ART-Linuxの複数コア利用機能による ソフトウエアディペンダビリティー向上

○加賀美 聡, 石綿 陽一, 西脇 光一, 梶田 秀司, 金広 文男, 尹 祐根, 安藤 慶昭,
佐々木 洋子, Simon THOMPSON, 松井 俊浩 (産業技術総合研究所，JST CREST)

Software Dependability using ART-Linux Multiple-core Utilization Function

○ S. Kagami, Y. Ishiwata, K. Nishiwaki, S. Kajita, F. Kanehiro, WK. Yoon, N. Ando,
Y. Sasaki, S. Thompson, T. Matsui (AIST, Japan and CREST, JST)

Abstract—This paper describes a software system dependability improvement for robot system such as redundant system, computationally low-cost log acquisition, online system monitoring and realtime function or I/O isolation. Those functions becomes possible using newly implemented ART-Linux multicore utilization functions in AMP and SMP fashion.

1. はじめに

近年、ロボットソフトウエアのオープンソース化が進み、OpenCV, OpenRave, Mobile Robot Programming Toolkit (MRPT) などのライブラリと、これらを統一的に扱うための ROS や RTM などのミドルウエアが開発されてきている。このようなライブラリは基本的に Linux または Windows に向けて開発されてきており、処理能力からみても x86 アーキテクチャのチップ必要とされる場合が多い。

一方でハードウエアとしては CPU は x86 系から組み込みプロセッサに至るまで、マルチプロセッサコア化してきており、ロボットに重要な実時間制御を SMP(Symmetric Multi-Processing) として行うものとして x86 系で Linux に対応したものとして Realtime Preemptive Patch が公開され、例えば Willow Garage の PR2 などで用いられてきている。SMP システムとは、Fig.1(left) に示すように、複数のコアの上に単一の OS が存在し、メモリを共有するシステムのことである。

しかし SMP システムにおける実時間 OS は、単一 CPU における RMS(Rate Monotonic Scheduling) のような確立されたスケジュール手法が存在せず、理論的な完全性が保証できない。このために実時間 SMP システムで動作するソフトウエアは、非実時間の通常の SMP のソフトウエアのように、単一 CPU を対象としたプログラミングとの互換性や移植性が一般的に低い。また全体として単一システムとして動作するために、開発時にまたは運用時の想定外のソフトウエアの挙動によりシステムの挙動が設計時に期待したものとは異なったり、OS を含めたシステム全体がストップしたりする危険性を常に含んでいる。

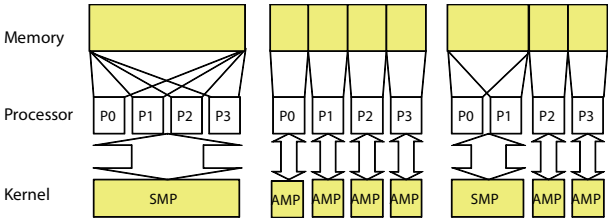

Fig.1 SMP sytem(left), AMP system(center) and SMP&AMP mixed system

実際に実世界で自律的に活動するロボットを実現するためには、多数の入出力、認識・計画・制御を行うさまざまな実行周期の実時間タスク、説明責任やデバッグのためのログ機能、非常用の安全システムなど、複雑な実時間処理システムを実現する必要があり、複数コアを SMP によって利用する際に、これらの全てのリソース配分を完全に設計することが困難になりつつある。

このような問題を解決するための方策として、AMP(Asymmetric Multi-Processing) により複数のコアにそれぞれ別々の実時間 OS を起動するシステムや、AMP と SMP 混在のシステムが考えられる。AMP システムとは、Fig.1(center) に示すように、複数のコアの上にそれぞれ別の OS が存在し、メモリも独立に用いるシステムのことである。また AMP と SMP の混在システムは、Fig.1(right) に示すように、任意の個数のプロセッサを SMP で、残りを別々に AMP システムが利用する物である。

これまで VxWorks[1], QNX[2] などの x86 系を対象とした組み込み専用の実時間 OS において、このような AMP と SMP 混在可能なシステムが実現されてき

SY0014/11/0000 - 0728

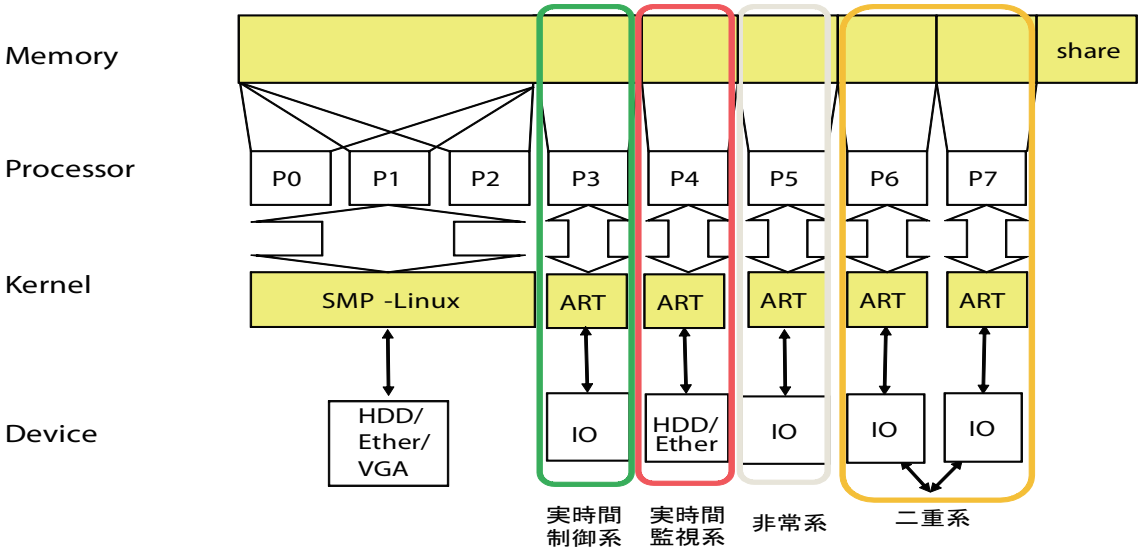

Fig.2 Proposed SMP& AMP mixtured Sytem

ているが、残念ながらこのような組み込みシステムでは、オープンソースのロボットソフトウエアを動作させるのには困難が伴う場合がある。一方で Linux で実時間な AMP と SMP 混在可能なシステムは存在していなかった。

そこで今回 JST CREST プログラム「実時間並列ディペンダブル OS とその分散ネットワークの研究」において、これまで 10 年以上に渡って開発を続けてきた Linux ベースの実時間 OS である ART-Linux を、複数の実時間 AMP と通常の非実時間 SMP の Linux が任意の割合で混在可能な OS を設計し、開発を行った。本論文はこの複数の実時間 AMP と非実時間 SMP の Linux が混在させることによって可能になるシステムの二重化、ログ作成、オンラインモニタリング、実時間処理部や実時間 I/O のアイソレーションなどのディペンダビリティー機能について考察する。

2. 実時間 OS: ART-Linux の変遷

著者らは, Intel の x86 アーキテクチャ用の Linux カーネル 2.0 に対応した実時間 OS である ART-Linux 2.0 を 98 年に設計・開発し [3, 4]、その後 00 年に Linux カーネル 2.2 に対応した ART-Linux 2.2[5]、 04 年に Linux カーネル 2.4 に対応した ART-Linux 2.4、 07 年にカーネル 2.6 に対応した ART-Linux 2.6[6] を設計・開発してきた。また 2004 年からソフトウエアを Sourceforge よりオープンソースとして公開してきている [7]。

これらの ART-Linux は、ユーザー空間から利用可能なシンプルな実時間システムコール `art_enter()`, `art_wait()` を持ち、小さなジッターで短い周期実行が実現できる特徴を持ち、ヒューマノイドロボット HRP-2[8] に採用されるなど、知能ロボット研究の発展に寄与してきた。

しかしこれまでの ART-Linux は単一 CPU にしか対応していなかった。そこでこの ART-Linux 2.6 をベースに、非実時間の SMP と実時間の AMP を任意に実現可能なシステムを設計し、開発を行った。

開発した ART-Linux は主として下記のような特徴を持つ。

- 各システムのメモリ配分はシステム起動時に任意にアサインでき、アサインしたメモリに AMP と SMP システムをリロケートとしてブート可能である。
- システム毎に仮想ネットワークデバイスと仮想シリアルコンソールを準備し、システムの外部との通信や、システムの OS のシリアルデバッグを可能としている。
- 各システムは独立のルートファイルシステムを用いてブートするため、各システムのカーネルおよびファイルシステムの最適化を図ることができる。仮想ネットワークデバイスを利用した NFS ブートも可能である。
- 各システムは共有のクロックを持つが、独立に動作し、別システムの挙動により実時間性を阻害されない。
- Fig.2 の右上の share に示すように、メモリの一部を共有領域として、どのプロセッサからも見える領域を準備し、この領域を通じて、システム間で高速に通信が可能な仕組みを導入している。
- デバイスを任意のシステムに振り分けが可能である。

本論文の原稿作成時には、システムは内部ユーザおよび協力ユーザーにおいてテスト中であり、検証が終わり次第、公開を予定している。

3. 実時間 AMP と非実時間 SMP の混在したロボットシステムの構成法

本節では実時間 AMP と非実時間 SMP の混在により、単一 CPU での実時間システムやマルチコア環境での実時間 SMP システムに比較して、どのようなシステムのディペンダビリティーの向上が可能になるかを論じる。

Fig.2 に、開発した ART-Linux により可能となるシステムの構成例を示している。この図では現在利用可能な CPU の例として 8 個のコアがあるものを挙げ、そのうちの 3 個で通常の非実時間な Linux が動作し、また残りの 5 つのコアに対してそれぞれ独立に実時間 OS である ART-Linux が動作している例を示している。

以下では、Fig.2 を例として、各システムの概要を述べる。

3·1 非実時間 SMP-Linux

Fig.2 では、P0〜P2 のプロセッサにアサインされた非実時間の SMP システムとして示している。図中に示すように、SMP の Linux にはデバイスとして通常のディスクやグラフィックス、ネットワークなどをアサインしている。このシステムの役割としては下記のようなものが考えられる。

- ユーザーインターフェース等。
- グラフィックハードウエア、イーサネットを始めとしてロボットの直接制御に用いない IO を行うハードウエアの処理。これらは多数の割り込みを発生させることから実時間システムにとっての障害となりやすい。
- 長期のプランニング、地図作成、モデルマッチングによる認識処理などの、ロボットの知能処理のうちでも非実時間の処理を行う。これらの結果は、仮想ネットワークあるいは共有メモリを通じて、実時間制御系などに伝える。
- 実時間用途に開発されていない多くのオープンソースライブラリの利用。

本システムはメモリアサインメント部を除けば、ほぼ通常の SMP-Linux であるために、通常の Linux のために開発されたものが、改変なしにそのまま動作するという特徴がある。

3·2 実時間制御系

Fig.2 では、P3 のプロセッサにアサインされた AMP 実時間システムとして示している。制御のために必要な IO ボードをアサインしている。本システムは他のオーバーヘッドがなく、制御のみに専念することが出来るために、低いジッターによる高い実時間制御性能や、リソース配分の簡単化によるシステム設計の容易さと障害の起きにくさが期待できる。

次に述べる実時間監視系のために、内部状態を共有メモリに書き出すことによりシステムの監視をオンラインで行うことができる。

3·3 実時間監視系

Fig.2 では、P4 のプロセッサにアサインされた AMP 実時間システムとして示している。この実時間監視系はログを保存するためのディスクや外部のシステムに警告するための別のネットワークデバイスをアサインしている。

本システムは他の SMP&AMP システムの内部状態を共有メモリを通じて監視し（他のシステムは自分で共有メモリに状態を書き出す必要がある）、ログを残すとともに、他のシステムの異常をリアルタイムで検知し、後に述べる非常系に知らせるなどの機能を果たす。

3·4 非常系

Fig.2 では、P5 のプロセッサにアサインされた AMP 実時間システムとして示している。非常系も独自の IO システムをアサインしている。前述の実時間監視系を通じて、あるいは共有メモリや仮想ネットワークを通じた他システムからの通知や他システムの監視、あるいは IO を介して得られるセンサの値などから、システムの異常を検知し、モーター電源断を始めとする緊急停止などの安全策を行う。

このシステムを他のシステムから独立にすることにより、非常システムが他の実時間システムに阻害されて動作しないなどの影響を避けることができる。また一方で、例えばヒューマノイドロボットの歩行、また高速で走行している車、アームで人の上に重量物を持ちあげている、などのいきなり電源断すると被害が大きくなるような状況では、その状況を回避するための計算機能と能力を有している必要がある。そのような状況に備えて、実時間計算能力を他のシステムから独立に温存することが、本システムの目的である。

3·5 二重系

複雑なソフトウエアの論理検証等は未だに部分的に（あるいはサンプリングベースにしか）行えないためにシステムの信頼性を確保する方法として二重系による方法が良く用いられている。

ここでは本システムによる二重系の構成法について述べる。Fig.2 では、P6〜P7 のプロセッサにアサインされた AMP 実時間システムとして示している。２つのシステムは、それぞれ独立に IO ボードをアサインす

ることも、IO ボードへのアクセスを排他的に制御することも可能であり、柔軟な二重系を構成することが可能である。(ただし排他的アクセスには IO ボードの制限も存在する)

ロボットのシステムが複雑になり、実世界において安全に行動するためには、二重系や多重系による信頼性の確保が重要になると考えている。

この機能を利用する例として、LTTng を利用し、システムに優先度逆転が発生していないかどうかを Time Series Theory によりオンラインで監視する手法を実現した [9]。

4. おわりに

本論文では、ロボットのソフトウエアシステムの構築において、単一 CPU での実時間システムや、複数コアを利用した実時間 SMP システムにおいて本質的に存在する問題点について述べたのち、これらの欠点を乗り越えるためのアイデアとして、複数の実時間 AMP と通常の非実時間 SMP の Linux が任意の割合で混在させるシステムの構成法について述べた。またロボットシステムの構成法の例を挙げて、既存のシステムと比較して、重要な実時間の処理は必要に応じて完全にアイソレーション可能なシステムの利点を述べた。

現在、JST CREST プログラム「実時間並列ディペンダブル OS とその分散ネットワークの研究」において、提案する複数の実時間 AMP と通常の非実時間 SMP の Linux が任意の割合で混在可能な OS を設計し、開発を行った。

現状では linux 2.6.32〜36 のカーネルと Ubuntu10.04LTS, Debian GNU/Linux 5.0 のディストリビューションに対応した開発を行っており、svn へのコミット総数は 2869 回である。

最大では DELL の T5500 (Intel Xeon E5504 2GHz 4core Dual, memory 12GB) および M6500 (Intel Core i7 X940 2.13GHz 4core を HT で利用, memory 8GB) の 2 つのシステムで、AMP システムが 8 つまたは SMP で 4 つと AMP システムが 4 つの動作を確認している。

現在は内部および外部ユーザーにおけるテスト段階にあり、ヒューマノイドロボット HRP2, 車輪型ロボット Pen2, Segway などで実験を行っており、検証が終わり次第公開を予定している。

参考文献

1) Wind River. *VxWorks*. http://www.windriver.com/vxworks.
2) QNX Software Systems Ltd. *QNX Realtime Operating System*. http://www.qnx.com.
3) 石綿陽一, 松井俊浩, 國吉康夫. 高度な実時間処理機能を持つ Linux の開発. 第 16 回日本ロボット学会学術講演会予稿集, pp. 355–356, 1998.
4) 石綿陽一, 松井俊浩. 汎用 OS とデバイスドライバを共有できる実時間オペレーティングシステム. 信学技報, CPSY 97-119, pp. 41–48, 1998.
5) 石綿陽一. SMP カーネルに基づく ART-Linux の安定化と実時間処理性能の測定. 第 3 回計測自動制御学会システムインテグレーション部門講演会講演論文集, 神戸, 12 2002.
6) 石綿陽一, 加賀美聡, 西脇光一, 松井俊浩. シングル CPU 用 ART-Linux 2.6 の設計と開発. 日本ロボット学会誌, Vol. 26, No. 6, pp. 546–552, 9 2008.
7) *ART-Linux*. http://www.dh.aist.go.jp/jp/research/assist/ART-Linux/, http://sourceforge.net/projects/art-linux/.
8) Kenji Kaneko, Fumio Kanehiro, Shuuji Kajita, Hirohisa Hirukawa, Toshikazu Kawasaki, Masaru Hirata, Kazuhiko Akachi, and Takakatsu Isozumi. Humanoid Robot HRP-2. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA2004)*, pp. 1083–1090, 2004.
9) Midori Sugaya, Ken Igarashi, Masaaki Goshima, Shinpei Nakata, Kimio Kuramitsu, Yoichi Ishiwata, and Satoshi Kagami. Extensible Online Log Analysis System for Improving Adaptation Cycles. In *13th European Workshop on Dependable Computing*, Pisa, Italy, 5 2011.