Xinu メッシュ上のハードウェア・イン・ザ・ループ進化計算

設計を進化させ、実機の実測で適応度を評価する

—— 3 世代 Raspberry Pi クラスタでの負荷分散設計探索と sim-to-real gap ——

児玉靖司
法政大学経営学部

2026 年 7 月 20 日

Abstract

「アプリ/ソフトウェアの**設計**を進化計算で探索し、各候補を**実機の Xinu メッシュで実際に走らせて**適応度を測る」枠組みを構築した。対象は、N-Queens(N=14) のワークロードを A53/A72/A76 の異種 3 ボード (計 12 コア) へどう配分すれば **makespan(最遅ノードの完了時間)** が最小になるか、という実行設計問題である。実機の列別コストから作った**コストモデル**を適応度に GA を回すと、モデル上は最速の設計が**実機では 44% も遅い**—— GA がモデルの穴を突く **sim-to-real gap** が生じた。較正しても純モデル GA はオーバーフィットし、**実機測定だけが真の順位**を与える。そこで**ハードウェア・イン・ザ・ループ** (モデルが候補を提案し、実機測定が勝者を決める) を採り、真のチャンピオン (均衡 3 分割、実機 1 427 ms) を同定した。本手法は、設計空間 GA の予測を実機実測で検証・確定する `aice-pi-evolution` 系譜の中核ループを、今回構築した 12 コア実機メッシュ上で閉じる。

Contents

1 背景と目的

`aice-pi-evolution` は、進化計算 (GA/MAP-Elites) で**言語・カーネル・アプリの設計**を探索する枠組みである。従来その適応度は LLM レビュアや静的モデルに依存していたが、それらは不完全で、GA は容易にその穴を突く。本プロジェクトは適応度を**実機の Xinu メッシュでの実測**に置き、「設計を進化 → 実機で評価」のループを閉じることを目的とする。土台は別レポート (*mesh_report*) で構築した、3 ボード 12 コアの自己形成メッシュである。

2 問題設定と手法

2.1 設計ゲノム

N=14 の N-Queens は先頭クイーンの列 0..13 ごとに独立な部分木を持つ。ゲノムは **各列をどのボードで評価するか**を表す長さ 14 の割当て $\mathrm{assign}[c] \in \{\mathrm{rpi3}, \mathrm{rpi4}, \mathrm{rpi5}\}$ とする。各ボードは割当てられた列を **4 コア・ワーカプール** (`/nqpart`) で処理し、非連続な列は複数回の連続ラン (逐次) になる。

2.2 適応度：実機コストモデルと実測

実機コスト行列 (表1) を `/nqpart` で採取した。列 c を 1 コアで解く時間 $\mathrm{cost}[b][c]$ である。ボードの処理時間はモデルでは

$$\mathrm{board\_time}(b) = \mathrm{SCALE} \cdot \sum_{\text{連続ラン}} (\text{4 コア block 分割の最重チャンク}) \; + \; \mathrm{RUN\_OVH} \cdot (\text{ラン数}),$$

$\mathrm{makespan} = \max_b \mathrm{board\_time}(b)$。SCALE=1.36(D キャッシュ off の 4 コア同時実行によるバス競合) と RUN_OVH=320 ms(1 ラン当たりの HTTP/ディスパッチ) は**実測 2 点から較正**した。実測適応度は、設計を実機に配って makespan を計測し、解の総和が真値 (365 596) と一致するかを検証する。

Table 1: 実機コスト行列 (単一コア、N=14、ms)。列 8–13 は対称で 5–0 に一致

列	0	1	2	3	4	5	6/7
rpi3 (A53)	565	636	688	703	729	739	752
rpi4 (A72)	638	718	777	792	820	832	847
rpi5 (A76)	273	307	335	341	349	354	367

注目点。branchy な backtracking では **rpi3(A53) が rpi4(A72) より速く**、rpi5(A76) は両者の約 2.3 倍。異種混在ゆえ「均等分割」は最適でない。

3 結果

3.1 モデル GA のオーバーフィットと sim-to-real gap

コストモデルを適応度に GA を回すと、モデル上のチャンピオンは**最速ボード rpi5 に列を集中**させる設計に収束した。ところが**実機で測ると逆に遅い** (表2、図1)。未較正モデルの「最良」(1 049 ms モデル) は実機で **2 150 ms**、較正後も rpi5 偏重設計は実機 ∼2 050 ms で、均衡分割 (実機 1 427 ms) に劣る。rpi5 に 14 列中 12 列を積むと、4 コアが D キャッシュ off で一斉にメモリ競合し、孤立測定のコスト和が予測するより遥かに遅くなる。

Table 2: 構成別のモデル値 vs 実機実測 (best-of-2、クールダウンあり)。総和は全て検証済み

設計	モデル makespan	**実機 makespan**	解総和
round-robin	6 788	4 586 ms	✓
naive-contiguous	2 566	2 273 ms	✓
evolved-GA(rpi5 偏重)	1 775	∼2 050 ms	✓
hand-tuned(均衡 3 分割)	1 775	**1 427 ms**	✓

3.2 ハードウェア・イン・ザ・ループによる決着

そこで**モデルは候補の提案に限り、勝者は実機測定で決める**方式を採った。複数シードの GA チャンピオンとベースラインを実機で (クールダウンつき best-of-2 で) 測り、最速を選ぶ。結果、真のチャンピオンは**均衡 3 分割** (rpi3 [10,14) / rpi4 [0,4) / rpi5 [4,10)、実機 1 427 ms) であった。最速の rpi5 に重い中央 6 列、遅い 2 ボードに軽い端 4 列ずつ、という配置である。

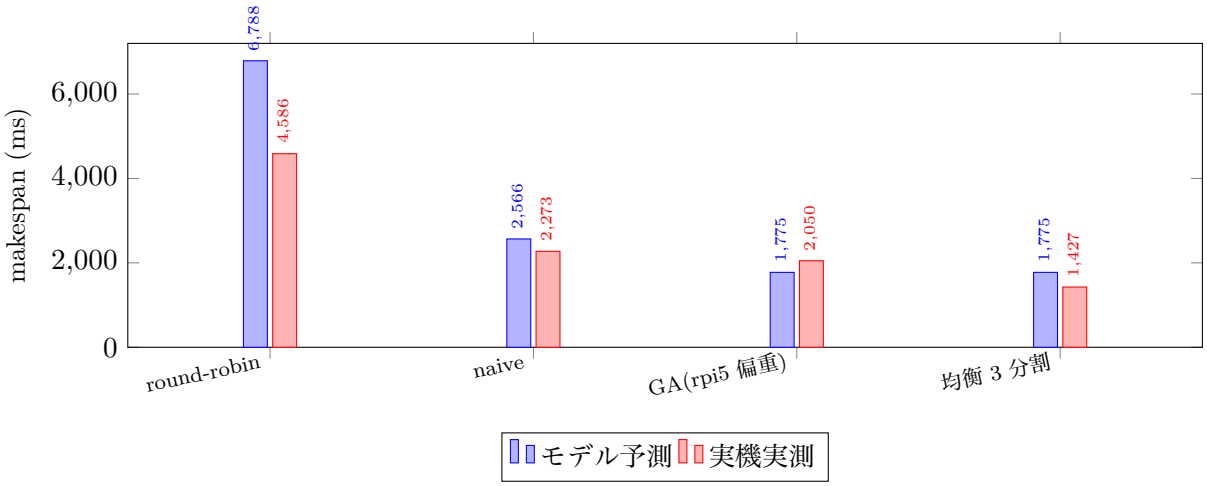

Figure 1: モデル予測 vs 実機実測。モデルは rpi5 偏重設計と均衡分割を同点 (1 775) と見るが、実機では均衡分割 (1 427) が偏重設計 (∼2 050) を明確に上回る。**モデルだけでは真の順位を誤る**。

3.3 実機実測を適応度にした GA

適応度そのものを実機実測にした GA(pop 10 × 6 世代、makespan をメモ化) も試した。機構は動作したが、**測定ノイズ** (連続高負荷でボードがスロットリング) と評価コストの高さで収束が遅く、チャンピオンは 2 487 ms にとどまった。すなわち**素朴な実測適応度 GA は高価かつノイジー**で、実用上は「モデル提案＋実機選抜」が優る。

4 考察

- **モデルは必ず不完全で、GA はその穴を突く**。孤立測定に基づくコストモデルは、4 コア同時実行のメモリ競合や逐次ランのオーバヘッドを取りこぼす。GA はそこを最大限に悪用し、**モデル上だけ速い**設計を生む。
- **実機測定だけが真の順位を与える**。本問題では近似最適が「均衡 3 分割」という人手解に一致した。GA が人手を上回れなかったのは問題が小さい (14 列・3 ボード) ためで、価値は **sim-to-real ループを閉じた方法論**にある。同じループは `xinu_rpi3_evolution` で `worker_pool` SMP を実装前に予測し、後に実機で dining ×3.99 を実証して裏付けた。
- **測定ノイズの制御が要**。連続高負荷のスロットリングを避けるクールダウンと best-of-2 で、実機順位は安定した。

5 結論と展望

Xinu メッシュを**進化計算の適応度評価器**として使い、設計を進化させ実機で測る sim-to-real ループを実機上で閉じた。純モデル GA のオーバーフィットを実測が暴き、ハードウェア・イン・ザ・ループが真のチャンピオンを同定した。今後は、人手最適化が困難な**より大きな問題** (大きい N、ノード増、コア数・分割法などの実行時可変軸をゲノムに追加) と、ボード側**汎用適応度ルート** (オンボード C-JIT / AIPL で任意の進化プログラムを採点) により、負荷分散設計から **アプリそのものの進化**へ広げる。