Gülçin Gedik, Robert Schöne
ZIH, CIDS, TU Dresden, Germany

A Taxonomy of Application Properties for Mixed-Precision Autotuning

Properties shape tuning approach and feasibility

1. Motivation: Promise vs Reality

HPC systems increasingly rely on reduced-precision hardware for peak throughput and energy efficiency largely driven by machine learning workloads, but scientific applications cannot automatically benefit from it.

Promise:
- Faster Computation
- Better Energy Efficiency
- Leverage Modern Hardware

Reality:
- Tedious Process
- Domain Expertise
- Unpredictable Cost

Key question: Which application properties make an HPC code a good or poor candidate for tool-based mixed-precision tuning?

Generic Mixed Precision Autotuning Workflow Phases:

Profiling → Sensitivity Analysis → Search Space Reduction → Search & Evaluate → Transform

Primary Cost Drivers: (Defines tuning effort)

Instrumentation Overhead · Validation Runs · Cast/Compiler Interference · Portability Risk

Design Levers: (Defines tuning procedure)

Granularity & Clustering · Validation Oracle · Integration Stage · Objectives & Metrics

3. Why Some Codes, Not Others?

Success Cases:
- ✓ Weather & Climate
- ✓ Geospatial Modeling
- ✓ Quantum Chemistry

Struggles:
- × Molecular Dynamics
- × Comp. Mechanics
- × Fluid Dynamics

Key Insight:
It's NOT about the tool—application **properties shape tuning** feasibility.
The same workflow stage can be trivial for one application and prohibitive for another.

Our approach: Examine applications through their properties; tuning feasibility depends on these properties and their interdependencies.

Property	Primary Workflow Impact
Structural	**Profiling & Search:** Tuning granularity and analysis cost.
Behavioral	**Validation:** Single-run reference vs. statistical validation.
Numerical	**Search Reduction:** Conservative pruning vs. aggressive exploration.
Performance	**Evaluation:** Bottleneck type and expected payoff.

4. Before You Tune: Decision Gates & Stress Case

An application is well-suited when most gates **pass**.
Stress case: Unstructured FE Krylov solver (sparse SpMV)

Gate	Krylov solver verdict
G1 [Structural] Tuning atoms exist? Hotspots localizable into precision-editable kernels	× Deep, data-dependent call structure
G2 [Numerical] Correctness check available? Affordable verification metric exists	✓ Residual-based verification
G3 [Behavioral] Stable behavior? Non-determinism bounded for reliable validation	× Convergence-based, run-to-run variance
G4 [Numerical] Recovery possible? Robust recovery enables safe exploration	× Error accumulates across iterations
G5 [Performance] Hardware benefit? Bottleneck transparent; casts unlikely to erase gains	× Memory-bound, cast overhead salient

Result: 4/5 gates fail ⇒ expensive profiling & validation, limited safe pruning, fragile payoffs.

Suitability must be judged holistically, not by any single property.

2. Four Application Properties That Shape Tuning Feasibility

STRUCTURE — Where can we tune?

What it captures:
Code organization, data flow, and parallel boundaries
- Kernels / loops / data structures define tuning atoms.
- Irregular access, deep calls, and communication boundaries raise analysis cost.
- Tuning granularity controls search-space size and cast density.

BEHAVIOR — Is the execution trajectory changed?

What it captures:
Runtime progression, control-flow stability, and variability
- Fixed iteration → final-output validation may suffice.
- Convergence/adaptive logic → track trajectories, stalls, false convergence, fallback frequency.
- Input-sensitive or non-deterministic runs → multiple inputs, repeated runs, or statistical validation.

NUMERICAL — What error is acceptable?

What it captures:
Sensitivity, recovery, and correctness criteria
- Significant variables, residuals, invariants, or domain scores define the oracle.
- Recovery mechanisms enable safer exploration.
- Ill-conditioning or strict constraints force conservative pruning.

PERFORMANCE — Will it even help?

What it captures:
Hardware support, bottlenecks, and conversion effects
- Compute-bound regions can exploit low-precision units.
- Memory / communication-bound regions depend on reduced data movement.
- Casts, compiler effects, and portability risks may erase gains.

5. False Friend: When It Doesn't Pay Off

Even at kernel level, the taxonomy exposes hidden costs.

```
void filter2x2(const float* in,
               float* out, int w, int h,
               const float filter[2][2])
  {
#pragma omp parallel for
  for (int y = 0; y < h - 1; ++y) {
    auto i0 = in + y*w, i1 = i0 + w;
    auto o  = out + y*(w-1);
#pragma omp simd
    for (int x = 0; x < w - 1; ++x)
      o[x] = i0[x]   * filter[0][0]
           + i0[x+1] * filter[0][1]
           + i1[x]   * filter[1][0]
           + i1[x+1] * filter[1][1];
  }}
```

Taxonomy Mapping:

Property	Kernel characteristic
Structural	Regular dense stencil; OMP parallel loop
Behavioral	Fixed iteration count; deterministic output
Numerical	Simple arithmetic; no ill-conditioning
Performance	→

3/4 properties *look* favorable — the **performance** bottleneck shift makes it a false friend.

6. False Friend: Instruction-level Example

Instruction-level view: lowering compute precision does not always pay off.

Cast overhead surpasses the theoretical gain.

Target: Intel x86 + F16/AVX512 Sapphire Rapids

Bottleneck: Memory-bound at fp32 storage; reduced precision compute arithmetic in fp16 brings conversion overhead. Keeping storage at fp32, compute at fp16 does not pay off, even when processing 2× elements per vector.

Operation	Instruction	ΔRT Cycle Demand
Load fp32 registers (i0, i1)	—	+0
Convert fp32 → fp16	8×`vcvtps2phx` + 4×`merge`	+12
Compute in fp16	1×`mul` + 3×`fma`	−2
Convert fp16 → fp32	2×`vcvtph2psx` + 1×`extract`	+3
Store fp32 result	—	+0
Net	conversion/shuffle dominates compute	

Best-case SIMD: 2×16 FP32 → 32 FP16 per full ZMM
15 conversion/pack operations per 32 outputs vs. only 2 cycles gained in arithmetic.

7. Conclusions & Future Work

Summary of Contributions:
- **Application-centric framework** for understanding mixed-precision autotuning feasibility
- **Systematic taxonomy** organizing properties into four categories
- **Impact matrix** explaining why tuning difficulty varies fundamentally across applications
- **Practical checklist** with five decision gates for assessing application readiness
- **Foundation for fairer tool comparisons** by explicitly considering application property space

Future Directions:
- Empirically validate and develop quantitative metrics and thresholds for decision gates
- Study convergence of in-place tuning with algorithmic mixed-precision redesign

Acknowledgments: Supported by NHR-Verein e.V. through NHR Graduate School (German BMFTR & Free State of Saxony, Grant No. NHR2025SN).
Contact: gulcin.gedik@mailbox.tu-dresden.de

References & DOI:
[1] Gülçin Gedik, Christian von Elm, Robert Schöne, "A Taxonomy of Application Properties for Mixed-Precision Autotuning", ICPE 2026
DOI: 10.1145/3777911.3801102