「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」

H22 年度
実績報告

平成 18 年度採択研究代表者

中島　達夫

早稲田大学　理工学術院　教授

高機能情報家電のためのディペンダブルオペレーティングシステム

§1. 研究実施の概要

本研究チームでは，DEOS フレームワーク D-fops の内の D-Visor と D-System Monitor を担当している．本報告書では，平成22年度の取り組みに関して報告する．中島チームでは，3つの研究課題に関して取り組んでいる．1つ目は，SPUMONE という仮想化層，2つ目は，モニタリングシステム，3つ目は，仮想化層向けの検証システムである．

SPUMONE 仮想化層に関しては，本年度は，2つの課題に取り組んだ，1つ目の課題は，組込みシステムにおける利用で必要不可欠となるリアルタイム性の改善である．2つ目の課題は，仮想化層の保護である．これにより，組込みシステムに適した仮想化層の実現が可能となる．モニタリングサービスに関しては主にモニタリングサービスとゲスト OS 間の同期問題に関して取り組んだ．最後の，仮想化層のための検証技術に関しては，いくつかのケーススタディを通して仮想化層の検証が可能なことを示した．

§2. 研究実施体制

(1)「早稲田大学」グループ

①研究分担グループ長：中島　達夫（早稲田大学理工学術院、教授）

②研究項目

- 組込みシステム向け D-Visor： SPUMONE の開発
- D-System Monitor としての整合性モニタリングに関する開発

(2)「筑波大学」グループ

①研究分担グループ長：追川　修一　（筑波大学システム情報工学研究科、准教授）

②研究項目

- D-System Monitor 実行環境 API の開発
- D-Visor のための検証技術の開発

§3. 研究実施内容

組込みシステム向け D-Visor としての SPUMONE の実装:

D-Visor の1つである SPUMONE は情報家電等の組込みシステムに適した仮想化層である. 現在, D-fops のプロトタイプシステムが利用している KVM は PC やサーバ向けの仮想化層であり, インテル x86 アーキテクチャが提供する仮想化支援のためのハードウエアを前提としているので, それ以外の組込みシステムが利用しているプロセッサに KVM を移植することは困難である. そのため, 本チームではより組込みシステムに適した仮想化層の開発をおこなっている. SPUMONE は, 組込みシステムにおける使用に適しており, 複数のオペレーティングシステムをマルチコアプロセッサ上で動作することを可能とする. 仮想コアと物理コアのマッピングを動的におこなうことを可能とすることにより, リソースの有効利用を可能とする.

D-Visor として SPUMONE を利用するため, 時間隔離と空間隔離の提供が必要不可欠である. しかし, 従来提案されている空間隔離方式は, オーバヘッドが大きいため多くの組込みシステムにおける利用には適していなかった. また, 時間隔離の提供のためには, リアルタイムスケジューラの提供だけではなく, 割り込み遅延の低減が重要となる. 本年度は, 仮想化層の信頼性を向上するための新しい仮想化層アーキテクチャの提案と実装, 割り込み遅延を低下させるための新しい方式の提案をおこなった[2,6,7,8, 9, 11, 13, 14].

時間隔離の実現:

SPUMONE の大きな特徴はゲスト OS の変更を最小限にして, 低オーバヘッドの仮想化を実現することである. しかし, 従来の割り込み遅延を改善する手法はカーネル内の割り込みイネーブル／ディスエーブル命令を置き換える必要があり, コードの変更量を増大させていた.

組込みシステムの RTOS 上では, 性能を重視してプログラミングをおこなっていたため, アプリケーションが直接割り込み禁止している場合もあった. そのため, リアルタイムアプリケーションの変更なしに割り込み遅延の応答性を改善することは極めて困難であった.

本年度の研究により開発した手法は, マルチコアプロセッサの機能を利用することにより, 割り込み遅延の改善をおこなう. 以下の左の図に示すように, RTOS と Linux が物理コアを共有する状況を考える. RTOS の CPU リソース使用率が低い場合, 資源の有効利用のため, 複数の OS が物理コアを共有することは必要不可欠である. しかし, Linux が割り込み禁止命令を実行することにより, RTOS の遅延が大幅に大きくなる可能性がある. 提案する方式では, Linux が使用する仮想コアを右の図が示すように, 別なコアにマイグレーションすることにより, 割り込み遅延を改善する. マイグレーションのコストは 50μs 程度であり, マイグレーションの導入によるオーバヘッドはシステムの性能にほとんど影響を与えない. 割り込み遅延を低下させる原因は, Linux カーネルが

実行する割り込み禁止命令である. 割り込み禁止命令は Linux カーネル内でのみ呼び出されるため, Linuxカーネルを呼び出すシステムコールを実行中の時だけ, RTOSとLinuxを同一の物理コア上で動作させないことにより, 割り込み遅延が発生しないことになる. そのため, SPUMONE 内に仮想コアマイグレーション機能を実装した. また, 本方式を RP1 マルチコアプロセッサ上で評価をおこなった結果, 今まで数十ミリ秒であった割り込み遅延の最悪値が 30 マイクロ秒にまで低下することが明らかになった.

仮想コアマイグレーション機能は, より効率が高い時間隔離を実現する可能性がある. 特に, リアルタイム性を犠牲にせずに消費電力を大幅に改善する可能性を提供する. 本研究テーマに関しては, よりリソース効率を向上する(特に消費電力を改善する)ために仮想コアマイグレーションを有効に利用する仮想コアスケジューリング法に関して来年度に検討をおこなう.

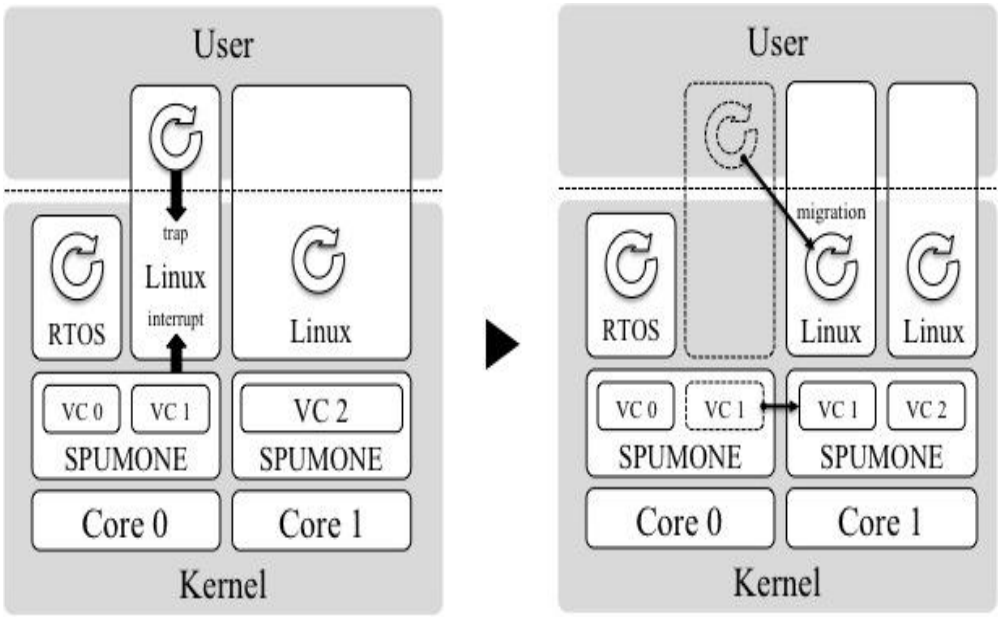

図1: 仮想コアマイグレーションを利用した時間隔離方式

空間隔離のための仮想化アーキテクチャ:

SPUMONE は組込みシステムが要求するオーバヘッドの最小化とゲスト OS の変更を最小限とするため, ゲスト OS カーネルと仮想化層を同一の特権レベルにて実行している. そのため, Linux 等のゲスト OS カーネルに外部から悪意があるプログラムが挿入されると, 容易に仮想化層自体を攻撃することによりシステムを崩壊することが可能となる. 従来のアプローチでは, 仮想化層とゲスト OS を異なるアドレススペースに置き, ゲスト OS が仮想化層を呼び出す時は, 特別な命令を用いることが一般的であった. しかし, このアプローチは, 特に時間隔離の実現の面で, 困難さを提供する. また, オーバヘッドが大きく, 多くのリソース制約や消費電力を考慮する必要がある組込みシステムには適していない. SPUMONE が提供する空間隔離方式は, マルチコアプロセッ

サが提供するローカルメモリを利用する.

マルチコアプロセッサ向けの SPUMONE は, プロセッサ毎に異なる仮想化層のインスタンスを生成する. つまり, 各コアを分散システムのノードと見なし, 各仮想化層のインスタンス間ではデータ構造の共有をおこなわない. そのため, コアの数が増えても共有データを保護するための同期がオーバヘッドになることはなく, コア数に無関係にシステムをスケールアップすることが可能となる. この特性は, 今後のメニーコア時代に向けても好ましい特性である. この特性は, スケーラビリティだけではなく, 信頼性を向上するためにも極めて有効である. 下図に示すように, SPUMONE は各仮想化層のインスタンスを各コアが提供するローカルメモリ内に配置する. ローカルメモリは他のコアからはアクセスすることが出来ないため, 他のコア上で実行される仮想化層に影響を与えることが出来なくなる. つまり, Linux カーネルからは RTOS を動作させるために使用されている SPUMONE をアクセスすることが出来なくなる. つまり, SPUMONE を攻撃することで RTOS の動作に影響を与えることはできなくなり, それがシステム全体の信頼性を向上することを可能とする. 本研究では, 提案する方式を RP1 マルチコアプロセッサ上に実装し, オーバヘッドなく提案する手法が実現出来ることを示した.

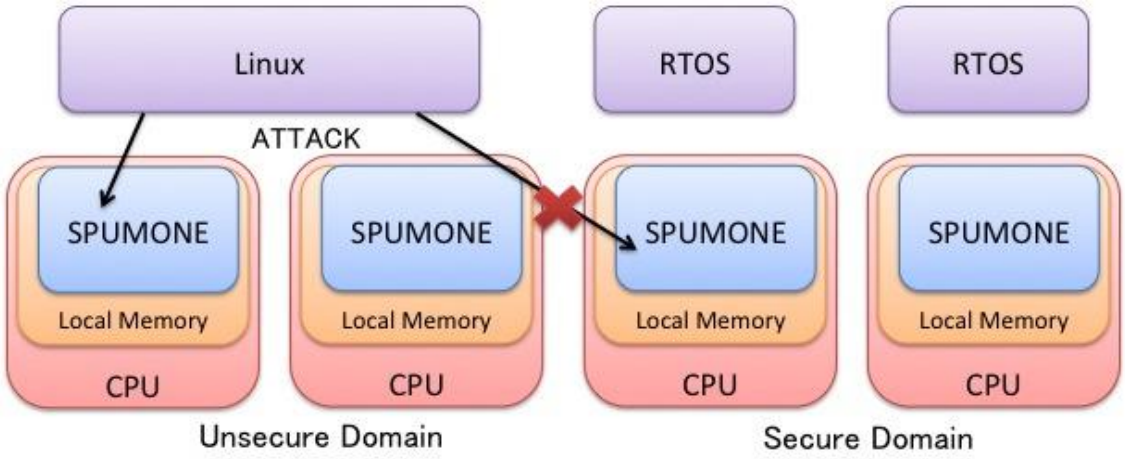

図2: ローカルメモリを利用した仮想化層の実装

D-System Monitor としてのモニタリングシステム:

モニタリングシステムはカーネル内のデータ構造の整合性をモニタリングすることにより, OS カーネル全体の整合性を保証する. 現在のモニタリングシステムは, キーとなるカーネルデータ構造毎に整合性をチェックするための関数を記述する. モニタリングシステムはそれらの関数を定期的に呼び出すことにより, カーネルの整合性が崩壊していないことを保証する. カーネルの整合性はアプリケーションの動作の正しさを保証するための前提であり, D-Case のエビデンスとして保証する必要があるプロパティである. 昨年までの研究では, モニタリングシステムにプロセスデータ構造の整合性監視をおこなわせることにより, カーネル rootkit の検出が可能であることを示した[10].

現状のモニタリングシステムの問題点は, モニタリングシステムと並行に OS カーネルがデータ構

造を変更するため，モニタリングシステムの整合性チェックが間違って整合性が失われたと報告することである．一般的には，ロック等の同期機構を利用することにより，以上の問題を解決するが，モニタリングシステムのためにOSカーネルを変更することはできない．そのため，モニタリングシステムはロックを利用しない楽観的同期手法を利用する[4]．本方式では，モニタリングシステムが整合性チェックをおこなっている最中にゲストOSがシステムコールを発行した場合，整合性チェックをはじめからやり直す．そのため，データ構造の同時アクセスにより一貫性が取れていない状態でデータ構造をアクセスすることがなくなるため，モニタリングシステムが間違って整合性が崩壊したと報告する確率を大きく改善することが可能となる．

モニタリングシステムを攻撃されたゲストOSから保護することは，モニタリングシステムを正常に動作させるために必要不可欠である．本研究では，マルチコアプロセッサが提供するローカルメモリを利用することにより，モニタリングシステムを保護する．つまり，下図に示すように，モニタリングシステムはLinuxを動作させる物理コアと異なるコア上で動作させる．そのため，Linuxカーネルを攻撃する悪意があるプログラムが挿入された場合でも，Linuxカーネルがモニタリングシステムを改変することは不可能となる．

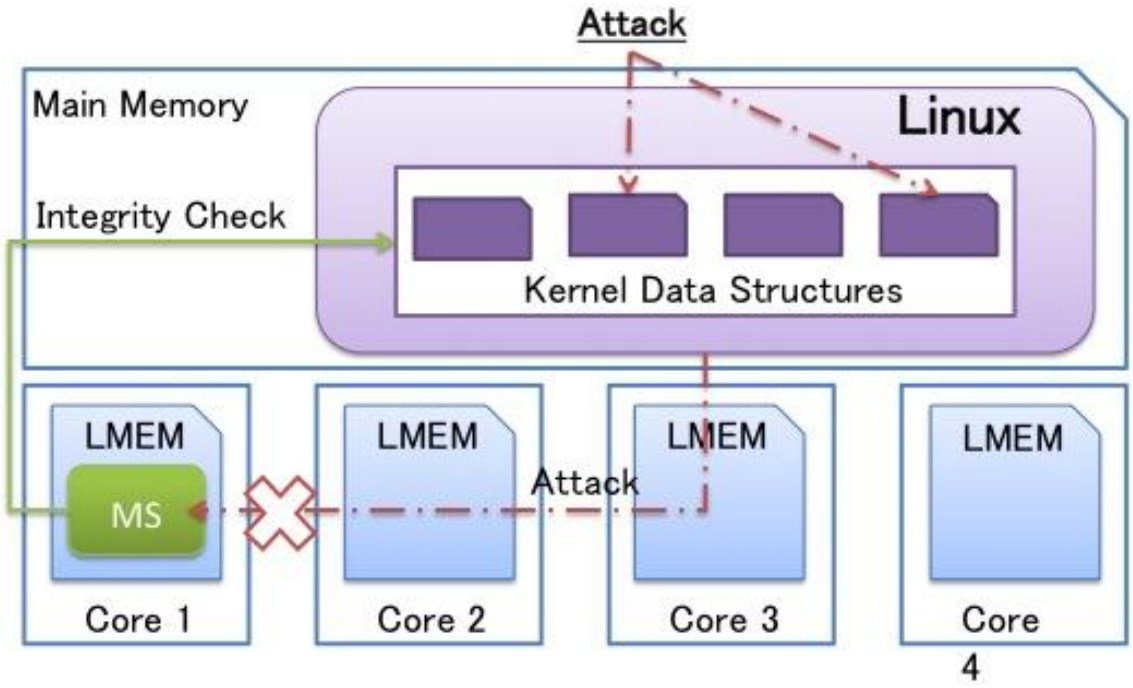

図3： モニタリングシステムアーキテクチャ

D-System Monitor 実行環境 API：

D-System Monitorとしては、異なる脅威に対応するために、OSカーネルの異なる部分を検査対象とする、複数種類の実装が出てくることになる。D-System Monitorを使用するシステムは、そのシステムが立ち向かう必要のある脅威に対応したD-System Monitorの実装を選択し、使用

できることが求められる。従って、複数の D-System Monitor 実装が同時に使用可能でなければならない。さらに、システムの要求により、使用する D-Visor も変わってくるが、D-Visor ごとに D-System Monitor を実装し直すことは避けなければならない。そこで、D-Visor からは独立した、目的の異なる複数の D-System Monitor を同時に動作可能にする D-System Monitor の実行環境を定義する API を策定している。

本年度は、D-System Monitor として、河野チームが開発している OS カーネルの振る舞いからキーロガーを検出する機能、およびルートキットを検出する機能について調査を行い、上記のモニタリングシステムも含め、全て同時に動作可能にできるかどうか、またどのような API が必要になるか検討を行った。その結果、河野チームが開発している機能については I/O のモニタリングが必要であり、またモニタリングシステムは OS カーネル内のメモリをアクセスする機能が必要であることがわかった。さらに、I/O のモニタリングについては、異なる機能が同一の I/O デバイスをモニタリングする場合にも対応する必要があることもわかった。そこで、D-System Monitor 実行環境 API としては、OS カーネル内のメモリアクセス、I/O デバイスのモニタリング機能を提供する必要がある。OS カーネル内のメモリアクセスとしては、既に Xen 用に定義されている XenAccess を用いることが検討されている。I/O デバイスのモニタリング機能は XenAccess は定義していないため、同一デバイスのモニタリングを可能にするコールバック形式の API を策定中である。

D-Visor のための仮想化層検証技術:

D-Visor は、D-fops が必要とするシステムの論理分割を行い System Container を実現するため、アイソレーション機能の提供が必要不可欠となっている。そこで、D-Visor のための仮想化層検証技術は、DEOS プロジェクトで研究開発されているモデルチェッカ（DEOS モデルチェッカ）を用いた静的検証により、D-Visor のアイソレーション機能を検証可能にすることを目的としている。本年度は、ページフォルト時におけるシャドウページテーブルのアップデート処理、シリアルラインデバイスに対する操作に着目し、ソースコードに仕様記述を追加し、それぞれの妥当性を静的に検証した。これらにより、D-Visor およびハードウェアデバイスの空間的アイソレーションが検証できることを示した。

シャドウページテーブルとは VMM で管理する、ゲスト OS のページテーブル（ゲストページテーブル）のコピーであり、プロセッサが仮想から物理へのアドレス変換のために使用するのはシャドウページテーブルである。ゲスト OS は自身のメモリ領域内に確保したゲストページテーブルを操作し、VMM はシャドウページテーブルを操作する。2 つのページテーブルの一貫性を保つための 1 つの契機となるのが、ページフォルトである。ページフォルト発生時に、VMM はゲストページテーブルにおけるページフォルトが発生したアドレスのエントリを参照し、シャドウページテーブルの対応するエントリにコピーする（シャドウページテーブルのアップデート）。この時、シャドウページテーブルに含まれる VMM のエントリを破壊してしまうと、アイソレーションが行われず、システム全体の障害要因となってしまう。また、アップデートするエントリのページテーブル内のインデックスも、有

効な範囲内である必要がある。これらの 2 点について仕様記述を行い、操作の妥当性を検証した。また、VMM のエントリを破壊してしまう可能性のあるソースコードでは、モデル検査器が不正であると報告することも確認した。

```
page_addr_t cr2 = read_cr2();
/*@
 $assumes ((cr2.l2 >= 0) && (cr2.l1 >= 0) && (cr2.offset >= 0) &&
 (cr2.l1 < NUM_PTE) && (cr2.offset < NUM_PAGE_OFFSET));
 */
if (cr2.l2 < VMM_TOP_PDE)              // protects VMM from an update.
     update_guest_paging_at(&cr2);
else
     panic("CR2 out of range");
```

図4： VMM におけるシャドウページテーブルアップデートの妥当性チェック例

2 つめの検証例は、ゲスト OS がデバイスへの操作を正しい順序で行っていることを VMM がチェックし、そのチェックからデバイスへの操作順序が不正である場合に、デバイスへの操作を中断し、ゲスト OS への通知が行われることを保証するものである。これにより、不正な操作によりデバイスが異常状態になることからのアイソレーションが可能になる。このチェックが確実なものであることを検証により保証することで、特にゲスト OS がパススルーによりデバイスを操作する場合に、ゲスト OS のバグまたは悪意のある操作によりデバイスを使用不可能な状態にすることを防ぐことができる。このための仕様記述を、シリアルラインの入出力に対して行い、確実にゲスト OS への通知が行われることを検証した。

Obtain a flag that indicate if the buffer is empty.

```
/*@
 $ensures((ser_tx_ready == 1) ||
          (ser_tx_ready == 0));
*/
int ser_is_ready_to_send(void)
{
     int thre = hw_uart_lsr_thre();

     if (thre)
          ser_tx_ready = 1;
     else
          ser_tx_ready = 0;

     return thre;
}
```

Send a character if the buffer is empty, or norify the VM if not.

```
/*@
 $ensures($old(ser_tx_ready) == 0
          ==> need_to_kill_vm == 1);
*/
void ser_send(int ch)
{
     if (ser_tx_ready == 0)
          need_to_kill_vm = 1;

     ser_tx_ready = 0;
     hw_uart_tx(ch);
}
```

図5： シリアルライン操作の妥当性チェック例

§4. 成果発表等

（4－1） 原著論文発表

●論文詳細情報

(1) Ki Duk Kwon, Midori Sugaya, Tatsuo Nakajima, “KTAS: Analysis of Timer Latency for Embedded Linux Kernel”, International Journal of Advenced Science and Technology(IJAST), Vol. 18, 2010.

(2) Yuki Kinebuchi, Kazuo Makijima, Takushi Morita, Alexandre Courbot, and Tatsuo Nakajima, "Composition Kernel: A Software Solution for Constructing a Multi-OS Embedded System," EURASIP Journal on Embedded Systems, vol. 2010, Article ID 458085, 8 pages, 2010. DOI:10.1155/2010/458085.

(3) 菅谷 みどり 中島 達夫, ``情報家電向け Linux の CPU 資源制限システムの設計と実装", 電子情報通信学会論文誌 D, Volume J93-D No.10 , Oct. pp.2001-2010, (2010)

(4) H. Shimada, A. Courbot, Y. Kinebuchi, T. Nakajima, "A Lightweight Monitoring Service for Multi-Core Embedded Systems", In Proceedings of the 13th Symposium on Object-Oriented Real-Time Distributed Computing, 2010, DOI: 10.1109/ISORC.2010.12

(5) Yuki Ohno, Sayaka Akioka, Midori Sugaya, Tatsuo Nakajima, "A Library-Based Tool to Improve CPU Assignment for Multicore Processor-Based Pervasive Servers," In Proceedings of IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications, pp.114-123, IEEE Press, (2010), DOI: 10.1109/RTCSA.2010.21

(6) Tatsuo Nakajima Yuki Kinebuchi Alexandro. Courbot Hiromasa Shimada T-H. Lin Hitoshi Mitake, "Composition Kernel: A Multi-core Processor Virtualization Layer for Rich Functional Smart Products", In Proceeding of The 8th IFIP Software Technologies for Future Embedded and Ubiquitous Systems(SEUS2010), pp. 227-238, LNCS 6399 Springer, (2010), DOI: 10.1007/978-3-642-16256-5_22.

(7) Tatsuo Nakajima, Yuki Kinebuchi, Hiromasa Shimada, Alexandre Courbot, Tsung-Han Lin, Temporal and Spatial Isolation in a Virtualization Layer for Multi-core Processor based Information Appliances, 16th Asia and South Pacific Design Automation Conference, pp.645-652, 2011, DOI:10.1109/ASPDAC.2011.5722268

(8) Shuichi Oikawa, Jin Kawasaki: Simultaneous Logging and Replay for Recording Evidences of System Failures. In Proceedings of the 8th IFIP Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2010), Springer-Verlag LNCS 6399, pp. 143-154, 2010. (DOI:

10.1007/978-3-642-16256-5_15)
(9) Jin Kawasaki, Shuichi Oikawa: Co-Locating Virtual Machine Logging and Replay for Recording System Failures. In Proceedings of the 10th IEEE International Conference on Computer and Information Technology, pp.1352-1357, 2010. (DOI: 10.1109/CIT.2010.242)
(10) Sun Lei, Tatsuo Nakajima, "Online Self-Diagnosis Self-Recovery Infrastructure for Embedded Systems", International Journal of Security and Its Applications, Vol. 4, No. 4, 2010.
(11) Hitoshi Mitake, Yuki Kinebuchi, Alexandre Courbot, Tatsuo Nakajima, Coexisting Real-Time OS and General Purpose OS on an Embedded Virtualization Layer for a Multicore Processor, In Proceedings of the 26th ACM Symposium On Applied Computing, pp.629-630, 2011.
(12) Hiromasa Shimada, Yuki Kinebuchi, Tsung-Han Lin, Alexandre Courbot, Tatsuo Nakajima, Design Issues in Composition Kernels for Highly Functional Embedded Systems, In Proceedings of the 26th ACM Symposium On Applied Computing, pp.338-345, 2011.
(13) Tsung-Han Lin, Yuki Kinebuchi, Alexandre Courbot, Hiromasa Shimada, Hitoshi Mitake, Chen-Yi Lee, Tatsuo Nakajima, Hardware-assisted Reliability Enhancement for Embedded Multicore Virtualization Design, In Proceedings of the 14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, pp.241-249, 2011. DOI: 10.1109/ISORC.2011.37
(14) Yuki Kinebuchi, Hitoshi Mitake, Yohei Yasukawa, Takushi Morita, Alexandre Courbot, Tatsuo Nakajima. A Study on Real-time Responsiveness on Virtualization Based Multi-OS Embedded Systems. In Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems, pp.369-378, 2011.

(4－2) 知財出願

① 平成22年度特許出願件数(国内 0 件)
② CREST 研究期間累積件数(国内 0 件)