並列オブジェクトによる大規模システムの実現——Second Life システム，Twitter システム，分子動力学アプリの場合——

米澤　明憲†　　前田　俊行†

Large Scale Applications of Concurrent Objects ——Cases for Second Life System, Twitter System, and Nano-Bio Applications——

Akinori YONEZAWA† and Toshiyuki MAEDA†

あらまし　1970 年代から米澤を中心に研究が始められた「並列オブジェクト」の概念や，それに基づいたソフトウェアシステムの構築法が，最近になりばく大な実ユーザがリアルタイムで利用する Web 系システムやスーパコンピュータ上の実用アプリケーションの開発に使われるようになった．ここでは，そのような開発を可能とする並列オブジェクトの特徴的な機能や性質を明らかにする．また，到来し始めたマルチコア，メニコア時代のプログラミングモデルの一つとしての並列オブジェクトモデルにも言及する．

キーワード　並列オブジェクト，並列プログラミング，Web アプリケーション，スパコンアプリ，マルチコアアーキテクチャ

1. まえがき

我々は今，大量の MPU がラップトップから超高性能コンピュータまでを構成していく，いわゆるマルチコア・メニコア時代の入り口に立っている．そして，スパコンの飛躍的進歩とあいまって，コンピュータの処理能力も今後爆発的に増大しこれが更なる情報爆発を生む時代に突き進んでいる．

更に爆発する情報・データを制御し，その中から有用な情報の抽出・加工を可能にするのは，やはり処理能力の爆発である．処理能力の爆発はコンピュータのハードウェア性能の爆発だけでは獲得できない．いうまでもなくそれには，大規模な並列・分散処理のためのプログラミングやソフトウェア開発方式の確立が不可欠である．

本論文は，VLSI の量産の可能性が示唆され始めた今から 30 年以上前の時代に米澤が考えた「並列オブジェクト」という概念のいくつかの側面を紹介するものである．その当時からの遠い将来，印刷物を刷るようにコンピュータ（CPU）が生産され，それが安価に利用できるようになったとき，我々は一体どのようなプログラミング方式によって，そのばく大な処理資源を効率良くかつ容易に利用することできるのか．そのことを考えた結果が「並列オブジェクト」である．

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻，東京都 School of Information Science and Technology, University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–0033 Japan

2. 並列オブジェクトとは

並列オブジェクトを，直観的に理解してもらうための準備として，ソフトウェア部品の一つである「オブジェクト」の概念を簡単に説明する．

2.1 ソフトウェアシステムの構成

メールサーバ，ファイルサーバ，あるいは Web サーバなどのいわゆる「サーバ」や Windows や Linux のような OS などは，プログラムの集まりである，ソフトウェアシステムである．もちろん様々な分野で開発される応用ソフトウェアもソフトウェアシステムである．

一般に，ソフトウェアシステムは多数のモジュール，すなわちソフトウェア部品から構成される．これは，車が多数のパーツ・部品を組み合わせて作られているのと同じである．例えば，Mozilla や IE などのようないわゆる Web ブラウザはかなり大きなソフトウェアシステムである．これらは，図 1 にある方形の箱で示したような多数のモジュールから構成されており，例

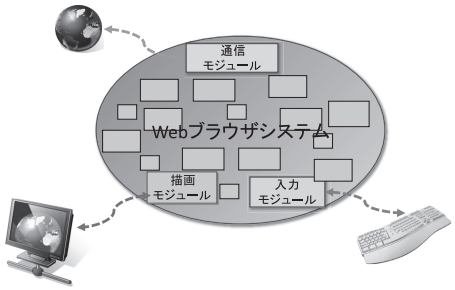

図 1　ソフトウェアシステムは多数のモジュール（部品）により構成される．
Fig. 1　Software consists of many modules.

えば通信，キーボード入力処理，ディスプレイへの出力などの機能やブラウザ特有の機能を実現する種々のモジュールがある．

2.2　モジュール（ソフトウェア部品）の要件

サイズが大きく強力な機能を備えたソフトウェアシステムを，仕様どおりに効率良く構築し，それを低コストで維持・保守するためには，システムを構成する多数のソフトウェア部品が，以下に述べるような性質を満たす，すなわち高いモジュール性をもった「形式」に従って開発されている必要がある．

（1）　部品は，明確なインタフェースをもつ

（2）　部品は，それが組み込まれる位置の周辺にある他の部品からの独立性が高い

（3）　部品は，他の部品と組み合わせて，より大きい部品を作ることができる

（4）　部品は，同等な機能をもつ他の部品によって置き換えることができる

計算機の黎明期から，あるサイズ以上のプログラムを開発するために，サブルーチン，手続き，関数などの機能がプログラミング言語に備えられた．これらの機能を実現するプログラム部分が，ソフトウェア部品，すなわちモジュールとみなされ，これらをメインプログラムから呼び出すことによりソフトウェアシステムが構成・実行される．

更に，1970 年代半ばごろに設計されたプログラミング言語では，アプリケーション領域からの必要に応じてプログラマ自身が新しいデータ型を定義・実現できるようになった．これが抽象データ型である．

2.3　オブジェクト

1990 年代半ば以降から開発されている大半のソフトウェアシステムは，いわゆる「オブジェクト指向」法に基づいて開発されている．この方法は「オブジェクト」と呼ばれる形式のモジュールを組み合わせてシステムを構築していくものである．オブジェクトと呼ばれるモジュールの形式は，基本的には前項で述べた抽象データ型を実現するソフトウェアモジュールの形式で，その構成はおおよそ図 2 のようなものである．

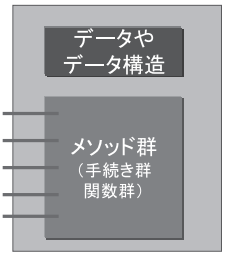

図 2　オブジェクト形式
Fig. 2　Object format.

オブジェクト形式のモジュールは，ひと固まりの変数やデータ構造とそれに対して参照・更新などの操作をするメソッドと呼ばれる手続き・関数群で構成され，このモジュール内にあるデータは，ここで定義されているメソッドを用いる以外にはアクセスできない．

データは指定されたメソッド以外では操作することは許されず，また各メソッドもその呼出し（あるいは起動）の仕方が定められている．オブジェクト形式は，この意味でデータとそれを操作する手続き群が密閉あるいはカプセル化され，ソフトウェアシステムのモジュールを形成している．（ここではプログラムのコードを共有・再利用するためのオブジェクト指向言語特有のインヘリタンス（継承）機構については割愛する．）

2.4　並列オブジェクト

このオブジェクト形式を一般化し，並列・分散処理に向けた計算モデル・プログラミングモデルの基礎となったものが「並列オブジェクト」である．別の言い方をすれば，並列オブジェクトは分散・並列処理を行うソフトウェアシステムの部品形式のことである．

並列オブジェクトは，直観的に説明すると，オブジェクト形式に（仮想的な）CPU を埋め込んだ形式と見ることができる（図 3 参照）．

すなわち並列オブジェクトは，データとそれを操作する手続きと操作を実行する CPU が一つのモジュールとして密閉されたものである．一つの並列オブジェクトは，一つのオブジェクトに一つの CPU を埋め込んだものとみなせる（図 4 参照）．

並列オブジェクトでは，その中に密閉されているメソッドを起動するには，メソッドの名前を指定したメッ

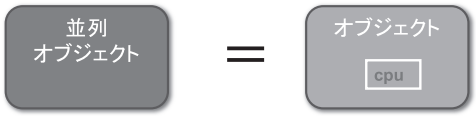

図 3　並列オブジェクトの直観的説明
Fig. 3　Intuitive explanation of concurrent objects.

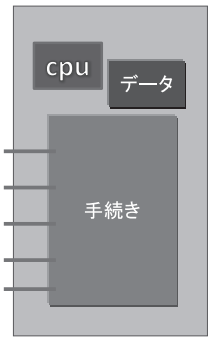

図 4　並列オブジェクト形式
Fig. 4　Concurrent object format.

セージをその並列オブジェクトに送る．メッセージが到着すると，並列オブジェクトは，指定されたメソッドを自分のもつ CPU を用いて実行する．

並列オブジェクト群で構成されるソフトウェアシステムでは，並列オブジェクト同士のメッセージのやり取りで，計算・情報処理が進行する．その際，メッセージ送信は基本的には非同期的で，メッセージを送った並列オブジェクトはそのメッセージがあて先である並列オブジェクトに到着し，指定したメソッドが起動されることを確認することなく，メッセージ送信以降の実行を直ちに継続する．この非同期通信によるメソッドの起動は，ソフトウェアシステム内の計算・処理の並列度を高め，モジュール（並列オブジェクト）間の独立性を強めるのに大きく貢献する．

2.5　並列オブジェクトによるシミュレーション

1970 年代半ばころ，米澤が並列オブジェクトを構想するときにはある構想が念頭にあった．将来，計算機は様々な問題領域内の世界をモデル化し，それをソフトウェアシステムとして表現・実行し，その世界を丸ごとシミュレーションするであろう．そのとき，その世界に登場する事物を並列オブジェクト群で表現し，事物の間のインタラクションを並列オブジェクト間のメッセージの伝送で表現・シミュレートすれば，世界のもともとの構造をあまりひずませることなくソフトウェアシステムとして表現できるのではないか．この考えはそれほど大きな間違いはなかったと思われる（図 5 参照）．

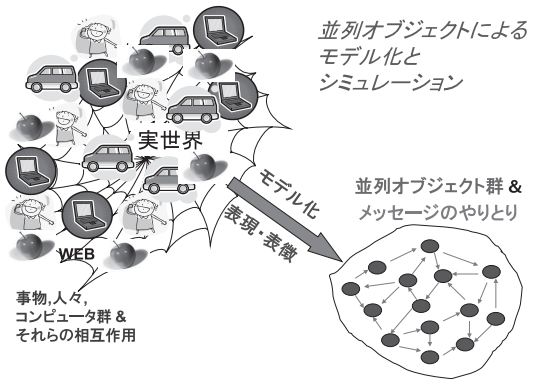

図 5　並列オブジェクトによるモデル化とシミュレーション
Fig. 5　Modeling and simulation with concurrent objects.

3.　並列オブジェクトの特性

本章では，前章でその概要を述べた並列オブジェクトについてより詳しく説明する．具体的には，六つの特徴的機構と性質について説明する．

3.1　メッセージ駆動型（message-driven）処理

並列オブジェクト内に密閉されているメソッド（手続き）群を起動するには，並列オブジェクトにメッセージを送る必要がある．メッセージを受け取った並列オブジェクトは，メッセージの中に指定された名前に従って，その名前をもつメソッドを起動し実行する．起動されたメソッドの実行が終了すると，その並列オブジェクトは次のメッセージが到着していれば，そのメッセージに従って次のメソッドの実行を始めるが，メッセージが到着していなければ，並列オブジェクトは休眠状態に入る．このように並列オブジェクトは基本的には，メッセージが到着することがきっかけで計算・情報処理が引き起こされる，メッセージ駆動型の処理方式をとる（図 6 参照）．

3.2　非同期メッセージ送信

並列オブジェクト間のメッセージ送信は基本的に非同期的（asynchronous）である．非同期通信に加えて，実行をいったん停止し，何らかのメッセージが到着すれば実行を再開するという処理が許されれば，非同期通信をもとに，同期通信も実現できることはよく知られている（図 7 参照）．また，並列オブジェクトに定義されているメソッドを起動するための非同期的メッセージ送信とともに，単純に値や情報を並列オブジェ

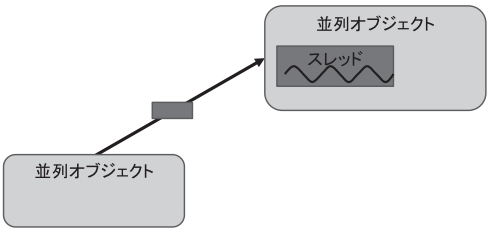

図 6　メッセージ駆動処理
Fig. 6　Message-driven processing.

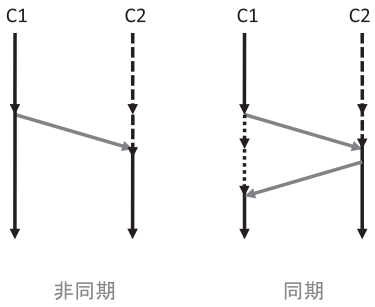

図 7　同期/非同期通信
Fig. 7　Synchronous/asynchronous communication.

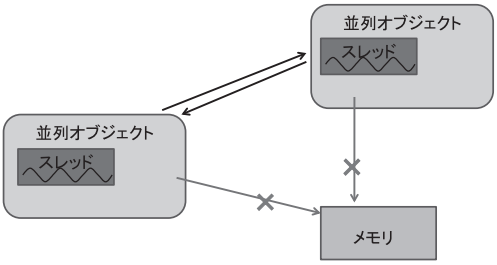

図 8　並列オブジェクト間の情報交換
Fig. 8　Information exchange between concurrent objects.

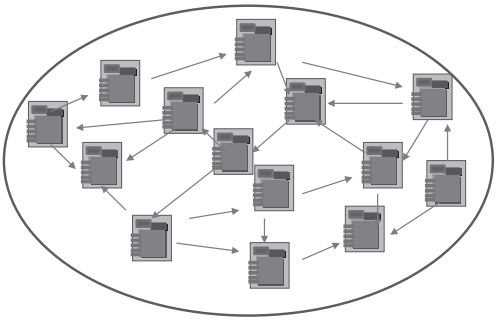

図 9　並列オブジェクトによる並列化
Fig. 9　Parallelization with concurrent objects.

クトに渡すための非同期的メッセージ送信もある．

3.3　情報の交換はメッセージ送信のみ

並列オブジェクト同士の情報交換は，転送されるメッセージを介するという手段のみが許される．この結果，プログラムシステムが並列オブジェクトのみで構成されるなら，ソフトウェアモジュール間の情報のやり取りは，並列オブジェクト間のメッセージのやり取りのみによって行われる．

つまり，図 8 にあるように，並列オブジェクト間のメッセージのやり取りによる情報交換は許されるが，並列オブジェクトのメソッド実行中に read/write 操作を直接共有のメモリに施すことは許されない（図 8 の × 印）．

3.4　多数の並列オブジェクトによる超並列性

並列オブジェクトの枠組みでは，メッセージ通信が非同期でかつメッセージの到着により並列オブジェクトが起動される．このため，並列オブジェクトが例えば二つあれば，それぞれが一つの仮想 CPU（スレッド）をもつので，この系の並列度は 2 となる．このように，並列オブジェクトをモジュールとして構成されるソフトウェアシステムでは，使われている並列オブジェクトの数だけの並列性が系の中に存在し得ることになる．

これまでの説明では，並列オブジェクト内には一つだけの仮想的な MPU（あるいはスレッド）があることを前提としている．この前提をくずし，複数のスレッドも許すような亜種の並列オブジェクトの枠組みもあるが，その場合並列オブジェクト内の並列性を許すことになり，同一オブジェクト内でスレッドの排他制御やデッドロック回避など，伝統的な並列プログラミングに内在する問題点を引き継ぐことになり，並列オブジェクトの利点が少なからず減少することになる．

3.5　安全な軽量プロセス（lock，unlock 操作が不要）

伝統的な並列プログラミングでは，複数のスレッドが同じデータにアクセスする場合に，古いデータや更新中の中間状態のデータを読み出してしまうような，いわゆるレース状態が引き起こされたり，スレッド間の同期の不具合からデッドロックを引き起こす危険がある．これを防ぐために，プログラマは lock/unlock 操作を用いて注意深く並列プログラミングを行わなければならない．しかしながら，多数のスレッドを利用し高度な処理をする場合には，レースやデッドロックが決して生じないという確信をプログラム開発者がもつことは大変困難である．

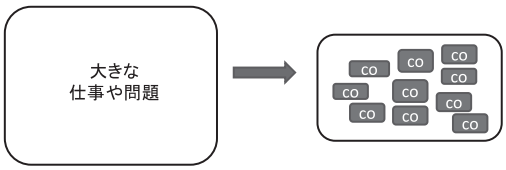

図 10　並列オブジェクトを用いた問題の分割
Fig. 10　Division of a problem by concurrent objects.

並列オブジェクトを用いた並列プログラミングでは，複数のスレッドによる共有データへの直接のアクセスは存在しない．また並列オブジェクトのメッセージ送信の方式によって，lock/unlock 操作を使うことなく同期を実現できる．このために，並列オブジェクトをベースにした並列プログラミングでは，レース状態やデッドロックが生じないプログラムを開発することが容易に可能になる．

3.6　並列オブジェクトを用いたモデル化と実装

2.5 でも触れたとおり，並列オブジェクトの考え方は，本質的に複数の事物が並列的に動作・相互作用（インタラクション）する実世界の諸問題のモデル化に適していると考えられるが，実際にモデル化したモデルをソフトウェアシステムとして効率的に実装・実行できるのは，一つの大きな問題を，相互作用が局所的なものに限定されるような大量の事物に分割できる場合である．

例えば，大量の粒子・物質の物理的シミュレーション（多体問題など）や，自律的に動作する複数のエージェントを用いることで複雑なシステムをモデル化するマルチエージェントシステムなどを，並列オブジェクトを用いて実装することが可能である．

また，問題の分割をより抽象化して工夫することで，更に多くの問題を並列オブジェクトの考え方でモデル化・実装することが可能である．例えば，CKY 構文解析や遺伝的アルゴリズムなどは，データ間の競合などを考慮してうまく問題を分割することで，並列オブジェクトを用いて効率的な並列化が実現できる．

次章では近年行われた並列オブジェクトの大規模な応用例をいくつか紹介する．

4.　大規模応用例

並列オブジェクトをベースに，米国を中心にそれぞれ特徴的な三つの大規模システムが構築されている．

4.1　Second Life システム

リンデン社の Second Life システムは，世界で数百万以上のユーザをもつ 3D 仮想世界構築ツールで，経済活動，教育プログラム，種々の娯楽等の分野で利用され，発表当時に大きな話題となり，現在もいろいろな分野で着実に利用されている．

Image from “Programming Second Life with the Linden Scripting Language” by Jeff Heaton (http://www.devx.com/opensource/Article/33905)

図 11　Second Life システム
Fig. 11　Second Life system.

このシステムでは，ユーザが構築する仮想世界におけるアバターをはじめとする登場物及び登場物間の相互作用は，それぞれ並列オブジェクトとその間のメッセージのやり取りとしてプログラミングされるように，ユーザが使う記述言語（スクリプト言語）と豊富なライブラリが用意されている．

Second Life システムで，ユーザに提供されるプログラミング形式（言語）として並列オブジェクトが採用されている理由は，並列オブジェクトがもつ自然で強力なモデル化能力にある．Second Life の（仮想）世界の各登場物は状態とそれに依存する行動からなるが，それらは並列オブジェクト内のデータ・値群と，メソッド群で容易に表現できる．このため一つの単体としての登場物を，並列オブジェクトという単体のソフトウェアモジュールとしてモデル化・表現することは直接的で自然である．

4.2　Twitter システム

140 文字以内の短いメッセージ（つぶやき）を仲間やコミュニティに送り合う Twitter システムは，既に新しいコミュニケーションのメディアになりつつある．米国の大統領の意見や国際的な草の根的世論を伺い知る手段から，学会などでの講演に対する意見や反応を聴衆の間で即時に交換するツールとして活用されている．

このようなシステムには，最大で毎秒数万個を超え

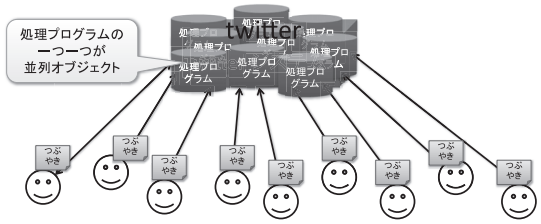

図 12　並列オブジェクトによるつぶやきの処理
Fig. 12　Processing tweets with concurrent objects.

るメッセージを即時にそれ以上の数のユーザに配信する処理を可能にする技術が要求される．

Twitter システムのバックエンド，特にメッセージキューシステムのプログラムを作成した Robey Pointer によれば [7]，ユーザからの接続ごとに並列オブジェクトを生成し，その並列オブジェクトにつぶやきを処理させている．ここで重要なことは，このつぶやき処理の中核 kestrel [8] は，Scala 言語の並列オブジェクト（Actor）ライブラリを用いて，1500 行程度のプログラムで記述されている点である．このように，並列オブジェクトを用いることで，高速な処理と簡潔なプログラム記述が可能となっている．

4.3　分子動力学シミュレーションプラットホーム（NAMD）

イリノイ大学の計算機科学者のグループと理論生物学者のグループが共同開発した，スパコン向けのナノスケール分子動力学のシミュレーションプラットホーム NAMD は，並列オブジェクトをベースとしたプログラミング言語・実行系 Charm++ [9] によって実装されている．

Charm++ は同大の Sanjay Kale 教授を中心に 1990 年代から開発が開始され，最近では分子動力学のみならず，量子化学，天文学のシミュレーションプラットホームの開発にも使われている．その実行時系は，並列オブジェクトの並列計算機内のノードをまたがる移動により動的分散を行わせるフレームワークも提供しており，並列オブジェクトシステムとしても先進的である．Charm++ を用いて開発されたアプリケーションは，2008 年には米国の二つの主要スパコンセンターが保有する資源の 10%から 20%を占めるに至っている．また，米澤らが 1980 年代半ばに提案しその後の様々の角度から研究・開発を行った言語系 ABCL [3] は，Charm++ に少なからぬ影響を与えている．

テキサス大学のスパコンセンターでは 2009 年の前半に NAMD を用いて新型ウイルスの分子動力学的解析を行い，その抗タミフル性を解析し顕著な成果を得たことが報道されている．

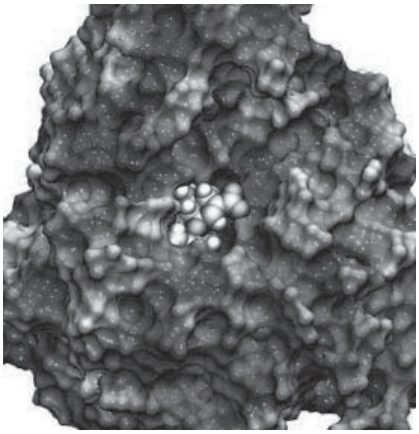

Image from http://www.ncbi.nlm.nih.gov/rrn/RRN1015

図 13　H5N1 鳥インフルエンザノイラミニダーゼとタミフルのシミュレーションの可視化
Fig. 13　Visualization of simulation of H5N1 avian flu neuraminidase with Tamiflu.

このように，近年の米国の HPC（高性能計算）の研究分野では，大量の MPU 上で動作するプログラムを効率的に作成・実行するための新しいプログラミング言語・処理系の模索が（並列オブジェクトに限らず）幅広く行われている [9], [11]〜[14].

例えば，X10 [11] は DARPA の HPCS プログラムのもとで IBM が研究開発しているプログラミング言語である．X10 は Java プログラミング言語のサブセットを，並行性や非同期通信・計算，大域アドレス空間分割などで拡張したようなプログラミング言語であり，Blue Gene を含む将来のスパコン上で実用することを目指している．また同様に Cray は Chapel というプログラミング言語の研究開発を行っている [12]．並列オブジェクトは，このような新たなプログラミング言語研究・開発の先駆的存在であり，今後その重要性は更に増していくものと予想される．

5. むすび

本論文は並列オブジェクトに関するこれまでの研究のうち，一般人にも理解しやすい部分を紹介したものである．これらの研究により米澤は，2008 年 7 月には，国際オブジェクト技術協会（AITO，スイス・ベルン在）からダール・ニゴール賞を受賞，また 2009 年

11 月には紫綬褒章を受章した．これは，共同研究者である元スタッフや学生諸氏に負うところ極めて大である．特に，柴山悦哉，J.-P. Briot，渡辺卓雄，松岡聡，小林直樹，田浦健次朗，増原英彦の諸氏にこの場を借りて，改めて深謝させて頂く．

文　献

[1] 米澤明憲，“私のソフトウェア研究,” コンピュータソフトウェア，vol.21, no.5, pp.20–31, Oct. 2004.
[2] A. Yonezawa, J.-P. Briot, and E. Shibayama, “Object-oriented concurrent programming in ABCL/1,” Proc. OOPSLA86, pp.258–268, Oct. 1986.
[3] A. Yonezawa, ABCL: An Object-Oriented Concurrent System, MIT Press, 1990.
[4] A. Yonezawa, Specification and Verification Techniques for Parallel Program based on Message Passing Semantics, TR191 (Ph.D. Thesis), MIT Laboratory for Computer Science, 1977.
[5] 米澤明憲，“ACTOR 理論について,” 情報処理，vol.20, no.7, pp.580–589, 1979.
[6] A. Yonezawa and M. Tokoro, eds., Object-Oriented Concurrent Programming, MIT Press, 1987.
[7] R. Pointer, private communication, Aug. 14 2009.
[8] Kestrel system. http://github.com/robey/kestrel/tree/master
[9] S. Kale and S. Krishman, “CHARM++: A portable concurrent object oriented system based on C++,” ACM SIGPLAN Notices, vol.28, no.10, pp.91–108, Oct. 1993.
[10] M.T. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L.V. Kale, R.D. Skeel, and K. Schulten, “NAMD: A parallel, object-oriented molecular dynamics program,” International Journal of High Performance Computing Applications, vol.10, no.4, pp.251–268, 1996.
[11] P. Charles, C. Grothoff, V. Saraswat, C. Donawa, A. Kielstra, K. Ebcioglu, C. Von Praun, and V. Sarkar, “X10: An object-oriented approach to non-uniform cluster computing,” Proc. OOPSLA2005, pp.519–538, 2005.
[12] Chapel: The Cascade High-Productivity Language, Cray, http://chapel.cray.com/
[13] Fortress, Sun Microsystems, http://projectfortress.sun.com/
[14] CUDA, NVIDIA Corporation, http://www.nvidia.com/object/cuda_home.html
[15] Second Life, Linden Ltd. Co, http://secondlife.com/
[16] J. Purbrick and M. Lentczner, “Second life: The world’s biggest programming environment,” Invited talk, OOPSLA 2007, Portland Oregon, Oct. 2007.
[17] C. Hewitt, “A universal actor formalism for artificial intelligence,” Proc. IJCAI, Stanford, pp.235–245, Sept. 1973.
[18] C. Hewitt and H. Baker, Jr., “Laws for communicating parallel processes,” Proc. IFIP Congress, 1977.
[19] G. Agha, A Model for Concurrent Computation in Distributed Systems, MIT Press, 1987.
[20] 米澤明憲，“並列オブジェクト指向言語 ABCL/1 による並列処理記述とその枠組みの研究,” 信学論（D），vol.J71-D, no.8, pp.1415–1422, Aug. 1988.
[21] 米澤明憲，柴山悦哉，J.-P. Briot，本田康晃，高田敏弘，“オブジェクト指向に基づく並列情報処理モデル ABCM/1 とその記述言語 ABCL/1,” コンピュータソフトウェア，vol.2, no.1, pp.9–23, April 1986.

（平成 22 年 2 月 18 日受付）

米澤　明憲

1970 東大・工卒．1977 MIT 計算機科学科博士課程了．Ph.D. in Computer Science．現在東京大学情報理工学系研究科教授．日本ソフトウエア科学会理事長，フェロー，功労賞受賞，ドイツ国立情報科学技術研究所（GMD）科学顧問，内閣府総合規制改革会議委員を歴任，現在（独）産業技術総合研究所情報セキュリティ研究センター副センター長，東京大学情報基盤センター長を兼務，21 期日本学術会議会員．マイクロソフト本社 Security Academic Advisory Board メンバ．2008 年国際オブジェクト技術協会（AITO）Dahl-Nygaard 賞受賞，2009 年船井業績賞，2009 年秋紫綬褒章．

前田　俊行

2006 東京大学大学院情報理工学系研究科博士課程了．博士（情報理工学）．現在，東京大学大学院情報理工学研究科コンピュータ科学専攻助教．