# 動的な環境変化に対応するインタプリタインタフェースの提案

3N－6

石口栄一†　早川栄一‡　高橋延匡‡
†拓殖大学大学院工学研究科
‡拓殖大学工学部

## 1　はじめに

近年、計算機の高機能化、小型化によりコンピュータの利用形態がウェアラブルコンピューティング環境や組込みシステムなどに代表されるように多様化してきている。このためハードウェアの構成が変わりやすい。このような環境に対応する柔軟なシステム構成としては、マイクロカーネルアプローチやOSのほとんどをアプリケーションとして実行することで強力な拡張性を提供するExokernel[1]などがある。しかし、実行サイズの肥大化や、OS自体の拡張性の限界という問題がある。

また、JVMをベースにしたシステム構成では、アプリケーションの再利用性は向上するが、開発言語がJava言語に縛られてしまうことになる。

我々は、ソフトウェアの移植性と柔軟性をもたせるため複数のインタプリタをサポートするインタプリタインタフェースを提案する。近年の急速なCPUパワーの向上により実現可能な時代が来ると考えている。

## 2．対象モデル

本研究では、ウェアラブルコンピュータを用いたセンサデバイスを管理するシステムを対象としている。センサデバイスとしては、温度センサ、赤外線センサ、万歩計、GPS、通信シグナル、物理的な変化するデバイス（バイブレーションなど）から構成すると仮定する。本システムとして対象としているのは、人の体に装着されたセンサを用いるシステムであり、そのセンサデバイスから生体情報を取得し、管理する。アプリケーションイメージを図1に示す。

このような場合、センサの追加や削除などのようにハードの構成が変わることがあるためアプリケーションのカスタマイズやシステムのコンフィグレーションが必要になる。これらの要求を満たすためにはシステム自体はより柔軟なものでなければならない。

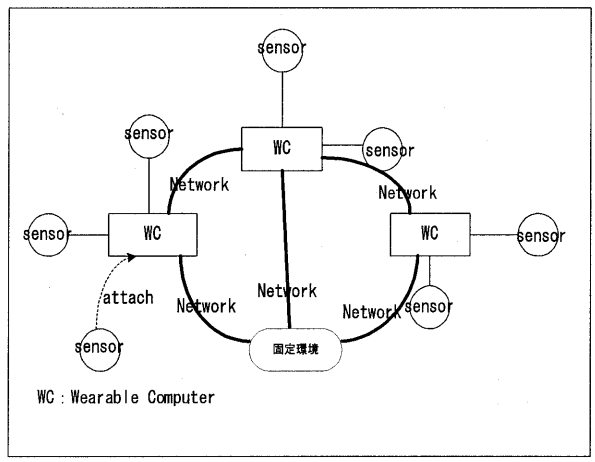


図1　アプリケーションイメージ

## 3　インタプリタインタフェースについて

先に述べたように、ウェアラブルコンピューティング環境では、より柔軟なシステム基盤が必要である。近年のアプリケーションが利用するリソースの量は非常に巨大なものになってきている。これらをウェアラブルコンピュータのような移動環境のストレージ内に蓄えるのはそのサイズやバッテリなどの制約上困難である。

また、従来から提供されているLKM（Loadable Kernel Module）やダイナミックローダなどの手法は、システムやアプリケーションをある程度小さくすることが可能だが、安全性に問題があり、ウェアラブルコンピューティング環境やモバイルコンピューティング環境のような移動環境の要求を満たすものではない。移動環境には、実行環境のみを提供すればいいので、プログラム開発環境などはもたない。アプリケーションのカスタマイズをおこなう場合などは、いったん分散システム上の計算機でカスタマイズをしたアプリケーションを移動環境へもっていくことで解決する。そこで、ウェアラブルコンピューティング環境をはじめとする小型のコンピューティング環境におけるアプリケーション記述言語として、我々はスクリプト言語を用いるアプローチをしている。従来のカーネルコードのうちよりコアな部分以外をインタプリタ化とすることにより柔軟なシステム構成を実現する。インタプリタを用いる利点としては次のようなものがある。

---

A proposal of Interpreter Interface based system in dynamically changeable environment
Eiichi ISHIGUCHI †, Eiichi HAYAKAWA ‡ and Nobumasa TAKAHASHI ‡
† Graduate school of Engineering, Takushoku University
‡ Faculty of Engineering, Takushoku University

(1) APやシステム自体の拡張性を容易に実現可能

モジュールの再コンパイルやリンクの必要がなくアプリケーション自体の拡張性を容易に実現可能である。

(2) APやシステム自体のサイズが小さくなる

アプリケーション自体のサイズが小さくなるため小型計算機のような小さなストレージ内に容易に収まる。また、ネットワークを用いてアプリケーションのダウンロードなども可能になる。

(3) セキュリティの向上が見込まれる

スクリプトを用いることにより、プログラマへの自由度は減るが、カーネルやアプリケーションはインタプリタが提供すること以外は実現できないため、セキュリティ、信頼性が向上する。

## 4 基本設計

### 4．1 全体構成

本システムの全体構成を図2に示す。本システムでは、複数のスクリプトインタプリタを共存して動かす基盤であるInterpreter Coreを提供する。

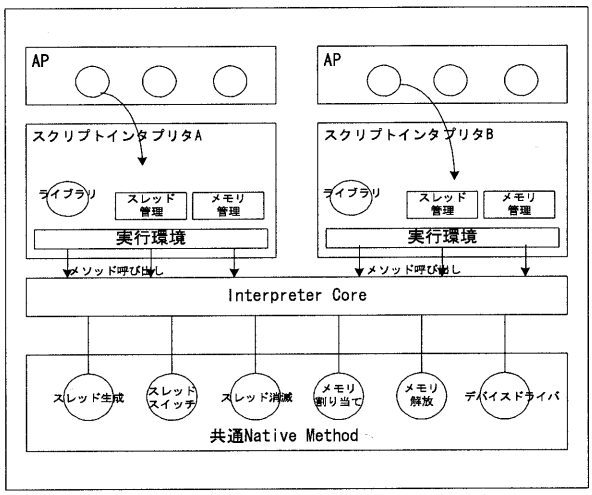


図2 全体構成

### 4．2 Interpreter Core

Interpreter Coreは、ハードウェアとスクリプトインタプリタの間に位置するソフトウェアであり各インタプリタにおけるハードウェアの制御をし、インタプリタインタフェースの実行環境の基盤をつくる。

また、上位の複数のスクリプトインタプリタで提供するスレッド管理、メモリ管理、デバイス管理の基盤となる共通ネイティブメソッドを提供する。Interpreter Coreでは、次のようなメソッドを提供する。

(1) スレッド操作系
- スレッド生成
- スレッド消滅
- スレッド放棄
- スレッドスイッチ

(2) メモリ操作系
- メモリ確保
- メモリ解放
- メモリ書き込み
- メモリ読み出し

(3) デバイス操作系
- タイマドライバ
- その他デバイスドライバ

(4) 通信／結合系
- インタプリタ間のバインド機構
- 通信機構

### 4．3 スクリプトインタプリタ

スクリプトインタプリタはアプリケーションの実行環境を提供する。アプリケーションからはスクリプトインタプリタであるVMが提供する資源にしかアクセスはできない。

## 5 TinyVMによる試作

H8プロセッサ用JVMのサブセットであるTinyVM[2]を用いて試作を行う。通常JVMはマシン依存の部分はJNI(Java Native Interface)が提供するインタフェースを用いてネイティブコードを呼び出している。本システムで提供するInterpreter Coreではスレッドスイッチなどのコアなネイティブコードを共通ネイティブメソッドとして提供する。VM自体は基本的に解釈実行するだけである。共通ネイティブメソッドを呼び出すクラスのインスタンスが生成されると、そのネイティブメソッドがメモリにロードされる。

## 6 おわりに

本稿では、ウェアラブルコンピューティングのような小型計算機環境におけるインタプリタインタフェースについて述べた。Interpreter Coreを用いることにより、豊富にあるインタプリタのソフトウェア環境を使うことができる。今後の課題としては、本システムの実現と移植性の評価があげられる。

## 参考文献

[1] Enger, D.R., Kaashoek, M. F., O'Toole Jr.: Exokernel : An Operating System Architecture for Application-Level Resource Management, In the Proceeding of the 15th ACM Symposium on Operating System Principles, 1995.

[2] TinyVM : http://tinyvm.sourceforge.net/