Bridging Runtime Gaps in LLVM: Vendor-Agnostic Dispatch for ML Kernels

S. Akash | IIT Patna · CERN GSoC · vLLM contributor

EuroLLVM Developers' Meeting · Dublin 2026

The Gap

MLIR compiles one `gpu.module` to 3+ GPU vendors, but picks the first compatible binary at runtime.

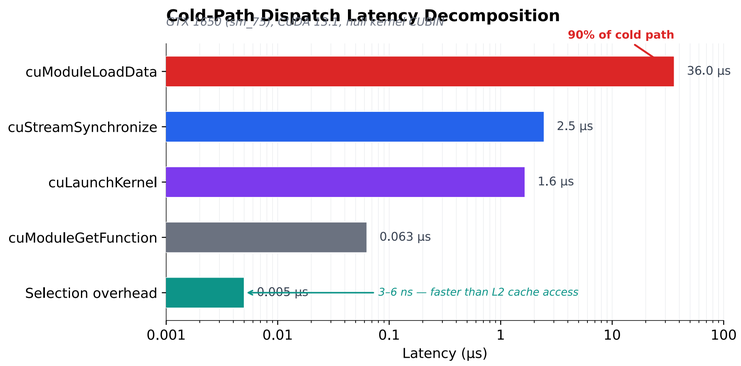
OffloadBinary carries N device images. The runtime loads the first image that doesn't fail. No metadata vocabulary. No measurement. No "best-compatible" mechanism.

Upstream Evidence

PR / Issue	What it shows
#148286	XeVM: new vendor images arriving fast
#186088	liboffload uses first-wins selection
#185663	`isMetadataCompatible`: no policy
#75356	Chapel users need dispatch
#88170	RFC: policy slot explicitly empty

5 independent signals pointing at the same gap.

Phase 1: Dispatch Measurement



36 µs	3--6 ns	< 0.02%
cuModuleLoadData (90% of cold path)	Selection overhead	Dispatch cost vs. kernel load

Selection is free relative to driver costs. So: what information should drive it?

Metadata Vocabulary

Key	Purpose
`min_sm`	Min CUDA compute capability
`min_gfx`	Min AMD GFX version
`requires_features`	Tensor cores, matrix units, etc.
`variant_priority`	Explicit ordering for tie-breaking
`variant_tag`	Human-readable variant name

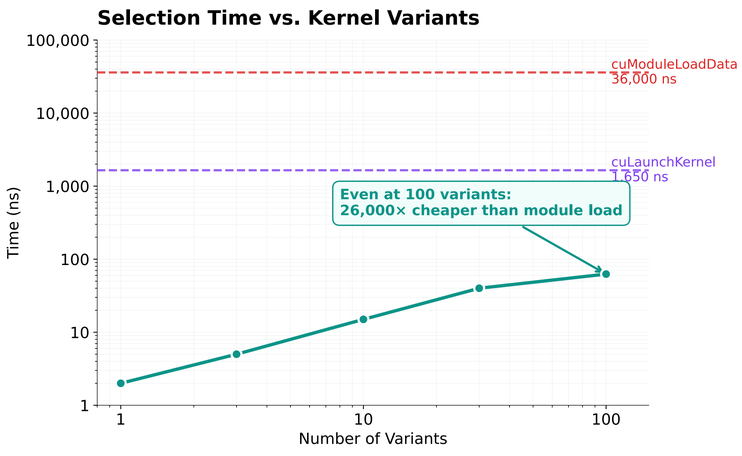
Related Work

System	Vend.	Meta	Pol.	Data	Ups.
IREE	●	●	●	●	●
chipStar	●	●	●	●	●
Proteus	●	●	●	●	●
liboffload	●	●	●	●	●
CPU FMV	●	●	●	●	●
Ours	●	●	●	●	●

● Yes ● Partial ● No

First to combine all five.

Selection Scales Linearly



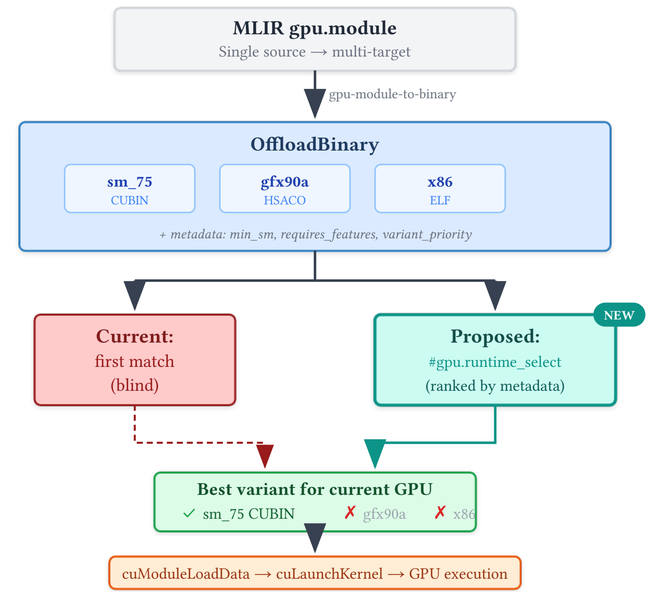
Even at 64 variants, selection stays under 400 ns. Three orders of magnitude below driver overhead.

The Insight

Since selection costs **3–6 ns**, the question is not *whether* to dispatch, but **what information drives it**. We formalize kernel variant selection as a **multi-armed bandit** problem.

Phase 1 measured the dispatch. Phase 2 makes it **intelligent**.

System Architecture



End-to-end: MLIR embeds metadata in OffloadBinary; at runtime, the MAB profiler scores each variant and dispatches the best match.

The MAB Formulation

Arms: N pre-compiled kernel variants (typically $N < 10$)
Reward: Negative execution time: $r_i = -t_{\text{exec}(v_i)}$
Context: $(kernel\_name, shape, device\_id) \rightarrow$ cacheable key
Objective: Minimize cumulative regret $R_T = \sum_{t=1}^{T} \left(\mu^* - \mu_{a_t}\right)$

Structural Properties:

$N < 10$	$\sigma^2 < 5\%$	**Cacheable**
arms (few variants)	low noise	contexts reuse

Why This Is a Degenerate Bandit

Standard MAB algorithms (UCB1, Thompson Sampling) are designed for $N \to \infty$ or non-stationary rewards. GPU kernel dispatch has **none** of these complexities:

Exhaustive exploration ($N \times w$ warmup samples) followed by **permanent exploitation**.
Regret: $O(N)$ constant, provably optimal for this problem class.
UCB1 and Thompson Sampling are unnecessary.

With $N = 3$ variants and $w = 3$ warmup rounds, convergence is guaranteed in **9 dispatches**. After that, every dispatch is optimal with zero marginal regret.

The Algorithm

```
fn dispatch(ctx, variants[N], warmup=3):
  key = (ctx.kernel, ctx.shape, ctx.device)
  if key in cache: return cache[key]
  if stats[key].count < N*warmup:
    arm = stats[key].count % N
    t = time(variants[arm])
    stats[key].update(arm, t)
    return variants[arm]
  cache[key] = argmin(stats[key].median)
  return cache[key]
```

Three phases: **Cold** (no data) ⟶ **Explore** (round-robin warmup) ⟶ **Exploit** (permanent lock).

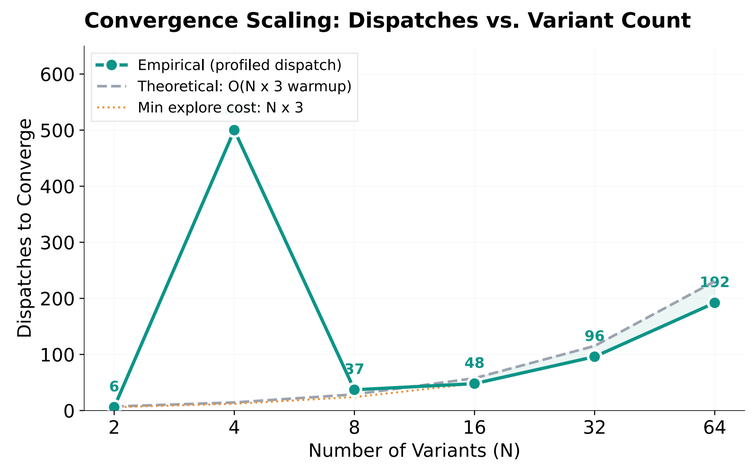
Key Findings

F1: Module loading dominates cold dispatch at 90%; selection is essentially free.
F2: GPU dispatch is a *degenerate* bandit; exhaustive exploration provably optimal.
F3: Convergence in 9 dispatches; zero marginal regret after lock-in.
F4: Linear variant scaling works from 2 to 64+ variants without changes.

Dispatch Latency Breakdown

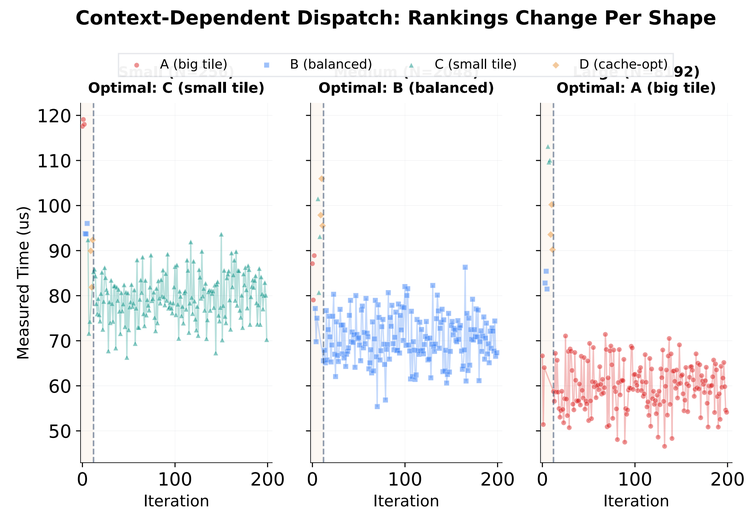
Operation	Latency	Share
cuModuleLoadData (cold)	36.0 µs	89.6%
cuModuleLoadData (warm)	9.6 µs	–
cuModuleGetFunction	63 ns	0.2%
cuLaunchKernel	1.65 µs	4.1%
cuStreamSynchronize	2.45 µs	6.1%
Hot-path total	**4.1 µs**	**launch+sync**
Selection overhead	**3–6 ns**	**< 0.02%**

Scaling with Variants



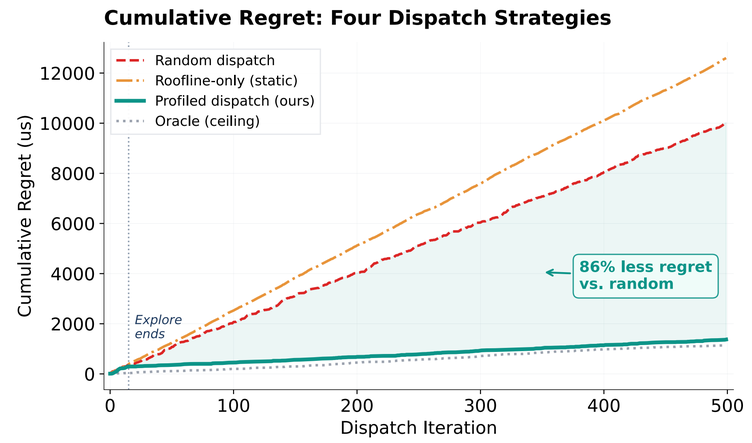
Profiled dispatch achieves 83% of oracle with 7.3x less regret than random.

Context-Dependent Selection



Different shapes converge to different optimal variants; context matters.

Profiled vs. Baselines



83%	O(N)	7.3x
Of oracle performance	Regret bound (constant)	Less regret than random

With 5 near-identical variants (53% spread, 6% noise), profiled dispatch converges and achieves near-oracle performance, 7.3x better than random, robust to noise.

Contact: 2201ee54_sakash@iitp.ac.in · drakathakash@gmail.com