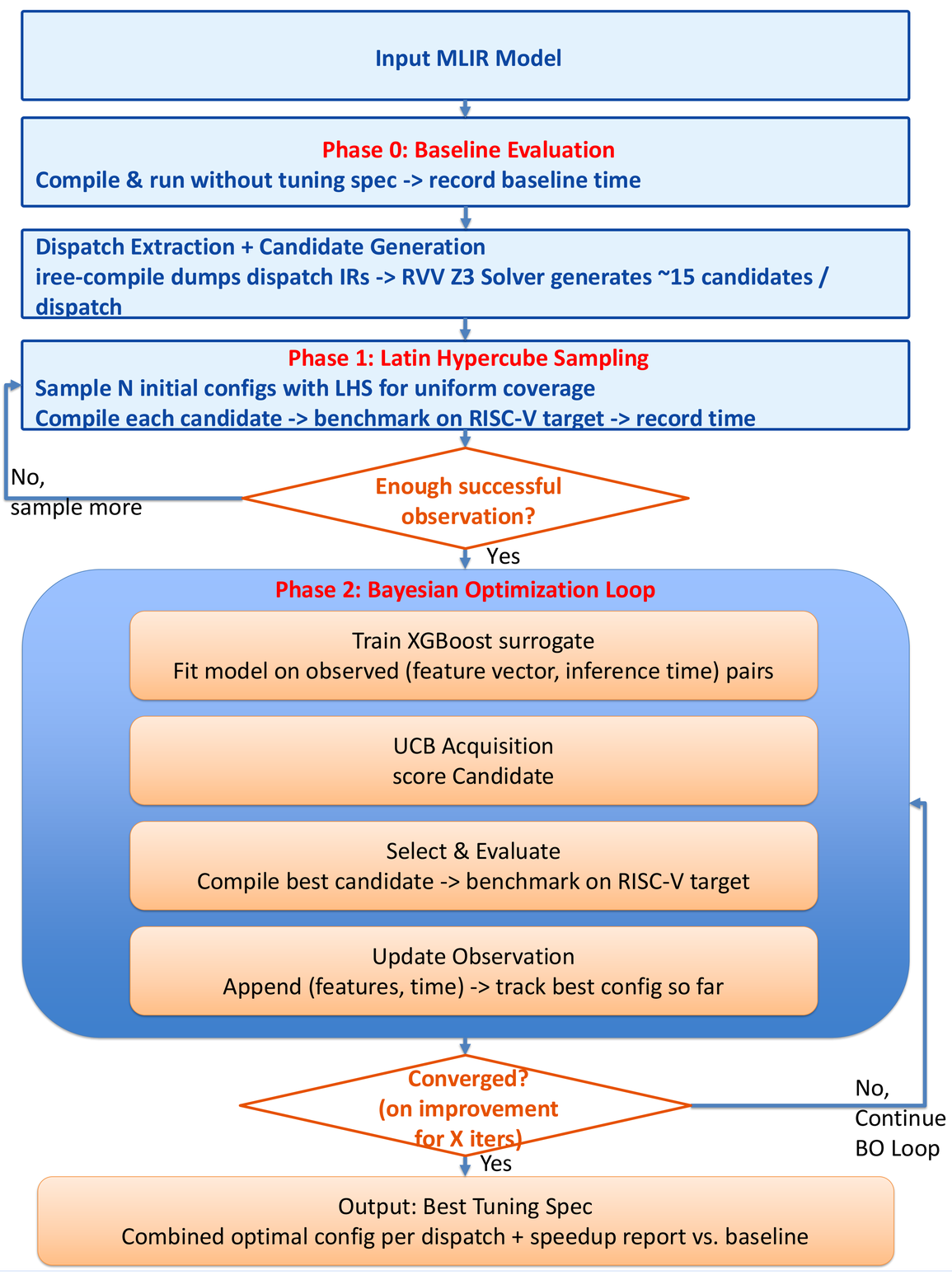
A CPU Autotuning Pipeline for MLIR-IREE

Chun-Lin Huang, Fu-Jian Shen, Kathryn Chapman, Jenq-Kuen Lee
Department of Computer Science, National Tsing-Hua University, Taiwan

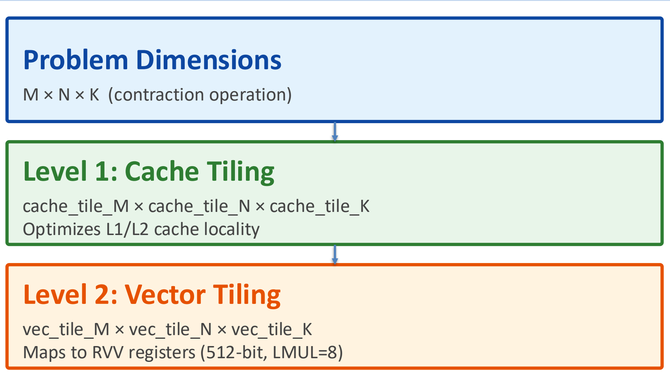
Motivation

- The IREE compiler includes CPU-oriented optimizations, but its default scheduling strategies may not always be optimal for every workload and target.
- RISC-V processors with Vector Extension (RVV) introduce a vast tiling & vectorization parameter space that is hard to tune manually.
- For **ResNet-18**, the combined dispatch-level **search space reaches ~10^{14} configurations** — exhaustive search is infeasible.
- Goal: An automated, sample-efficient tuning framework that finds near-optimal configurations with minimal overhead.

End-to-End Auto-Tuning Pipeline



Hierarchical Double-Tiling For RISC-V



Z3 constraints enforce:

- Cache tile product ≤ cache capacity
- Vector tile ≤ cache tile (hierarchical)
- Divisibility & vector-length alignment

Target Hardware

Target Triple	`riscv64`
ABI	`lp64d (double-precision float)`
Vector Extension	`RVV 1.0 (+v, +zvl512b)`
Vector Width	`512-bit (configurable up to 4096-bit)`
LMUL	`8× (register grouping)`
ISA Extensions	`+m, +a, +f, +d`
Benchmark Device	`Banana Pi BPI-F3 (SpacemiT K1)`

XGBoost + Bayesian Optimization

Three-phase approach:

Phase 1: Latin Hypercube Sampling (LHS)

- Divides each dimension into **n** equal intervals for uniform coverage
- Better exploration than random sampling in high-dimensional spaces
- Default: 10 initial samples

Phase 2: Surrogate-Guided Search

1. Train XGBoost surrogate model on observed evaluations
2. Compute UCB acquisition: $\alpha(x) = -\hat{f}(x) + \lambda \cdot d(x, D_{obs}) \cdot \sigma_y$
3. Generate 500 candidates (25% mutation + 75% random)
4. Select highest UCB score → compile & benchmark on target
5. Update observations and retrain model

Feature Engineering (17 features per dispatch):

Raw (11): M, N, K, cache tiles, distribution, vector tiles
Derived (6): log(FLOPs), log(cache_tile_size), vec_M/cache_M, vec_N/cache_N, cache_K/K, vec_K/cache_K

Phase 3: Early Stopping

- Patience: 20 iterations without improvement
- $\lambda = 0.1$ (exploration weight, configurable)

Experiment Result

CNN models compiled via IREE targeting RISC-V (RV64GCV, 512-bit RVV). Benchmarked remotely on Banana Pi BPI-F3 (SpacemiT K1) via SSH. Baseline is compiled with IREE's default scheduling (no tuning spec applied). All inference times report the **median of 5 runs**.

Model	Params	Baseline(ms)	Tuned(ms)	Speedup
MLP	custom	0.557	0.424	**1.31x**
MNASNet 0.5	2.2M	552	114	**4.84x**
ResNet-18	11.7M	3128	2812	**1.11x**
SqueezeNet 1.1	1.2M	2359	593	**3.98x**
MobileNetV2	3.5M	1784	383	**4.66x**
ShuffleNet V2	2.3M	220	41.1	**5.35x**
EfficientNet-B0	5.3M	2220	583	**3.81x**

Key Observations:

- Auto-tuning achieves **1.11x–5.35x** speedup across **7** CNN models on RISC-V, with an average of **3.58x.**
- Lightweight models with depthwise separable convolutions (ShuffleNet, MNASNet, MobileNetV2) benefit most, achieving **4.66x–5.35x** speedup.
- The XGBoost-guided Bayesian search converges within **50–150 iterations** (2–40 minutes), making it practical for on-device tuning exploration.
- Models with large convolution kernels (ResNet-18) show limited gains due to constrained search space on RISC-V's vector register capacity.

Conclusion

Contributions:

- Fully automated tuning pipeline for IREE on RISC-V with RVV support.
- Hierarchical double-tiling strategy modeling cache and vector register levels.
- XGBoost + Bayesian Optimization with LHS for sample-efficient search in 10^{14}-scale spaces.
- Our auto-tuning framework achieves **1.11x–5.35x speedup** across 7 CNN models on RISC-V, with an average of **3.58x**, by jointly optimizing two-level tiling parameters through XGBoost-guided Bayesian search in under 40 minutes per model.

Ongoing Work

Custom RISC-V FP8 Data Type & Numerical Precision Tuning

- Our lab is developing a custom FP8 (8-bit floating-point) data type as a RISC-V ISA extension, targeting low-power, high-throughput ML inference.
- Plan to integrate FP8 support into the auto-tuning runner, enabling joint optimization of both tiling configurations and numerical precision.
- Numerical precision tuning: automatically search the trade-off between inference accuracy and throughput by selecting FP32, FP16, or FP8 per dispatch.
- This enables a multi-objective search space combining scheduling decisions (tiling, vectorization) with precision decisions (data type selection) — a unique co-optimization opportunity on RISC-V.

Other Directions

- Extend to Transformer-based models (attention dispatch tuning).
- Transfer learning: reuse tuning knowledge across similar dispatches and models.
- Support for longer RISC-V vector lengths (1024-bit, 2048-bit) on next-gen hardware.