Xinu メッシュを AIPL 駆動リアルタイム OS 化する計画と課題

フィジカル AI（アーム型ロボット）向け分散 RTOS への道筋

—— 3 世代 Raspberry Pi クラスタの実測を踏まえた検討——

児玉靖司
法政大学経営学部

2026 年 7 月 20 日

Abstract

ベアメタル Embedded Xinu を、AIPL（自作アクター言語）で記述する**フィジカル AI 向けリアルタイム OS（RTOS）**へ発展させ、最終的にアーム型ロボットを **12 台の Xinu ノード**で制御する構想がある。本書は、その前段として**現状の Xinu を RTOS 化するうえでの改良余地**を、本セッションで得た 3 世代 Raspberry Pi（A53/A72/A76）クラスタの実測に基づいて洗い出し、問題点・原因・対策・優先順位を整理する。結論として、現行ポートは**協調型スケジューラ・不安定なタイマ割り込み・GC やデマンドページングによる非有界遅延**という、RTOS の前提を欠く箇所を持つ。一方 Xinu の**小ささ**は WCET 解析・安全性の見通しにおいて強みであり、「巨大 RTOS を削る」より「小さい Xinu に実時間保証と AIPL 型を足す」方が、AIPL 一貫路線に整合し、差別化になる。本書は 4 段階のロードマップと、各段階の受入基準（実測ジッタ等）を示す。

Contents

1　背景と目標

1.1　目標像

- **言語**：制御・知覚・計画を原則 AIPL（型推論・効果推論を持つアクター言語）で記述する。
- **OS**：Xinu を土台に、**ハード実時間**（周期制御ループ、境界化された割り込み遅延）を提供する RTOS 化する。
- **対象**：アーム型ロボット等のフィジカル AI。最終形は **12 ノードの Xinu** を接続し、関節・センサ・計画をノードに割り当てる分散制御。

1.2　本書の立場

「12 台接続」に進む前に、**1 ノードの Xinu が RTOS 要件を満たすか**を先に固める。分散化はその上に載る。以下では単一ノードの RTOS 課題（第 3 節）と、分散・並列計算機としての課題（第 4 節）を分けて述べる。

2　現状（本セッションの実測から）

3 世代 Pi（A53=Pi3 / A72=Pi4 / A76=Pi5）で、ワーカープール型 SMP・キャッシュ実験・WiFi メッシュを実機検証した。RTOS 化の観点で関係する事実を挙げる。

Table 1: RTOS 化に関係する現状の実測事実

観測	内容と含意
スケジューラ	起動ログ `sched: cooperative ctxsw`。**協調型（非プリエンプティブ）**。
タイマ割り込み	`clktime` が常に 0 を返す＝**周期割り込みが安定発火せず**。フリーランカウンタ直読で回避したが、RTOS には割り込み遅延・ジッタの保証が要る。
マルチコア	ワーカープール型：core0 が全 OS ＋制御、core1–3 は IRQ マスクの計算専用。runtime は**非リエントラント**＝対称 SMP 不可。計算律速は ×3.99 スケール。
D-cache	既定 OFF（コヒーレンシのため）。本セッションで**明示キャッシュ管理**により多コア D-cache を 3 世代で成立させ、単一コア再帰処理が ×14–×16 高速化。
メモリ	AIPL に**約 8 秒周期のアクタ GC**、Pi5 に**デマンドページング**——いずれもハード実時間の非有界遅延源。
待ち機構	ワーカープールの `SMP_WAIT_LIMIT`=2×10^9 スピン等、**非有界の待ち**が散在。
ネットワーク	WiFi IBSS メッシュ（`MANET` 10.0.0.n）＋ MANET/UDP。本セッションでは各台参加後も **近隣数 0（セル未収束）** ＝ WiFi メッシュは**非決定的**で制御平面に不適。

3　RTOS 化の課題（単一ノード）

3.1　致命的ギャップ：スケジューリングとタイミング

① **協調型スケジューラ**。ロボット制御（例：1 kHz サーボループ）は **固定優先度プリエンプティブ**（Rate Monotonic、必要なら EDF）が必須。共有資源（ロックフリー・メールボックス等）で**優先度逆転**が起こるため**優先度継承**を要する。

② **タイマ割り込みの不安定**。`clktime=0` は周期割り込みが安定発火していなかった証拠。RTOS では周期 tick と**割り込み遅延（latency ＋ jitter）を実測・上限保証**しなければ成立しない。精密デッドラインには tickless ＋ one-shot タイマ。

3.2　決定性を壊す要素の除去

③ **GC の停止**。AIPL の周期 GC は制御ループを殺す。RT パスは**無割当**（アクタ・メッセージを事前確保）、または**インクリメンタル／リージョン GC**。GC する best-effort 領域と GC しない RT 領域を分離する。

④ **デマンドページング**。ページフォルト＝非有界遅延。RT タスクのメモリは **ピン留め（フォルト禁止）**にする。

⑤ **非有界のスピン／タイムアウト**。`SMP_WAIT_LIMIT` 等の巨大ループはジッタ源。すべての待ちを**有界・イベント駆動**へ。

3.3 実時間 I/O とマルチコアの使い方

⑥ **RT ドライバ級の新設**。現行 USB/net は best-effort。ロボット用に PWM ／ステップ生成（ハードタイマ）・エンコーダ捕捉・I2C/SPI/CAN の**有界トランザクション**を「WCET 保証つき RT ドライバ」として分離する。

⑦ **AMP（非対称）パーティショニング**。非リエントラント runtime ゆえ対称 SMP は不可。コア／ボードに**役割固定（制御コア／センサコア／計画コア）**する AMP が適合し、「12 Xinu」構想（関節・サブシステムごと独立 RT パーティション）と一致する。制御コアは裸で隔離。

3.4 安全（フィジカル AI ＝人・機械を傷つけ得る）

⑧ **ウォッチドッグ・デッドラインミス → セーフ状態**。締切超過を検出したらモータを安全トルク／ブレーキへ。1 ノード障害を波及させない**フォールト封じ込め**。現在未整備。

⑨ **Capability ＋ MMU 隔離**。既存構想の Capability を拡張し、**MMU 領域でアクタを隔離**（MMU は既に稼働）。バグった AIPL アクタが制御ループを壊せないようにする（「そのアクタは自関節しか触れない」）。

3.5 AIPL を「実時間言語」にする（差別化点）

⑩ **タイミングを型／効果で表現し、スケジューラビリティを静的検証**。アクタに周期・デッドライン・WCET を宣言 → コンパイラが**優先度付き RT タスクへ写像**。**効果型で「この RT アクタは割当／ブロック／ GC ／ send をしない」を証明**（③④⑤を言語で担保）。Rate Monotonic 上限で**スケジュール可能性を静的判定**する。これは「AIPL で書く RTOS」を他にない形で強くする。

4 分散・並列計算機としての課題

12 ノードを「全体として 1 台の並列計算機／ロボット」として使うための課題。

⑪ **決定的インターコネクト＋時刻同期**。本セッションで WiFi IBSS メッシュは参加後も**近隣数 0（セル未収束）**で、**非決定的**。制御平面は **CAN ／ EtherCAT ／ TSN Ethernet**、WiFi は非 RT の副系に。多関節協調には **PTP（IEEE 1588）級の時刻同期**が要る（現行の「ACK 方式で時刻同期不要」は throughput 向け）。

⑫ **異種対応の分散スケジューラ（動的・2 階層）**。A76>A72>A53 の実測速度で取り分を按分し、**クロスボード work-stealing**（空いた板が忙しい板から盗む）。オンボードのワーカープール ＋ Pi3 の least-loaded 分配を一般化。**Mac 中心の星型（SPOF）を P2P 化**。

⑬ **集団通信（MPI 相当）**。`broadcast/scatter/gather/reduce/barrier` を実装。木構造の scatter/reduce により台数増加でも効く。持続接続バイナリ RPC で HTTP/1.0 の毎回接続オーバヘッドを排す。

⑭ **タスク粒度**。オンボード損益分岐は 5–11 µs だが、**ネット越えは ms オーダ**。クロスボードで配るタスクは $\geq$ **約 100 ms の純計算**でないと損。D-cache 本番化（残課題は USB DMA バウンス）で計算/通信比が上がり、より小さいタスクでも分散が黒字化する。

Table 2: 段階的計画と受入基準

段階	内容	受入基準（測って合否）
P1	**プリエンプティブ固定優先度スケジューラ＋タイマ IRQ 堅牢化**（①②）。固定周期タスクの release jitter を µs で計測するハーネスを併設。	1 kHz 周期タスクの release jitter が上限内（例 $< 100\,\mu s$）、割り込み遅延を実測・提示。
P2	**決定性の除去要素を潰す**：RT パス無割当＋ GC 分離、ページピン、非有界待ちの撲滅（③④⑤）。	RT パスで GC/フォルト/非有界待ちが 0（効果型または計測で確認）。
P3	**AMP パーティショニング**で制御コアを隔離（⑦）＋**安全**（WD ／セーフ状態、Capability ＋ MMU 隔離）（⑧⑨）。	制御コアが best-effort 負荷から隔離、故意フォールトでセーフ状態へ遷移。
P4	**決定的ノード間通信＋時刻同期**（⑪）、**分散スケジューラ／集団通信**（⑫⑬⑭）、**AIPL の timing/effect 型**（⑩）を並行整備し、12 ノードへ拡張。	2 ノード協調運動のジッタ上限、reduce の台数スケール、AIPL でのスケジューラビリティ静的判定。

5 ロードマップ（優先順）

投資対効果が最も高いのは P1 である。「プリエンプティブ化＋タイマ／割り込み遅延の実測」で**制御ループのジッタを数値で押さえる**ことが、RTOS を名乗るための最初の一歩であり、以降のすべての基礎になる。

6 リスクと未解決問題

- **WiFi メッシュの非決定性**：本セッションで実証（近隣 0・未収束）。制御に使うなら有線 RT フィールドバスへ移す判断が要る。WiFi は監視・非 RT データに限定。
- **D-cache 本番化の壁**：多コア D-cache は成立させたが、本番投入には USB DMA バッファの出自（ヒープ／スタック由来）を 64 B 境界化・バウンスする改修が残る。
- **AIPL runtime の RT 化**：GC 分離と効果型による無割当保証は言語・処理系両面の設計課題。
- **認証・保証**：フィジカル AI では WCET と安全の**論証**が要る。Xinu の小ささは有利だが、AIPL 生成コードまで含めた WCET 解析の方法論が要る。

7 結論

現行の Xinu ポートは、**協調型スケジューラ・不安定タイマ・GC ／ページングによる非有界遅延**という点で、まだ RTOS 要件を満たさない。だが Xinu の**小ささ**は WCET 解析・安全論証に有利で、「**小さい Xinu に実時間保証と AIPL 型を足す**」方針は、御社の AIPL 一貫路線に整合し差別化になる。まず **P1（プリエンプティブ化＋割り込み遅延の実測）**に着手し、制御ループのジッタを数値で押さえることを推奨する。分散（12 ノード）化は、単一ノードの実時間保証を固めた上で、決定的インターコネクトと動的分散スケジューラの上に載せるのが順序として正しい。