Vol.2021-ICS-204 No.1
2021/9/15

エージェントシステム試作プラットフォームMiLogの実行環境の多様性確保に関する一考察 –20年間の環境変化からみた考察–

福田 直樹[1,a)]

概要：本稿では，エージェントシステム試作プラットフォーム MiLog におけるこれまでの 20 年以上にわたる開発の履歴のなかから，特に実行環境の変遷やその変化に対応できるようにするための実行環境の多様性の確保に関する側面からの考察を述べる．特に本稿では，この MiLog の開発のこれまでの 20 年程度となる過程について，そのソフトウェアプラットフォームの実行環境の多様性と変遷という視点からその知見を述べると共に，開発当初に想定した実行環境の将来的な多様性について，どのように開発の過程に反映されてきたか，そこで想定されなかったことは何かについて，著者の主観も交えて考察する．

1. はじめに

ソフトウェアの試作や実装を伴う研究において，事後的な検証を後年になってから行う場面だけでなく，そのソフトウェア資産を長期的に継続して活用していくという側面からも，そのソフトウェア・プログラム等の長期的・継続的な利用・実行可能性の確保は１つの重要な課題となりうる [1][2]．たとえば，30 年後における知能システムの進展 [3] という命題を考えてみた場合，個々の技術や方向性には当時の想像にはなかったものが使われることもあるが，一方で驚くほど正確にその未来を予見し本質を突いていたと考えられるものもある．研究活動およびその成果物・副産物として得られたものに対して，その事後的な価値を現代の視点から振り返ることは重要である一方で，それらの研究を実施に遂行している場面においては，事後的に見れば明らかに見えるようなその分野の研究の最先端が，必ずしもその先端の研究を進める場面で強く自覚されないということも起きうる [4]．このことは，その将来において事後的なあるいは長期的な活用が期待されるソフトウェア資産等であっても，必ずしもその開発・研究過程で自覚的にそれを意識したものとなることが自明ではないということを意味する．

本稿で述べる MiLog[5] と名付けられたソフトウェアは，これまで著者が 20 年にわたり開発を行ってきたエージェントシステム試作プラットフォームであり [2]，個々のエージェントに対して論理型言語実行処理系を備え，エージェント間でのクエリを起点とした相互通信ができ，エージェントを他のコンピュータ上で動作する実行環境へ，その実行スレッド動作状態を保ったまま移動させる強いモビリティ（強モビリティ，Strong Mobility）を実現している [6]．

MiLog は Java 言語を用いて実装されており，図 1 にも示される通りそのプログラミング・実行モニタリング環境としてのグラフィカルユーザインタフェース (GUI) を含むものであるが，これまでに，MacOSX, Windows, Linux, Solaris, および OpenBSD 上などといった異なる種類の OS や CPU アーキテクチャ上での動作実績があるほか，ほぼ同様の互換性を持つ Java VM が動作する初期のハンドヘルド PC(WindowsCE) などでも動作し，現在でも Raspberry Pi3/4B および Jetson Nano/Xavier NX 等の組み込み向けの小型コンピューティングデバイス上での動作も確認されている．オープンソースのプログラミング言語処理形においても多くの CPU アーキテクチャや OS に対応する事例は見られるが，その中で，プログラミング・実行モニタリング環境としての GUI を実装に含んでいてなお前述のように多様な OS/CPU アーキテクチャなどのプラットフォームに対応し 20 年程度以上の長期間にわたって開発・運用された事例は，著者が知りうる限りでは他に例が思いつかない．

MiLog の開発に当たっての経緯や履歴についてはすでに文献 [2] で概要を述べており，その 20 年程度にわたる開発履歴として残った開発過程に関する記録データについて

[1] 静岡大学 学術院情報学領域
Shizuoka University, Johoku, Hamamatsu 432–8011, Japan
[a)] fukuta@inf.shizuoka.ac.jp

© 2021 Information Processing Society of Japan

も，その解析方法の検討を進めている [1]．それらの検討過程でも明らかになってきているが，これまでの MiLog の研究開発・運用過程を振り返った時に，20 年程度の時間の経過に伴って，対応できた実行環境となる OS/CPU プラットフォームの多様性は事前の想像を大きく超えるものであったと考えている．

本稿では，この MiLog の開発のこれまでの 20 年程度となる過程について，そのソフトウェアプラットフォームの実行環境の多様性と変遷という視点からその知見を述べると共に，開発当初に想定した実行環境の将来的な多様性について，どのように開発の過程に反映されてきたか，そこで想定されなかったことは何かについて，著者の主観も交えて考察する．

2. 準備

本節では，文献 [2] で述べた MiLog プラットフォームの開発および運用・応用の履歴について，簡潔にまとめる．

2.1 プラットフォーム開発の履歴

MiLog プラットフォームの開発の構想は 1998 年 3 月頃に著者が主体として考えられたもので，途中様々な議論を経て，モビリティの実装まで含めた基本的な実装が 2000 年頃までに行われてきている．その後，2004 年頃までに NAT の外側に出て通信・移動できるようにする機構の拡張が行われた [2]．

MiLog プラットフォームの開発では，その開発における履歴を `memo.txt` という 1 つのテキストファイルに継続して記録してきている [2]．この記録は本稿執筆時点で 8666 行であり，記録自身が開始されたのは 1998 年 10 月 28 日からで，MiLog の論理型言語処理系コア (Prolog サブセット) の実装と Web アクセス対応ライブラリの試作をほぼ完了しマルチエージェント環境への拡張を進めつつあった時期からの開発のほぼ全ての履歴を記録している．

この記録によれば，1999 年 8 月 31 日の段階でエージェント間クエリをコンピュータをまたいで TCP/IP 通信により実行できる機構が実装されると同時にエージェントの強いモビリティの初期の試作動作が確認されている．1999 年 11 月 28 日に，BiddingBot システムの試作を MiLog を用いて行うための組込述語の追加実装を行った記録があり，少なくともこの段階で後述の BiddingBot システムの基礎的な概念設計をある程度 MiLog 上で行えていたと考えられる [1]．

これ以後に大きな動きのあった実装上の変更があったのは，2004 年 1 月 24 日での NAT を抜けてエージェントが移動できる機構の実装完了の記録であり，この段階以降の改変の記録は本稿執筆時点で 2151 行が確認されていることから，記録の行数の割合で 70 パーセント以上がこの時期までに行われている．最も最近の記録は，本稿執筆時点では 2021 年 1 月 18 日のものであり，HTTPS に関連した最新のセキュリティ機構の実装および Java のバージョン 15 以降で発生する非互換性の問題に追従するためのコードの改訂が記録されている．

2.2 活用の履歴

すでに文献 [2] でも述べた通り，MiLog プラットフォームは，モビリティの実装が完了するよりも前から，主にマルチエージェント的なソフトウェアのプロトタイピングに使用されてきており，その初期にプロトタイピングに実際に使用された事例の代表的なものには，協調型のマルチエージェントを用いたインターネットオークション入札支援システムである BiddingBot[7] がある．MiLog 自身が持つモバイルエージェント機能の開発強化に向けた機能拡張としては，iML(スタンドアロン GUI 構築のための拡張)[8]，MiPage(エージェントの Web アプリケーション化のための拡張)[9]，MiNet(P2P アドホックネットワーク構成を用意にするための拡張)[10] が，MiLog をプラットフォームコアとして開発され発表されている．また，MiLog プラットフォームのモビリティを機能として用いた事例としては，P2P 環境下でのファイル共有検索における意味レベル照合の効率化に適用した事例 [11] があり，関連する発表で情報処理学会より山下記念研究賞を受けている [*1]．

文献 [2] でもすでに述べているが，直接的なターゲットアプリケーション開発ではなく，研究遂行上の計算プロセスの管理および結果の解析に MiLog が用いられた事例としては，組合せオークションの勝者決定の高速化に関わる一連の研究 [12][13][14][15][16][17][18] がある．実行環境の多様性が重要となった場面としては，データの解析にあたって大量のメモリを必要とする場面があり，研究開始当初より MacPro(1st Generation, Early 2008) に 32GB のメモリを搭載してその処理を実行できるようにしていた．このサイズのデータを処理する MiLog 言語のコアである MiLogEngine の基本設計は，MiLog のコア機能の実装を行った 2000 年頃よりほぼ変わっておらず，大容量データ処理への対応には計算過程における内部 ID の表現を 32bit 整数から 64bit 整数に拡張したのみで済み，それをそのまま 64bit Java VM 上で動作させることでこのサイズのデータを処理させることができた [2]．その他，研究支援という側面からは，著者自身のホームページのサーバとしての運用等のほか，本項を含めて著者が筆頭として執筆したほぼすべての LaTex 原稿のコンパイル処理を MiLog で記述したプログラムにより行っている．

このような経緯により，著者は MiLog を研究遂行のみでなくその支援過程まで含めた複数の方向性の異なる目的に対してこれまでに使用を継続してきており，その過程で，

[*1] http://www.ipsj.or.jp/award/yamasita2011-detail.html#ics

© 2021 Information Processing Society of Japan

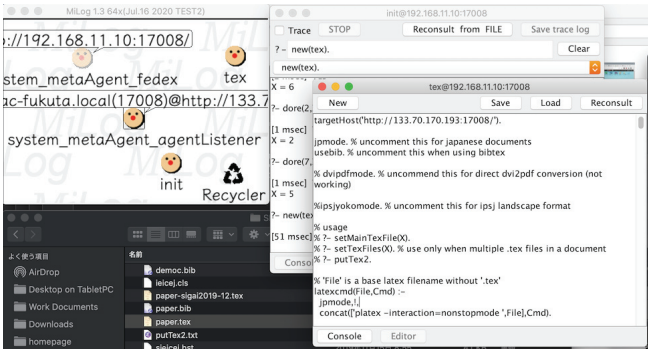

図 1 MiLog の基本的なユーザインタフェースの執筆時最新と思われる動作環境の 1 つにおける動作例 [2]

その必要性および偶発的な要因により，多様な実行環境上での使用・運用を行ってきた．この運用過程における知見や気づきについて，次節で述べる．

3. 動作環境とその多様性

MiLog の基本的なユーザインタフェースについて，本稿執筆時での実行動作環境 *2 における動作例 [2] を，図 1 に示す．MiLog では，動作中のエージェントがエージェントモニタ（図 1 左側）上にアイコンとして表示されるとともに，それぞれのエージェントに対して専用のコンソールウィンドウのような GUI（図 1 右側）が割り当てられ，エージェントが他の実行環境へネットワークを通じて移動した際にはそのコンソールウィンドウがそのまま移動に追従し，移動先での結果出力および操作を受け付けることができる．

ここで，ほぼ同一の CPU アーキテクチャ系列でおよそ 20 年程度の生産時期の違いのあるハードウェア実行環境として，Sun Ultra10(1998 年発売モデル，UltraSPARC-IIi 333MHz 256MB メモリ) と SPARC T5-2 Server(2017 年まで生産，SPARC T5 3.6GHz total 256CPU-threads dual-socket 256GB メモリ) を例に，ほぼ同一のソフトウェア実行環境上における動作をさせた *3

図 2 は，Ultra10 上で MiLog 実行環境を動作させた例である．一方で，図 3 は，SPARC T5-2 Server 上で MiLog 実行環境を動作させた例である．Ultra10 は，1990 年代に発売されたハードウェアであるが，当時より 64bit CPU を搭載しており，一部の実装メモリ容量上の制限事項を除けば現行に近い 64bit バイナリを動作させることが可能である．Ultra10 に搭載される CPU である SPARC-IIi から，

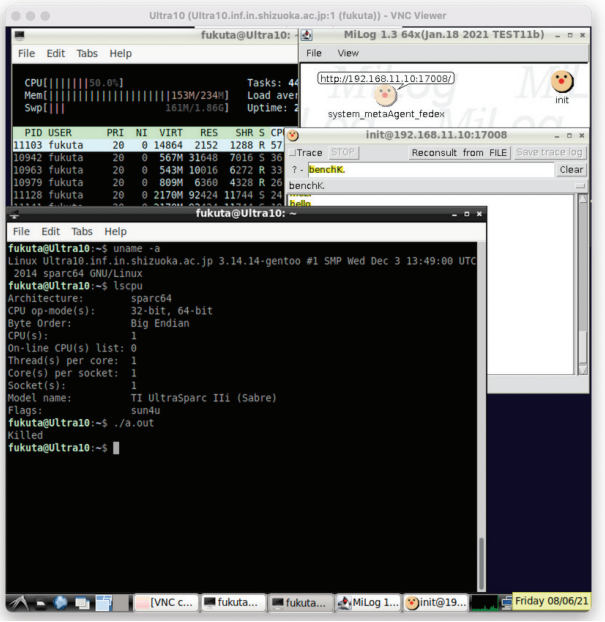

図 2 Sun Ultra10(1998 年発売) 上における 64bit Linux 上での MiLog の動作

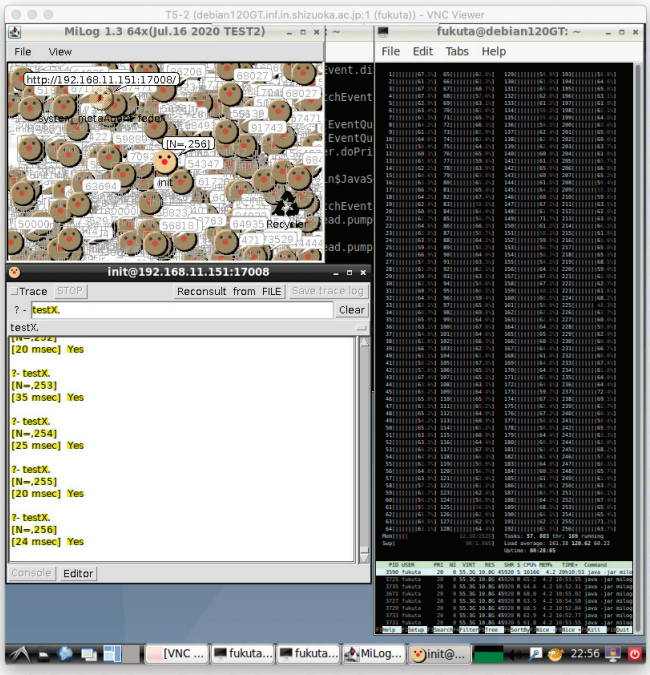

図 3 SPARC T5-2 Server (2017 年まで生産) 上における 64bit Linux 上での MiLog の動作

SPARC T5-2 Server に搭載される CPU である SPARC T5-2 では，動作クロックはおよそ 10 倍 (333MHz - 3.6GHz)，搭載 CPU のソケットあたりのコア数/CPU スレッド数は 16 倍/128 倍となっており，搭載メモリは本稿で使用した機材で 1000 倍 (256MB - 256GB) であり，最大搭載メモリようでもほぼ同様である．このように，ほぼ同一の CPU アーキテクチャが必要な互換性を保ったまま 10 倍から 1000 倍という単位で 20 年程度の期間にスケールする例が実際に

*2 この例では，Octa-core CPU, 64GB メモリを搭載した，本稿執筆時点で購入可能な最新機種の 1 つである 16inch MacBookPro 2019 上で動作する macOS 11 Big Sur をその実行動作環境としている．

*3 ブートローダーのメモリ的制約により，起動時の Linux カーネルのみ Ultra10 では 2014 年時点のものを用いた以外は，Debian SPARC64 port に基づく 64bit SPARC Linux 環境および独自ビルドの OpenJDK12 を用いて，両者でほぼ同一のバイナリを同一のディスクを用いて起動し動作させた．

© 2021 Information Processing Society of Japan

あり，そのような環境で全く同一のバイナリが動作可能であり続けるということを，実際に確認することができた．図 3 での実行例では，SPARC T5-2 上での動作例においてベンチマークプログラムを用いて 256CPU-threads をほぼフルに稼働させることができている．

4. おわりに

本稿執筆時点では，著者にはさらに上位の CPU を搭載した比較可能なモデルを実行環境に用意することはできなかったが，少なくともおよそ 10-1000 倍で性能がスケールした場面における同一のソフトウェアプラットフォームの相互運用を実機で確認することができた．このことから，ソフトウェアプラットフォームの運用時にアーキテクチャの互換性が維持された場合には，10-1000 倍といったオーダーでの性能のスケールが起きることが 20 年程度の時間経過があれば実際にあり，そこで動作可能なソフトウェアプラットフォームに何が起きるのかを検証可能であることがいえる．これは同時に，設計時にそのオーダーの性能のスケールが起きることを想定する必要がある場面が実際にあることも示している．

本稿では，CPU アーキテクチャ等が大きく変わる場面にどのように対処可能なのかについては触れていない．この点についての追加の考察と検討は，今後の課題である．

謝辞 本開発をここまで継続するにあたって，開発初期の著者の学生時代において様々な有益なデザイン上の助言や励ましをいただいた恩師や協力いただいた当時の所属研究室の諸氏に，深く感謝する．また，これまでにこのような継続的な研究を自由に行えるような優れた環境を与えてくれた所属組織およびその構成員らに，心より感謝を述べたい．

参考文献

[1] Fukuta, N.: An Analysis of Long-term Development Process on the Agent System Prototyping Platform MiLog – A Report over 20 Years of Its Development –, *Proc. the 35th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI2021)*, pp. 4N2–IS–3b (2021). (Work in progress session, single-page abstract).

[2] 福田直樹：エージェントシステム試作プラットフォーム MiLog 開発の履歴とその考察，電子情報通信学会技術報告 AI-2020-21(2021-01)，pp. pp.42–45 (2021).

[3] 川村秀憲，大知正直，清　雄一，福田直樹，横山想一郎：2050 年の知能システム，情報処理，Vol. 61, No. 5, pp. 482–483 (2020).

[4] 福田直樹：AI の研究最前線はあとでそれと分かるもの，電子情報通信学会情報・システムソサイエティ誌， Vol. 25, No. 4, pp. 8–9 (2021).

[5] Fukuta, N., Ito, T. and Shintani, T.: MiLog: A Mobile Agent Framework for Implementing Intelligent Information Agents with Logic Programming, *Proc. of the First Pacific Rim International Workshop on Intelligent Information Agents (PRIIA2000)*, pp. 113–123 (2000).

[6] Fukuta, N., Ito, T. and Shintani, T.: A Logic-based Framework for Mobile Intelligent Information Agents, *Poster Proc. of the Tenth International World Wide Web Conference(WWW10)*, pp. 58–59 (2001).

[7] Ito, T., Fukuta, N., Shintani, T. and Sycara, K.: BiddingBot: A Multiagent Support System for Cooperative Bidding in Multiple Auctions, *Proc. of the Fourth International Conference on Multi Agent Systems(ICMAS'2000)*, pp. 399–400 (2000).

[8] Fukuta, N., Mizutani, N., Ozono, T. and Shintani, T.: iML : A Logic based Framework for Construction Graphical User Interface on Mobile Agents, *Lecture Notes in Artificial Intelligence* (Bartenstein, O., Geske, U., Hannebauer, M. and Yoshie, O., eds.), Vol. 2543, Springer-Verlag, pp. 36–50 (2003).

[9] 福田直樹，大圃忠親，新谷虎松：知的モバイルエージェントによる Web ページ構築フレームワーク MiPage の実装，人工知能学会論文誌，Vol. 17, No. 3, pp. 348–353 (2002).

[10] Yamaya, T., Shintani, T., Ozono, T., Hiraoka, Y., Hattori, H., Ito, T., Fukuta, N. and Umemura, K.: MiNet: Building Ad-hoc Peer-to-Peer Networks for information Sharing based on Mobile Agents, *Proc. of 5th International Conference on Practical Aspects on Knowledge Management(PAKM2004)*, pp. 59–70 (2004).

[11] Fukuta, N.: A Mobile Agent Approach for P2P-based Semantic File Retrieval, *Journal of Information Processing*, Vol. 20, No. 3, pp. 607–613 (online), DOI: 10.2197/ipsjjip.20.607 (2012).

[12] 福田直樹：留保価格を導入した大規模複数ユニット組合せオークションの近似価格決定に関する一考察，電子情報通信学会論文誌，Vol. J98-D, No. 6, pp. 948–961（オンライン），DOI: 10.14923/transinfj.2014SWP0026 (2015).

[13] Fukuta, N.: An Approach to VCG-like Approximate Allocation and Pricing for Large-scale Multi-unit Combinatorial Auctions, *Journal of Information Processing*, Vol. 21, No. 1, pp. 9–15 (online), DOI: 10.2197/ipsjjip.21.9 (2013).

[14] Fukuta, N. and Ito, T.: An Experimental Analysis of Biased Parallel Greedy Approximation for Combinatorial Auctions, *International Journal of Intelligent Information and Database Systems*, Vol. 4, No. 5, pp. 487–508 (online), DOI: 10.1504/IJIIDS.2010.035773 (2010).

[15] Fukuta, N. and Ito, T.: Fine-grained Efficient Resource Allocation Using Approximated Combinatorial Auctions–A Parallel Greedy Winner Approximation for Large-scale Problems, *Web Intelligence and Agent Systems: An International Journal*, Vol. 7, No. 1, pp. 43–63 (2009).

[16] 福田直樹，伊藤孝行：短時間再割り当てを考慮した組み合わせオークション勝者決定の高速近似手法，コンピュータソフトウェア (日本ソフトウェア科学会論文誌)，Vol. 25, No. 4, pp. 208–225 (2008).

[17] 福田直樹，伊藤孝行：組み合わせオークションにおける多数入札時での勝者決定の近似解法に関する一考察，電子情報通信学会論文誌，Vol. 90-D, No. 9, pp. 2324–2335（オンライン），入手先 ⟨http://hdl.handle.net/10297/2080⟩ (2007).

[18] Fukuta, N. and Ito, T.: Towards Better Approximation of Winner Determination for Combinatorial Auctions with Large Number of Bids, *Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology(IAT2006)*, pp. 618–621 (2006).

© 2021 Information Processing Society of Japan