Journal of Systems Architecture 169 (2025) 103598

目次はScienceDirectで閲覧可能

Journal of Systems Architecture

ジャーナルホームページ: www.elsevier.com/locate/sysarc

安全重要システム向けLinux：概説

Markel Galarraga [a,b,*], Charles-Alexis Lefebvre [a], Jon Perez-Cerrolaza [a], Jose A. Pascual [b]

[a] イケラン技術研究センター、バスク研究技術同盟（BRTA）、アラサテ／モンドラゴン、スペイン

[b] スペイン・バスク大学（UPV/EHU）情報学部、ドノスティア＝サン・セバスティアン

論文　情報

キーワード:
Linux
機能安全
安全上重要なシステム混合重
要度文献レビュー

ABSTRACT

次世代の安全重要システム（自動運転車など）は、高性能コンピューティングデバイス、多様なソフトウェアスタック、機械学習アルゴリズム、異なる安全重要度を持つソフトウェアアプリケーションを統合する、ますます複雑化するシステムである。 産業界と学術界は、組み込みシステムや重要分野（通信、銀行など）での広範な採用実績、およびコンピューティングデバイス・ソフトウェアスタック・機械学習ソフトウェアの幅広いサポートを背景に、これらの安全重要システムにおける汎用OSとしてのLinux活用に関心を高めている。しかし、体系的なエラー低減技術、ランダム故障耐性、時間的・空間的独立性といった安全基準の要件を満たすことは課題となる。 これは特に、異なる安全重要度のソフトウェアアプリケーションを統合する場合（混合重要度）に顕著である。本文献調査では、安全重要システム開発のためのLinuxの提案、分析、拡張に関する研究を検証する。また、これらの研究が焦点を当てる主な課題を特定する。最後に、主要な産業の取り組みの概要も提示する。

目次

* 連絡先著者：イケラン技術研究センター、バスク研究技術連合（BRTA）、アラサテ／モンドラゴン、スペイン。
メールアドレス：mgalarraga@ikerlan.es (M. Galarraga)、calefebvre@ikerlan.es (C.-A. Lefebvre)、jmperez@ikerlan.es (J. Perez-Cerrolaza)、joseantonio.pascual@ehu.eus (J.A. Pascual)。

https://doi.org/10.1016/j.sysarc.2025.103598
受理日：2025年7月22日；改訂版受理日：2025年9月19日；採択日：2025年10月10日
オンライン公開日 2025年10月17日

1. はじめに

近 年 、多様な産業分野において、自律走行車や自動列車運転システムといった先駆的な安全関連システムへの投資が進められている。これらは高性能コンピューティングデバイス（例：マルチコアプロセッサ、 GPU）、多様なソフトウェアスタック（例：ROS、CUDA ）、および異なる安全重要度（すなわち混合重要度システム）を持つ複雑なソフトウェアアプリケーション（例：AIアルゴリズム）を統合している。

しかし、プログラマブル電子システム（電子機器／ハードウェアとソフトウェア）で構成される安全重要システムは、その故障が人間や環境に壊滅的な結果をもたらす可能性があるため、厳格な安全認証基準に準拠して開発される必要がある。そのような基準の例としては、汎用産業用規格IEC 61508[1]や自動車用規格ISO 26262[2]が挙げられる。 したがって、基盤となるオペレーティングシステム（OS）は、適用可能な機能安全規格[3–5]に準拠した先進的な安全関連システムの統合を促進すべきである。

この点に関して、産業界と学術界の双方が、安全性が極めて重要なシステムへのLinux活用に関心を高めている[6]。その理由の一つは、Linux OSが組み込みシステムからスーパーコンピュータに至る幅広いコンピューティング領域で主導的なOSであることにある[7]。 さらに、通信や銀行業務といった重要アプリケーションや、宇宙船（例：SpaceX Falcon 9、Dragon）などの信頼性システムにおいても既に採用されている[6]。加えて、そのオープンソース開発モデルは、IEC 61508機能安全規格に基づく認証に潜在的に適していると見なされてきた[8]。

本稿では、安全上重要なシステムにおけるLinuxの活用に関する学術文献をレビューする。その目的のため、Linuxの安全関連用途の意図に応じて貢献を分類する。

(1) *Linux as a GPOS:* Linuxは、安全重要システム内で安全非関連汎用ソフトウェアアプリケーションを実行する汎用オペレーティングシステム（GPOS）である。主な課題は、システムの安全重要部分からの必要な分離と独立性を確保することである。

(2) *SCOSとしてのLinux：*Linuxは、安全関連および混合重要度ソフトウェアアプリケーションの安全な実行を支援することを目的とした安全重要度オペレーティングシステム（SCOS）です。主要な課題は、タイミング挙動、診断、フォールトトレランス、時間的・空間的独立性などの安全規格要件への準拠を確保することです。

学術文献のレビューに加え、Linux FoundationのELISA（Enable Linux In Safety Applications）プロジェクト、EB Corbos Linux for Safety Applications、RedHawk Linux RTOSといった主要な産業イニシアチブについても調査する。

本論文の残りの構成は以下の通りである：第2節では本研究を理解するために必要な主要概念を提示する。第4節では文献レビューに用いた方法論と分類体系を説明する。 第5章では文献レビューを開始し、混合重要度システムにおいてLinuxを汎用リアルタイムOS（GPOS）として利用した研究を紹介する。第6章ではLinuxを安全クリティカルOS（SCOS）として利用した研究を扱う。第7章では安全クリティカルシステムへのLinux導入に焦点を当てた主要な産業アプローチを提示する。第8章では文献レビューから抽出した主な課題を要約する。最後に第9章で本論文を総括する。

2. 背景

このセクションでは、当社の業務を理解するために必要な基本概念について説明します。

2.1. 安全関連システム

安全関連システムとは、機能安全の原則を通じて危険リスクを許容可能なレベルまで低減し、その対象環境の安全を確保するように設計された、システムの電気的、電子的、またはプログラム可能な電子的部分を指します。機能安全は、電気的、電子的、およびプログラム可能な電子システムの正しい動作によって確保される安全の側面を特に扱います。危険リスクを許容可能なレベルまで低減することで、対象環境が「許容できないリスク」に晒されることを防止することを目的としています。 安全はまた、「ユーザーおよび環境に対する壊滅的な結果の不在」とも定義される[9]。

機能安全規格は、安全関連システムの重要度レベルに応じて、リスク評価の要求事項、技術、および手法を定義する。規格IEC 61508は安全を「許容できないリスクからの解放」と定義する（IEC 61508-4 Ed2 3.1.11項）。 一方、ISO 26262規格では安全を「不合理なリスクの不在」（ISO26262-1 第1版 1.103項）と定義している。 両規格ともリスクを定量化可能な尺度として扱い、「安全性」をリスクを許容可能なレベル以下に低減することと定義している。これは、当該システムによって生じる「許容できないリスク」に、この文脈のいかなる部分も晒されてはならないことを意味する。 例えば、自動運転車のケースでは、「許容できないリスクからの自由」は、コンテキスト内の様々なステークホルダーに適用される：(i) 車内の乗員・動物（乗客、該当する場合の運転者）、(ii) 車外の乗員・動物（歩行者、自転車利用者、他の運転者）、(iii) インフラと環境（他の車両、道路構造物、財産、野生生物など）。

さらに、各機能安全規格は、リスクを許容可能なレベル以下に維持するための重要度レベルを定義している。例えば、IEC 61508は安全度水準（SIL）を1（最も低い重要度）から4（最も高い重要度）まで定義している。 自動車分野のISO 26262では、これに類似した自動車安全完全性レベル（ASIL）をA（最低重要度）からD（最高重要度）まで定義している。航空電子機器分野では、RTCA DO-178Cが設計保証レベル（DAL）をDAL E（最低重要度）からDAL A（最高重要度）まで定義している。

しかし、これらの規格の多くは15年以上前に策定されたものであり、安全要件や技術、手法は新たな技術の発展に追いついていない。最先端の安全関連システムと比較すると、現代システムの複雑化（大規模なコードベース、周辺機器の増加、機能の追加など）は、従来のリスク評価手法にとって課題となっている。 これらの現代システムは、外部通信、セキュリティ上の懸念による更新の必要性、適応学習メカニズム（例：機械学習）、多層ソフトウェアアーキテクチャなど、複雑な相互依存関係と挙動を持つ複数の相互作用するコンポーネントで構成されている。自律走行車は、こうしたシステムの代表的な例である。その結果、安全基準を最先端の安全関連複雑システムに適用することはますます困難になっている。

2.2. Linuxと機能安全

LinuxはオープンソースOSであり、スマートフォンや組込みシステムからクラウドコンピューティング、スーパーコンピュータに至るまで、様々な分野の安全関連以外のアプリケーションの大半を占めている[6,7]。安全基準を満たすために開発されたわけではないが、Linuxは広範なハードウェア・ソフトウェアサポートを備えた巨大なエコシステムの恩恵を受けており、これがその普及につながっている。 プロプライエタリOSではなくLinuxを使用することで、(i) ライセンス料が不要なため開発・導入コストを削減できる、(ii) オープンソースの性質により脆弱性への耐性が潜在的に高まり信頼性が向上する、(iii) コンピュータサイエンス分野の多くの専門家がこのOSの経験を有するため、強力なコミュニティとサポートにアクセスできる、といった利点がある。その結果、

産業界では安全関連システムへのLinux採用に関心がますます高まっている。産業界の関心と並行して、学術界からの関心も増加している。この点に関して、序論で述べたように、我々は安全重要システムにおけるLinux利用に焦点を当てた研究を特定し、システムにおけるLinuxの役割に基づき主に二つのグループに分類した：(1) GPOSとしてのLinux、および
(2) SCOSとしてのLinux

特に留意すべき点は、特定のLinuxバージョンを機能安全に適したものとする試みがこれまで行われていないことです。Linuxは完成品ではなく、進化を続けるプロジェクトです。したがって、安全関連プロジェクトで使用するための独立したコンポーネントとして認証することはできません。Linuxを組み込んだ各システムは、システム全体として正当性を説明し、認証を受ける必要があります。 さらに、安全重要システムにおけるLinux利用を可能とするプロジェクトは、Linuxカーネルのみに焦点を当てるのではなく、ライブラリ、ツール、およびOSを構成するその他全ての要素を考慮に入れる必要がある。

安全規格は、要求されるSILに応じて適用すべき特定の開発プロセス、文書化手法、仕様策定方法、検証・試験活動を定義する。SILが高いほど、これらの要求事項は厳格になる。例えば、最高レベルの完全性（IEC 61508のSIL4、ISO 26262のASIL D、DO-178CのDAL Aなど）では、規格は以下のことを要求する：

- 厳格な仕様策定とトレーサビリティ：全ての安全要求事項は正式に仕様化され、設計・実装・検証・妥当性確認の各成果物を通じて追跡可能な形で関連付けられなければならない。
- 開発活動は、明確に定義され文書化された安全ライフサイクルに従う必要があり、レビューや評価における役割、責任、独立性が定められていること。
- 完全な検証とテスト。具体的には、DO-178Cにおける修正条件/決定カバレッジ（MC/DC）、体系的な故障注入、網羅的テストなどの要件を満たすこと。
- 開発および検証ツールの認定と証拠。
- バージョン管理、変更影響分析、再検証を厳格に管理すること。

コミュニティによって開発され、絶えず進化を続けるLinuxのようなオープンソースOSでこれらの要件を満たすことは困難である。Linuxカーネルは4000万行以上のコードで構成され、世界中の数千人の貢献者によって開発されており、プロセス、ドキュメント、設計成果物に対する集中管理が存在しない。 その結果、安全規格における高完全性レベルが要求するトレーサビリティ、プロセス保証、体系的な検証は根本的に欠如している。これが、Linuxベースの安全重要システムに対する認証取得が、学界と産業界の両方で取り組まれている未解決課題である理由を説明している。

最後に、安全重要システムにおけるLinuxの採用は、ますます複雑化する安全重要システムの利用という産業ニーズに応えるものである。したがって、Linux利用への関心は、そうした複雑性を支えるプラットフォーム・ライブラリ・プログラム等の必要性に応えるものだ。この点に関して、他の研究者らはこれらのシステムの様々な側面について文献レビューを行っており、我々は第3章でその概要を概説する。

2.3. *Linuxとリアルタイム*

安全関連システムは予測可能な動作を示す必要があり、これは時間予測性も意味する。IEC 61508は、時間的独立性を確保するため、決定論的スケジューリング手法と厳格な優先度ベースのスケジューリングの使用を推奨している（IEC 61508-3 第2版 附属書F.5）。この要件が、多くの安全重要システムにリアルタイム制約が課される理由である。 例えば、自動車のアンチロック・ブレーキ・システム（ABS）は特定の時間枠内で作動しなければならない[4]。したがって、SCOS（安全クリティカルシステム用OS）はリアルタイムOS（RTOS）であるが、全てのRTOSがSCOSであるわけではない。

リアルタイムシステムは、出力が指定された期限内に確実に提供されるよう設計されており、速度よりも信頼性を優先する。重要なのは、リアルタイムシステムであることが必ずしも高速システムを意味しない点であり、むしろ期限の遵守が本質である。リアルタイムシステムは、期限遵守の程度に基づいて分類できる。

そして、順守しない場合の結果。*ソフトリアルタイムシステム*では、指定された期限を過ぎても結果が生成・使用され続ける可能性があるが、サービス品質（QoS）は低下する。この柔軟性により稼働率は高まるが、システム全体のパフォーマンスは低下する可能性がある。*堅実なリアルタイムシステムは*より制約の強いアプローチで動作する：システムが予定通りに結果を届けられない場合、期限後の出力は使用不能となるため結果を生成しない。ただしこの失敗は深刻な結果を招かず、信頼性が不可欠であるものの、重大な影響なく期限超過を許容する。対照的に、*厳格なリアルタイムシステムは*柔軟性のない期限が特徴である。 これらの厳格な期限遵守は絶対条件であり、未達時は重大な故障を引き起こす可能性がある。このため、安全関連リアルタイムシステムはハードリアルタイムシステムに分類される[10]。

標準的なLinuxカーネルはRTOSではないが、それをRTOSに変換する手法がいくつか存在する。本調査で示した通り、マイクロカーネルやLinuxカーネルと並行して動作するコカーネルを利用する手法もあれば、PREEMPT_RTパッチのようにカーネル自体を修正してリアルタイム性を付与する手法もある。 このパッチはバージョン6.12以降、メインラインLinuxカーネルに統合されている。LinuxをRTOSへ変換する手法の詳細については、Reghenzaniら[10]およびStruhárら[11]による調査報告を参照されたい。前者はPREEMPT_RTパッチに重点を置き、後者はLinuxにおけるリアルタイムコンテナベース仮想化に焦点を当てている。

3. 関連研究

Linuxが本調査の中心である一方、安全重要システムのあらゆる構成要素がその信頼性と認証可能性に役割を果たす。OSに加え、基盤となるハードウェアアーキテクチャ、GPUなどのアクセラレータ、ミドルウェア、AIフレームワークなども安全重要システムにおいて独自の役割を担い、固有の課題を抱えている。 産業界が構築を目指す安全重要システムの複雑性が増すにつれ、これらの側面は近年学術界で徹底的に研究されてきた。この観点から、本節ではこれらの側面を探求した他の調査研究をまとめ、安全重要な文脈で解決策を見出す際にはシステム全体を考慮する必要性を強調する補完的な視点を提供する。

Perez-Cerrolazaら[4]は、異なるデバイス抽象化レベル（ナノスケール、コンポーネント、デバイス）における基本的な安全要件に取り組む研究貢献を分類し、概説している。彼らは、マルチコアシステムの安全認証が課題である理由を説明し、基本的な安全技術要件である時間的・空間的独立性、信頼性、診断カバレッジなどの側面に焦点を当てている。

GPUに関しては、Perez-Cerrolazaら[5]による調査が、GPUデバイスのランダムなハードウェア故障、系統的故障、および実行の独立性に対処する研究を分類し概説している。これは、共有ハードウェアリソースを持つGPUデバイス上で、複雑で並列かつ計算負荷の高いソフトウェア機能を、異なる安全重要度レベルで統合する課題に焦点を当てている。

安全重要システムにおけるハイパーバイザーの活用については、Lozanoら[12]がレビューを行っている。彼らは安全重要システムの基盤としてハイパーバイザーを利用する研究を包括的にレビューし、各ハイパーバイザーの情報を収集・分類するとともに相互比較を行っている。

学術研究の調査に加え、専門家へのインタビューによる産業調査も実施されている。Kassab [13] の調査は産業における安全重要ソフトウェアのテスト手法を収集し、Pedersenら[14]は主要な産業ニーズと課題を特定・分析している。

特定の産業分野における調査も実施されている。本稿で述べる通り、自動車および航空宇宙分野が、Linuxの活用に焦点を当てた研究の主要な領域である。

表1

文献調査の結果。

ライブラリ	検索パラメータ	発見
ACM	タイトル、要旨、キーワード	59
Google Scholar	タイトル	20
IEEE	タイトル、要旨、キーワード	356
Inspec	タイトル、要旨、キーワード	398
Science Direct	タイトル、要旨、キーワード	29
Scopus	タイトル、要旨、キーワード	703
Springer Link	タイトル	77
Web of Science	タイトル、抄録、キーワード	290
合計		*1932*
重複の除去		*1336*

安全上重要なシステムであり、これはLinuxに特有ではなく、安全上重要なシステム全般に当てはまる。 したがって、これらの分野における安全クリティカルシステムに焦点を当てた調査は存在する。自動車分野では、Padenら[15]が都市部車両の計画・制御アルゴリズムに関する最新技術を研究し、各手法を並列比較している。一方、Rabeら[16]は機械学習自動車アプリケーション開発の方法論を探求し、研究状況の概要を示しつつ主要課題を特定している。航空電子工学分野では、Boglettiら[17]が航空機の安全重要駆動装置（飛行表面アクチュエータ、燃料ポンプ、発電機）の電動化を研究し、そこから派生する主要な技術的課題と研究テーマを提示している。 最後に、Perez-Cerrolazaら［3］は産業・輸送分野におけるAIベース安全重要システムの開発を研究している。具体的には、最先端AIと安全基準を融合させる課題に取り組む研究を収集し、AIベース安全重要システム開発における課題・技術・手法を分析している。

安全とセキュリティは異なる基準で評価されるものの、特に現代の安全重要システムにおいては強く結びついている。このため、一部の研究者は安全重要システムのセキュリティ側面を研究している。Kriaaら［18］は、安全とセキュリティの両方を考慮した産業施設設計とリスク評価に関する産業界および学術界のアプローチを調査している。 Lisobaら[19]は、システム開発の初期段階に焦点を当て、安全性とサイバーセキュリティの共同解析手法に関する文献レビューを提供している。Kavallieratosら[20]は、安全性とサイバーセキュリティの共同設計に関する追加手法を特定することで先行文献レビューを拡張し、この分野における最近の進展を研究している。 先行研究は、最先端の安全重要システムにおける安全とセキュリティの共同設計の必要性、および両者を考慮した方法論のさらなる研究の必要性について合意している。 この点に関して、本調査ではLinuxベースの安全重要システムについて、安全面のみに焦点を当て、セキュリティ面は対象外としていることに留意する必要がある。これは、特にLinuxに焦点を当てた場合、安全とセキュリティが現在もなお、異なる目標を持ち、異なる認証基準に対応する、別個でありながら相互に関連する側面であるためである。

Linuxに話を戻すと、安全上重要なシステムにおけるその利用は、セキュリティ、リアルタイム動作、分離といった他の側面と必然的に結びついている。 これらの側面を焦点に置いた文献調査を行った著者もいる。Procopio [21] は、安全性とセキュリティの統合分析に焦点を当てた研究やプロジェクトをまとめ、両方を考慮した規格を紹介し、安全性とセキュリティの認証に準拠するRTOSを列挙し、最後に、Linuxベースのシステムで安全認証への準拠を目指すプロジェクトはセキュリティ認証も考慮すべきだと主張している。 Linuxのリアルタイム動作に関しては、Reghenzaniら[10]の調査が、PREEMPT_RTパッチに特に焦点を当てつつ、リアルタイムLinuxシステム構築のための最新アプローチをまとめている。 一方、Struhárら[11]は、Linuxコンテナベース仮想化にリアルタイム特性を導入する取り組みに関する文献をレビューし、分離機能と併せたLinuxのリアルタイム性に焦点を当てている。

これらの調査と比較した場合、本研究の新規性は、安全重要システムにおいてLinuxを使用している、または使用を計画している文献をレビューし、Linuxが安全重要システムでの使用において直面する課題を特定し、Linuxのシステム内での役割と各研究が焦点を当てる課題に基づいて研究を分類した点にある。文献レビューに加え、安全重要システムでのLinuxの活用を目指す主要な産業努力の概要と、Linuxの安全重要システムでの使用に関する産業の取り組みの概要を記載したセクションを含める。

安全重要システムでの使用における課題を特定し、各研究をLinuxのシステム内での役割と焦点となる課題に基づいて分類している点にある。文献レビューに加え、Linuxベースの安全重要システムを目指す主要な産業努力の概要をまとめたセクションを含め、それらを特定された課題と関連付けている。

4. 方法論と分類体系

4.1. 検索方法論

本文献レビューの方法論は、以下のデータベースにおける文献検索から構成される：ACM、Google Scholar、IEEE、Inspec、Science Direct、Scopus、Springer Link、Web of Science。検索クエリは「Linux AND safety」とし、タイトル、抄録、キーワードで検索を実施した。Google ScholarとSpringer Linkではこれらの検索パラメータが使用できなかったため、タイトルのみを対象とした。 「機能安全」や「安全重要」ではなく「安全」で検索したのは、文献内で「安全システム」や「安全関連システム」など様々な表現が用いられているためである。したがって、多くの誤検出が生じることを承知の上で、これら全てを包含する語句が必要であった。

結果は表1に示す。検索は2025年3月に実施した。ご覧の通り、検索語を含む1336件の論文を発見した。これらを精査し、安全上重要なシステムでLinuxを使用する論文を特定し、該当しないものを除外した。 例えば、メモリ安全性に焦点を当てた研究や、機能安全を扱っているものの安全上重要なシステムでのLinux使用を意図していない研究などが含まれていた。第3節で説明した通り、セキュリティ面に焦点を当てた研究も除外対象とした。選別後、102件の研究が対象として残った。

4.2. 分類体系

本調査で対象文献を分類するために用いた分類体系を提示する。前述の通り、分類はLinuxの想定用途に基づく：

(1) *Linux as a GPOS:* Linuxは、安全上重要なシステム内で安全非関連汎用ソフトウェアアプリケーションを実行する汎用目的オペレーティングシステム（GPOS）である。
(2) *SCOSとしてのLinux*：Linuxは、安全関連ソフトウェアアプリケーションおよび混合重要度ソフトウェアアプリケーションの安全な実行を支援することを目的としたSCOSである。

最初の分類の後、各カテゴリーはさらに細分化される。研究の分類を表2にまとめる。LinuxをGPOSとして使用する最初のケースでは、システム内の安全重要コンポーネントからLinuxを隔離する方法をさらに分類する：

(1) *ハイパーバイザーは*仮想化レイヤーを提供し、同一ハードウェア上で複数のOSまたはベアメタルプログラムを同時に実行可能にすることで、安全上重要なコンポーネントを効果的に分離する。

表2

研究の主要分類体系の概要

主要カテゴリ	サブカテゴリー	作品
Linux as a GPOS	ハイパーバイザー	[22–39]
	複数カーネル	[40–46]
	異種ハードウェア	[47–54]
SCOSとしてのLinux	認証可能性	[6,21,55–66]
	リアルタイム性と決定論的タイミング特性 フォールトトレランスとリカバリ	[67–92] [93–111]
	Linuxベースの安全重要システム設計	[112–121]

表3

研究対象領域。

領域	汎用オペレーティングシステムとしてのLinux	SCOSとしてのLinux
自動車	[22,23,28,30,32–34,36,37,39,43,44,47,49,50,52,53]	[6,71,72,75,79,82,93,113–116,118]
航空電子機器	[24–26,31]	[56,61,62,68,70,73,85,90,100,104]
医療	–	[96]
核医学	–	[48,57]
ロボティクス	–	[78,92,103]
スペース	–	[61,62,112]
未指定	[27,29,35,40–42,45,46,51]	[21,55,58,59,63–65,67,69,74,76,77,80,86–89,91,94,95,97–99,101,102,105–111,117,119–121]

(2) *マルチカーネルアプローチ*では、複数のカーネルを並列に実行し、各カーネルが異なるタスクを処理することで、安全上重要な機能を汎用オペレーティングシステム（GPOS）から分離する。

(3) *ハードウェアベース*のアプローチは、GPOSとSCOSの分離を強制するために専用ハードウェアコンポーネントを利用する。

第二のケースであるSCOSとしてのLinuxにおいては、機能安全に関連するLinuxの具体的な側面を分類します。これらは以下の通りです：

(1) *認証可能性*：LinuxをSCOSとして使用するシステムの認証可能性を指す。

(2) *リアルタイム性と決定論的タイミング挙動*：遅延の測定やタスク完了の期限保証に加え、マルチコアおよび混合重要度システム向けのリソース割り当てとタスク実行順序を提案するスケジューリング中心のアプローチを含む。

(3) *耐障害性と回復性*：システム内の障害の注入、検出、軽減、許容に関連するもので、安全上重要な機能が正しく動作し続けられることを保証します。

(4) *Linuxベースの安全重要システム設計*。これには、LinuxをSCOSとして使用するために提案された設計パターンとアーキテクチャが含まれる。

一般的な分類に加え、ここでは二次的な分類をいくつか提示する：研究対象の分野別（表3）、Linuxをリアルタイム化するためのアプローチ（実施されている場合）（表4）、および対象プラットフォームの命令セットアーキテクチャ（ISA）（表5）に基づく分類である。

表3に示すように、これらの研究が位置づけられる主な分野は自動車分野であり、Linuxを汎用オペレーティングシステム（GPOS）としても安全制御システムオペレーティングシステム（SCOS）としても利用している。航空電子機器分野が次に多い分野である。これら二つの分野は、安全上重要なシステムにLinuxの汎用機能を組み込む可能性に最も関心を示している。

リアルタイムアプローチに関しては、表4に示す通り、特にLinuxがSCOSである場合にPREEMPT_RTが最も広く採用されている。各アプローチの特性に関する詳細は、Reghenzaniら[10]による調査で確認できる。 注目すべきは、PREEMPT_RTパッチを除き、他のアプローチはLinuxの各役割ごとに固有である点だ。この分離は予想されるもので、LinuxがGPOSとして使用される場合のアプローチは、リアルタイムタスクを処理する第二のカーネルを追加することに依存するため、その部分を

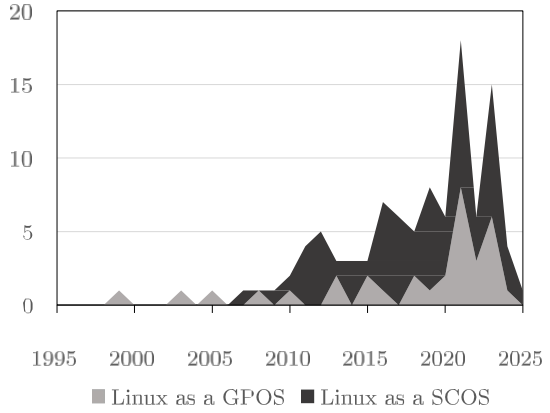

図1. Linuxの使用形態（SCOSまたはGPOS）別に分類した、年次別出版物の積み上げ棒グラフ。

システムは安全上重要な機能を担い、Linuxを二次的な汎用オペレーティングシステム（GPOS）として使用する。Linuxを安全オペレーティングシステム（SCOS）として用いるアプローチの場合、Linuxカーネルをリアルタイム対応に改造し、安全上重要な機能を担わせる。

対象プラットフォーム、特にISA（表5参照）に関しては、LinuxがGPOSでもSCOSでも、64ビットARMv8-Aとx86-64が最も多く採用されているアーキテクチャであることがわかる。これは学術研究においてこれらのプラットフォームが広く利用可能であることに起因する。

最後に、図1に年ごとの研究件数を可視化した。これを見ると、特に過去5年間において、安全重要システムにおけるLinuxの利用に対する関心が高まっていることがわかる。過去5年間で汎用オペレーティングシステム（GPOS）としての利用が増加しているものの、2000年以前から既にそうしていた研究も一部存在した。 LinuxをSCOSとして利用する研究は2007年頃に始まり、その後徐々に増加し、過去5年間で急増している。

5. 混合重要度システムにおける汎用安全制御システムとしてのLinux

汎用性と幅広いプラットフォームサポートにより、Linuxは多くの混合重要度システムにおいて、独立した安全重要コンポーネント（例：SCOS）と並行してGPOSとして使用されてきた。主な課題は、Linuxが安全重要機能に干渉しないことを保証することである。

表 4

リアルタイムLinuxアプローチ（指定されている場合）。複数のアプローチを採用する研究もあることに留意。

リアルタイムアプローチ	Linux as a GPOS	Linux as a SCOS
FIN.X-RTOS	–	[104]
階層的スケジューリングパッチ	–	[76]
Linux/RK	–	[90]
LITMUS^{RT}	–	[83]
プリエンプトRT	[24,26,28,31]	[69,71,73,78,79,81,87,89,94,96,105,115,121]
RTAI	[43,45,46,103]	-
RTLinux	[40,42]	-
SPark	[42]	-
Xenomai	[41,44,103]	

-

表5

ターゲット命令セットアーキテクチャ（指定されている場合）。なお、一部の研究は複数のプラットフォームおよびISAを対象としている。

ターゲットISA	ビット	Linux as a GPOS	Linux as a SCOS
ARMv7-A	32	[22,31,47,48,52,54]	[69,73,97,106,107,110,113]
ARMv8-A	64	[23,29,30,36,50,51,53]	[6,63–65,67–72,75,84,85,91,92,107–109,113,115,118]
PowerPC	32	[42,45]	[120]
RISC-V	64	[29,30,35]	[119]
SPARC (V7、V8)	32	–	[112]
x86	32	[25,27,40,42,43]	[74,82,93,94,116]
x86-64	64	[24,26,28,34,37,41,44,49]	[66,76,78–80,83,86–90,94,95,98,99,103,111]

例えば、分離された安全上重要なコンポーネントには、マルチコプターのタイミングおよび安全上重要な飛行制御タスクが含まれ、一方、視覚ベースのナビゲーションタスクはLinux上で実行される[24]。 分離された安全上重要なコンポーネントの別の例としては、車両の安全上重要なECU機能（先進運転支援システム（ADAS）、車載情報娯楽システム（IVI）、計器クラスター（IC）の重要な部分など）が挙げられる。これらは安全性とリアルタイム性を保証して動作し、非重要な部分はLinux上で動作する[34]。

この調査の範囲内で、分離課題に対処するための3つのカテゴリーの解決策が浮上している：

(1) *ハイパーバイザー*：ハイパーバイザーは仮想化レイヤーを提供し、同一ハードウェア上で複数のOSまたはベアメタルプログラムを並行実行可能にすることで、安全重要コンポーネントを効果的に分離する。

(2) *マルチカーネル*: 安全上重要なタスクの決定論的スケジューリングを保証するため、マルチカーネル手法が採用されている。これにより、安全上重要な機能とLinux汎用機能を時間領域で分離する。

(3) *異種ハードウェア*: 汎用OS（GPOS）と安全OS（SCOS）の分離を強制するために、特定の異種ハードウェアアーキテクチャおよびコンポーネント（例: Linux + FPGA、Linux + 安全CPU）が提案されている。

これらのソリューションにはそれぞれトレードオフがある。ハイパーバイザーはシステムリソースをパーティション分割または仮想化することで空間的・時間的な分離を提供する。これにより、安全性が重要なシステムの機能と並行してLinuxを実行できる。しかし、仮想化に伴う追加の複雑性、認証作業、潜在的なパフォーマンスオーバーヘッドが生じる。 マルチカーネル方式はリアルタイムカーネルをLinuxと統合し、低遅延応答を必要とするアプリケーションに対応するとともに、Linux上で動作する必要があるアプリケーション向けに両OS間の連携を可能にします。ただし、Linuxとリアルタイムカーネルがメモリ、ドライバ、割り込みコントローラを共有することが多いため、障害隔離性が低下するという代償を伴います。このため、Linuxのバグや誤動作がシステム全体に影響を及ぼす可能性があります。 最後に、異種ハードウェア設計は、ARM TrustZoneなどの特定のハードウェア技術に依存してシステムリソースを分割する。干渉を最小限に抑えながら強力な分離性と決定性を提供するが、より複雑なシステム統合と専用ツールチェーンを必要とし、他のソリューションと比較して柔軟性が制限される可能性がある。

システム全体として認証を取得するため、この種のソリューションではハイパーバイザー、セカンドカーネル、ハードウェアも認証を受ける必要がある。この点において、認証を念頭に置いた複数のソリューションが開発されている。例えば、ARMv8アーキテクチャ向けのPikeOSは、EN 50128およびEN 50657（鉄道）のSIL4、ならびにIEC 61508（産業用）のSIL3認証を取得している。 別の例として、seL4マイクロカーネルは「seL4の形式証明の強度は、ISO26262の最高レベルであるASIL-Dが要求する水準を大幅に上回る」と主張している[122]。ハイパーバイザーや異種ハードウェアなどの利用に関する詳細は、第3章で提示された調査研究を参照されたい。

本節で分類した研究の概要を表6に示す。

5.1. *ハイパーバイザー*

ハイパーバイザーは、物理デバイス（例：CPU、メモリ）の抽象化レイヤーを提供するとともに、リソーススケジューリングや割り当てといった機能群を備えたソフトウェア層である。これにより、同一ハードウェア上で複数のOSやベアメタルアプリケーションを並行実行可能としつつ、相互の分離と制御されたリソース共有を実現する。この方式により、システムは安全性が重要な機能とそうでない機能を同一プラットフォームに統合できる。 ハイパーバイザーには主に2種類ある：物理ハードウェア上で直接動作するタイプI（ベアメタル）ハイパーバイザーと、OS上で動作し仮想化に依存するタイプII（ホステッド）ハイパーバイザーである。サーバーなど他の用途で使用されるハイパーバイザーとは異なり、安全性が重要なシステム向けに提案されるハイパーバイザーの主な目的は、仮想化ではなくシステムの分割とゲスト間の分離である。 このため、この文脈ではハイブリッド型も存在するものの、タイプIハイパーバイザーが最も一般的である。

この点に関して、ソースコードサイズを最小化することで認証可能性に焦点を当てた、この種のハイパーバイザーがいくつかの著者によって開発・提案されている。特定のハイパーバイザーは、代わりに静的パーティショニングを用いることで、仮想化のためのハードウェア拡張への依存を回避している[35]。 静的パーティショニングとは、ゲストシステムにシステムリソースを事前に割り当て、実行中に変更しない手法を指す。静的パーティショニング型ハイパーバイザーの他の二例として、*Jailhouse* [31] と *Bao* [30] が挙げられる。前者はシステム起動に Linux を活用するのに対し、後者は自身で起動処理を行う。

既存のツール上に構築された他のハイパーバイザーも存在する。例えばKVMエコシステムを基盤とするKHV [29]が挙げられる。さらに、自動車分野向け（*DriveOS* [34]、*Automotive Hypervisor* [39]）や航空電子機器分野向け（*FlyOS* [24, 26]）など、特定分野向けのハイパーバイザーも設計されている。航空電子機器分野では異なるアプローチが提案されている

表6

Linuxを汎用オペレーティングシステム（GPOS）として使用した調査対象研究の概要。

カテゴリ	目標	使用技術	特徴	作品
ハイパーバイザー	建築設計	自動車向けハイパーバイザー	各機能が実行される場所を選択するための異なるモードをサポート	[39]
		CLARE	Linuxはディープラーニングプロセッサユニット（DPU）と連携し、AIアルゴリズムを実行する	[25]
		JailhouseとCLARE	マルチSoCアーキテクチャ	[23]
		KVM	アポロ制御コンポーネントをLinuxからErikaへ移行	[28]
		手動ハードウェア設定	Linux における視覚ベースのナビゲーションタスク、フライト Quest RTOS における制御	[24]
			Linux での視覚ベースのナビゲーションタスク、Quest RTOS での飛行制御	[26]
		概念実証	同一プラットフォームに利便性と安全機能を統合した概念実証プラットフォーム	[33]
		Xen	LinuxはXen制御ドメイン上で動作する	[22]
		XtratuM	安全性が重要なJavaアプリケーションのためのプラットフォーム	[27]
	通信	VOSYSmonitor	ネットワークチャネ ルの物理層に構築され、複雑性を低減	[36]
		Xen	オープンソース実装の既存ドライバの再利用を可能にします	[38]
	パーティショニングハイパーバイザー設計	Bao	スタンドアロン	[30]
		DriveOS	Quest OSによって起動される	[34]
		刑務所	Linuxによって起動	[31]
		KHV	KVMをベースとするが、起動後はLinuxに依存しない	[29]
		概念実証	一時的なパーティショニング	
		[32]VOSYSmonitoRV	ハー	
			ドウェア仮想化拡張機能は不要	[35]
	システムの再起動	Quest-V	サスペンドとレジュームの仕組み	[37]
マルチカーネル	共同カーネル設計	RT-Linux	Linuxカーネル内でリアルタイムAdaタスクを実行アドレス空間	[40]
		SParK	検証・妥当性確認の労力を最小化するためにシステムを分割	[42]
	設計方法論	RTAI	分散型安全重要リアルタイムアプリケーションのための方法論の設計	[46]
	機能評価		時間トリガ型スケジューラを用いたRTAI/LXRTのパーティショニング機能評価	[43]
	機能拡張		タイムトリガード環境におけるTMOモデルのサポートアーキテクチャ	[45]
		Xenomai	リアルタイムコンテナの階層的グループスケジューリング	[41]
			時間的障害を防止するための監視	
		[44]	CPU動作モード	安全
異種ハードウェア	アーキテクチャ設計	上重要な機能を実行するためのx86 SMM	[49]	
			安全上重要な機能を実行するためにARM TrustZone上に構築	[54]
		ヘテロジニアスSoC	ハードウェアおよび両OSの構成オプションによる分離	[47]
			異種SoCの活用による安全重要システム機能、ハイパーバイザーによるGPOSの分離	[52]
		プログラマブルロジックSoPC	FPGA内の全ての安全上重要な機能	[48]
			FPGAに実装されたRISC-Vで動作する安全上重要な機能	[50]
	レイテンシの評価	ヘテロジニアスSoC	異なるアーキテクチャの比較（OpenAMP有無）	[51]
			OpenAMP非搭載	
	ネットワークィルタ		重要トラフィックと非重要トラフィックを分離するネットワークフ	[53]
			非重要トラフィックを分離するネットワークフィルタ	

シュナイダー[32]は、ハイパーバイザーのように動作し共有リソースへのアクセスを管理するミューテックス形式のパーティション管理者を採用している。

他の著者らは、Xen [22]、Jailhouse [23]、Xtratum [27]、CLARE [23,25] などのハイパーバイザーに基づく安全クリティカルシステムの設計アプローチを提案している。これらのハイパーバイザーはLinuxの分離を支援する。

異なるSCOS（リアルタイムOS）から、Erika [22,23]、FreeRTOS [25]、PaRTiKle [27] を含む。自動車分野では、次世代ソフトウェア定義車両（SDV）向けに具体的な実装が提案されており、Quest-Vハイパーバイザーを用いてLinuxとQuest RTOSを分離する[37]、KVMを用いてLinuxとErika RTOSを分離する[28]といった例がある。

これらの異種アーキテクチャは、ゲスト OS に対して透過的になるように設計されており、自動車の安全関連システムの潜在的な進化形となっています [33]。

ゲスト間の通信手法も提案されている。例えばVOSYSVirtualNet[36]は、セキュア領域と非セキュア領域間の低遅延仮想ネットワークリンクである。これはVOSYSmonitor[123]ハイパーバイザー上で動作し、各領域ごとに1つのバッファとハイパーバイザーへのシグナルによって実装される。ハイパーバイザーが通信の調整を担当する。 同じ目的を持つ別のアプローチがXenハイパーバイザー向けに提案されている[38]。これはvirtio、RPMsg、Xenバス上に構築された、Xenハイパーバイザーの非特権ゲスト間におけるドメイン間通信（IDC）のためのフレームワークであり、ゲスト間のUDP通信と比較して最大3倍のレイテンシ改善が可能である。

5.2. *複数カーネル*

複数のカーネルアプローチは、Linuxの機能を活用しつつ決定論的なタイミング動作を保証するために採用されてきた。本調査における「複数カーネル」とは、標準的なLinuxカーネル（いわゆるバニラカーネル）と、コカーネルと呼ばれる二次的なカーネルを並行して実行することを指す。 標準カーネルは非決定論的かつ安全非関連タスクを処理し、コカーネルは時間決定論的かつ安全クリティカルな機能を管理する。ハイパーバイザーとは異なり、コカーネルは標準Linuxカーネルと並行して動作し、システムリソース（例：中央処理装置（CPU）、メモリ、ハードウェア割り込み）を直接制御することで、決定論的なリアルタイム応答を保証する。

一般的に、コカーネルは次のように動作します：Linuxの下位層である割り込み処理の最下位レベルで動作します。したがって、すべてのハードウェア割り込みはコカーネルを経由し、コカーネルが直接処理するかLinuxに転送するかを決定します。この判断は、割り込みがリアルタイムタスクに関連するか否かに基づいて行われます。 これにより、2つの実行ドメインが形成される。すべての重要機能を担うリアルタイムドメインはコカーネルで実行され、汎用機能はLinuxドメインで実行される。コカーネルとLinux間の通信は通常、専用APIや共有メモリを介して行われる。この通信方式とコカーネル固有の機能は、各ソリューションごとに実装依存となる。

最後に、Linuxは完全公平スケジューラ（CFS）からリアルタイムFIFOやラウンドロビン（RR）スケジューラまで標準的なスケジューラ群を利用できるのに対し、コカーネルは通常、ハードリアルタイムタスクを実行可能な単純で予測可能な優先度ベースのスケジューラのみを備えています。

Linux向けの最も一般的な共同カーネルは以下の通りである：

- Xenomai [41]（特に高重要度タスクの制御と監視に重点を置いたものを含む）[44]。
- RTLinuxおよびその派生版（例：ベアメタルベースのsaRTL [42]、あるいはマルチタスクランタイムシステムGNARL [40]のリアルタイムAdaタスク実行用カスタム派生版）
- RTAI ベースの設計方法論およびフレームワークは、時間トリガーアーキテクチャ(TTA)、時間トリガープロトコル、および時間的および空間的干渉を低減するための適応を実装している [43、45、46]。

5.3. *異種ハードウェア*

特定のハードウェアアーキテクチャは、安全上重要な機能をLinux GPOSから分離することで安全性の保証を強化できる。調査文献では、マルチコアSoC（System-on-a-Chip）、マルチコアSoPC（System-on-Programmable-Chip）、専用CPU操作の3種類のハードウェア構成が見られる。 異種アーキテクチャ上でハイパーバイザーを動作させる混合アプローチなど、異なる手法が存在することは重要である。しかし、調査対象の文献ではこれらの手法は確認されなかったため、本調査には含まれていない。

市販の汎用（COTS）ヘテロジニアス・マルチコアSoCプラットフォームは、この分離のために専用CPUを提供できる。 この構成では、安全上重要な機能は、リアルタイム処理ユニット（RPU）またはマイクロコントローラユニット（MCU）と呼ばれる独立したコアセットで実行され、Linux OS は、アプリケーションプロセッサユニット（APU）またはマイクロプロセッサユニット（MPU）と呼ばれる汎用コアで実行されます。 RPUは安全重要機能の展開において、ロックステップ動作やAPUとの干渉回避など複数の利点を提供する[51,52]。RPUの例としてはCortex-RおよびCortex-Mベースのコアがあり、これらはベアメタル環境またはRTOS（FreeRTOS[51]、TI-RTOS[52]）上で安全機能を実行可能である。μC/OS-II [53]）。また、リソースをゲストOS間で分配する「手動」アプローチも存在する。通常のCOTSプラットフォームのリソースを、ハードウェア、ブートローダー、OS自体の設定によってゲストOS間で分配する手法も存在する[47]。これらのプラットフォームの主な欠点は、RPUが通常、APUと比較して機能やリソースが制限されている点である。

COTS SoPCデバイスは、FPGAと同様に、SoCデバイスに加えてプログラマブルロジック（PL）領域を提供し、カスタム知的財産（IP）ブロックを配置できる。 安全グレードのデバイスでは、PLがLinux APUから分離されており、干渉を受けないことが保証されています。このタイプのデバイスでは、カスタムIPブロックが、モニター[48]や、安全上重要な機能を実行するためにRTOS（例：AUTOSAR[50]）を実行する専用ソフトコアプロセッサなどのフォールトトレラントコンポーネントを実装できます。 プログラマブルロジックは柔軟性を高める一方で、ハードウェア記述言語（HDL）でのプログラミングが必要となり、設計の徹底的な検証が求められるため、開発の複雑性は増す。

専用ハードウェアの最後のカテゴリーは、異なるCPUコアではなく特定のCPU動作モードを利用し、安全上重要な機能をLinux GPOSから分離する[49,54]。具体的には、安全上重要な機能を同じコア上で「保護」モードで実行する。 x86システム管理モード（SMM）やARM TrustZoneなどのこれらのモードは、ハードウェアリソースの分離を強制し、通常の実行環境からの独立性を確保することで、安全機能を効果的にホストできます。これらのシステムは追加のコアを必要としないため、開発コストを削減できます。ただし、適切な分離を確保することは他のアプローチほど単純ではありません。

6. SCOSとしてのLinux

安全クリティカルなシステムやアプリケーション（例：自動運転）の複雑化に伴い、LinuxをSCOSとして採用し安全クリティカル機能を担わせる取り組みが進められている。これには本調査対象研究が解決を目指す複数の課題が存在する。具体的には以下の課題である：

(1) *認証可能性*：Linuxのオープンソースかつ汎用的な性質により、その開発プロセスと特性は非常に独特である。このため、安全性が重要な規格の多くの要求事項やガイドラインと整合性が取れない。それでもなお、著者らは認証の可能性を研究し、準拠のための代替ルートをいくつか提唱している。

(2) *リアルタイム性と決定論的タイミング特性*：汎用オペレーティングシステム（GPOS）として、Linuxはリアルタイム動作を保証しない。ただし、PREEMPT_RTパッチやその他の手法によりこれを実現可能であり、これらはLinux上で安全クリティカル機能を実行するための基盤技術として活用できる。 さらに、リソース競合（CPU、メモリなど）によるタイミング遅延を検知・回避する方法も必要であり、安全クリティカルタスクの決定論的タイミング動作を保証できる適切なスケジューラも不可欠である。

(3) *フォールトトレランスとリカバリ*：あらゆるシステムでランダムな障害は発生しますが、その影響を排除することは安全上重要なシステムにおいて特に重要です。これはLinuxベースの安全上重要なシステムにも当てはまります。

(4) *Linuxベースの安全重要システム設計*：Linuxベースの安全重要システムは、Linux本体だけでなく、ハードウェア、場合によってはハイパーバイザー、ミドルウェア、ライブラリなど、より多くのコンポーネントで構成される。安全重要認証は個々のコンポーネントではなくシステム全体に対して行われるため、システム定義と設計段階が極めて重要である。

本節で分類した研究の概要を表7および表8に示す。

6.1. 認証可能性

COTS オープンソースソフトウェア、特に Linux は、一般的な IEC 61508 に基づき、安全関連システムでの使用が提唱されています。これは、まず「実使用実績による証明」ルート[58]、次に、同規格のルート 3_S（「準拠非準拠開発」としても知られる） [59] を通じて実現されています。 このルートでは、Linux の開発は規格のガイドラインに従っていないものの、その要件は満たしている、という考え方に基づいています。

同じ基準、特にそれが要求する100%のテストカバレッジに焦点を当てると、Linuxでは従来のコードの静的解析は実現不可能であり、代わりに統計的手法が使用できるという主張がなされている[6,63,64]。 同様に、統計的手法に必要な収集データに対する異なるファイルシステムや負荷の影響に関する研究[65]、およびハイパーバイザー（ACRN）の影響に関する研究[66]も行われている。

Linuxの認証可能性に関する他の提案としては、オートマトンに基づく形式検証アプローチ[55]、OpenGL SC 2.0.1（Open Graphics Library – Safety Critical）準拠の3Dグラフィックスドライバの実装[60]、Linuxベースの安全重要システム構築時にセキュリティ認証も考慮に入れる提案[21]などがある。

特定の分野における標準規格についても、Linux の使用と認証可能性に関して研究が進められている。

- *航空宇宙分野*[61,62]: 著者らは、Linuxが航空宇宙分野で広く利用されるためには少なくとも4つの課題に対処しなければならないと主張している：「高速であること、決定論的であること、組み込みシステム向けであること、そして保証されていること」である。その後、保証について詳しく述べ、保証のための最も有望な解決策はリバースエンジニアリングであると提案している[61]。 また、安全重要アプリケーションにおけるLinux活用（ELISA）プロジェクトの新設航空宇宙ワーキンググループを紹介している[62]。
- *民間航空* [56]: 著者らは、民間航空における認証のためのリアルタイムOS、プログラミング言語、ツール、およびプロセスの最新状況をレビューしている。彼らは、これまでの著者たちがLinuxをCO-178B規格に対する認証の候補とみなしていることを示し、その手順には、テスト計画や手順、テストおよび検証手順を含む要件および設計文書といった、文書による証跡の作成が必要になると示唆している。
- *原子力発電所*[57]：著者らはLinuxの機能性とISO/IEC 23360:2006（Linux Standard Base（LSB））を分析し、IEC 62138:2004（原子力発電所）と比較した。 著者らは、原子力発電所の計装制御（I&C）システムに Linux を使用するための正当性は、実質的かつ骨の折れる検証、妥当性確認、認証、および技術サポートなしでは不十分であると結論づけている。

6.2. リアルタイムおよび決定論的タイミング動作

Linux カーネルのリアルタイムおよび決定論的タイミング動作は、Linux システムにおけるレイテンシおよびデッドラインの分析、特定されたタイミング関連の問題を修正するための提案、適切なタイミング動作を研究および/または保証する方法など、徹底的に研究されてきた。

プレーンなLinuxとPRE-EMPT_RTパッチ適用版Linuxの比較は、いくつかの研究[69,81]で行われている。他の研究では同様であるが、Dockerコンテナ[79,80]やROS2やOROCOS[78]などのミドルウェアを組み込んだシステムである。 研究では、このパッチがLinuxのタイミング特性に与える好影響について一致している。また、パッチ適用時にはDockerがタイミング特性にほとんど影響を与えないことも示されている[79,80]。さらに、安全クリティカルなアプリケーションなど高重要度タスクではミドルウェアの使用を推奨しない[78]。その理由は、ミドルウェアがリアルタイムシステムにおいて制御が不可欠な重要なシステム特性を隠蔽するためである。

Linuxカーネルのタイミング特性を測定する手法の開発（eBPF [71] または統計的手法[67] による）およびアクティブモニタリング [77] も、リアルタイム安全重要システムには必要であると主張されている。 こうした測定手法の研究により、一部の著者らは CUDA フレームワーク [82,83] におけるタイミング遅延を発見・修正し、さらには x86 のシステム管理割り込み (SMI) が原因であることを突き止め、x86 プラットフォームの安全上重要なアプリケーションへの適合性について疑問を投げかけている [83]。 Linuxカーネルのタイミング面における欠点は、Zuepkeら[73]によって高速ミューテックス（futex）の実装においても発見されており、彼らはハードリアルタイムおよび安全クリティカルシステムに適したPikeOSにおけるfutex実装の修正を提案しているが、これはLinux実装の機能の一部のみを提供するものである。

リソース予約（CPU、メモリなど）も、Linuxシステムにおいて重要タスクのリアルタイム動作を保証する手段として用いられてきた。 例えばCPU予算制約[74]は、安全上重要なリアルタイムタスクにCPU時間を予約することでその適切な動作を保証する。Linuxコンテナとそのオーケストレーションは別の実現手法であり、REACT[76]ではコンテナのリアルタイム要件を考慮し、適切なCPU割当量と予算を割り当てることで要件を満たすようにオーケストレーションされている。 Linuxコンテナリソースの割り当てと制限に用いられる制御グループ（cgroups）は、Linuxベースの混合重要度システムにおいて、メモリ要求のスロットリングを実装し、重要アプリケーションのメモリ競合レイテンシを低減するためにも利用されている[75]。 cgroups とは別に、ハードウェアレベルで動作するメモリ帯域幅スロットリング機構である MemGuard も、メモリ競合によるクリティカルタスクの実行時間の予測不可能な変動を排除するために使用されています [84]。 予測可能実行モデル（PREM）[72, 124]は、リソース競合によるタイミング遅延を防ぐ全く異なるアプローチである。プログラマによる注釈に基づき、実行ファイルにリソース使用関連情報を含めることで、低レベルな仲裁機構に依存せず高レベルでリソース競合を解決する。この手法により、生成された実行ファイルのリソースアクセス挙動は高度に予測可能となる。 最後に、冗長性と多様性といった広く用いられる技術もLinuxシステムで採用されている[68,70]。その考え方は、タイミングの異常値が冗長な実行によって覆い隠され、全ての冗長な実行が同時に失敗する確率は極めて低いというものだ。

6.2.1. スケジューリング

スケジューリングは、安全性が重要なシステムのタイミング保証において極めて重要な役割を果たす。そのため、これまでに多くの研究の焦点となってきた。

Linuxのデフォルトスケジューラは CFS です。これは汎用スケジューラであり、次にスケジュールされるタスクは、その重みに対してCPU時間をより少なく受け取ったタスクとなります。したがって、その目的はCPU時間を公平に分配することであり、デッドラインやレイテンシに対する決定論的な保証を提供することではありません。 このため、カーネルにはいくつかのリアルタイムスケジューリングポリシーも組み込まれています。SCHED_FIFOとSCHED_RRは固定優先度スケジューリングを実装し、常に最優先タスクを実行します。両者の違いは、前者がタスクがブロックまたはイールドするまで実行するのに対し、後者は同じ優先度のタスクに対してCPUタイムスライスを使用する点です。 SCHED_DEADLINEは、最優先期限先頭（EDF）スケジューリングを実装する別のリアルタイムスケジューリングポリシーである。ただし、このポリシーは優先度を使用しない。代わりに、期限、周期、およびタスクが出力を生成するために必要なCPU時間量である実行時間の3つのパラメータを使用する。これにより、タスクが正しくスケジューリングされることが保証され、

表7

LinuxをSCOSとして使用している調査対象作品の概要（その1）。

カテゴリ	目標	特性	Linux に
認証可能性	認証戦略	おける IEC 61508 の「実証済み使用」ルートに関する研究	[58]
		民間航空電子機器向けLinux認証のための文書化戦略	[56]
		Linux標準ベースは、大幅な作業なしでは原子力発電所には不十分	[57]
		Linux向けIEC 61508「非準拠開発の評価」ルート	[59]
		安全性とセキュリティは共同で管理すべきである	[21]
		リバースエンジニアリングがLinux認証における最も現実的な手法である	[61]
	機能拡張	安全基準に準拠したグラフィックスデバイスドライバの実装	[60]
	形式検証	Linuxカーネルのためのオートマトンモデルに基づく形式検証アプローチ	[55]
	取り組み	ELISAプロジェクトの概要	[62]
	テストカバレッジ	Linuxシステムコールの実行パスの最大数に関するノンパラメトリック推定	[6]
		Linuxシステムコールの実行経路の最大数のパラメトリック推定	[64]
		Linuxシステムコールの未検証実行経路発見の確率とリスク推定	[63]
		Linuxシステムコール実行パスにおけるストレスとファイルシステムの影響	[65]
		Linuxシステムコール実行パスにおけるACRNハイパーバイザーの影響	[66]
リアルタイムおよび決定論的タイミング特性	比較	2種類の異なるボードにおける標準版Linuxとリアルタイム Linuxの比較	[69]
		Dockerコンテナの有無による標準版とリアルタイムLinuxの比較	[79]
		バニラ版およびリアルタイムLinuxにおけるOROCOSとROS2の比較	[78]
		各種ボードおよび異なる負荷条件下における標準版とリアルタイムLinuxの比較	[81]
	スケジューリング	安全性と性能のバランスを取るスケジューリング指標	[92]
		リアルタイム・ワンギャング・アット・ア・タイム・スケジューリング	[91]
		熱環境を考慮したスケジューリング	[85]
		クリティカルタスクのタイミング要件を満たすための過負荷状態下でのスケジューリング	[90]
		モードおよび重要度を考慮したスケジューリング	[86]
		並列リアルタイムタスクのための混合重要度フェデレーテッドスケジューリングアルゴリズム	[89]
		リアルタイムタスクの耐障害性スケジューリング	[88]
		時間およびイベントトリガー型スケジューリング	[87]
	タイミング保証	予測可能な実行モデルをARMベースのヘテロジニアスプラットフォームに適用	[72]
		メモリ競合を予測し、重要度の低いタスクをスロットルする	[75]
		タイミング多様性、タイミング異常値を隠蔽する二重モジュラー冗長性	[70]
		タイミングダイバーシティ向け高速回復モデル	[68]
		Linuxシステムコール実行パスのpWCET推定	[67]
		CPU予算によるリソース予約	[74]
		リアルタイムコンテナオーケストレーター	[76]
		アクティブ監視によるタイミング違反の予測と回避策の実行	[77]
		CUDA、pthreads、Linux、およびQNXのPOSIXインターフェースにおけるタイミング遅延の原因を特定し軽減	[82]
		メモリ帯域幅制御システム	[84]
	タイミング測定	Linux実装に基づく新しいfutex実装	[73]
		bpftrace (eBPF) によるレイテンシ測定の可能性の検討	[71]
		リアルタイムLinuxおよびDocker環境下におけるネットワーク遅延の評価	[80]
		CUDAのタイミング不具合の原因を調査し、システム管理割り込みが原因であることを発見	[83]

表8

LinuxをSCOSとして使用した調査対象研究の概要（その2）。

カテゴリ	目標	特性	作品
フォールトトレランスとリカバリ	比較	Linuxとベアメタルにおける CPU レジスタのビット反転の影響の比較	[97]
		並列化あり/なしのマルチコア Linux におけるフォールトトレランス比較	[110]
		シングルコアベアメタルと並列化有無のマルチコア Linux 間のフォールトトレランス比較	[106]
	故障検出	ロボットシステムのための故障検出・診断スキーム	[103]
		複数の OS レベルリソースを監視し異常を検出することでアプリケーション障害を検知	[100]
	障害注入	CPU レジスタに故障を注入し、アプリケーションレベルでの影響を調査	[93]
		アプリケーションへの影響を最小限に抑えたソフトウェア障害注入	[94]
		アーキテクチャのエラー報告レジスタへの障害注入	[98]
		カーネルに属する任意のメモリアドレスでビット反転を実行	[99]
		アーキテクチャのエラー報告レジスタに障害を注入する（ハイパーバイザーおよび OS レベルで）	[95]
		ファイルシステムのモデルを含め、障害注入による境界条件ケースへの到達を可能にする	[104]
		ハザードと潜在的原因を特定するための SPTA 解析、およびハザードシナリオをシミュレートするソフトウェア障害注入フレームワーク	[96]
		カーネルパスをモデル化し、各状態にタイミング遅延を挿入 最もクリティカルな状態と許容可能な最大タイミング遅延を特定	[105]
		クリティカルなアプリケーションに特定の汎用 CPU レジスタプールを使用させる	[108]
	障害緩和	重要なアプリケーションに対し、2つの異なるアーキテクチャにおいて、汎用 CPU レジスタの特定のプールを使用させる	[107]
		障害注入、ソフトエラー結果とシステムアーキテクチャパラメータの相関分析のための機械学習、および緩和手法を含むフレームワーク	[109]
		2つのコアでの冗長実行用ライブラリ（多様性確保のための段階的実行を含む）	[111]
		ビット反転によるエラーを防止するための、ミューテックスのカウント変数に対する三重冗長化	[102]
		Linux スケジューラの構造体を修正し、ビット反転の影響を軽減する	[101]
Linux ベースの安全重要システム設計	実証機	オペレーター-コントローラー-モジュール概念に基づくアーキテクチャ	[113]
		Linux 上で実装された AUTOSAR Adaptive	[118]
		ハードウェアの完全仮想化による認証済みソフトウェアの再利用実現	[112]
	設計アプローチ	プラトーニングのための分散型・オブジェクト指向・コンポーネントベースアプローチ	[115]
		周期的制御システムのコンポーネントを動的に更新する方法	[120]
		SIL2LinuxMP に基づくガイドライン：Jailhouse、Linux、AI フレームワークを用いた RISC-V ボードにおける安全重要システムの構築	[119]
		ホストとして KVM、重要タスクに AUTOSAR、汎用タスクに Linux/Android を採用	[116]
		Android デバイスを用いた安全重要機能実装ガイドライン	[121]
		論理的分離：ある機能が他の機能の動作によって予期せぬ挙動を引き起こされない状態	[117]
		自動運転車における最近の自動車開発におけるデザインパターン	[114]

スケジューラがタスクの追加を許可する場合、システム内で他のタスクが実行中であっても、各タスクが必要とするCPU時間を把握しているため、それ以上のタスク追加を拒否することが可能です。

これらのスケジューラがLinuxをGPOSからRTOSへと近づける一方で、カーネルの非プリエンプティブ領域やシステムサービス・割り込みによる干渉が、たとえこれらのスケジューラを使用してもLinuxをRTOSとしないことを留意すべきである。PREEMPT_RTパッチがLinuxをRTOSとするために修正する点がまさにこれらである。

この文脈において、安全性が極めて重要なシステムでの利用に向けて、Linuxのスケジューリング機能を修正または拡張する取り組みが存在する。いくつか著者らは、安全上重要なタスクのためにCPUコアを予約するスケジューリング手法を提案している[89,91]。他方、スケジューラが考慮すべき特定の指標の使用を主張する研究もある。例えば、適時計算に基づく新たな安全性能指標（SP指標）[92]や、過負荷耐性指標 *であるダクティリティ*[90]などである。 *延性*とは、過負荷時にタスクを完全に放棄するのではなく、機能を段階的に低下させることで低重要度タスクの実行を継続するシステムの能力を指す。

最後に、安全クリティカルシステムにおけるタスクスケジューリング時に二次的側面を考慮に入れる提案がなされている。その中には、プラットフォームの過熱を防止できる熱感知型スケジューリングプラットフォーム[85]や、考慮可能なスケジューラが含まれる。

マルチモーダル混合重要度システムにおけるモードと重要度の両方の変更[86]、複数のリソース指向ヒューリスティックに基づく同期と信頼性を共同管理するスケジューリング手法[88]、および時間トリガーとイベントトリガーのスケジューリングを統合するスケジューラ[87]。著者らは、これらが安全重要システムにおいてしばしば両方必要であると主張している。

6.3. フォールトトレランスとリカバリ

安全クリティカルシステムは、他のシステムと同様にランダムな故障の影響を受けやすい。しかし安全クリティカルシステムでは、こうした故障が人間や環境の安全に壊滅的な結果をもたらす可能性があるため、その影響を軽減する技術が必要である。Linuxベースの安全クリティカルシステムも例外ではなく、多くの研究者がLinuxベース安全クリティカルシステムにおけるこうしたエラーの影響軽減に注力している。 この観点から、我々は以下の4つの重点領域を特定する：

- *フォールトトレランス*：研究では、LinuxシステムとLinux非搭載システムにおけるランダムエラーの影響を比較し、Linuxがそのようなエラーがシステムに影響を与えるのを阻止する能力を調査することを目的としている[97, 106,110]。これら研究はいずれも、ベアメタルシステムはLinuxシステムよりもエラーが発生しやすく、Linuxが生成する例外処理がエラーの伝播や他部位での顕在化を阻止するのに有用である点で一致している。これはベアメタルシステムで生じる現象である。
- *障害検出*：著者らはLinuxシステムにおける障害検出手法を提案している。これらはOSレベルのリソースを監視して異常を探知する方法[100]と、コンポーネントベースのロボットシステム向け診断スキーム[103]から構成される。
- *障害軽減*：障害軽減技術は複数の研究者によって提案されている。その中には、Linuxカーネル内の特定情報フィールドの冗長性に基づく手法[101, 102]や、重要アプリケーションの冗長かつ多様な実行に基づく手法[111]が含まれる。他の研究者は異なるアプローチを採用し、重要アプリケーションにプロセッサレジスタの特定プールを使用させることを強制している。 この手法はレジスタ割り当て技術（RAT）[107,108]と呼ばれている。最後に、SOFIA [109]は、ソフトエラー結果とシステムアーキテクチャパラメータを相関させるための故障注入機能と機械学習、ならびに故障緩和技術（具体的には三重モジュラー冗長性（TMR）とRAT）の実装を含むフレームワークである。
- *障害注入*：Linuxシステムにおける障害注入のフレームワークと技術は、広範な研究対象となってきた。著者らが提案したフレームワークの一部は、基盤となるハードウェア（特にアーキテクチャのエラー報告レジスタ（MCA））への障害注入に依存している[93,95,98]。他のフレームワークはソフトウェアエラーを注入し、Linuxカーネルモジュールとして構築されている[94,99]。 最後に、モデルベースの障害注入手法を提案する著者もいる。これらの提案の中には、システム理論的プロセス解析（STPA）を用いて危険要因とエラーの潜在的原因を特定し、それらの危険シナリオをシミュレートするもの [96]、ファイルシステムのモデルを故障注入プロセスに組み込みコーナーケース到達を可能にする手法[104]、カーネルパスをモデル化し各状態にタイミング遅延を挿入することで最もクリティカルな状態と許容可能な最大タイミング遅延を特定する手法[105]などが挙げられる。

6.4. Linuxベースの安全重要システム設計

安全重要認証は個々のコンポーネントに対してではなく、システム全体に対して付与される。したがって、安全重要システムにおけるLinuxの使用には、単に「安全な」Linuxバージョンだけでなく、ハードウェア、システム設計、ミドルウェアなど、より広範な要素が必要となる。この観点から、Linuxベースの安全重要システムを構築する方法に関する提案がなされてきた。

SIL2LinuxMP [125] 論理分離アーキテクチャは、そのようなシステムを構築するための基盤として提案されている [117,119]。このアーキテクチャは Linuxカーネルの機能を利用して、Linux上で（論理的に）分離されたパーティションを作成し、安全上重要な機能と汎用機能を分離する。論理的分離とは、空間的・時間的分離を必ずしも意味せずとも、ある機能が他の機能の動作によって予期せぬ（危険な）挙動に陥らないことを指す。この論理的分離とハイパーバイザーレベルでの分離（Jailhouseによる）を組み合わせる手法も提案されている[119]。 Messnarzら[114]による、最近の自動運転車開発に見られる設計パターン研究に基づく予測も同様の方向性を示唆している。彼らは将来、車両の各構成要素（ステアリング、モーターなど）が、車両の中央コンピュータとして機能し運転入力を生成するLinuxサーバー上で動作するアプリによって制御されると予測している。 特定の目的、特に自動車システムに焦点を当てたLinuxベースのシステム設計提案も存在する。これには、安全上重要な機能を実装するためのAndroidシステム利用ガイドライン[121]、車両隊列走行のためのソフトウェア設計[115]、テレマティクスゲートウェイのアーキテクチャ[116]、動的更新 [120]。

最後に、安全上重要なシステムにおけるLinuxの能力を示すデモ機を構築した著者もいる。

- ラジコンカーにおける先進運転支援システム（ADAS）アプリケーション [113]。
- 同一プラットフォーム上でADASとIVIを実行するためのLinux上のAutosar Adaptiveの実装 [118]。
- 完全な仮想化をサポートし、Linux カーネルの拡張として実装されたタイプ II ハイパーバイザー [112]。これは、将来のスタンドアロンハイパーバイザーの概念実証である。

7. 産業アプローチ

安全産業におけるLinuxをSCOSとして活用する関心の高まりを受け、様々なプロジェクトやワーキンググループが誕生している。本節では、LinuxをSCOSとして活用する取り組みに焦点を当て、現在この目的でLinuxを採用している産業アプローチを検証することで、調査範囲を拡大する。本節では学術プロジェクト以外の取り組みも扱うため、第4節で概説した方法論の範囲外となる。

重要な取り組みの一つが、Linux FoundationのELISAプロジェクトである。[126]（安全アプリケーションにおけるLinuxの活用）は、「企業がLinuxベースの安全重要アプリケーションを構築・認証しやすくすること」を目的としている。この共同取り組みは、航空電子機器分野のボーイング、ボッシュ・ホンダ・日産などの自動車メーカー、NASAやEASA（欧州航空安全機関）などの航空宇宙機関を含む主要組織によって支援されている。目標は、ベストプラクティスの確立とLinuxの堅牢性向上により、安全上重要なアプリケーションへの適性を確保することである。さらに、次世代航空宇宙、自動車、医療機器に焦点を当てた取り組みを、各分野別のワーキンググループが主導している。

この取り組みに先立ち、オープンソースオートメーション開発研究所（OSADL）が主導する SIL2LinuxMP プロジェクト [125] は、COTS マルチコアプラットフォーム上で動作する組み込み GNU/Linux システムの本質的な最小構成要素（ブートローダ、ルートファイルシステム、カーネル、C ライブラリインターフェース）の認定に焦点を当てていました。

本プロジェクトは、複雑な安全関連システムをSIL2安全規格IEC61508に適合させることを目的とし、特に既存ソフトウェアのルート3s（IEC61508-2 第2版 第7.4.2.2項）に基づく適合性達成を目標とした。 SIL2LinuxMPプロジェクトは、時間的・資源的分離を重視したSIL2アーキテクチャを提案し、構成要素の多様性を活用して保護層分析（LOPA）による保証を提供する。

ELISAとSIL2LinuxMPはいずれもLinuxをSCOSとして対象とし、*認証可能性を*主な重点としている。しかし現在までに、いずれの取り組みもLinuxベースシステムの完全な認証を実証していない。特にELISAは、SIL2LinuxMPが築いた基盤を基盤とし、SPCベース認証手法や戦略といった学術的進歩を取り入れることが可能であり

論拠や戦略といった学術的進歩を取り入れることが可能であり、これらは特別関心グループ（SIG）内でさらに発展させられるだろう。

自動車業界は、表3に列挙された研究件数からも明らかなように、SCOSとしてのLinux活用に特に強い関心を示している。Automotive Grade Linux（AGL）[127]イニシアチブはLinux Foundationの一環であり、「自動車メーカー、サプライヤー、テクノロジー企業を結集し、コネクテッドカー向け完全オープンソフトウェアスタックの開発と採用を加速する」ことに焦点を当てている。 このプロジェクトはトヨタ、メルセデス・ベンツ、フォルクスワーゲンなど主要自動車メーカーの支援を受けている。AGLは車載ソフトウェアの多様な応用分野、特に先進運転支援システム（ADAS）、機能安全、自動運転に焦点を当てており、Linuxがその中核コンポーネントとして機能している。

テスラは自動運転システムの基盤OSとしてLinuxを採用している[128]。現在利用可能な機能は、J3016ガイドライン[129]に基づき、最高でSAEレベル2に分類される。このガイドラインでは、運転自動化のレベルを0（AEBなどの支援機能）から5（あらゆる条件下で自律走行する状態）まで分類している。 テスラのオートパイロットは、自動車分野におけるLinuxの大規模導入事例として最も注目されている。SAEレベル2の自動化ではその有効性が明らかだが、より高いレベルへの拡張には、より厳密なタイミング保証と安全性の実証が必要となる。冗長性やスケジューリングに関する学術的な提案は潜在的な技術を提供しているが、テスラの独自システムのため、有効性の直接比較は困難である。

エレクトロビットのEB corbos Linux for Safety Application [130]は、自動車向け高性能コンピューティング（HPC）システム向けにISO 26262 ASIL-B/D認証を取得するために設計されたUbuntuベースのOSです。このOSは、CA TÜVによって評価された安全アーキテクチャ上に展開され、ASIL-B対応のEB corbosハイパーバイザーを活用して、決定論的スケジューリング、ハードウェア仮想化、メモリ保護を保証します。 システム完全性を維持するため、EB corbos Linux for Safety Applicationsは特定のユーザースペース初期化、厳格なアプリケーションメモリ保護、およびカーネルによるユーザースペース変更を防止する監視措置を備えています。さらに、プロセッサコンテキストスイッチを監視し、意図しない変更から保護します。 このアプローチは、セクション5で概説したように、安全パーティションの分離・監視・モニタリングを強制するためにハイパーバイザーと組み合わせて*Linuxを汎用OSとして*使用する手法に似ているものの、EB corbos Linuxの特異性は、安全パーティション自体に安全機能を担うLinuxカーネルが含まれている点にある。ハイパーバイザーは監視と故障封じ込めを強化する一方で、複雑性、コスト、認証における潜在的なリスクを追加する。 したがって、EB corbos Linux*はLinuxベースの安全重要システム設計*に関する学術提案への一歩を示すものの、Linuxを直接SCOSとして採用する上で依然として存在するギャップを浮き彫りにしている。

Linuxのもう一つの認証済みバージョンとして、Codethink Trustable Repro-ducible Linux（CTRL OS）[131]がある。これはCA Exidaからベースライン安全評価を取得している：「CTRL OSに適用されたプロセスフレームワークの評価により、SIL 3におけるIEC 61508の関連安全要件が満たされ、このベースライン安全ケース評価によりプロセス適合性の論証が完了していることが示された」詳細は多く明かされていないが、CTRL OSはLinuxカーネル、systemd、glibc、gccなどフリーかつオープンソースのコンポーネントとツールで構築されたLinuxベースのOSおよび関連ソフトウェア開発キットである。マルチコアマイクロプロセッサを基盤とする重要度が高いシステムおよび／または混合重要度システムへの統合を想定して設計されている。

CTRL OSは、仮想化などの追加コンポーネントに依存しないプロセス保証に焦点を当て、*LinuxをSCOSとして*位置付ける取り組みである。 この意味で、特定された課題のいくつかに対処する点で、第6節で概説した学術研究と一致している。認証可能性をサポートするフレームワークを提供し、Linux自体に直接依存する設計アプローチを推進する。しかしながら、実際の完全な認証プロセスを検証するエンドツーエンドのユースケースはまだ不足している。

航空宇宙分野にも専用のソリューションが存在する。Space Grade Linux [132] はELISAイニシアチブの一部であり、特別な機能として関心グループ。その目標は、決定論的リアルタイム処理、低遅延、耐障害性などの機能を備えたLinuxベースのSCOSを設計することである。ドイツ宇宙運用センター（GSOC）は宇宙船運用センターにLinuxを利用しており、NASAはインジェニュイティ火星ヘリコプターのGPOSとして採用した[133]。 このケースでは、Linuxは混合重要度アーキテクチャ内のQualcomm Snapdragon 801コアを基盤とする航法コンピュータで利用されている。安全上重要な操作は、デュアルロックステップ機能を備えたARM Cortex-R5プロセッサによって管理される[134]。 SpaceXもまた、Dragon 9宇宙船にLinuxベースの飛行ミッションクリティカルコンピューターを組み込んでいる。このシステムは3つのデュアルコアx86プロセッサで構成されるアーキテクチャに依存しており、各コアは独立したLinuxインスタンス上で同一の計算を個別に実行する。これらの計算結果は、何らかのアクションが取られる前に投票される。

航空分野では、安全性が極めて重要なリアルタイムOSにおける空間・時間分割にARINC 653規格を採用している。ARINC 653は、パーティション間の分離を強制しパーティション間通信をサポートするため、APEX（アプリケーション・エグゼクティブ）と呼ばれる分割メカニズムとAPIを規定している。 実際の運用では、このような混合重要度環境においてLinuxが汎用オペレーティングシステム（GPOS）として頻繁に採用される。例えばSysgo PikeOSやWind River VxWorks 653といった商用ソリューションでは、分離カーネルによる干渉防止を保証しつつ、ARINC 653準拠パーティションと並行してLinuxパーティションをホスト可能である。 こうした進歩にもかかわらず、航空電子機器においてLinuxをSCOSとして使用するには、特に時間的・空間的分離の保証、共有リソース上の干渉の制限、認証のための十分な保証の提供といった重大な課題が残っており、これらは第6節で議論される中心的な問題である。 学術的な取り組みでは、Linux ベースの ARINC 653 対応パーティションおよびスケジューラ [85、135、136]、ハイパーバイザーベースのアプローチ [137] などを提案し、これらの課題の解決を試みている。

防衛分野において、Redhawk Linux RTOS [138] はミッションクリティカルなアプリケーション向けの商用オープンソースSCOSとして提案されている（すなわち、第2節で記述される安全規格に基づく安全クリティカルではない）。

ミッションクリティカルなアプリケーション向けの商用オープンソースSCOSとして提案されている（すなわち、第2節で説明する安全規格に基づく安全クリティカルではない）。Linuxベースでありながら、特定の最適化プラットフォーム上で決定性を保証し、5マイクロ秒未満のレイテンシを維持すると主張している。米海軍のイージス兵器システムなど、同海軍の様々な防衛プログラムに広く導入されている。

Linuxを基盤とし、特定の最適化プラットフォーム上で決定性を保証し、5マイクロ秒未満のレイテンシを維持すると主張している。米国海軍のイージス兵器システムなど、様々な防衛プログラムに広く導入されている。

このシステムは、ロッキード・マーティン社が主導しています。さらに、カーチス・ライト社は、同社の Partvus DuraCOR 312 ミッションコンピュータにも同じ RTOS を実装しています。

8. 安全上の課題における Linux

これまでのセクションでは、安全関連システムにおける Linux の用途と、SCOS としての Linux を使用する場合の課題について確認してきました。このセクションでは、以下の研究課題 (RQ) について答えを出します。

RQ 学界および産業界における既存のソリューションは、SCOS としての Linux の採用に関する主要な課題にどのように対処しており、どのようなギャップが残っているのか？

第5節、第6節、および第7節では、学術研究と産業研究の両方を検討することで、安全重要システムにおけるLinux利用の現状課題を特定した。これらの課題から、安全関連システム内でのLinux利用における以下の主要な課題を抽出した。

(1) *認証可能性：*SCOSは通常、システムの認証を確保するための認証パッケージを提供する（例：Greenhills Integrity RTOS SIL3事前認証済みカーネル）。認証パッケージがなくても、LinuxをSCOSとして使用することを正当化する手法は文献で見出せる。 例えば、SIL2LinuxMPプロジェクトは、IEC 61508のルート3に従うことでLinuxをSIL 2レベルまで認証可能と提案している。さらに、Allendeら[6,63,64,117]の研究およびそれを発展させた研究[65–67]は、統計的または確率論的手法を今後の道筋として提案している。 最後に、産業用Linuxの2つのバージョン、EB Corbos LinuxとCTRL OSが安全上重要な認証取得を主張している点に留意することが重要である。

(2) *リアルタイム性と時間的決定性：*セクション2.3で説明したように、安全性が極めて重要なシステムでは時間的決定性を確保することが鍵となる。これは、いかなるタイミング遅延も回避するために、リソースに対する決定論的なスケジューリングと制御を必要とする。 この要件を Linux に導入するには、Linux 完全プリエンプティブカーネル (PREEMPT_RT)、共同カーネルアプローチ (Xenomai、RTAI など)、またはカスタマイズされた Linux カーネル (Redhawk Linux など) を使用することで実装されています。 プラットフォームに大きく依存するものの、最新のデータでは、プリエンプティブカーネルは共同カーネルアプローチに比べて 10 倍以上の高いジッタを示している。

(3) *耐障害性：*安全上重要なシステムではランダムなエラーが壊滅的な結果を招く可能性があるため、Linuxベースの安全重要システムにおける障害は徹底的に研究されてきた。文献では主に四つの側面が焦点となっている：(i) 耐障害性：Linuxシステムと Linux非搭載システムの耐障害性を比較した研究では、Linuxシステムがベアメタルシステムよりもエラー発生率が低いことが判明している。 (ii) 検出と (iii) 緩和：著者らは障害を監視・停止するためのLinuxカーネルの修正を提案している。また、特定のプロセッサレジスタプールを強制的に使用させることで、重要アプリケーションとハードウェアの相互作用方法への修正を提案する著者もいる。 最後に、(iv) 注入では、ハードウェアレベルとソフトウェアレベルの両方で障害が注入されており、形式手法やモデリングが含まれる場合もある。この点に関して、4つの側面すべてに対処できるツールや手法を作成することが主な課題であり、一部の著者によってすでに試みられている[109]。

(4) システム*分割と重要リソースへのアクセス：*混合重要度プラットフォームでは、ハイパーバイザー、デュアルカーネル、異種ハードウェアなど、共有ハードウェアへのアクセスを保証するメカニズムを用いてシステムの重要部分を隔離する。 したがって、LinuxをSCOSとして使用するには、Linux外部または内部の分離メカニズムが必要となる。これらは、同一プラットフォーム上でLinuxをGPOSとして、別のSCOSを適切に分離して実行するために広く利用されている。ただし、これらの手法（特にハイパーバイザー）は、SCOSとして動作するLinuxを別のGPOS（または別のLinuxインスタンス）から分離するためにも用いられる。 コンテナベースの論理的分離も、同一Linuxカーネル上で動作する混合重要度システムを構築する可能性のある手法として提案されている。自動車分野では、Elektrobit社のEB corbos Linuxが、システムを分割し、品質管理（QM）用と安全重要タスク用の2つのLinuxカーネルを実行可能にする独自開発のハイパーバイザーを基盤としている。

9. 結論

本文献レビューでは、安全重要システムにおけるLinuxの活用に焦点を当てた学術研究の概要を提示した。研究対象を、システム内におけるLinuxの役割に基づき分類し、同一システム内の別の安全重要部分から分離された汎用オペレーティングシステム（GPOS）としてLinuxを利用する研究と、安全重要機能を実装するためにLinuxを利用する研究とを区別した。 さらに、Linuxの焦点となる側面に沿って研究をグループ化し、機能安全におけるLinux利用の主な課題を特定した。LinuxをGPOSとして使用する場合、SCOSからの分離が主要な課題である。これを実現するために、研究では(i)ハイパーバイザー、(ii)マルチカーネル手法、(iii)ハードウェアベースの分離が用いられていることを確認した。 一方、Linuxを安全重要OSとして使用する場合、多くの側面が研究されており、我々はこれを(i)認証可能性、(ii)リアルタイム性と決定論的タイミング特性、(iii)耐障害性と回復性、(iv)Linuxベースの安全重要システム設計に分類した。さらに、安全重要システムにおけるLinux活用を試みる主要な産業アプローチを概説した。 最後に、文献および産業の取り組みから、安全クリティカルシステムにおけるLinux利用に関する主な課題を特定した。

CRediT著作者貢献声明

Markel Galarraga: 原稿作成（原稿草稿）、調査、原稿作成（校閲・編集）。**Charles-Alexis Lefebvre:** 原稿作成（校閲・編集）、原稿作成（原稿草稿）、調査。**Jon Perez-Cerrolaza:** 原稿作成（校閲・編集）。**Jose A. Pascual:** 原稿作成（校閲・編集）。

利益相反に関する宣言

著者らは、本論文で報告された研究に影響を与えた可能性のある既知の競合する金銭的利害関係や個人的関係がないことを宣言する。

謝辞

本研究は、プロジェクトMEDUSA（CER-20231011、CERVERA卓越ネットワーク、スペイン科学・革新・大学省より欧州連合復興・レジリエンスメカニズム（RRM）を通じて技術開発・革新公社（CDTI）が資金提供）およびバスク自治州政府経済開発・インフラ局の支援を受けています。
政府（エマイテックプログラム及びプロジェクトKK-2023/00012、KK-2023/00090、KK-2024/00030、KK-2024/00068、並びに統合グループ助成金IT1504-22）。また、MICIU/AEI/10.13039/5011により資金提供された助成金PID2023-152390NB-I00の支援を受けています。
22）。また、MICIU/AEI/10.13039/501100011033 および「FEDER 資金」による助成金 PID2023-152390NB-I00 によっても支援されています。

データ利用可能性

本論文で記述された研究にはデータは使用されなかった。

参考文献

[1] 国際電気標準会議、IEC 61508 (1-7): 電気/電子/プログラム可能な電子安全関連システムの機能安全（第2版、 版）、2010年。

[2] 国際標準化機構、ISO 26262 (1-10): 道路車両
— 機能安全、2018年。

[3] J. Perez-Cerrolaza, J. Abella, M. Borg, C. Donzella, J. Cerquides, F.J. Cazorla, C. Englund, M. Tauber, G. Nikolakopoulos, J.L. Flores, 産業および輸送分野における安全重要システムのための人工知能：調査報告, ACM Comput. Surv. 56 (7) (2024) http://dx.doi.org/10.1145/3626314.

[4] J. Perez-Cerrolaza, R. Obermaisser, J. Abella, F.J. Cazorla, K. Grüttner, I. Agirre, H. Ahmadian, I. Allende, 安全重要システム向けマルチコアデバイス：調査報告, ACM Comput. Surv. 53 (4) (2020) http://dx.doi.org/10.1145/3398665.

[5] J. Perez-Cerrolaza, J. Abella, L. Kosmidis, A.J. Calderon, F. Cazorla, J.L. Flores, 安全クリティカルシステム向けGPUデバイス：調査報告, ACM Comput. Surv. 55 (7) (2022) http://dx.doi.org/10.1145/3549526.

[6] I. アジェンデ、N.M. ギレ、J. ペレス＝セロラサ、L.G. モンサルベ、J. ピーターソン、R. オーバーマイサー、Linuxベース次世代自律安全関連システムのための統計的テストカバレッジ、IEEE Access 9 (2021) 106065–106078, http://dx.doi. org/10.1109/ACCESS.2021.3100125.

[7] The Linux Foundation, 2021 Linux foundation annual report. New Hori-zons for Open Source, 2021, https://www.linuxfoundation.org/resources/ publicationslinux-foundation-annual-report-2021. (最終アクセス日: 2025年3月27日).

[8] A. Platschek, N. Mc Guire, L. Bulwahn, 「Linuxの認証：SIL2LinuxMPにおける3年間の教訓」, Embedded World, 2018年.

[9] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, 耐障害性と安全性を備えたコンピューティングの基本概念と分類法, IEEE Transactions on Dependable and Secure Computing 1(1) (2004) 11–33, http://dx.doi.org/10.1109/TDSC.2004.2.

[10] F. Reghenzani, G. Massari, W. Fornaciari, リアルタイムLinuxカーネル：PREEMPT_RTに関する調査, ACM Comput. Surv. 52 (1) (2019) http://dx.doi.org/10.1145/3297714.

[11] V. Struhár, M. Behnam, M. Ashjaei, A.V. Papadopoulos, リアルタイムコンテナ：調査、Fog Computing and the Internet of Things ワークショップ、2020年、 URL https://api.semanticscholar.org/CorpusID:216086064.

[12] S. Lozano, T. Lugo, J. Carretero, 安全上重要なシステムにおけるハイパーバイザーの使用に関する包括的調査, IEEE Access 11 (2023) 36244–36263, http://dx.doi.org/10.1109/ACCESS.2023.3264825, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85153367542&doi=10.1109%2fACCESS.2023.3264825&partnerID=40&md5=b400a143ab34e61934623d2fe545d8cd.

[13] M. Kassab, Testing practices of software in safety critical systems: Industrial survey, in: ICEIS 2018 - Proceedings of the 20th International Conference on Enterprise Information Systems, vol. 2, 2018, pp. 359–367, http://dx.doi.org/10.5220/0006797003590367, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85047779106&doi=10.5220%2f0006797003590367&partnerID=40&md5=e27b9f16b26c56c372cea91f8fd5bcd6.

[14] J. Pedersen Notander, M. Höst, P. Runeson, 柔軟な安全重要ソフトウェア開発における課題 - 産業的定性調査, Lect. Notes Comput. Sci. (人工知能講座・バイオインフォマティクス講座を含む) 7983 LNCS (2013) 283–297, http://dx.doi.org/10.1007/978-3-642-39259-7_23, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84884946145&doi=10.1007%2f978-3-642-39259-7_23&partnerID=40&md5=5e8e0bfb802853c4166671dc9099956bf5.

[15] B. Paden, M. Čáp, S.Z. Yong, D. Yershov, E. Frazzoli, 自動運転都市車両のための運動計画および制御技術の概説, IEEE Trans. Intell. Veh. 1 (1) (2016) 33–55, http://dx.doi.org/10.1109/TIV.2016.2578706, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041966386&doi=10.1109%2fTIV.2016.2578706&partnerID=40&md5=83612f8453c213d8ea4497c362b8df51.

[16] M. Rabe, S. Milz, P. Mader, 自動車分野における安全クリティカルな機械学習アプリケーションの開発方法論：調査報告, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2021, pp. 129–141, http://dx.doi.org/10.1109/CVPRW53098.2021.00023, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85115853735&doi=10.1109%2fCVPRW53098.2021.00023&partnerID=40&md5=c8f2f9f704b204d0365c6541521830 66.

[17] A . Boglietti, A. Cavagnino, A. Tenconi, S. Vaschetto, 安全上重要な電気機械と駆動装置：より電気化された航空機における現状調査, IECON Proceedings (Industrial Electronics Conference), 2009, pp. 2587–2594, http://dx.doi.org/10.1109/IECON.2009.5415238, URL https://www.scopus. com/inward/record.uri?eid=2-s2.0-77951645264&doi=10.1109%2fIECON.2009.5415238&partnerID=40&md5=25ac2eec58231269d9c82c4e936c833b.

[18] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, Y. Halgand, 産業制御システムにおける安全性とセキュリティを統合するアプローチの概観, Reliab. Eng. Syst. Saf. 139 (2015) 156–178, http://dx.doi.org/10.1016/j.ress.2015.02.008, URL https://www.sciencedirect.com/science/article/pii/S0951832015000538.

[19] E. Lisova, I. Šljivo, A. Čaušević, Safety and security co-analyses: A systematic literature review, IEEE Syst. J. 13 (3) (2019) 2189–2200, http://dx.doi.org/10. 1109/JSYST.2018.2881017.

[20] G. Kavallieratos, S. Katsikas, V. Gkioulos, サイバーフィジカルシステムのサイバーセキュリティと安全の共同設計—包括的調査, Futur. Internet 12 (4) (2020) http://dx.doi.org/10.3390/fi12040065, URL https://www.mdpi. com/1999-5903/12/4/65.

[21] G. Procopio, 「GNU/Linuxリアルタイム OS領域における安全性とセキュリティ」, P. Ciancarini, M. Mazzara, A. Messina, A. Sillitti, G. Succi (編), 『第6回防衛応用ソフトウェア工学国際会議論文集』, Springer International Publishing, Cham, 2020, pp. 245–254.

[22] A. Avanzini, P. Valente, D. Faggioli, P. Gai, 「Xenハイパーバイザーを介したLinuxとリアルタイムERIKA OSの統合」, 第10回IEEE産業用組込みシステム国際シンポジウム（SIES）, 2015年, pp. 1–7, http://dx.doi.org/10.1109/SIES.2015.7185 pp. 1–7, http://dx.doi.org/10. 1109/SIES.2015.7185063.

[23] A. Biondi, D. Casini, G. Cicero, N. Borgioli, G. Buttazzo, G. Patti, L. Leonardi, L.L. ベッロ、M. ソリエリ、P. ブルージオ、I.S. オルメド、A. ルオッコ、L. パラッツィ、M. ベルトニャ、A. チラード、N. マッツォッカ、A. マッツェオ、SPHERE: ヘテロジニアスプラットフォームに基づく次世代サイバーフィジカルシステムのためのマルチSoCアーキテクチャ、IEEE Access 9 (2021) 75446–75459, http://dx.doi.org/10.1109/ACCESS.2021.3080842.

[24] A. Farrukh、R. West、FLYOS：自律型マルチコプター向け統合モジュラー航空電子機器、2022 IEEE 第28回リアルタイム・組込み技術・応用シンポジウム（RTAS）、2022年、68–81頁、http://dx.doi.org/10.1109/RTAS54340.2022.00014。

[25] E. Cittadini、M. Marinoni、A. Biondi、G. Cicero、G. Buttazzo、ハイパーバイザー技術による異種プラットフォーム上の AI 搭載リアルタイムサイバーフィジカルシステムのサポート、Real-Time Syst. 59 (4) (2023) 609–635、http://dx.doi。 org/10.1007/s11241-023-09402-4。

[26] A. Farrukh, R. West, FlyOS: 自動マルチコプター向け統合モジュラー航空電子機器の再考, Real-Time Syst. 59 (2) (2023) 256–301, http://dx.doi. org/10.1007/s11241-023-09399-w.

[27] A. Gonzalez, W. Mata, A. Crespo, M. Masmano, J.M. Felix, A. Aburto, リアルタイム安全重要組み込みJavaアプリケーションをサポートするハイパーバイザーベースプラットフォーム, コンピュータシステム科学工学 28 (2013) URL https://api.semanticscholar. org/CorpusID:8668879.

[28] L. Belluardo, A. Stevanato, D. Casini, G. Cicero, A. Biondi, G. Buttazzo, 安全でセキュアな自動運転のためのマルチドメインソフトウェアアーキテクチャ, 2021 IEEE 第27回組み込み・リアルタイムコンピューティングシステムと応用国際会議 (RTCSA), 2021, pp. 73–82, http://dx.doi.org/10.1109/ RTCSA52859.2021.00017.

[29] C. Li, R. Guo, X. Tian, H. Wang, KHV: KVMベースのヘテロジニアス仮想化, Electronics 11 (16) (2022) http://dx.doi.org/10.3390/electronics11162631, URL https://www.mdpi.com/2079-9292/11/16/2631.

[30] J. マーティンズ、A. タヴァレス、M. ソリエリ、M. ベルトニャ、S. ピント、Bao: 現代のマルチコア組込みシステムのための軽量静的パーティショニングハイパーバイザー、M. ベルトニャ、F. テラネオ (編)、次世代リアルタイム組込みシステムに関するワークショップ (NG-RES 2020)、Open Access Series in Informatics (OASIcs)、77、Schloss Dagstuhl – Leibniz-Zentrum für Informatik、Dagstuhl、ドイツ、2020年、pp. 3:1–3:14, http://dx.doi.org/10.4230/OASIcs.NG-RES.2020.3, URL https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.NG-RES.2020.3 .

[31] R. Ramsauer, J. Kiszka, W. Mauerer, 混合重要度システムのための新規ソフトウェアアーキテクチャ, S. Keil, R. Lasch, F. Lindner, J. Lohmer (編), 半導体製造におけるデジタルトランスフォーメーション, Springer International Publishing, Cham, 2020, pp. 121–128.

[32] J. Schneider, 混合重要度システムにおける相互運用性の障壁の克服, J. Stjepandić, G. Rock, C. Bil (編), 多分野環境における持続可能な製品開発のための並行エンジニアリングアプローチ, Springer London, ロンドン, 2013, pp. 1093–1104.

[33] S. Patrick, 先進車載システム：未来に向けたリファレンスデザイン, ISBN: 01487191, 2016, http://dx.doi.org/10.4271/2016-01-0085.

[34] S. Sinha, R. West, DriveOSにおける統合車両管理システムへの取り組み, ACM Trans. Embed. Comput. Syst. 20 (5s) (2021) http://dx.doi.org/ 10.1145/3477013.

[35] F. Caforio, P. Iannicelli, M. Paolino, D. Raho, VOSySmonitoRV: Linux対応RISC-Vプラットフォームにおける混合重要度ソリューション, 2021年 第10回地中海組込みコンピューティング会議 (MECO), 2021年, pp. 1–4, http://dx.doi.org/10.1109/MECO52532.2021.9460246.

[36] J. Vetter, J. Fanguede, K. Chappuis, D. Raho, VOSYSVirtualNet: 混合重要度システム向け低遅延インターワールドネットワークチャネル, 2018 IEEE 第13回産業用組込みシステム国際シンポジウム (SIES), 2018, pp. 1–9, http://dx.doi.org/10.1109/SIES.2018.8442097.

[37] A. Farrukh, R. West, JuMP2start: ソフトウェア定義車両システム向け時間認識型ストップ・スタート技術, R. Pellizzoni (編), 第36回ユーロマイクロリアルタイムシステム会議（ECRTS 2024）、ライプニッツ国際情報学会議録（LIPIcs）第298巻、シュロス・ダグシュトゥール – ライプニッツ情報学センター、ドイツ・ダグシュトゥール、2024年、pp. 1:1–1:27, http://dx.doi.org/10. 4230/LIPIcs.ECRTS.2024.1, URL https://drops.dagstuhl.de/entities/document/ 10.4230/LIPIcs.ECRTS.2024.1.

[38] F. Lesniak, T. Harbaum, J. Becker, Xenハイパーバイザー上での低遅延ドメイン間通信, 2023 IEEE 16th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC), IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 340–346, http://dx.doi.org/10.1109/ MCSoC60832.2023.00057, URL https://doi.ieeecomputersociety.org/10.1109/ MCSoC60832.2023.00057.

[39] K. Kim, 「自動車プラットフォームにおけるハイパーバイザーベースのゲストOSと安全RTOSのアーキテクチャ設計」, 2023年適応・融合システム研究国際会議(RACS '23)論文集, 計算機学会(ACM), ニューヨーク州ニューヨーク, 米国, 2023年, http://dx.doi.org/10.1145/ 3599957.3606220.

[40] H. Shen, A. Charlet, T.P. Baker, Linux カーネル下における Ada マルチタスクの「ベアメタル」実装, M. González Harbour, J.A. de la Puente (編), Reliable Software Technologies — Ada-Europe' 99, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 287–297.

[41] M. Barletta, M. Cinque, R.D. Corte, 混合重要度システムにおけるリアルタイムコンテナのための階層的スケジューリング, 2021 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW, 2021, pp. 286–287, http://dx.doi.org/10.1109/ISSREW53611.2021.00082.

[42] S. Ghaisas, G. Karmakar, D. Shenai, S. Tirodkar, K. Ramamritham, SParK: 統合リアルタイムシステムのための安全パーティションカーネル、K. Sachs, I. Petrov, P.E. Guerrero (編)、アクティブデータ管理からイベントベースシステムへ—アレハンドロ・ブフマン60歳記念論文集、Lecture Notes in Computer Science, vol. 6462, Springer, 2010, pp. 159–174, http://dx.doi.org/10.1007/978-3-642-17226-7_10.

[43] R. Obermaisser, B. Leiner, リアルタイムLinuxに基づく時間トリガ型オペレーティングシステムの時間的・空間的分割, 2008年 第11回 IEEE 国際シンポジウム オブジェクトおよびコンポーネント指向リアルタイム分散コンピューティング (ISORC), 2008年, pp. 429–435, http://dx.doi.org/10.1109/ISORC.2008.10.

[44] D. Loche, A. Générès, M. Lauer, J.-C. Fabre, 混合重要度システムにおける時間的障害予防のための実行時監視と制御, 2021年 第17回欧州信頼性コンピューティング会議, EDCC, 2021年, pp. 53–60, http://dx.doi.org/10.1109/EDCC53658.2021.00015.

[45] R. Obermaisser, E. Henrich, K. Kim, H. Kopetz, M. Kim, 2つの補完的な時間トリガ技術TMOとTTPの統合, From Specif. Embed. Syst. Appl. 184 (2005) 211.

[46] R. Obermaisser、P. Peti、分散型組み込みリアルタイムシステムの迅速なアプリケーション開発のためのフレームワーク、IEEE Region 8 EUROCON 2003 収録。 Computer As a Tool., vol. 1, 2003, pp. 80–84 vol.1, http://dx.doi.org/10.1109/ EURCON.2003.1247983.

[47] J. Schneider、T. Nett、デュアルコアシステム上で Linux と AUTOSAR を並行して実行することによる IVI および ADAS 機能の統合に関する安全上の問題、掲載：Automotive - Safety & Security 2014, Gesellschaft für Informatik e.V., Bonn, 2015, pp. 55–68.

[48] C. Toner, H. Boukabache, G. Ducos, M. Pangallo, S. Danzeca, M. Widorski, S. Roesler, D. Perrin, CERNにおける放射線モニタリングシステム向け28nm ZYNQ SoCベースの耐障害性FPGA設計, Microelectron. Reliab. 100–101 (2019) 113492, http://dx.doi.org/10.1016/j.microrel.2019.113492, URL https://www.sciencedirect.com/science/article/pii/S0026271419304822. 第30回欧州電子デバイス信頼性・故障物理・解析シンポジウム.

[49] N. Mouzakitis, M. Paolino, M.D. Grammatikakis, D. Raho, 混合重要システム向けX86システム管理モード(SMM)評価, S. Saponara, A. De Gloria (編), 『産業・環境・社会に浸透するエレクトロニクス応用』, Springer International Publishing, Cham, 2021, pp. 164–170.

[50] L. クオモ、C. スコルディーノ、A. オッタヴィアーノ、N. ウィストフ、R. バラス、L. ペニーニ、E. グイディエリ、I.M. サヴィーノ, 次世代自動車用ECU向けRISC-Vオープンプラットフォームの構築, 2023年 第12回地中海組込みコンピューティング会議 (MECO), 2023年, pp. 1–8, http://dx.doi.org/10.1109/MECO58584.2023.10154913.

[51] S. Alonso, J. Lazaro, J. Jimenez, L. Muguira, U. Bidarte, リアルタイム組込みSoCプラットフォームにおけるOpenAMP フレームワークの評価, 2021年 第36回回路・集積システム設計会議 (DCIS), 2021年, pp. 1–6, http://dx.doi.org/10.1109/DCIS53048.2021.9666157.

[52] S. Karthik, K. Ramanan, N. Devshatwar, S. Paul, V. Mahaveer, S. Zhao, M. Vishwanathan, C. Matad, ハイパーバイザーベースの統合コックピットソリューションへのアプローチ, 収録: 2018 IEEE 第8回国際コンシューマーエレクトロニクス会議
- 2018.8576222.

[53] K.-B. Gemlau, N. Sperling, R. Ernst, 性能アーキテクチャにおける混合重要度通信スタックのための効率的なタイミング分離, 2022 IEEE 第27回新興技術と工場自動化国際会議, ETFA, 2022, pp. 1–8, http://dx.doi.org/10.1109/ETFA52439.2022.9921445.

[54] T. Van Eyck, H. Trimech, S. Michiels, D. Hughes, M. Salehi, H. Janjuaa, T.-L. Ta, 「Mr-TEE: 混合重要度コードの実用的な信頼実行」, 第24回国際ミドルウェア会議: 産業トラック論文集, Middleware '23, 計算機学会, ニューヨーク州ニューヨーク, 米国, 2023年, pp. 22–28, http://dx.doi.org/10.1145/3626562.3626831.

[55] D.B. de Oliveira, T. Cucinotta, R.S. de Oliveira, 「Linuxカーネルのための効率的な形式検証」, P.C. Ölveczky, G. Salaün (編), 『ソフトウェア工学と 形式手法』, Springer International Publishing, Cham, 2019年, pp. 315–332.

[56] A. Kornecki, J. Zalewski, リアルタイム安全重要システム向けソフトウェアの認証：現状, Innov. Syst. Softw. Eng. 5 (2) (2009) 149–161, http://dx.doi.org/10.1007/s11334-009-0088-1.

[57] A. Andryushin, V. Durnev, A. Chernyaev, 原子力発電所におけるオペレーティングシステム利用の問題点, Ann. Nucl. Energy 70 (2014) 87–89, http://dx.doi.org/ 10.1016/j.anucene.2014.03.009, URL https://www.sciencedirect.com/science/ article/pii/S0306454914001212.

[58] N. Mc Guire, 「IEC 61508 の文脈における安全重要システムのための Linux」, 『 』, 第 9 回リンツ・リアルタイム Linux ワークショップ論文集, vol. 5, 2007.

[59] N. マクガイア、I. アジェンデ、「複雑システムの認証へのアプローチ」、2020年 第50回 IEEE/IFIP 国際信頼性システム・ネットワーク会議 ワークショップ集 (DSN-W)、2020年、 pp. 70–71, http://dx.doi.org/10.1109/ DSN-W50199.2020.00022.

[60] N. Baek, OpenGL SC 2.0.1 標準仕様に準拠したグラフィックスデバイスドライバの実装, 応用工学技術国際ジャーナル (ロンドン) 5 (3) (2023) 143–149,
2.0.1標準仕様に準拠したグラフィックスデバイスドライバの実装、Int. J. Appl. Eng. Technology (London) 5 (3) (2023) 143–149, http://dx.doi.org/10.61485/IJAeT/v5-3-2023-18, 出版社著作権: © 2023, Roman Science Publications and Distributions. All rights reserved..

[61] S.H. VanderLeest, Avionics Linux, in: 2023 IEEE/AIAA 42nd Digital Avion-ics Systems Conference, DASC, 2023, pp. 1–6, http://dx.doi.org/10.1109/ DASC58513.2023.10311247.

[62] S.H. VanderLeest, K. Stewart, 航空宇宙アプリケーションにおけるLinuxの実現, 2023 IEEE/AIAA 第42回デジタル航空電子システム会議 (DASC), 2023年, pp. 1–6, http://dx.doi.org/10.1109/DASC58513.2023.10311338.

[63] I. アジェンデ、N.M. ギレ、J. ペレス、L.G. モンサルベ、J. フェルナンデス、R. オーバーマイサー、「安全ソフトウェアのテストカバレッジのためのLinuxカーネル実行経路の不確実性の推定」、2021年欧州設計・自動化・テスト会議・展示会（DATE）、2021年、 pp. 1446–1451, http://dx.doi.org/10.23919/DATE51398.2021.9473951.

[64] I. Allende, N. Mc Guire, J. Perez, L.G. Monsalve, R. Obermaisser, Linuxベースの安全システムに向けて—ソフトウェア実行パスカバレッジのための統計的アプローチ, J. Syst. Archit. 116 (C) (2021) http://dx.doi.org/10.1016/j.sysarc.2021.102047.

[65] Y. Chen, X. Tang, S. Xu, F. Zhu, Q. Zhou, T.-H. Weng, 異なるシナリオにおけるLinuxカーネルの実行経路非決定性の分析, Connect. Sci. 35 (1) (2023) 2192442, http://dx.doi.org/10.1080/09540091.2023.2192442.

[66] R. Shao, Y. Wu, L. Liu, Y. Chen, R. Zhou, Q. Zhou, ACRN上におけるLinuxカーネルの動的実行経路解析, 2024中国自動化会議(CAC), 2024, pp. 4433–4438, http://dx.doi.org/10.1109/CAC63892.2024.10865482.

[67] M. Galarraga, C.-A. Lefebvre, J. Perez-Cerrolaza, J.A. Pascual, Linuxベースの安全クリティカルシステムに向けて—Linux システムコールの実行時間変動性解析, J. Syst. アーキテクチャ 156 (2024) 103266, http://dx.doi.org/10.1016/ j.sysarc.2024.103266, URL https://www.sciencedirect.com/science/article/pii/S1383762124002030.

[68] A. Christmann、R. Hapka、R. Ernst、自律システムのためのタイミング多様性の形式解析、2023 Design, Automation & Test in Europe Conference & Exhibition 、 DATE 、 2023 、 pp. 1–6 、 http://dx.doi.org/10.23919/DATE56975。 2023.10137030。

[69] G.K. Adam, シングルボードコンピュータ上におけるLinuxカーネルのリアルタイム性能と応答遅延測定, Computers 10 (5) (2021) http://dx.doi.org/ 10.3390/computers10050064, URL https://www.mdpi.com/2073-431X/10/5/ 64.

[70] R. Hapka, A. Christmann, R. Ernst, 「タイミング多様性による高性能プラットフォームの不確実性の制御」, 2022 IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, 2022, pp. 212–219, http://dx.doi.org/10.1109/RTCSA55878.2022.00029.

[71] L. Thomeczek, A. Attenberger, J. Kolb, V. Matousek, J. Mottok, LinuxカーネルeBPFトレースを用いた安全上重要なレイテンシ要因の測定, ARCS Workshop 2019; 第32回国際コンピューティングシステムアーキテクチャ会議, 2019, pp. 1–8.

[72] P. Houdek, M. Sojka, Z. Hanzálek, 「ARMベースの異種プラットフォームにおける予測可能な実行モデルに向けて」, 2017 IEEE 26th International Symposium on Industrial Electronics, ISIE, 2017, pp. 1297–1302, http://dx.doi.org/10.1109/ISIE.2017.8001432.

[73] A. Zuepke, R. Kaiser, 決定論的フューテックス：WCETと境界付き干渉問題への対応, 2019 IEEE リアルタイム・組込み技術応用シンポジウム, RTAS, 2019, pp. 65–76, http://dx.doi.org/10.1109/ RTAS.2019.00014.

[74] A. Alonso, E. Salazar, J. López, 処理能力が限られたシステムにおける予測可能性向上のための資源管理, 2010 IEEE 第15回新興技術・工場自動化会議, ETFA 2010, 2010年,
pp. 1–7, http://dx.doi.org/10.1109/ETFA.2010.5641339.

[75] J. Kim, P. Shin, S. Noh, D. Ham, S. Hong, メモリ要求スロットリングとLinux Cgroupによる安全重要アプリケーションのメモリ干渉遅延低減, 2018年 第31回IEEE国際システムオンチップ会議, SOCC, 2018年, pp. 215–220, http://dx.doi.org/10.1109/SOCC.2018.8618555.

[76] V. Struhár, S.S. Craciunas, M. Ashjaei, M. Behnam, A.V. Papadopoulos, REACT: リアルタイムコンテナオーケストレーションの実現, 2021年 第26回 IEEE 新興技術と工場自動化国際会議 (ETFA), 2021年,
pp. 1–8, http://dx.doi.org/10.1109/ETFA45728.2021.9613685.

[77] W. Dong、C. Zhao、S. Shu、M. Leucker、安全性とセキュリティが重要なソフトウェアのための予測的アクティブモニタリング、Sci. 中国情報科学 55 (12) (2012) 2723–2737, http://dx.doi.org/10.1007/s11432-012-4739-8.

[78] S. Barut, M. Boneberger, P. Mohammadi, J.J. Steil, ロボティクス応用におけるROS 2とOROCOSのリアルタイム性能比較評価, 2021 IEEE国際ロボティクス・オートメーション会議 (ICRA), 2021, pp. 708–714, http://dx.doi.org/10.1109/ICRA48506.2021.9561026.

[79] P. Masek, M. Thulin, H. Andrade, C. Berger, O. Benderius, 自動運転大型車両を例としたリアルタイムソフトウェア向けサンドボックス化ソフトウェア展開の体系的評価, 2016 IEEE 第19回 インテリジェント交通システム国際会議, ITSC, 2016, pp. 2398–2403, http://dx.doi. org/10.1109/ITSC.2016.7795942.

[80] G. Albanese, R. Birke, G. Giannopoulou, S. Schönborn, T. Sivanthi, リアルタイムアプリケーションのコンテナ化展開におけるネットワークオプションの評価, 2021年 第26回 IEEE 新興技術と工場自動化国際会議 (ETFA), 2021年, pp. 1–8, http://dx.doi.org/10.1109/ETFA45728.2021.9613320.

[81] T.F. De Barrena, A. Garcia, J. Franco, J.L. Ferrando, 産業用エッジAIにおけるリアルタイムLinuxの影響, M. Dassisti, K. Madani, H. Panetto (編), 『革新的インテリジェント産業生産と物流』, Springer Nature Switzerland, Cham, 2025, pp. 486–504.

[82] S. Liu、R. Wagle、J.H. アンダーソン、M. Yang、C. Zhang、Y. Li、「今日の自律性：遅延が発生しやすいブラックボックスの多さ」、R. Pellizzoni (編)、 第 36 回 Euromicro リアルタイムシステム会議 (ECRTS 2024)、Leibniz International Proceedings in Informatics (LIPIcs)、 vol. 298 、 Schloss Dagstuhl – Leibniz-Zentrum für Informatik 、 Dagstuhl 、 ドイツ 、 2024 年 、 pp. 12:1–12:27 、 http://dx.doi.org/10. 4230/LIPIcs.ECRTS.2024.12 、 URL https://drops.dagstuhl.de/entities/document/ 10.4230/LIPIcs.ECRTS.2024.12。

[83] R. Wagle、Z. Tong、R.L. Sites、J.H. アンダーソン、予測可能な GPU 実行をお求めですか？ Beware SMIs!、2023 IEEE 29th International Conference on Parallel and Distributed Systems、 ICPADS 、 2023 、 pp. 2100–2109 、 http://dx.doi.org/10.1109/ ICPADS60453.2023.00285。

[84] E. Seals, M. Bechtel, H. Yun, BandWatch: ヘテロジニアスマルチコア向けシステム全体のメモリ帯域幅制御システム, 2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, 2023, pp. 38–46, http://dx.doi.org/10.1109/RTCSA58653.2023.00014.

[85] O. Benedikt, M. Sojka, P. Zaykov, D. Hornof, M. Kafka, P. Š˙ ucha, Z. Hanzálek, 航空電子分野におけるMPSoC向け熱感知スケジューリング：ツールと初期結果, 2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, 2021, pp. 159–168, http://dx.doi.org/10.1109/RTCSA52859.2021.00026.

[86] D. de Niz, L.T. Phan, Partitioned scheduling of multi-modal mixed-criticality real-time systems on multiprocessor platforms, in: 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium, RTAS, 2014, pp. 111–122, http://dx.doi.org/10.1109/RTAS.2014.6925995.

[87] G. Gala, I. Kadusale, G. Fohler, Linuxカーネルにおける時間トリガーとイベントトリガーの統合スケジューリング, 第17回組み込みリアルタイムアプリケーション向けOSプラットフォーム年次ワークショップ, OSPERT 2023, 2023年, http://dx.doi.org/10.48550/arXiv.2306.16271.

[88] J.-J. Han, Z. Wang, S. Gong, T. Miao, L.T. Yang, リソース認識型スケジューリングによる信頼性のあるマルチコアリアルタイムシステム：利用率上限と分割アルゴリズム, IEEE Trans. Parallel Distrib. Syst. 30 (12) (2019) 2806–2819, http://dx.doi.org/10.1109/TPDS.2019.2926455.

[89] J. Li, D. Ferry, S. Ahuja, K. Agrawal, C. Gill, C. Lu, 並列リアルタイムタスクのための混合重要度フェデレーテッドスケジューリング, 2016 IEEE リアルタイム・組込み技術応用シンポジウム, RTAS, 2016, pp. 1–12, http://dx.doi.org/10.1109/RTAS.2016.7461340.

[90] K. Lakshmanan, D. De Niz, R.R. Rajkumar, G. Moreno, 混合重要度サイバーフィジカルシステムにおける過負荷プロビジョニング, ACM Trans. Embed. Comput. Syst. 11 (4) (2013) http://dx.doi.org/10.1145/2362336.2362350.

[91] W. Ali, H. Yun, 「RT-Gang: 安全重要システム向けリアルタイムギャングスケジューリングフレームワーク」, 2019 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 143–155, http://dx.doi.org/10.1109/RTAS.2019.00020, URL https://doi.ieeecomputersociety.org/10.1109/RTAS.2019.00020.

[92] A.H. Sifat, X. Deng, B. Bharmal, S. Wang, S. Huang, J. Huang, C. Jung, H. Zeng, R. Williams, 自動ロボットにおける計算意識を可能にする安全性能指標, IEEE Robot. Autom. Lett. 8 (9) (2023) 5727–5734, http://dx.doi.org/10.1109/LRA.2023.3300251.

[93] R. Amarnath, S.N. Bhat, P. Munk, E. Thaden, システムコールのソフトエラー信頼性を評価するフォルト注入アプローチ, 2018 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW, 2018, pp. 71–76, http://dx.doi.org/10.1109/ISSREW.2018.00-28.

[94] G. Cabodi, M. Murciano, M. Violante, リアルタイム組込みアプリケーションの信頼性解析のためのソフトウェア障害注入の強化, ACM Trans. Embed. Comput. Syst. 10 (2) (2011) http://dx.doi.org/10.1145/1880050.1880060.

[95] M. Cinque、A. Pecchia、仮想化マルチコアシステムにおけるハードウェア障害の注入について、J. Parallel Distrib. Comput. 106 (2017) 50–61, http://dx.doi.org/10.1016/j.jpdc.2017.03.004, URL https://www.sciencedirect.com/science/ article/pii/S0743731517300849.

[96] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson, R. Iyer, ロボット遠隔手術システムのシステム理論的安全評価, in: F. Koornneef, C. van Gulijk (編), 『コンピュータの安全性、信頼性、セキュリティ』, Springer International Publishing, Cham, 2015, pp. 213–227.

[97] L.G. カサグランデ、F.L. カステンズミット、オペレーティングシステム有無で開発された組込みソフトウェアにおけるソフトエラー解析、2016年 第17回ラテンアメリカテストシンポジウム（LATS）、2016年、pp. 147–152、http://dx.doi.org/10.1109/LATW.2016。7483355.

[98] A. Lanzaro, A. Pecchia, M. Cinque, D. Cotroneo, R. Barbosa, N. Silva, マルチコアシステム評価のための予備的障害注入フレームワーク, F. Ortmeier, P. Daniel (編), コンピュータの安全性、信頼性、およびセキュリティ, Springer ベルリン・ハイデルベルク, ベルリン, ハイデルベルク, 2012年, pp. 106–116.

[99] A.D. Velasco、B. Montrucchio、M. Rebaudengo、「KITO ツール：Linux カーネルデータ構造における障害注入環境」、Microelectron. Reliab. 60 (2016) 153–162, http://dx.doi.org/10.1016/j.microrel.2016.02.011, URL https://www.sciencedirect.com/science/article/pii/S0026271416300300.

[100] A. Bovenzi, S. Russo, F. Brancati, A. Bondavalli, アプリケーションソフトウェア障害検出のためのOSレベル異常の特定に向けて, 収録: 2011 IEEE International Workshop on Measurements and Networking Proceedings (M&N), 2011, pp. 71–76, http://dx.doi.org/10.1109/IWMN.2011.6088494.

[101] A.D. Velasco, B. Montrucchio, M. Rebaudengo, スケジューラのカーネルデータ構造に対する強化アプローチ, ARCS 2017: 第30回国際会議「: コンピューティングシステムアーキテクチャ」, 2017年, pp. 1–4.

[102] A.D. Velasco, B. Montrucchio, M. Rebaudengo, 互排ロックカーネルデータ構造のためのTMR技術, 2017年 第18回IEEEラテンアメリカテストシンポジウム, LATS, 2017, pp. 1–6, http://dx.doi.org/10.1109/LATW.2017.7906745.

[103] M.Y. Jung, P. Kazanzides, 構成要素ベースのロボットシステム向け故障検出と診断, 2012 IEEE International Conference on Technologies for Practical Robot Applications, TePRA, 2012, pp. 1–6, http://dx.doi.org/10.1109/TePRA.2012.6269387.

[104] D. コトロネオ、D. ディ・レオ、R. ナテッラ、R. ピエトランツォーノ、航空電子分野向けオペレーティングシステムの状態ベース堅牢性テストに関する事例研究、収録：F. Flammini, S. Bologna, V. Vittorini (編), 『コンピュータ安全・信頼性・ セキュリティ』, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 213–227.

[105] R. Shahpasand, Y. Sedaghat, S. Paydar, 組込みリアルタイムオペレーティングシステムのステートフル堅牢性テストの改善, 2016 6th International Conference on Computer and Knowledge Engineering, ICCKE, 2016, pp. 159–164, http://dx.doi.org/10.1109/ICCKE.2016.7802133.

[106] G.S. Rodrigues, F. Rosa, Á.B. de Oliveira, F.L. Kastensmidt, L. Ost, R. Reis, 「Linuxおよび並列化APIを実行するソフトエラー環境下におけるARMプロセッサの耐障害性手法の影響分析」, IEEE Trans. Nucl. Sci. 64 (8) (2017) 2196–2203, http://dx.doi.org/10.1109/TNS.2017.2706519.

[107] J. Gava, R. Reis, L. Ost, RAT: 軽量でアーキテクチャ非依存のシステムレベルソフトエラー軽減技術, A. Calimera, P.-E. Gaillardon, K. Korgaonkar, S. Kvatinsky, R. Reis (編), VLSI-SoC: Design Trends, Springer International Publishing, Cham, 2021, pp. 235–253.

[108] J . Gava, R. Reis, L. Ost, RAT: 軽量システムレベルソフトエラー軽減技術, 2020 IFIP/IEEE 第28回超大規模集積回路国際会議, VLSI-SOC, 2020, pp. 165–170, http://dx.doi.org/10.1109/ VLSI-SOC46417.2020.9344080.

[109] J. Gava、V. Bandeira、F. Rosa、R. Garibotti、R. Reis、L. Ost、SOFIA: ソフトエラーの早期評価、識別、および軽減のための自動化フレームワーク、J. Syst. Archit. 131 (2022) 102710, http://dx.doi.org/10.1016/j.sysarc.2022.102710, URL https://www.sciencedirect.com/science/article/pii/S1383762122002028.

[110] G.S. Rodrigues, F.L. Kastensmidt, R. Reis, F. Rosa, L. Ost, 「ARM Cortex-A9デュアルコアにおけるフォルト注入下でのpthreadsとopenmpの使用影響分析」, 2016年 第16回欧州放射線及びその部品・システムへの影響に関する会議 (RADECS), 2016年, pp. 1–6, http://dx.doi.org/10.1109/ RADECS.2016.8093180.

[111] F. Mazzocchetti, S. Alcaide, F. Bas, P. Benedicte, G. Cabo, F. Chang, F. Fuentes, J. Abella, SafeSoftDR: 安全上重要なタスクのためのソフトウェアベースの多様な冗長性を可能にするライブラリ, FORECAST Workshop (with HiPEAC Conference), 2022.

[112] H. Joe, H. Jeong, Y. Yoon, H. Kim, S. Han, H.-W. Jin, 宇宙機フライトコンピュータ向け完全仮想化マイクロハイパーバイザー, 2012 IEEE/AIAA 第31回デジタル航空電子システム会議 (DASC), 2012年, pp. 6C5–1–6C5–9, http://dx.doi.org/ 10.1109/DASC.2012.6382393.

[113] A. Prakash, L. Krawczyk, C. Wolff, APP4MC RaceCar: タイミング動作の評価と検証のための実用的な ADAS デモ機, in: Proceedings of the 2nd Eclipse Research International Conference on Security, Artificial Intelligence, Architecture and Modelling for Next Generation Mobility, 2021.

[114] R. Messnarz, G. Macher, J. Stolfa, S. Stolfa, 高度に自律的な車両（システム）設計パターン – 故障時動作性と高水準の安全・セキュリティの実現, in: A. Walker, R.V. O'Connor, R. Messnarz (編), Systems, Software and Services Process Improvement, Springer International Publishing, Cham, 2019, pp. 465–477.

[115] S. Mouelhi, D. Cancila, A. Ramdane-Cherif, 「接続車両隊 列向け自律制御システムの分散オブジェクト指向設計」, 2017年 第22回複雑コンピュータシステム工学国際会議, ICECCS, 2017年, pp. 40–49, http://dx.doi.org/10.1109/ICECCS.2017.32.

[116] Z. Gu, Z. Wang, S. Li, H. Cai, 仮想化技術に基づく自動車テレマティクスゲートウェイの設計と実装, 2012 IEEE 第15回オブジェクト/コンポーネント/サービス指向リアルタイム分散コンピューティング国際シンポジウムワークショップ, 2012, pp. 53–58, http://dx.doi.org/10.1109/ISORCW.2012.20.

[117] I. Allende, N. Mc Guire, J. Perez, L.G. Monsalve, N. Uriarte, R. Obermaisser, マルチコアデバイスに基づく混合重要度組込みシステム開発のためのLinuxへの取り組み, 2019年 第15回欧州信頼性コンピューティング会議, EDCC, 2019年, pp. 47–54, http://dx.doi.org/10.1109/EDCC.2019.00020.

[118] M. Kotur、M. Dragojević、G. Velikić、I. Bašičević、「AUTOSAR適応コンテキストにおけるデジタルコックピット」、2018 IEEE 8th International Conference on Consumer Electronics - Berlin, ICCE-Berlin, 2018, pp. 1–4, http://dx.doi.org/10.1109/ ICCE-Berlin.2018.8576209.

[119] C. エルナンデス、J. フリー、R. パレデス、C.-A. ルフェーブル、I. アジェンデ、J. アベラ、D. トリラ、M. マッチニグ、B. フィッシャー、K. シュヴァルツ、J. キシュカ、M. ロンベック、J. クロッカーズ、N. マクガイア、F. ラムマーストファー、C. シュヴァルツル、F. ヴァルテット、D. リューデマン、M. ラバエン、SELENE: 高性能安全重要システムのための自己監視型信頼性プラットフォーム、2020年 第23回 ユーロマイクロ デジタルシステム設計会議 (DSD)、2020年、pp. 370–377、http://dx.doi.org/10.1109/DSD51259.2020。00066.

[120] M . Wahler, S. Richter, S. Kumar, M. Oriol, リアルタイム制御器のための非破壊的大規模コンポーネント更新, 2011 IEEE 第27回データ工学国際会議ワークショップ, 2011年, pp. 174–178, http://dx.doi.org/10.1109/ICDEW.2011.5767631.

[121] A. Armoush, D. Franke, I. Kalkov, S. Kowalewski, An approach for using mobile devices in industrial safety-critical embedded systems, in: G. Memmi, U. Blanke (Eds.), Mobile Computing, Applications, and Services, Springer International Publishing, Cham, 2014, pp. 294–297.

[122] T. seL4 Foundation, seL4 Proofs & Certification, 2025, https://sel4.systems/ Verification/certification.html. (最終アクセス日: 2025年9月15日).

[123] P. Lucas, K. Chappuis, M. Paolino, N. Dagieu, D. Raho, VOSYSmonitor, ARMv8-A上における混合重要度システムのための低遅延監視層, M. Bertogna (編), 第29回ユーロマイクロリアルタイムシステム会議（ECRTS 2017）、ライプニッツ国際情報学論文集（LIPIcs）第76巻、シュロス・ダグシュトゥール – ライプニッツ情報学センター、ドイツ・ダグシュトゥール、2017年、 pp. 6:1–6:18, http://dx.doi.org/10.4230/LIPIcs.ECRTS.2017.6, URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECRTS.2017.6.

[124] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, R. Kegley, COTSベース組込みシステムのための予測可能な実行モデル, 2011年 第17回 IEEE リアルタイム・組込み技術・応用シンポジウム, 2011年, pp. 269–279, http://dx.doi.org/10.1109/RTAS.2011.33.

[125] OSADL, SIL2LinuxMP: OSADL - Open Source Automation Development Lab eG, 2014, https://www.osadl.org/SIL2LinuxMP.sil2-linux-project.0.html. (最終アクセス日: 2023年11月30日).

[126] ELISA, ELISA - Advancing Open Source Safety-Critical Systems, 2019, https://elisa.tech/. (最終アクセス日: 2025年3月27日).

[127] Automotive Grade Linux, AGL - Automotive Grade Linux, 2012, https://www.automotivelinux.org/. (最終アクセス日: 2025年3月27日).

[128] S. McElligott、What OS Does Tesla Use?、2023、https://cars.usnews.com/cars-trucks/features/what-os-does-tesla-use#:~:text=The%20Tesla%2Dspecific%20operating%20systems,based%20projects%20such%20as%20Ubuntu. （最終2025年3月27日アクセス）。

[129] SAE International, 運転に関連する用語の分類と定義道路走行自動車向け自動化システム、技術報告書 J3016_202104, SAE International, 2021.

[130] Elektrobit, EB corbos Linux for Safety Applications, 2024, https://www.elektrobit.com/products/ecu/eb-corbos/linux-for-safety-applications/。（最終アクセス日（2025年3月27日に終了）。

[131] Codethink, Codethink Trustable Reproducible Linux (CTRL OS), 2025, https://www.codethink.co.uk/ctrl-os.html. (最終アクセス日: 2025年5月21日).

[132] ELISA, Space Grade Linux SIG. Building a space ready Linux distribution., 2019, https://elisa.tech/space-grade-linux-sig/. (最終アクセス日: 2025年6月17日).

[133] G. Emad, How Embedded Linux is used in Spacecrafts !, 2024, https://www.embeddedrelated.com/showarticle/1614.php. (最終アクセス日: 2025年3月27日) 2025年3月27日最終アクセス).

[134] H. アヴァード・F・グリップ、J. ラム、D.S. ベイヤード、D.T. コンウェイ、G. シン、R. ブロッカーズ、J.H. Delaune、L.H. Matthies、C. Malpica、T.L. Brown、A. Jain、A.M.S. マーティン、G .B. Merewether、NASA の火星ヘリコプター用飛行制御システム、AIAA Scitech 2019 Forum、2019年、p. 1289、http://dx.doi.org/10.2514/6.2019-1289、arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2019-1289. URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-1289.

[135] C. Kown, D. Kim, H. Joe, H. Kim, Linuxベースのメモリ効率型ARINC 653パーティションスケジューラ、in: Proceedings of the 2014 IEEE Emerging Technology and ファクトリーオートメーション、ETFA、2014年、pp. 1–5、http://dx.doi.org/10.1109/ETFA。2014.7005306.

[136] S. Han, H.-W. Jin, 完全仮想化に基づくARINC 653パーティショニング, 2011年 IEEE/AIAA 第30回デジタル航空電子システム会議, 2011, pp. 7E1–1–7E1–11, http://dx.doi.org/10.1109/DASC.2011.6096132.

[137] S.H. ヴァンダーリスト, ARINC 653 ハイパーバイザー, 収録: 第29回デジタル航空電子システム会議 ference, 2010, pp. 5.E.2–1–5.E.2–20, http://dx.doi.org/10.1109/DASC.2010.5655298.

[138] Concurrent Real-Time, Redhawk Linux RTOS, 2022, https://concurrent-rt.com/products/software/redhawk-linux/。 （最終アクセス日：2025年3月27日）。