2W-09

コンパニオンロボット開発支援プラットフォームの開発

杉崎　勇斗†　　田胡　和哉‡

東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻†　　東京工科大学コンピュータサイエンス学部‡

1　はじめに

近年，IoT 分野において Pepper や COZMO などのロボットに API を通して連携した人工知能の開発が積極的に行われている．しかし，このような開発を行う上での大きな問題点としてハードウェアと人工知能の接続を開発者自身が行う必要があり開発時の大きな障害となっている．そのため，ハードウェア開発者と人工知能開発者の双方に利便性を提供できるプラットフォームの開発が必要となる．そのプラットフォームを利用することでハードウェア開発者は開発したハードウェアをプラットフォームに接続し人工知能アルゴリズムを選び・適用することで，実際にデバイス上で人工知能アルゴリズムを利用することができる．人工知能開発者はハードウェアとの接続の手間を省くことで AI プログラムからロボットを操作し簡単にハードウェアと連携した人工知能の開発が可能となる．本稿ではこのような開発を簡単に行うことのできるプラットフォーム MAGI を開発したので，それについて述べる．

2　MAGI

2.1　概要

MAGI （Middleware for Artificial General Intelligence）とは玩具・家電等の身の回りのデバイスをクラウド上の人工知能から制御するシステムを簡単に開発できることを目的としたミドルウェアである．人工知能開発に必要なアルゴリズム以外の部分についてはプラットフォーム側で提供することで開発時の負担をできるだけ減らし，人工知能開発に集中して取り組むことができる環境を提供する．

Proposal of development support system for companion robots

† Yuto SUGISAKI
Bionics, Computer and media science, Entrepreneurship pro-gram, Tokyo University of Technology Graduate School.
‡ Kazuya TAGO
School of Computer Science, Tokyo University of Technology.

MAGI はそれぞれの技術要素をプラグイン式に追加することで機能の追加を行う．それと併せて人工知能の開発を行うことで自分の開発すべき部分にのみ注力して開発することが可能となる．また，常に実機と接続して開発を行うことは現実的でないため，開発時は基本的にシミュレータを用いて開発を行うこととなる．そして開発が終了して実機に適用する段階で出力先をシミュレータからロボットに切り替えることで，実機で簡単にテストすることが可能となっている．

2.2　構成

MAGI は，IoT や人工知能の実現に必要となる技術要素を多数の計算機で簡単に動作させるための仕組みである．MAGI の構成図を 図 1に示す．

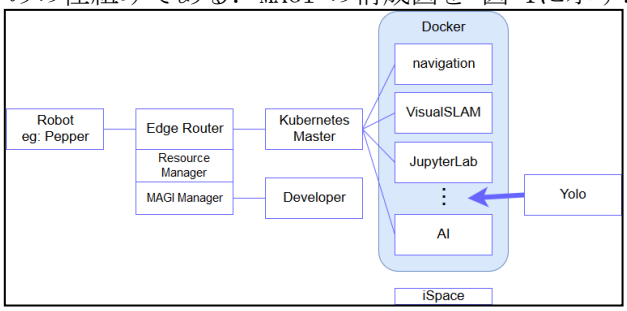

図 1　MAGI の構成図

開発者はまず Edge Router に Wi-Fi を通して接続しそのネットワーク内で提供されている MAGI Manager を通して管理 UI へアクセスする．その画面を通してロボット等の接続やクラウド上のリソースの管理などを行う．クラウド上では MAGI の根幹をなすプラグイン方式での環境の提供を行う．そのためのアプローチとしてコンテナ仮想化技術 Docker[1]の利用があげられる．Docker の特徴としてカーネル部分をベース OS と一部共有するためリソース消費量が少なく高速に起動することや，Dockerfile と呼ばれる Docker イメージの構成内容をまとめて記述するファイルを用いることで作成する Docker イメージを元に同一の環境を簡単に用意できるといった点があげられる．これを利用することで従来

必要であった環境構築の手間がなくなり，個々の環境を最低限にまとめた Docker イメージを利用することでそれぞれの環境を選択して利用することが可能となる．また，Docker イメージは Dockerhub 等を通して配布されており，他者が開発した Docker イメージを使うこともできるため，他者が開発した他のアルゴリズムと組み合わせて開発することも可能となる．

Docker コンテナそれぞれは独立した環境として起動するため環境に隔たりなくデータを共有する仕組みが必要となる．そのために iSpace という分散処理プラットフォームを開発し提供する．iSpace ではディクショナリ形式でデータの共有やストリーム・シグナルによる関数の実行などをサポートする．

3　実装

3.1　Edge Router

Edge Router は MAGI の環境の起点として利用され，この Edge Router からクラウド等のセットアップを自動で行う．また，MAGI Manager や Resource Manager の実行環境としても利用される．この Edge Router を介してクラウドとのネットワークを Wi-Fi にて提供する．加えて Docker Registry も提供し，利用者がオリジナル Docker イメージを利用したい際に提供する．

3.2　Resource Manager

Resource Manager は，プロジェクト・データボリューム・Docker のテンプレートや実行中のコンテナといった各種リソースを階層構造で管理している．これらのリソースは，Resource Manager のライブラリで提供されているコマンドや REST API を通して追加・編集・削除等を行うことができる．また，driver を作成することで様々なリソースに対応することが可能となっている．

3.3　MAGI Manager

MAGI Manager では Resource Manager の API をコールしその結果に応じてレンダリングを行うことから，サーバーサイドが不要で，フロントエンドの開発のみ行う．フロントエンドの開発は React を用いて行った．React とは Facebook 社が開発している UI に特化したライブラリで Virtual DOM と呼ばれるレンダリング機構を用いることで従来の DOM を書き換える方式と比較するとレンダリング速度が大幅に向上している．Virtual DOM では状態が変化した際に Virtual DOM の差分のみを再レンダリングする．これにより高速にレンダリングすることが可能となった．MAGI Manager では画面全体が更新されることが少なくそれぞれのリソースの状態のみを反映していけば良いため相性が良い．また，React では HTML の部品をコンポーネントして扱い画面を作成していく．これにより似たような画面を扱う際にコンポーネントを再利用することで，低コストで画面を作成することができる．

3.4　iSpace

iSpace は，Python の分散処理環境で，分散処理プログラムの実現において典型的に見られるプログラムを集約して実装したものである．iSpace ではディクショナリ形式でオブジェクトの共有を提供する．この名前空間にオブジェクトを登録し，すべてのノードから呼び出して利用することにより分散処理を実現する．

4　実装と運用

Edge Router は NUC と呼ばれる小型 PC に Ubuntu をインストールし Docker を用いて各種環境の構築・アクセスポイントの設定を行った．また，Resource Manager は Python を用いて開発を行った．管理画面となる MAGI Manager は React を用いて開発した．各 Manager は Docker イメージとして提供することで Edge Router 上で実行される．

これらはオンプレミス上で構築された Kubernetes[2]上で開発・運用されている．
現在はオンプレミス上での運用を行っているが，今後 Google が提供する Google Kubernetes Engine 上で実行し，東京工科大学のプロジェクト実習[3]の講義で当環境を利用した開発を行う．

5　おわりに

本稿では，人工知能開発に必要となる開発環境や各種アルゴリズム等をプラグイン式で提供し任意に選択して利用することができるプラットフォームを開発し提供した．当環境を利用することで様々なアルゴリズムをテストすることや人工知能の開発を行っていくことができると考えられる．

参考文献

[1] Docker Inc.：”Docker”
https://www.docker.com/ (2019/01/07)
[2] The Linux Foundation：”Kubernetes”
https://kubernetes.io/ (2019/01/08)
[3] 東京工科大学：”プロジェクト実習”
https://www.teu.ac.jp/gakubu/cs/project/index.html (2019/01/10)