Intercepting Functions for Memoization
A case study using transcendental functions

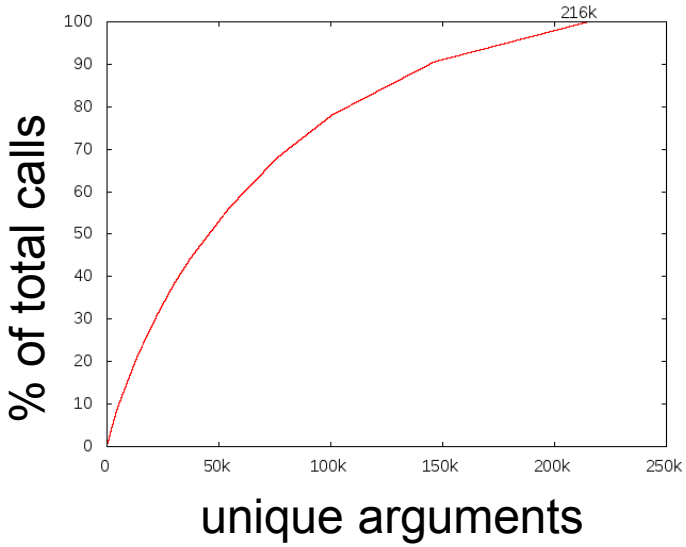
Arjun Suresh, Bharath Narasimha Swamy,
Erven Rohou, André Seznec

HiPEAC Conference, Prague, Jan 2016

Motivation
Function arguments repeat often

- Not necessarily poorly written or un-optimized code
- Example
  - calls to libm/j0 from ATMI Goh
  - total of 21.9M calls
  - 216k unique arguments

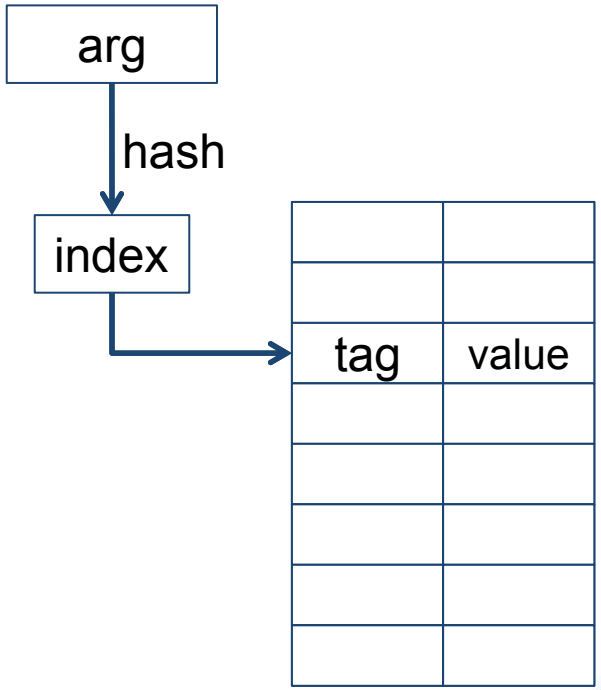
Size of sliding window	Captures
4 k	8 %
16 k	13 %
64 k	28 %
256 k	100 %



Opportunity: cache results

- Store tuples `(args, value)` in table
  - managed as a cache
- Lookup table before calling again
- Trade CPU time for storage

```
float fun(arg)
{
  index = hash(arg)
  if (table[index][0] ≠ arg) {
    table[index][0] = arg
    table[index][1] = original_fun(arg)
  }
  return table[index][1]
}
```



Condition: pure functions

- Pure function
  - returns the same result for the same input
  - no side-effect

Savings

```
float fun(arg)
{
  index = hash(arg)
  if (table[index][0] ≠ arg) {
    table[index][0] = arg
    table[index][1] = original_fun(arg)
  }
  return table[index][1]
}
```

miss

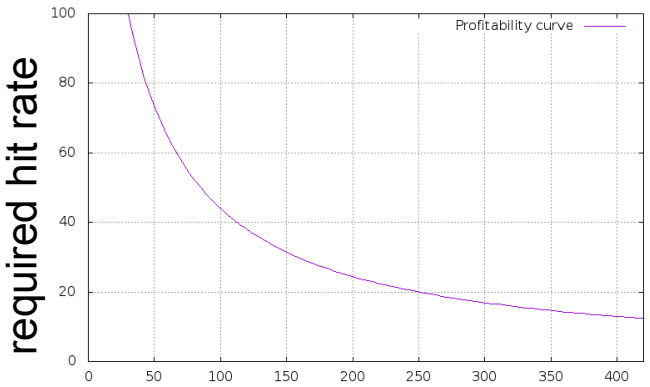
- Assume
  - T_f: execution time of function
  - t_h: time to retrieve data in table (hit)
  - t_{mo}: overhead in case of miss
  - H: fraction of hits
- We want:

hits misses

$$H \times t_h + (1 - H)(T_f + t_{mo}) < T_f$$

- We need:

$$H > \frac{t_{mo}}{T_f + t_{mo} - t_h}$$

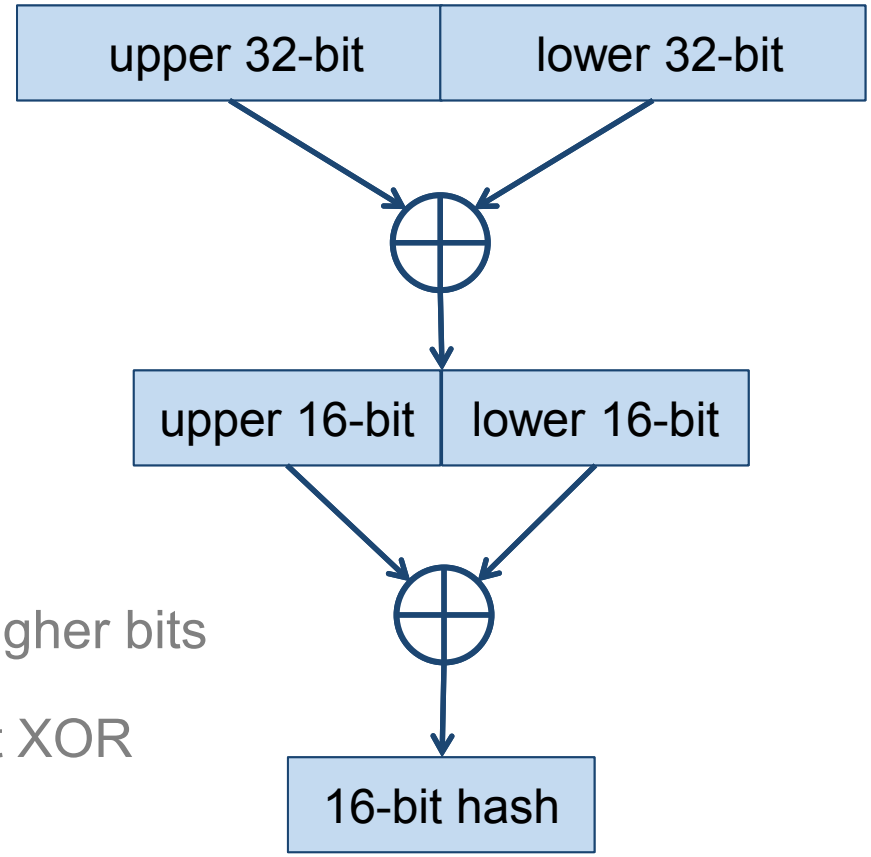


Implementation

- Hash function
- Select pure functions
  - focus on the math library `libm.so`
- Intercept library functions
  - rely on operating system mechanism `LD_PRELOAD`

Hashing

- Must be fast
- Simple XOR-ing
  - 6 SSE instructions
  - few collisions (good enough)
- For smaller indices, mask higher bits
- For multiple arguments, first XOR arguments

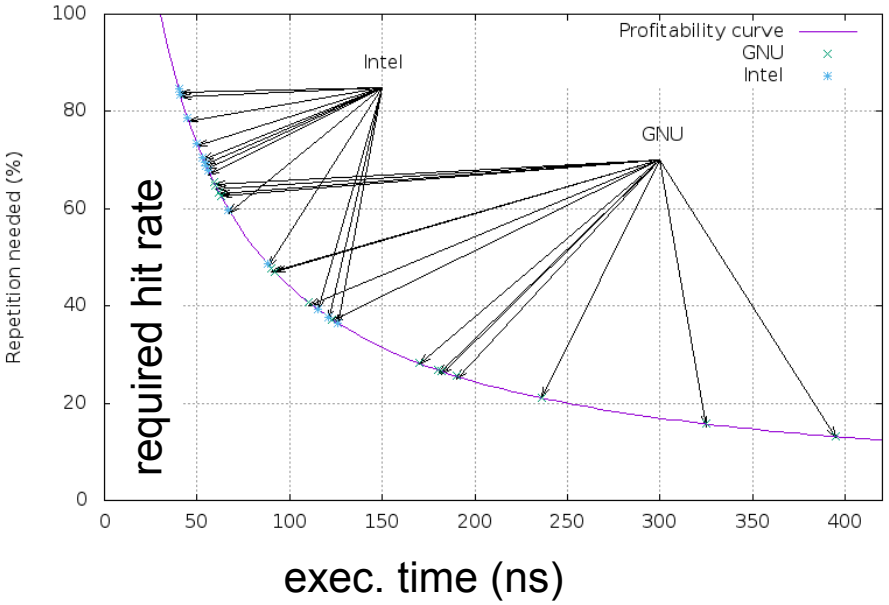


Pure functions: `libm.so`

- Standard math library on UNIX/Linux

Data on Intel Ivy Bridge, 2 GHz, GNU libm 2.2

function	hit time (ns)	miss overhead (ns)	exec time (ns)	hits needed
exp	30	55	90	48 %
log			92	47%
sin			110	41 %
cos			123	37 %
j0			395	13 %
j1			325	16 %
pow			180	27 %
sincos			236	21 %



Intercepting function calls

- Rely on OS feature
  - just set `LD_PRELOAD` env. variable
  - specify shared libraries to be loaded before others
  - handled by dynamic loader
- No recompilation, no source needed

```
% export LD_PRELOAD=/path/to/mylib.so
% ./benchmark
```

Experimental Setup

- Linux 3.11
- Intel Ivy Bridge, 2 GHz
  - GCC 4.8, `libm` 2.2
  - Intel icc 9
- ARM Cortex-A9, 1.2 GHz
  - GCC 4.7, `libm` 2.2

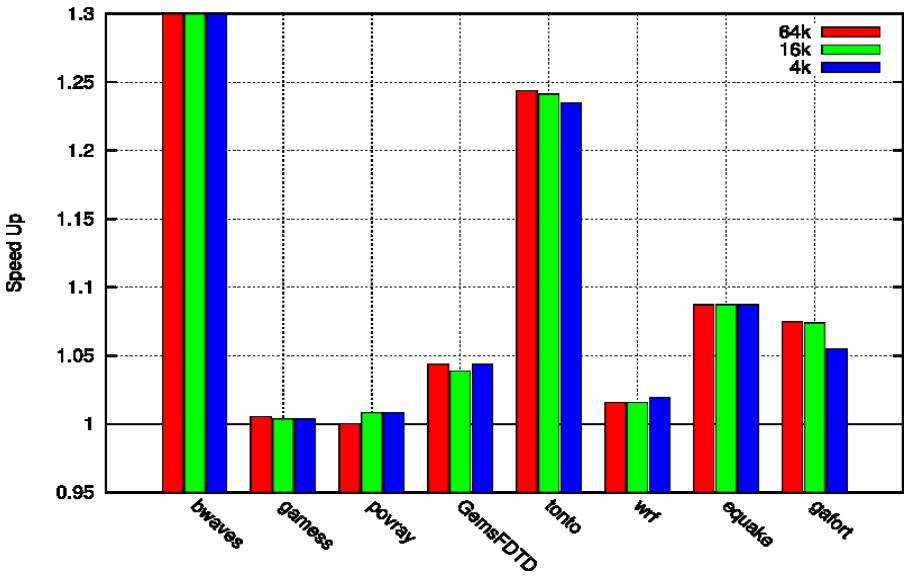
- Benchmarks
  - SPEC CPU 2006
  - SPEC OMP 2001
  - Splash 2x PARSEC
  - Barsky
  - ATMI
    - processor temperature
  - Population dynamics
    - convection-diffusion-reaction equations

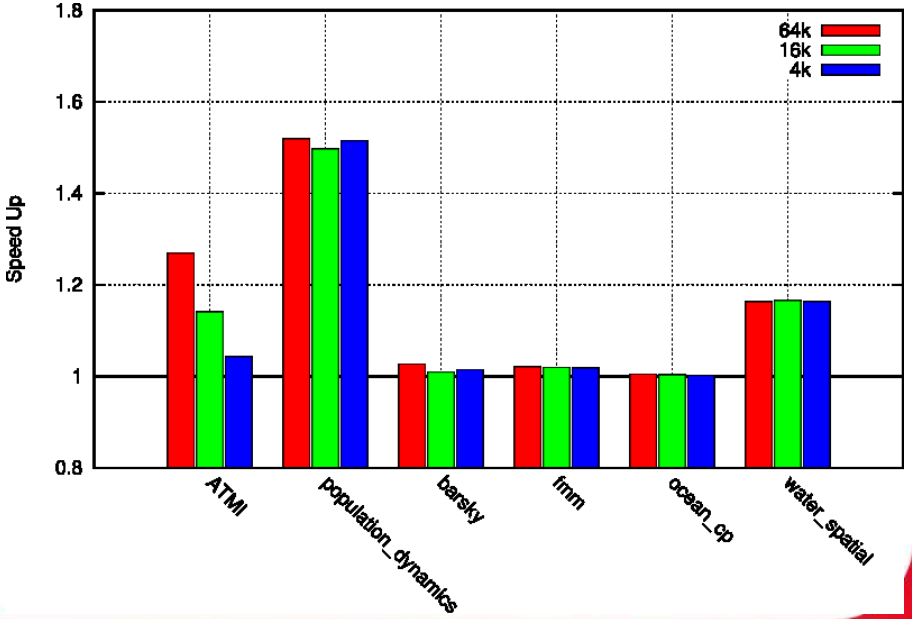
Results

- SPEC speedups +1% – +24%
- ATMI: +27%
  - extensive use of Bessel functions

    `j0` and `j1`
- Population dynamics: +52%
  - `exp`, `log`
- Outstanding: bwaves

~1.7





“Fixing” Performance bug

- `pow()` 10,000x too slow for some input values
  - bug known since 2012
  - `https://sourceware.org/bugzilla/show_bug.cgi?id=13932`
- In bwaves, we experience 1000x
  - for $0.75^{1.000...}$
- Memoization nicely covers for the extra latency
  - 1.7x speedup overall

Rounding modes

- Math functions are not, strictly speaking, pure
  - results depend on rounding modes
- Rounding modes are set by a function call
  - `fesetround()`
- Intercept, flush tables

Automatic Disabling

- Possible performance degradation, due to
  - lookup overhead (t_{mo})
  - poor argument locality
- Helper thread monitors hit rate
- Can disable memoization
  - change the address of the function in the GOT

  (Global Offset Table)

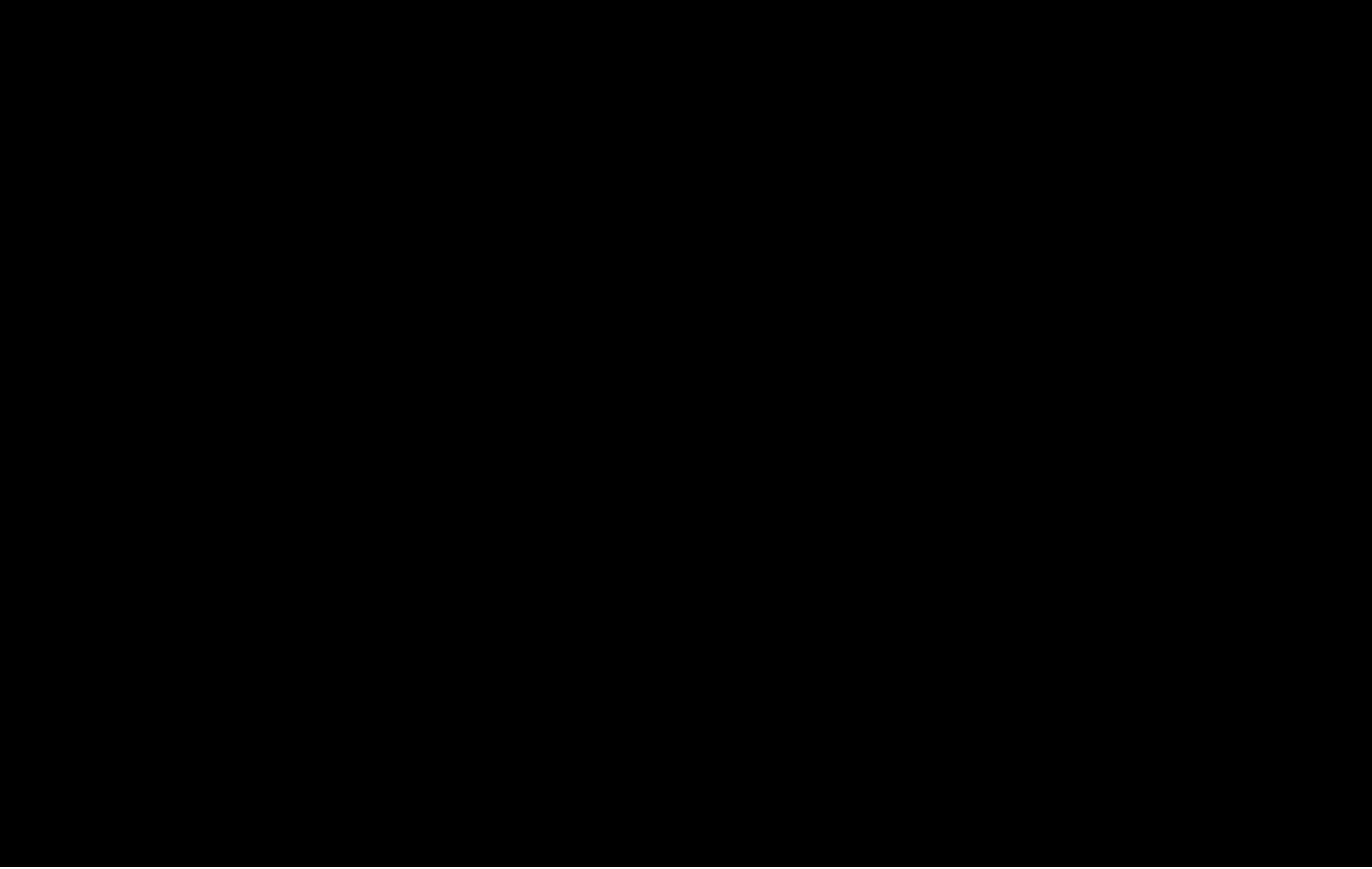
- Can restore later, to account for program phases

Summary

- Function arguments repeat often
- Memoization is an effective way to improve performance
- Transparent approach for the user

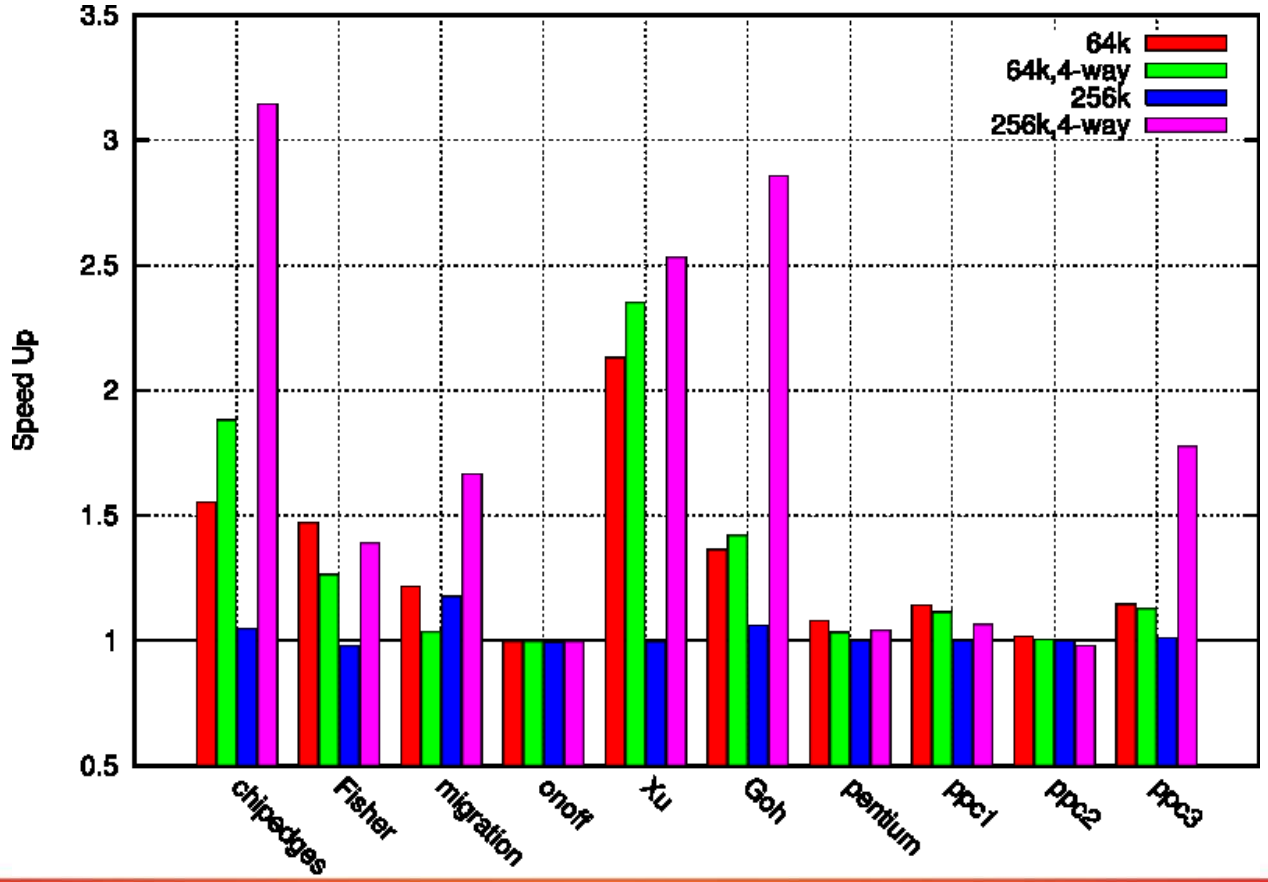
Future Work

- Compile-time detection of pure functions
  - automatic analysis
  - memoize user code
  - wrapper inlining
- Hardware support
  - specialized `call` instruction
  - lowers profitability threshold



Associativity

- Only beneficial for Bessel functions in ATMI



LD_PRELOAD

- Indirect calls to DLLs
- Actual address resolved

  by loader at run-time

  – GOT (Global Offset Table)