Node-RED ベースの分散アプリ開発環境の実装と開発資産の再利用に関する考察

仲道 耕二[†1] 大木 憲二[†1] 野村 佳秀[†1]

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．我々は以前，IoT 向けフローベース開発環境である Node-RED に基づくモデル変換を用いて，配備環境に適した実装を生成可能なシステムについて提案した[1]．本稿では，検討モデルの実装機能とその適用効果，適用領域について述べるとともに，分散環境向けアプリ開発における開発資産の再利用性ついて検討を行った．

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．

我々は以前, IoT 向けフローベース開発環境である Node-RED[2]に基づき，モデル変換を用いて配備環境に適した実装を生成可能なシステムについて提案した[1]．抽象度の高い IoT アプリのモデルを M2T（Model To Text）と M2M（Model To Model）の 2 種類のモデル変換技術によって配備環境に適した実装を生成可能なシステムを構築することで，コアロジックを記述した Node-RED フローから変換によって様々な環境で動作検証を行いながら開発可能な環境を目指すものである．

今回，我々は上記提案システムの具体的な実装として Node-RED ベースの分散アプリ配備環境を開発した．オリジナルの Node-RED に変更を加えないようにする為に，提案したモデル変換処理を npm モジュールとして実装し，配備指示ノードから配備先機器を指定できるようにした．また開発環境と配備先の機器間の接続状態や変換後のアプリの配備状態を確認できるデバイス管理機能を持たせた．

本稿では，上記 Node-RED ベースの分散アプリ配備環境の実装状況とその適用効果について述べるとともに，分散環境におけるソフトウェア資産の再利用性について検討を行った．

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

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される（図 1 参照）．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定義した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は，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．

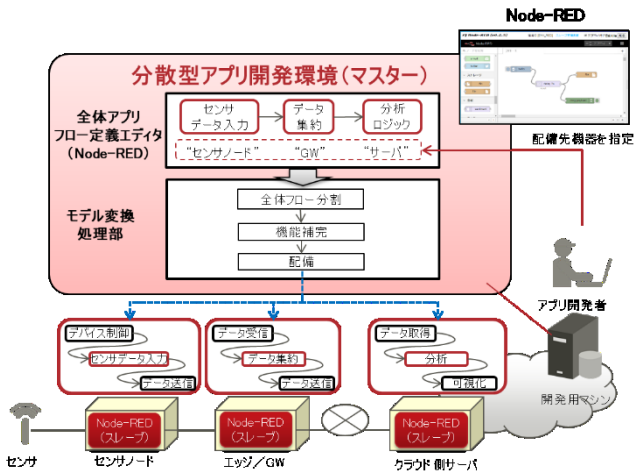

図 1 分散アプリ開発環境の構成概要

†1 (株)富士通研究所
Fujitsu Laboratories Ltd.

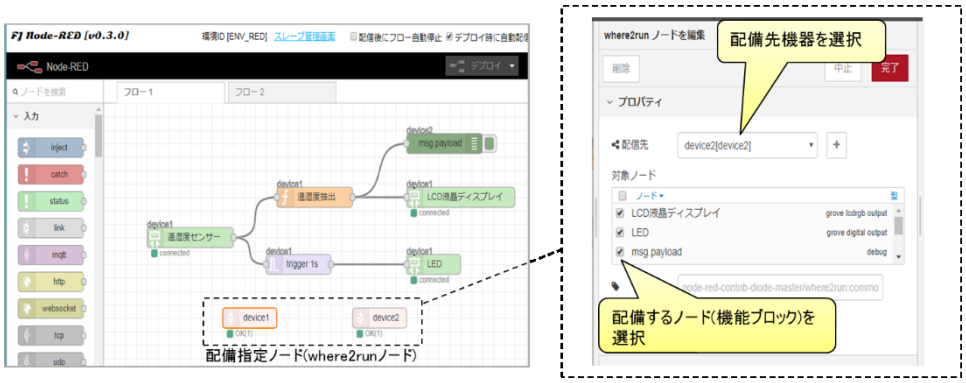

図 2 Node-RED によるアプリ定義画面と配備先／配備ノードの指定

(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 参照）．

マスターNode-RED 画面上でデプロイボタンを押下すると，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て必要な設定を行い，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) **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できる．

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．

このような特徴や制約を鑑み，現在本開発環境の適用領域として，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．今後，実際のスマート工場でのアプリ開発を本開発環境で行い，アプリ開発効率などに関する効果検証を行う予定である．

3. 再利用性の検討

本節では Node-RED で定義した全体アプリフローに着目し，その再利用性について検討する．

3.1 再利用性向上におけるモデル変換利用の意義

今回実装した分散アプリ開発環境では Node-RED で定義した全体フローを，指定した機器に対してモデル変換処理を行い適切に変換（分割して必要に応じて通信処理ノードを追加）し配備することで，既存の Node-RED では不可能であった分散環境への自動配備を実現した．モデル変換を用いることで，ロジック開発と実行環境向けアプリ開発を分離し，ロジック部分（全体フロー）の再利用性を高めることが可能になる．特にスマート工場のような，試作／実証／修正の繰り返しが必要な領域において，開発者は全体フローの流用によるロジック定義／修正に注力できるため開発効率の向上が期待できる．

3.2 再利用性向上に向けた課題

今回の実装におけるモデル変換機能では全体フローの分割と通信ノードの補完を 1 つの変換テンプレートにより実現している．しかし，今後変換により別のノードを追加したりするために変換テンプレートが増えることも考えられる．その場合，利用する全体フローに応じて適切な変換処理（テンプレート）を選択するケースが想定される．そのために変換テンプレートを利用者が選択したり，あるいは何等かの判断基準により自動で変換テンプレートを選択する機能が必要になると思われる．

4. おわりに

本稿では分散環境におけるアプリ開発のための実装機能について説明し，モデル変換を用いた Node-RED アプリフローの再利用性について検討を行った．今後は本開発環境の現場での適用し，効果の検証を行う予定である．

参考文献

[1]大木,仲道,野村,栗原,“モデル変換技術を用いた IoT アプリケーション開発と資産再利用性の検討”,WWS2018.
[2]“Node-RED”, https://nodered.org/