Jawara: 組み込み機器上の分散処理のための Lisp 処理系

寺西 裕一†

Android OS 上に試作した Lisp 処理系 Jawara について報告する．本処理系はインタプリタであり，コア機能のメモリ上でのサイズが比較的小さく，組み込み機器に適している．また，組み込み機器上でのスクリプトの動作を遠隔から制御する HTTP インタフェースを設け，Web ブラウザ経由でのインタラクティブなデバッグとメンテナンスを可能とした．さらに，オーバレイプラットフォーム PIAX の ID トランスポートモジュールを組み込むことで，遠隔コード実行を含む分散処理をネットワーク接続の詳細を隠蔽した上で記述可能とした．

1. はじめに

ユビキタスコンピューティング環境の実現には，実世界に遍在するセンサ，アプライアンス，端末などの組み込み機器において，情報の収集や制御を相互に協調しつつ行なう分散処理が必要となる．従来こうした機器は低機能なものがほとんどであったが，近年，CPU や通信モジュールの高性能化により，高機能な OS がスマートフォン等で動作可能となり，組み込み機器間で分散処理を行なえる基盤が整いつつある．

しかし，組み込み機器においてプログラムコードを実行させたい場合，通常，組み込み機器を接続したデスクトップ PC 等に特別な SDK やデバッグツールをインストールした上で開発を行なう必要がある．また，プログラムコードを機器に送り込んだあと，機器自体を再起動，または，実行中のコードを中断しなければ，新たなコードを実行することができず，機能書き換えにはオーバヘッドが生じる．

また，組み込み機器を広域に配備することを考えた場合，多様なネットワーク実装に対応した接続を想定する必要がある．例えば，TCP, UDP 等による通常の IP 接続においても，IPv4, IPv6 のバージョンの違いを意識する必要がある．また，NAT の配下への接続や，IP 以外のネットワーク，例えば ZigBee によるネットワーク接続も想定される．こうしたネットワーク実装毎の違いを考慮しつつ End-to-End の接続性を確保し，分散処理を行なうプログラムコードを書く労力は大きい．

これらの課題の解決を目指し，組み込み機器向けの汎用 OS である Android 上で動作可能な Lisp スクリプト言語処理系 ‘Jawara’ を試作した．

2. Jawara

Jawara はオープンソースプロジェクトとして筆者らが開発を行なっている．公開の詳細については，サイト 1) を参照されたい．Jawara の Lisp 言語処理コア部分は JKLD [2] のコードをベースとしている．ただし，実用性の観点から，マルチスレッド対応，マクロ機能，エラー処理，末尾再帰処理の追加，スクリプトエンジン向け対応，Java ブリッジ機能追加など全面的に書き直されている．

Jawara は JKLD 同様，インタプリタによる Lisp 処理系であり，Java 言語の Reflection 機構を用いてスクリプトを逐次実行していく実装形態をとっている．したがって，実行効率はそれほど高くないが，言語処理のコアに必要なメモリ量は小さく保てるため，メモリ容量に制約がある場合が多い組み込み機器に適している．また，必要な機能以外はメモリ上に読み込まずに済むよう，モジュールシステムを採用し，機能をモジュール毎にロードできるようにしている．

Jawara の Android 版の処理系は ‘Service’ として実現し，組み込み機器上で常駐する形態を取っている．ネットワーク接続の待ち受けを行なうプログラムを，前面で動作中の他アプリケーションの背後で動作可能である．また，WiFi，GPS，加速度センサ等の機器上の機能をあらかじめ定義された関数やマクロを通じてアクセス可能としており，機器固有の機能を利用したアプリケーションを作成できる．

3. 組み込み機器上における遠隔コード実行

Jawara において，定義されたプログラムを他の機器へ送信する機能，および，遠隔から送り込まれたプログラムを実行(eval) する機能を実装した．これにより，遠隔機器でのコードの実行(遠隔コード実行：Remote Code Evaluation)が実現できる．プログラムの関数定義の書き換えもプログラムとして実行できるため，実行中のプログラムを書き換えることも可能である．既存の Lisp 処理系においても，プログラムを遠隔で実行する機能を有するものがあるが，本処理系の特徴は，遠隔デバッグ機能 Remote-REPL を持つ点と PIAX ID トランスポートを統合している点にある．以下，それぞれについて述べる．

3.1 Remote-REPL

Lisp 処理系においては，逐次的なプログラミングやデバッグ等を行なうインタフェースとして，コンソール上でキーボードによるプログラム入力と結果の出力を繰り返

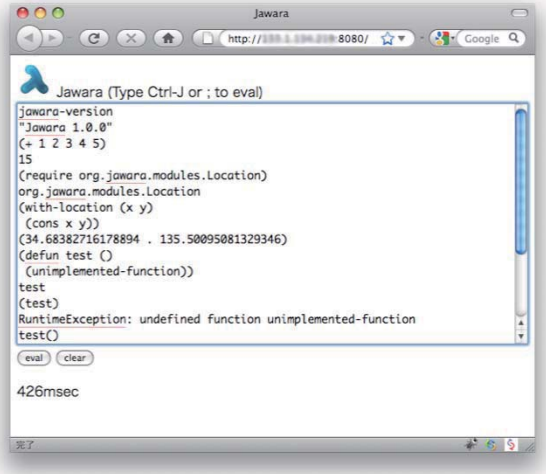

図 1 Remote-REPL

† 大阪大学大学院情報科学研究科，情報通信研究機構

すREPL(Read Eval Print Loop)が通常利用される．しかし，組み込み機器においては，必ずしもキーボードが利用できるわけではなく，ディスプレイが存在するとも限らない．また，Android には，遠隔からコンソールを利用できる shell 機能があるが，デスクトップ PC 等に専用の SDK のインストールが必要となり，REPL が動くとしても，デバッグできる環境が限られてしまう．

そこで Jawara では，組み込み機器におけるコード実行およびデバッグを簡単に行なえるよう，HTTP による遠隔コード実行インタフェースを設け，通常の Web ブラウザ経由でREPLを実行できる‘Remote-REPL’を実装した．図 1 は，Remote-REPL 実行の様子である．テキストボックス上の任意のカーソル位置で，Control + ‘J’ を押下することで，直前のプログラム式が評価される．実装上は，評価の際，テキストボックス内のプログラム式を JavaScript で構文解析して切り出し，Ajax により非同期で Jawara 処理系の HTTP 遠隔コード実行インタフェースに引き渡している．同じプログラムコードの再実行や，プログラムコードを少し変更しての再実行等のインタラクティブなデバッグ作業がテキストボックス上で行なえる．これは，Emacs における Lisp interaction mode と同様の操作感覚である．Remote-REPL により，機器上で稼働するプログラムの置き換えや動作の変更・確認が通常の PC 上で可能となり，デバッグや機能更新のオーバヘッドを軽減できる．

3.2 PIAX ID トランスポートの統合

PIAX[3]は，オーバレイネットワークプラットフォームとして IP や IP 以外を含む異種ネットワークを統合する ID/Locator 分離機構 [4] を実装しており，ノード(ピア)の識別子(以下，ID)と，ピアの接続位置を指し示す位置識別子(Locator)をアーキテクチャ上，別のレイヤで扱うことが可能である．Locator とは，ネットワーク接続に必要となる識別子(アドレス)を指し，TCP, UDP であれば，IP アドレスとポート番号が，ZigBee であれば，ZigBee アドレスが対応する．PIAX の ID/Locator 分離機構の実装には，ID トランスポートレイヤと Locator トランスポートレイヤが存在し，ID と Locator のマッピングを任意の方法で行なえるようになっている．PIAX に用意されている Locator モジュールとしては，TCP, UDP, NAT 越え機能付き UDP,ノード内モジュール間通信(Emulation)などがある．

Jawara は異種ネットワークを統合的に扱えるよう，この PIAX の ID トランスポート機能を取り込んだ．個々の機器には PIAX のピアとして ID が割り振られ，分散処理を実行するプログラムは，ID によりピアを識別する．分散処理を行なう各プログラムコードでは，ID 指定により遠隔コード実行を記述でき，Locator の詳細は考慮する必要がない．

ピア間で直接通信を行なう際，送信元の ID と Locator のマッピングが送信先のピアに伝播する．また，あるマッピングを知っているピアが，そのマッピングに対応するピアへ他のピアが接続しても良いとき，当該マッピングの ID を明示的に「紹介(introduce)」することによっても伝播する．

表 1 は，Jawara における ID トランスポート関連関数である(主要関数を抜粋)．`piax-eval`が指定したピアにおいて遠隔コード実行を行なう関数である．引数には，呼び出し元のピア，実行したい遠隔ピアの ID，および，遠隔に送り込むプログラムコードを指定する．

3.3 記述例

図 2 は，Jawara 上で P2P 型のフラッディングを実現する記述例である．この例は，

表 1 主な ID トランスポート関数

関数名	動作
`piax-open`	引数に指定された locator を用いてピアを立ち上げる．
`piax-tcp-locator`	引数に指定されたスペックに従い TCP の Locator をインスタンス化する
`piax-udp-locator`	引数に指定されたスペックに従い UCP の Locator をインスタンス化する
`piax-introduce-id`	自ピアが ID / Locator のマッピングを持つ ID を「紹介」する．
`piax-eval`	指定された ID を持つピアにおいて遠隔コード実行を行なう．
`piax-known-peer-id-list`	自ピアが ID / Locator マッピングを持つ ID のリスト．

- 自ノードが知る全てのピアに送信
- 同じピアは 2 度通らない
- 最大 7 ホップで終了

という条件で，ピアへのメッセージの送信を繰り返し，指定されたプログラム `prog` を実行する．最初の行から 19 行目までが，この動作を実現するためのコードである．15 行目から 18 行目が，遠隔コード実行に対応しており，通信可能なピアへメッセージを送信する動作を繰り返し実行する記述がなされている．`piax-local-peer` という変数は，関数 `piax-eval` 内の遠隔コードでのみ有効となる変数であり，遠隔コードを実行中の PIAX ピアを指している．ピア ID は文字列であり，17 行目にあるように，遠隔実行のコードでは quasiquote(`)された記述内で unquote(,)すれば送信することができる．エラー処理などを省いたコードではあるが，Locator の管理等は含まれず，意図する挙動がシンプルに記述できていることが分かる．22 行目から 27 行目は，PIAX ピアとして機器をネットワーク接続する指定である．ここでは UDP Locator によりピアを起動している．26, 27 行目において指定している seed-locator は既にネットワークに参加しているピア(シードピア)の Locator である．

4. 評価

本処理系により本格的な分散処理のアプリケーションが記述可能であることを示すため，分散ハッシュテーブル Kademlia[5]を実装した．定期的なデータ更新等の機能は除いたコアとなる機能のみの実装であるが，実質的な作業時間は半日〜1 日程度，約 250 行で記述できた．

プログラミング言語によって行あたりの記述密度，用意されるライブラリの機能が

```
;; 共通コード
(require 'piax)
(defvar flooding-ttl 7)

(defun flooding (peer prog)
  (flooding-forward peer nil prog))

(defun flooding-forward (peer visited prog)
  (eval prog)
  (for-each
   (lambda (id)
     (when (and (< (length visited) flooding-ttl)
                (not (equal id (piax-peer-id peer)))
                (not (member id visited)))
       (piax-eval peer id
         `(flooding-forward piax-local-peer
           ',(cons (piax-peer-id peer) visited)
           ',prog))))
   (piax-known-peer-id-list peer)))

;; ノード上のコード
(defvar peer
 (piax-open
  :locator
  (piax-udp-locator "192.168.1.2" 12367)
  :seed-locator
  (piax-udp-locator "192.168.1.1" 12367)))

(flooding peer '(display "hello"))
```

図 2　フラッディングの記述例

影響するため，一概に記述の容易性を評価することは難しいが，一つの指標として他の実装と同様のプログラムを記述するためにかかる行数の比較を行なった．

図 3 は，上記 Kademlia 実装のルーティング記述部分，Java による実装のひとつである Overlay Weaver(OW)[6]から Kademlia クラス実装部分，Erlang による実装のひとつである ermdia[7]，および，Python による実装のひとつである Entangled[8]から，上記と同等の内容に相当する記述部分を切り出し，行数を比較した結果を示している．いずれも，コメント行と空行を削除した行数である．なお，OW は別モジュールで実装されているルーティングの共通部分のコードの行数は含んでいない．

本処理系では，遠隔コード実行の記述によりネットワーク接続の詳細が隠蔽され，アルゴリズムの本質的な内容のみが記述されており，記述量が少なくなっている．OW が，ルーティングの共通部分のコードを含まないにも関わらず本実装よりも行数が多くなっているのは，Java 言語の必須定義(Exception ハンドリング，変数定義，import など)による影響もあるが，ID と Locator のマッピング相当(IDAddressPair)の管理を含んでいることも一因である．また，ermdia, Entangled は，UDP による通信機能を含む記述となっており，記述量が多くなっている．

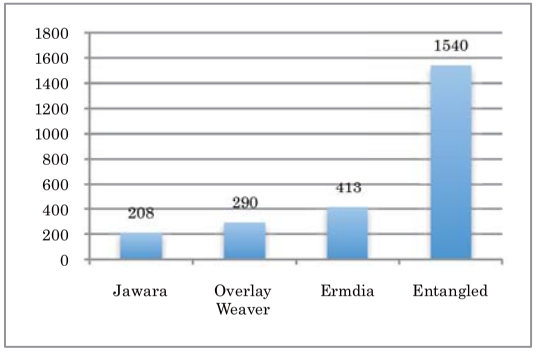

図 3　Kademlia 実装の記述量(行数)

5. まとめ

本稿では，組み込み機器上で動作し，分散処理の記述を簡潔に行なえる Lisp 処理系 Jawara について述べた．今後本処理系を用いた実用的な分散型アプリケーションの開発と処理系としての完成度の向上をはかる．特に現状では遠隔コード実行にアクセス制約等を設けておらず，セキュリティ上問題があるため，実用上は検討が必要である．

謝辞　本処理系の開発，及び検証は，日本電信電話株式会社 ＮＴＴサービスインテグレーション基盤研究所と国立情報学研究所の提供する研究設備，回線を利用した共同研究の一環として実施している．ここに記して謝意を示す．

参考文献

1) Jawara Project, “Jawara,” available at: http://jawara.org/
2) 湯浅太一, “Java アプリケーション組込み用の Lisp ドライバ,” 情報処理学会論文誌：プログラミング, Vol.44, No. 4, pp. 1-16, Mar. 2003.
3) “PIAX,” available at: http://www.piax.org/
4) 吉田 幹，寺西裕一，下條真司, “オーバレイネットワークにおける ID/Locator 分離機構,” 情報処理学会論文誌, Vol. 50 No. 9 pp. 1-14 Sept. 2009.
5) P. Maymounkov, D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric", *Proceedings of* IPTPS02, Mar. 2002.
6) 首藤一幸，田中良夫，関口智嗣, “オーバレイ構築ツールキット Overlay Weaver,” 情報処理学会論文誌：コンピューティングシステム, Vol.47, No. 12, pp.358-367, Sept. 2006.
7) “ermdia,” available at: http://sf.net/projects/ermdia/
8) “Entangled,” available at: http://entangled.sourceforge.net/