IoT 向け Node-RED ベースの分散アプリ開発環境の実装と評価

仲道耕二 [1] 大木憲二 [1] 野村佳秀 [1] 浅沼稔 [2] 舞田正朋 [2]

概要：IoT フロントやエッジネットワークなど，機器が複数配置された分散環境で動作するアプリの開発・運用に際しては，機器の入れ替えやアプリ機能の更新などに伴って，アプリを修正し再度機器に配備する手間・コストが課題となる．そこで我々，機器が多数配備された分散環境で連携動作する IoT アプリの効率的な開発や配備を行うために，フローベースのアプリ開発環境である Node-RED[1]で定義したフローを入力モデルとして利用し，モデル変換技術を用いて配備環境に適した実装を自動生成する方式を考案し実装したので報告する．さらに本稿ではアプリ開発効率への影響度を評価するために，スマート工場で向けアプリを例に，通常の Node-RED を用いて分散機器毎の個別のアプリを開発した場合と，分散提案手法を利用してアプリを開発した場合のアプリ規模の比較を行った．

Implementation and Evaluation for Node-RED based Application Development Environment for Distributed IoT Systems

KOJI NAKAMICHI[1] KENJI OKI[1] YOSHIHIDE NOMURA[1]
MINORU ASANUMA[2] MASATOMO MAIDA[2]

1. はじめに

センサの多様化やクラウドプラットフォームの進展により，大量データの収集，分析，活用を行う IoT システムの実用化が進んでいる．またセンサデータを全てクラウドに転送せず，センサに近いエッジ装置で処理して制御機器にフィードバックを返すことでリアルタイム性を高めるエッジコンピューティングへの期待も高まっている．このような IoT フロントやエッジ NW など，機器が複数配置された分散環境で動作するアプリの開発・運用に際して，機器の入れ替えやアプリ機能の更新などに伴ってアプリを修正し，再度機器に配備する手間やコストが課題となる．

そこで我々は上記課題を鑑み，機器が多数配備された分散環境で連携動作する IoT アプリの効率的な開発や配備を行うために，オープンソースのフローベースのアプリ開発環境である Node-RED[1]で定義したフローを入力モデルとして利用し，モデル変換技術を用いて配備環境に適した実装としてのアプリフローを自動生成する方式を考案し，これを Node-RED の拡張として分散アプリ開発環境を実装した．

また本環境を用いることによる，アプリ開発効率への影響度を評価するために，スマート工場で使用する製造状態撮影アプリを例に，通常の Node-RED を用いて分散機器毎の個別のアプリを開発した場合と，分散アプリ開発環境を利用してアプリを開発した場合のアプリ規模の比較を行った．

以降では，2 章において Node-RED ベース分散アプリ開発環境の機能の概要と特徴，および想定適応先について述べる．さらに 3 章において提案手法による効果評価について述べ，最後に 4 章において結論を述べる．

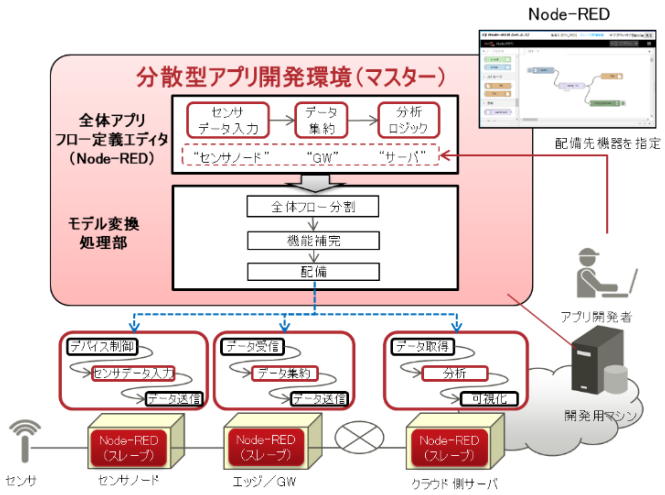

図 1 分散アプリ開発環境の構成概要

2. Node-RED ベース分散アプリ開発環境

2.1 Node-RED ベース分散アプリ開発環境の実装機能

実装した Node-RED ベースの分散アプリ開発環境の機能の特徴について述べる．

(1) Node-RED ベース分散アプリ開発環境の構成

1 (株)富士通研究所 2 (株)富士通テレコムネットワークス

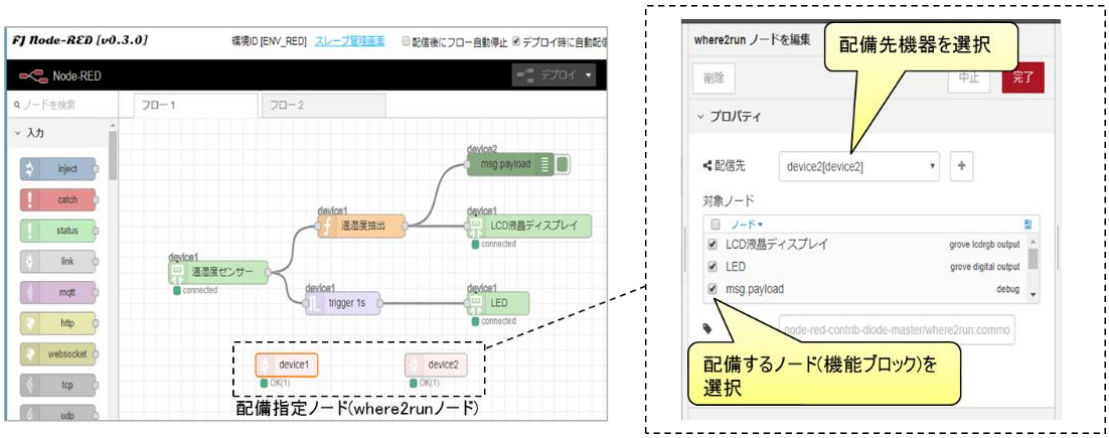

図 2　Node-RED によるアプリ定義画面と配備先／配備ノードの指定

実装した Node-RED ベース分散アプリ開発環境は開発環境（マスター）用ソフトウェアと実行環境（スレーブ）用のソフトウェアから構成の構成を図 1 に示す．マスターは Node-RED をベースに，定義したアプリフローに対してアプリ処理機能ブロックであるノードをどの配備先機器に配備するかを指定し，指定した機器に適した Node-RED フローに変換を行うモデル変換機能，ならびに配備先機器の状態を管理するスレーブ管理機能を持つ．一方，スレーブは Node-RED をベースに，マスター側から送付された変換済みの Node-RED フローの受信／起動処理を行う配信フロー制御機能が追加されたものである．

追加機能の実装に際し，サーバ側／クライアント側いずれもオリジナルの Node-RED 自身には手を加えないように，拡張機能を npm モジュールとして実装し npm インストールを行うことで拡張機能が追加できるように実装した．

(2) 配備先機器向け Node-RED フローの生成・配備

サーバ側の Node-RED で作成したフロー（全体フロー）に対して，フロー全体あるいは個別のノードをどの配備先機器で実行するのか指定するために，配備指定ノード（where2run ノード）を新たに追加した．ノード一覧から where2run ノードを選択して編集画面上に配置し，設定画面からどの配備先機器および配備ノードを選択することができる（図 2）．その際，配備機器は同時に複数選択することが出来る．これにより全体フローや一部のノードを同時に複数の配備先機器に配備指定することが可能となる．

マスター側の Node-RED 画面上でデプロイボタンを押下することで，全体フローはモデル変換処理により指定した配備先機器毎に全体フローを分割し，配備先機器間の通信のための MQTT ノードを自動で補完し，通信識別子としてトピック情報などの必要な設定を行い，生成した Node-RED フローを生成指定した配備先機器（スレーブ）へ配布する．

(3) モデル変換処理

本分散アプリ開発環境は,Node-RED で定義した全体フローに対してモデル駆動アーキテクチャ（MDA）に基づいたモデル変換処理を行う．Node-RED で定義した全体フロー（JSON データ）を入力モデルとし，まずこの JSON データを EMF(Eclipse Modeling Framework)形式へ一旦変換してから，変換ルールに基づいて配備先毎の Node-RED アプリを生成し各機器への配備までを行う．今回は変換ルールとして配備先機器間の通信処理および各配備先機器への配備処理を，変換処理記述言語の 1 つである ATL（Atlas Transfer Language）言語で定義した．また変換ルールは，運用条件などに応じて追加や変更も可能である．

(4) 配備先機器（スレーブ）管理機能

マスターとスレーブ間は websocket 通信により接続する．定期的に接続状態および変換後アプリフローの配備状態を監視する機能を持っており，管理画面を通じてスレーブの状態を把握することが可能である．

2.2 分散型アプリ開発環境の特徴と想定適用先

分散アプリ開発環境の特徴の 1 つは，定義した全体フローを元に複数の機器に分配して配備できることであり，複数の機器が物理的に広範囲に設置されている環境で，アプリ配備の効率化が期待できる．また，全体フローを元にアプリのロジック修正を行い，簡単に実環境に応じたアプリを生成して配備することができるため，機器の増設に応じたアプリの配備や，アプリの修正などが発生する環境において効率的なアプリ開発が行える．一方制約条件として，分散型アプリ開発環境は，マスター／スレーブともに Node-RED をベースにしているため, Node-RED が動作する機器を配置する必要がある．

このような特徴を鑑み，本開発環境の適用領域の 1 つとしてスマート工場を想定している．スマート工場では気温や湿度等の工場内環境の監視や，製造状態の監視，作業状

態の管理など様々な目的に応じて柔軟に計測機器を配置して効率的にアプリ開発を行う必要があるなど，本開発環境の特徴を生かせる条件が整っている．

3. スマート工場向け IoT アプリ開発に対する効果検証

3.1 効果検証の概要

分散アプリ開発環境を用いた場合の IoT アプリ開発への開発効率向上効果の検証を行った．ここでは定量的な効果検証として，工場向けの製造状態撮影アプリを例に，通常の Node-RED を用いて分散機器毎の個別のアプリを開発した場合と，分散アプリ開発環境を利用してアプリを開発した場合の開発工数の違いを比較するために，Node-RED フロー上のノード数によるアプリ規模の定量比較を行った．

3.2 製造状態撮影アプリの概要

今回，評価の為にスマート工場で使用する IoT アプリとして，製造状態撮影アプリの開発に対して分散アプリ開発環境を活用した．このアプリは工場の製品製造ライン上で製造される製品の基板上に複数個取り付けられる部品のロット番号やシリアル番号を，それぞれ複数のカメラで撮影して画像から文字を読み取り，その文字情報および画像ファイルをサーバに転送して保存するものである．本アプリで取得した文字情報や画像情報は，製造ラインの管理者が必要に応じて製造状態情報撮影＆保存アプリとは別の製造状態管理アプリ経由で参照し，製品の製造状態を追跡するために活用される．図 3 に製造状態撮影システムの構成概要を，図 4 に分散アプリ開発環境上で定義した製造状態撮影アプリの全体フローを示す．

製造状態撮影アプリの全体ロジックは，人による開始指示をトリガーとして，開始時刻設定やカメラへのパラメータ設定を行った後，カメラ撮影処理と撮影した画像からロット番号やシリアル番号を読み取ることで取得する．読み取った文字情報と画像ファイルをそれぞれサーバへ転送すると同時に GW 装置にもファイルとして保存する．

分散アプリ開発環境を用いた開発では、実際のシステム上の機器に対して必要なアプリを配備するために，全体フローに対して，配備指定ノード(where2run ノード)を用いて，全体フローの各ノードの配備先機器を指定する．カメラのパラメータ設定や撮影／OCR 処理を行う部分は、カメラが接続されたスレーブ装置に配備するように、1 つの配備指定ノードにおいて、複数のスレーブ装置を指定する．また取得した文字情報や画像情報をサーバに転送する処理などは、さらに別配備指定ノードを用いて GW 用スレーブ装置に配備するように設定する．これにより全体フローはモデル変換処理により，指定したスレーブ機器毎に分割され，各カメラ接続装置と GW 装置間の通信を行うための通信処理ノードが補完された Node-RED フローが生成され、各装置に配備される．

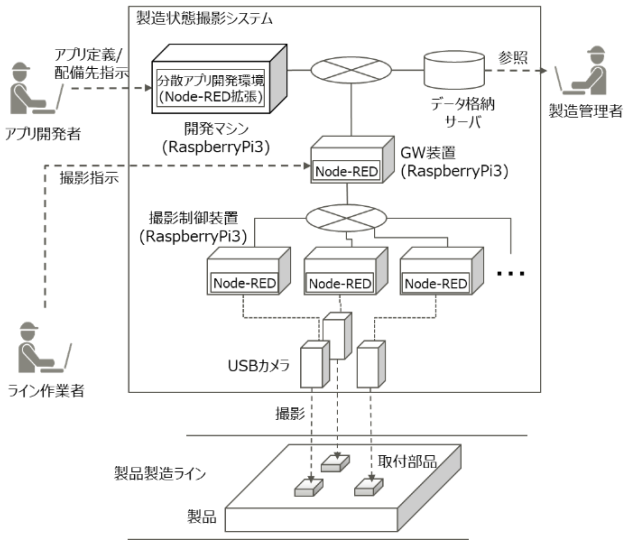

図 3　製造状態撮影システムの構成概要

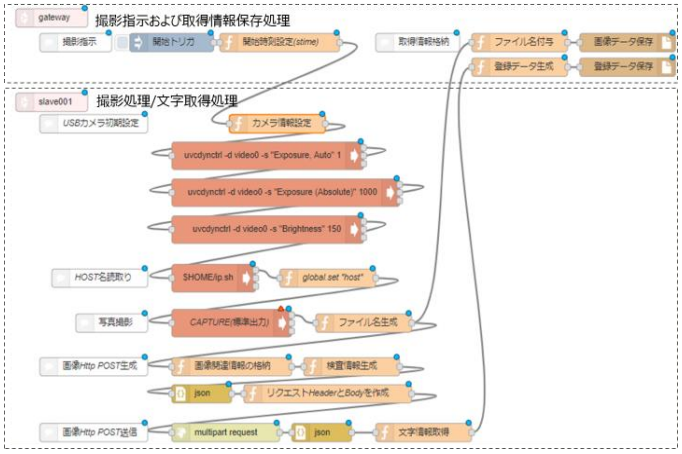

図 4　製造状態撮影アプリの全体フロー

3.3 アプリ規模の比較によるアプリ開発効率への影響評価効果検証の概要

分散アプリ開発環境によるアプリ開発効率への影響を評価するために，開発するアプリ規模の比較を行った．分散アプリ開発環境で定義した全体フローのノード数と，分散アプリを用いずに個別に製造状態撮影システム上の機器上で動作するアプリフローを開発して導入した場合の Node-RED フローのノード数の比較を行った．ただし，実際には個別機器で動作する Node-RED フローと分散アプリ開発環境によって個別機器にモデル変換によって配備される Node-RED フローは構造上同等であるため，変換後の配備フローのノード数を合計したものを個別機器で動作する Node-RED フローの合計ノード数として算出した．

図 5 に撮影制御装置（カメラを接続したスレーブ機器）の数に対する，製造状態撮影システム全体の Node-RED フローのノードの合計値の関係を示す．撮影制御装置数に依存してシステム全体の Node-RED アプリのノード数が増えることが分かる．つまりシステムの規模が増加するにつれて開発すべきアプリの規模が増加していることを意味している．

一方，分散型アプリ開発環境では，開発するアプリはマスター機器上の全体フローのみであり，実際のシステム上の撮影制御装置数によらず一定であることが分かる．個別の機器で動作するアプリフローは分散アプリ開発環境上で自動生成されるため,開発規模は一定かつ少ない量で済む．またシステムの規模が大きいほどより効率的な開発が可能となる．

このアプリ開発規模の削減効果の要因は，分散型アプリ開発環境の特徴であるモデル変換処理の 2 つの性質によるものである．まず，モデル変換処理において，機器間の通信処理部分はモデル変換処理で補完されるため，全体フロー上では定義する必要がない．そのため，最初から機器毎に Node-RED フローを定義する場合に比較して，通信に関するフロー処理が削減される．2 つ目の要因は，分散アプリ開発環境でフローやノードの配備先機器をしている配備指定ノードによって，同一のフローやノードを同時に複数の機器に配信指定できることに起因している．つまり多数の機器に同じフローやノードを配備したい場合でも全体フローとして 1 回定義すればよいため，個別に開発する場合に比較して開発規模は各段に削減できる．

またこれらの効果は配備機器が多くなればなるほど大きくなる．

4. 結論

本稿では、機器が多数配備された分散環境で連携動作する IoT アプリの効率的な開発や配備を行うことを目的に，定義した全体フローからモデル変換技術を用いて配備環境に適した実装を生成・配備する分散アプリ開発環境について提案した．

さらに，本環境を用いることによる，アプリ開発効率への影響度を評価するために，スマート工場で使用する製造状態撮影アプリを例に，通常の Node-RED を用いて分散機器毎の個別のアプリを開発した場合と，分散アプリ開発環境を利用してアプリを開発した場合のアプリ規模の比較を行った．その結果，分散アプリ開発環境を用いることで，通常の Node-RED で個別にアプリを開発する場合よりアプリ開発工数を削減できることが確認できた．また，その際削減されるのはモデル変換によって補完されるノード部分および複数の機器に配備される部分であるが，配備先機器数が多いほど削減効果が高くなることが分かった．

今回はアプリ開発効率への効果として，アプリ開発規模のみに着目して評価を行ったが，今後本提案方式の適用によるアプリ開発工程などへの影響評価を行う予定である．

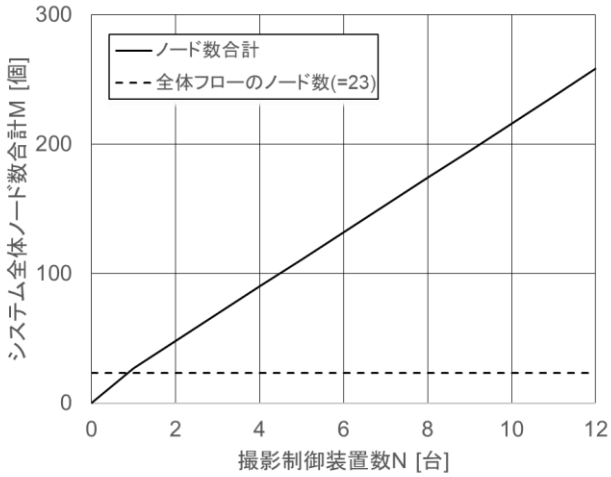

図 5　撮影制御装置数とシステム全体の Node-RED フローのノード数合計の関係

参考文献

[1] “Node-RED”, https://nodered.org/