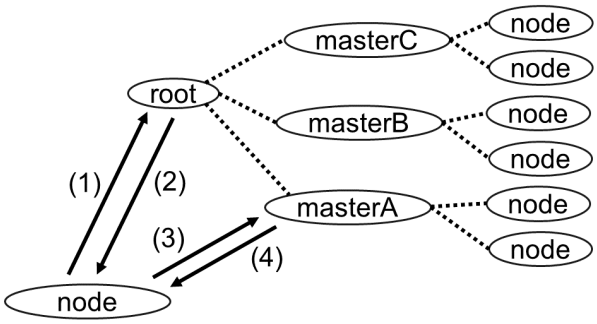

図 7 グループの形成手順

が，master ノード配下にあるノードの情報は管理しない．master ノードでは，それぞれのノードグループごとに所属する node の一覧を管理する．

masterA と node1 により構成されるグループ 1 がある場合を考える．この場合，まず node1 へのアクセスには root ノードが保持するグループ 1 のマスターノード masterA の IP アドレスを root ノードへ問い合わせる．root ノードは，masterA の IP アドレスを NS レコードとして保持しているため，DNS クライアントは次に masterA の IP アドレスへ問い合わせを行う．masterA は問い合わせに対して node1 の IP アドレスを返す．これにより node1 の IP アドレスがわかり node1 へアクセスが可能になる．

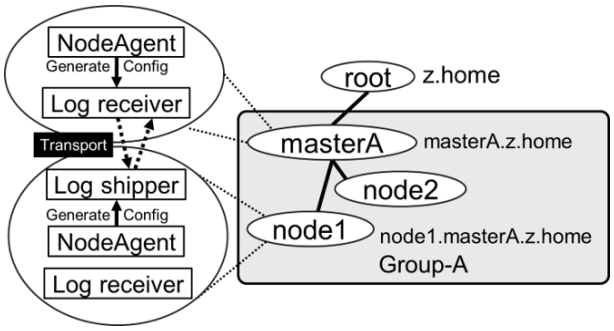

図 8 DNS によるノード管理

Node Agent

個々の IoT 機器にインストールする Node Agent を実装する．これは，DNS で取得した master ノード一覧から最適なノードを選び出し，参加要求を送信する．また，許可応答または拒否応答に応じて参加するグループを繰り返し探索する．

ログの複製

ログの複製には，Log Shipper の OSS である Fluentd を使用する．Fluentd では，ログの転送とログの受信の 2 つを行う．転送では，Node Agent が保持する情報をもとにログの複製先を含む設定ファイルを生成する．Fluentd はこの設定をもとに複製先で動作する Fluentd へログをリアルタイムで転送する．受信では，ポートを開放し転送されてきたログを受信し，ローカルのファイルに書き込む．

4.2 実験環境

研究室内の仮想化基盤上に表 1 の仮想マシンを作成し検証を行う．仮想マシンには Ubuntu をインストールする．root-node および master-node では DNS サーバを PowerDNS で構築する．master-node および node には新たに開発する Node Agent とログ送受信のための Fluentd をインストールする．

表 1 仮想マシンのスペック

VM 名	台数	CPU	RAM	Storage
node	4	1 コア	1GB	50GB
master-node	2	1 コア	1GB	50GB
root-node	1	1 コア	1GB	50GB

4.3 評価

分散システムは 1 時間あたり 120 - 200 件のログを生成することから，個別ノードから生成されたログを想定した 200 行のテキストファイルを作成する [10]．このテキストファイルに含まれるログは，研究室内の Elasticsearch に蓄積した Linux OS の仮想マシン 20 台から収集した syslog を使用する．syslog でやり取りされるメッセージのフォーマット例を表 2 に示す．生成したテキストファイルは Node Agent で生成した Fluentd 設定ファイルをもとに起動した Fluentd で別ノードへと転送する．

表 2 サンプル

omfwd: TCPSendBuf error -2027, destruct TCP Connection to elasticsearch-edge.a910.tak-cslab.org:5000 [v8.32.0 try http://www.rsyslog.com/e/2027]

ノードの可用性を評価するため，個別のノードの電源を落としたりネットワークを切断したりすることで可用性が維持されるかを確認する．

5. 議論

本研究では，個別ノードのスペックが同一である前提のもと提案を行った．しかし，実際の IoT では個々の機器のハードウェアスペックや内部で動作するソフトウェアも異なる．そのため，ログのレプリカの作成において，個別ノードのスペックや内部の状況を考慮する必要がある．この点は今後の研究課題として検討したい．

評価では syslog を使用したがログの種類によって 1 行あたりの長さや単位時間あたりの流量は異なる．そのため，ログの種類を変更して検証を行う必要がある．アプリケーションログには自作のアプリケーションや OSS で公開されているログジェネレータ，Python 製 Web アプリケーションを想定している．

現状ではログの複製タイミングがリアルタイムであるため，ログの量が増えた場合に転送先ノードが処理しきれず欠損や遅延が発生する可能性がある．これにはバッチ処理やバッファリング，再送制御を Fluentd へ設定することで改善されると考えられる．このパラメータは，場合によってはノード数に応じて適切な値を最適化により求める必要がある．

6. おわりに

本研究では，IoT 機器を対象に可用性に配慮したログの分散管理手法を提案した．サーバに比べネットワークや電源が不安定かつ，分散して配置されたノード間でレプリカを配置するため，ノードのグループ化およびログレプリカの動的配置，プル型アーキテクチャを提案した．本研究では，IoT 機器における可用性を考慮したログ管理手法を提案することにより，ログサーバを必要とせず安定的なログ管理を実現した．

参考文献

[1] Chuvakin, A.: The complete guide to log and event management, *White Paper* (2010).

[2] Wang, C., Kavulya, S. P., Tan, J., Hu, L., Kutare, M., Kasick, M., Schwan, K., Narasimhan, P. and Gandhi, R.: Performance Troubleshooting in Data Centers: An Annotated Bibliography, *SIGOPS Oper. Syst. Rev.* (2013).

[3] Floarea, A., V.Sg 但 rciu：Smart refrigerator: A next generation refrigerator connected to the IoT, *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–6 (2016).

[4] Pereira, R. I., Dupont, I. M., Carvalho, P. C., Sandro C.S.Juc 叩：IoT embedded linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant, *Measurement*, Vol. 114, pp. 286 – 297 (2018).

[5] Lloret, J., Tomas, J., Canovas, A. and Parra, L.: An Integrated IoT Architecture for Smart Metering, *IEEE Communications Magazine*, Vol. 54, No. 12, pp. 50–57 (2016).

[6] Koo, Y. and Kim, S.: Distributed Logging System for ROS-Based Systems, *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 1–3 (2019).

[7] Visoottiviseth, V., Jaralrungroj, U., Phoomrungraungsuk, E. and Kultanon, P.: Distributed Honeypot log management and visualization of attacker geographical distribution, *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 23–28 (2011).

[8] Lin, X., Wang, P. and Wu, B.: Log analysis in cloud computing environment with Hadoop and Spark, *2013 5th IEEE International Conference on Broadband Network Multimedia Technology*, pp. 273–276 (2013).

[9] Hossain, M. S., Kumar Bhowmik, P., Rahman, M. A., Uddin, M., Yousuf, M. A. and Abu Taher, K.: Develop a Model to Secure and Optimize Distributed File Systems for ISP Log Management, *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–6 (2019).

[10] Yuchao Chen, Weiming Wang and Ming Gao: Application of distributed safe log management in Small-Scale, High-Risk system, *2009 International Conference on Innovations in Information Technology (IIT)*, pp. 26–29 (2009).