Byfl: Veni, Vidi, Numerari

Scott Pakin

Applied Computer Science Group (CCS-7)

14 August 2015

UNCLASSIFIED

Outline

- Motivation
- Approach
- Examples
- Conclusions

Applications in the ~~Exascale~~ Timeframe

Rintascale Zettascale Udascale Xonascale Mingascale Ochascale Vundascale Yottascale Nenascale Tredascale Quexascale Wekascale Lumascale Peptascale Sortascale

- Want apps being developed today to run fast on tomorrow's supercomputers
- Architectural details are as yet unknown:
  - Cache sizes and associativities, memory and storage latencies and bandwidths, functional units per threads, threads per core, cores per socket, sockets per node, extent of coherence domains, on/off-chip network topologies, communication latencies and bandwidths, ...
- Can't simulate or model what we don't know
- Typical approach: Optimize for current supercomputers and hope that future supercomputers aren't *too* different

NNSA
National Nuclear Security Administration

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos
NATIONAL LABORATORY
EST. 1943

Applications in the Exascale Timeframe

- Want apps being developed today to run fast on tomorrow's supercomputers
- Architectural details are as yet unknown
  - Cache sizes and associativities, memory and storage latencies and bandwidths, functional units per threads, threads per core, cores per socket, sockets per node, extent of coherence domains, on/off-chip network topologies, communication latencies and bandwidths, …
- Can't simulate or model what we don't know
- Typical approach: Optimize for current supercomputers and hope that future supercomputers aren't *too* different

NNSA National Nuclear Security Administration

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos NATIONAL LABORATORY EST.1943

Hardware Performance Counters

- Provided by all modern processors
- Tally various microarchitectural events
  - Flops, cache misses, branch mispredicts, …
- Inform many performance-analysis tools
  - VTune, CrayPat, PAPI, HPCToolkit, …
- Pros
  - Detailed HW information, unintrusive
- Cons
  - Vary by processor, limited number usable at once, often highly unintuitive interpretations

UNCLASSIFIED

NNSA National Nuclear Security Administration

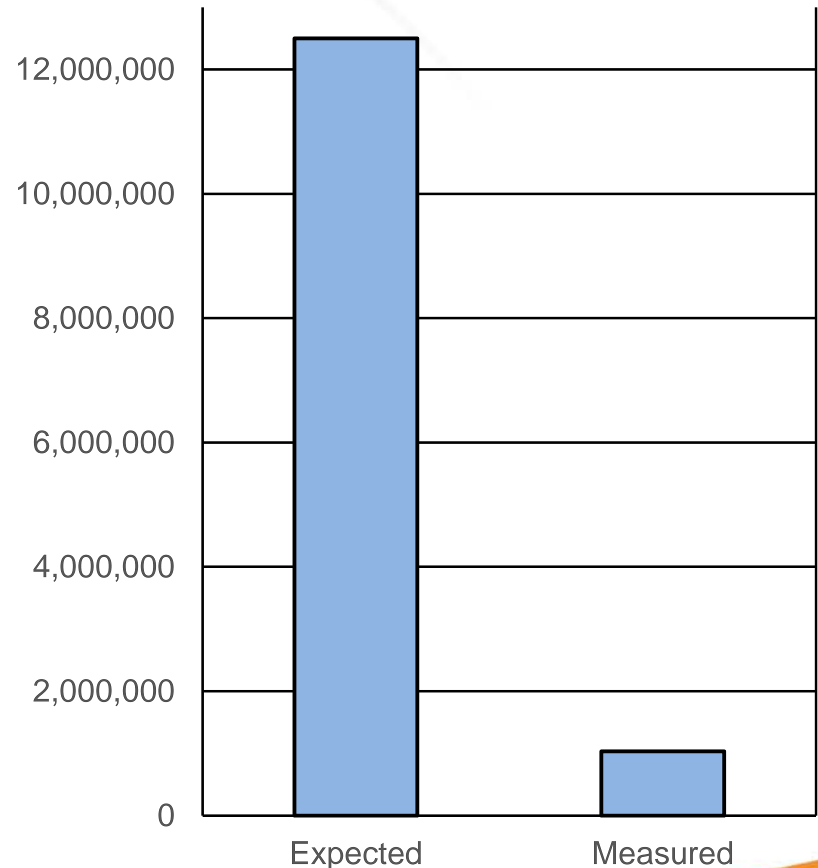
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos NATIONAL LABORATORY EST. 1943

Measuring Memory Accesses

```
for (i = 0; i < 100000000; i++)
  sum += array[i]; /* doubles */
```

- How many main-memory accesses?
  - Measure L3 cache misses
  - 100M accesses × 8B/access × 1 line/64B × 1 miss/line = 12.5M misses
- Performance counter results
  - Tally is only 1M
  - Why? Because prefetches don't count as misses on Intel processors



Measuring Floating-Point Operations

```
double apply_shape (int n, float *v, float *vs)
{
  double accum = 0.0;
  int i;

  for (i = 0; i < n; i++) {
    float x = fabsf(v[i]);
    vs[i] = (1.0f - x) * (x <= 1.0f);
  }
  for (i = 0; i < n; i++)
    accum += (double) vs[i];
  return accum;
}
```

SP map

DP reduce

- How many flops does the above perform?
  - Expect somewhere between 3 (`-`, `*`, `+=`) and 6 (also `fabsf()`, `<=`, `(double)`)

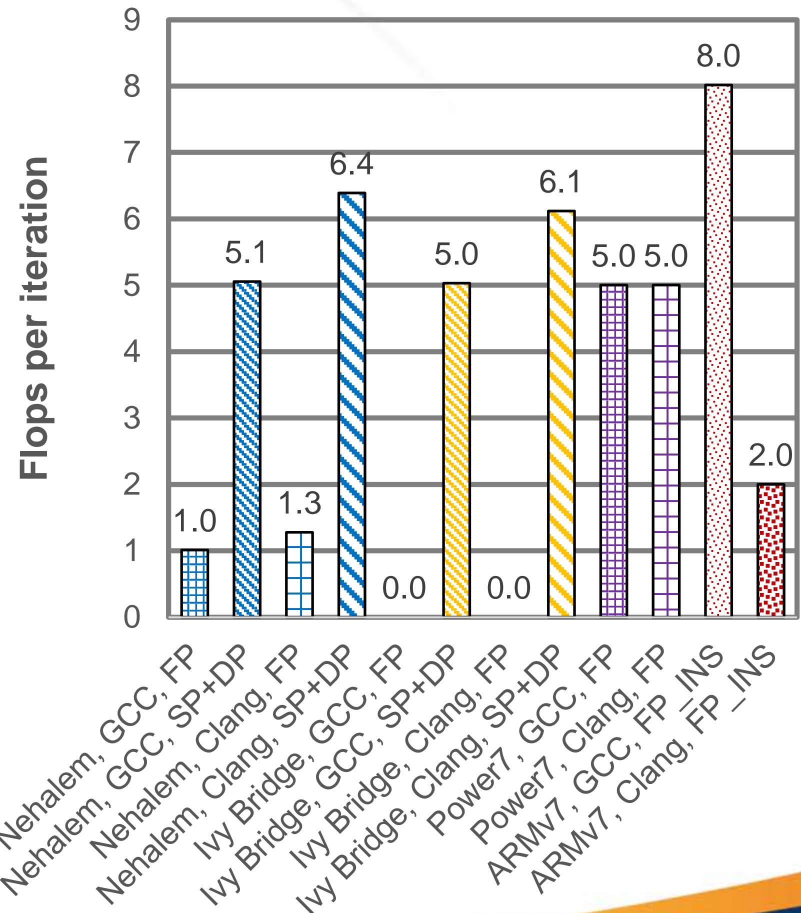
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

NNSA National Nuclear Security Administration

Los Alamos NATIONAL LABORATORY EST.1943

Measuring Floating-Point Operations

- Performance counter results
  - Range is from 0 to 8
  - Results depend on microarchitecture, compiler, and specific counters used
- Explanations
  - Vector flops not always counted as flops
  - FP register motion may count as flops



Key Insight

- Don’t need microarchitectural details for a first-order analysis of future application performance

Likely to perform well	Unlikely to perform well
Flop-heavy codes (actually, also integer-op-heavy codes)	Branch-heavy codes, especially unpredictable or divergent branches
Codes with lots of memory locality (temporal, perhaps also spatial)	Codes randomly accessing large amounts of memory
Data-parallel codes (e.g., vector or SIMT parallelism)	Highly serial codes

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA
NNSA National Nuclear Security Administration
Los Alamos NATIONAL LABORATORY EST.1943

Byfl Approach

- "Software performance counters"
- Instrument code at compile time
  - Tally operations of interest
  - Flops, integer ops, loads, stores, branches, …
  - Richer info than at run-time
- Accumulate counter values at run time
  - Want to handle "**while not** converged() **do** …"

```
for (i = 0; i < 100000000; i++)
  sum += array[i]; /* doubles */
```

```
for (i = 0; i < 100000000; i++) {
  sum += array[i]; /* doubles */
  num_loads++;
  num_flops++;
  bytes_loaded += 8;
}
```

Artist's conception. Transformation actually performed on the compiler's intermediate representation.

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

NNSA National Nuclear Security Administration

Los Alamos NATIONAL LABORATORY EST. 1943

Benefits

- No ambiguity regarding semantics
  - When in doubt, can read tool source code
- No measurement variability across architectures
  - A divide is a divide, not a reciprocal approximation
- Not limited to a given number of live counters
- No mutually exclusive counters
- Not limited to what the hardware can measure
- Not limited to scalar counters

UNCLASSIFIED
NNSA National Nuclear Security Administration
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA
Los Alamos NATIONAL LABORATORY EST. 1943

Analyzing Application Performance

- Somewhat different thought process
  - Not hardware-centric but application-centric

Ask not	Ask
How many cache (or TLB) misses did my app observe?	What is my app’s working-set size?
How many flops did my app perform per second?	How many flops did my app perform per load?
How many branches were mispredicted?	How many operations did my app perform per branch?

Basic Analysis

- More detail than is provided by HW counters

Parameter	Measurement (SNAP)
Load operations	25,952,958,723
Store operations	5,297,479,266
Floating-point operations	32,338,525,886
Integer operations	127,617,520,266
Function-call operations (non-exception-throwing)	33,202,227
Function-call operations (exception-throwing)	0
Unconditional and direct branch operations (removable)	1,109,576,789
Unconditional and direct branch operations (mandatory)	64,884,317
Conditional branch operations (not taken)	1,249,240,855
Conditional branch operations (taken)	10,986,614,142

Basic Analysis

- More detail than is provided by HW counters

Parameter	Measurement (SNAP)
Unconditional but indirect branch operations	0
Multi-target (switch) branch operations	10
Function-return operations	77,443
Other branch operations	0
Bytes loaded	295,721,649,378
Bytes stored	79,140,611,792
Unique addresses loaded or stored	128,318,469
Bytes needed to cover half of all dynamic loads and stores	1,928
Vector operations	11,208,258,983
Total vector elements	22,416,518,166

Example #1: “Hot” Library Calls

	SNAP		NuT	
1	54.4%	llvm.memset.p0i8.i64	20.1%	llvm.memcpy.p0i8.p0i8.i64
2	44.5%	fabs	12.0%	acos
3	0.1%	_gfortran_internal_pack	8.3%	log
4	0.1%	omp_get_num_threads	7.2%	asin
5	0.1%	omp_get_thread_num	7.2%	cos
6	0.1%	GOMP_barrier	7.2%	sin

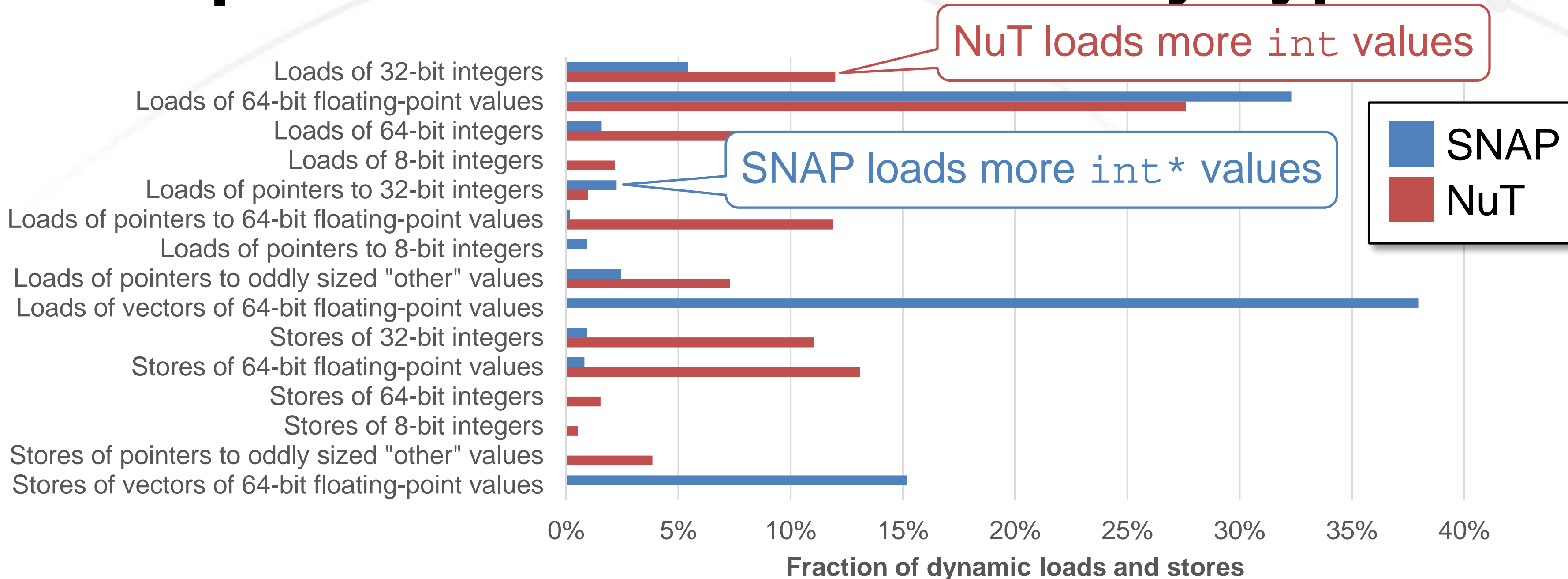
- What library calls get invoked most frequently?
  - SNAP: `memset()`, `fabs()`, and OpenMP calls
  - NuT: `memcpy()` and transcendentals

NNSA National Nuclear Security Administration

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

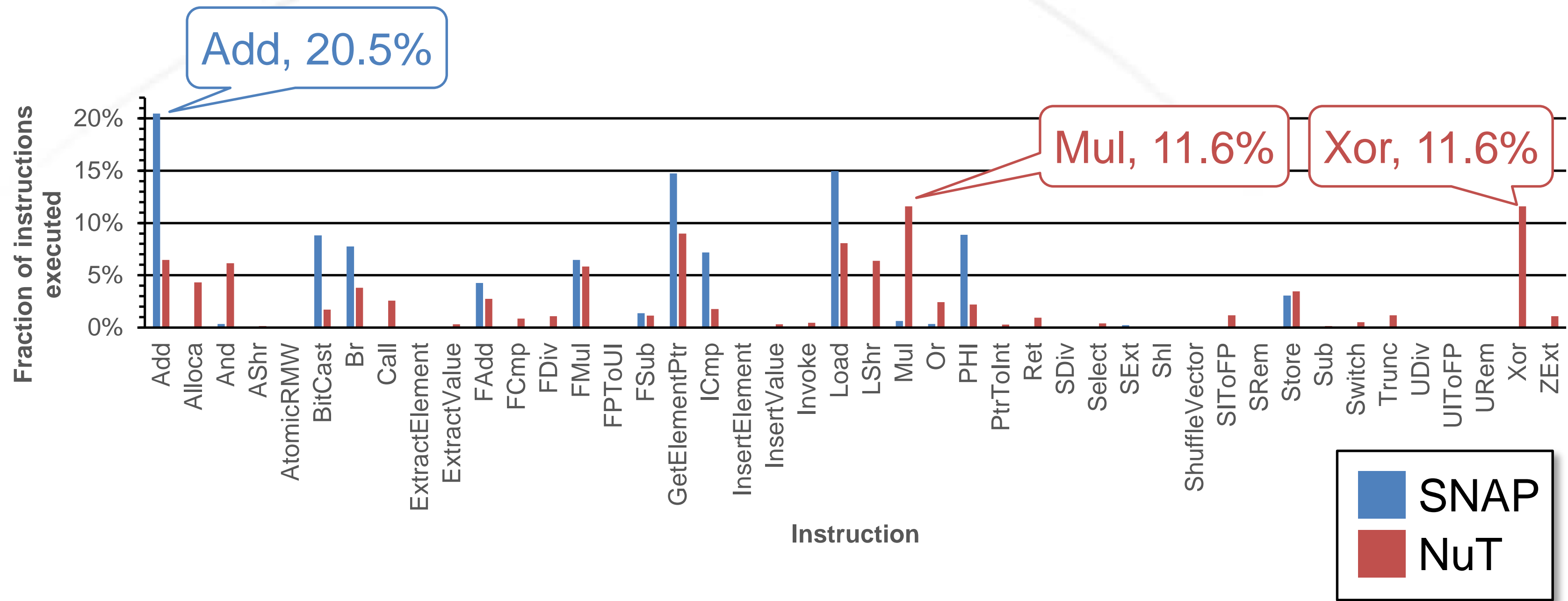
Los Alamos NATIONAL LABORATORY EST. 1943

Example #2: Loads and Stores by Type



- Are load and store data types as expected?
  - SNAP: loads of 64-bit FP vectors
  - NuT: loads of 64-bit FP scalars

Example #3: "Hot" Instructions



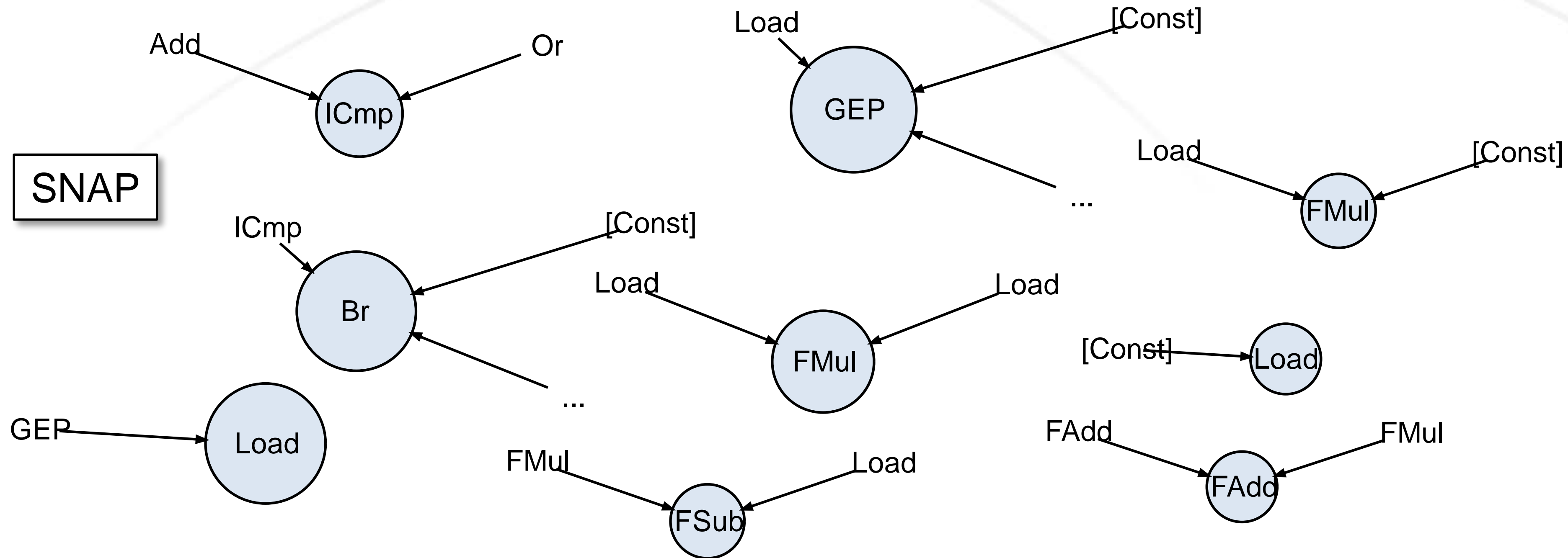
- Which instructions *should* the CPU run fast?
  - SNAP: integer adds
  - NuT: XORs and integer multiples

NNSA National Nuclear Security Administration

Los Alamos NATIONAL LABORATORY EST. 1943

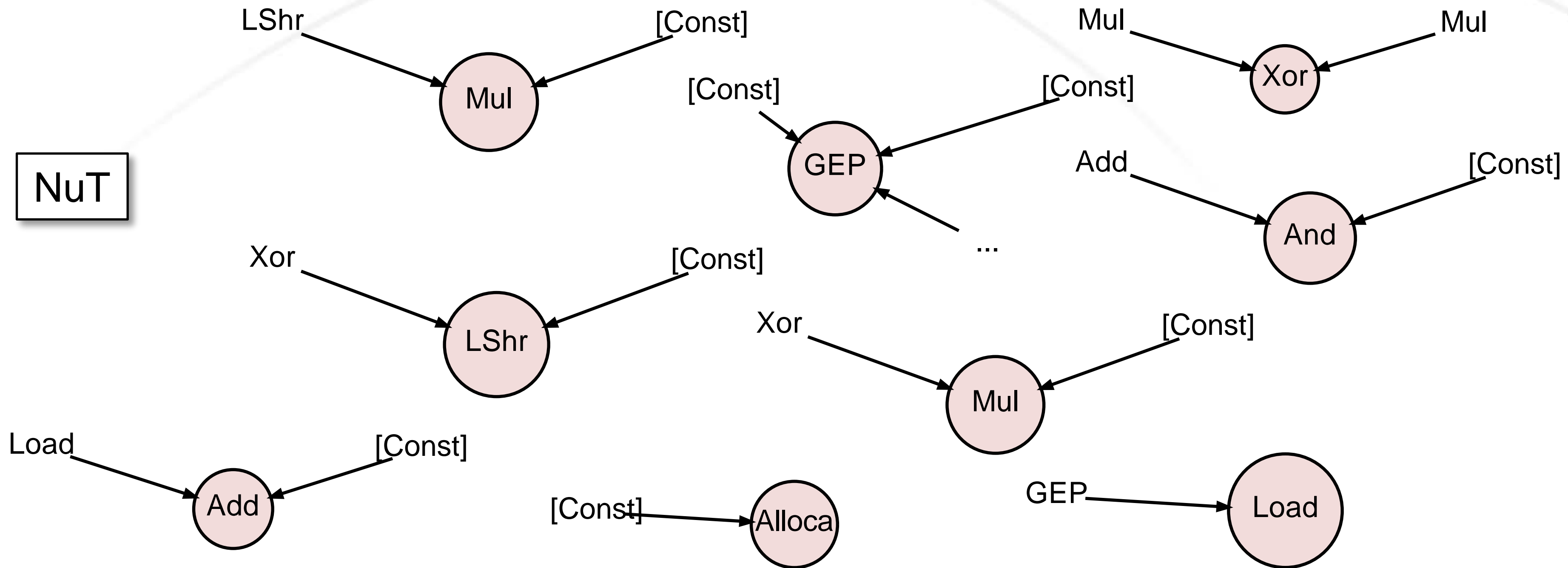
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Example #4: Instruction Dependencies



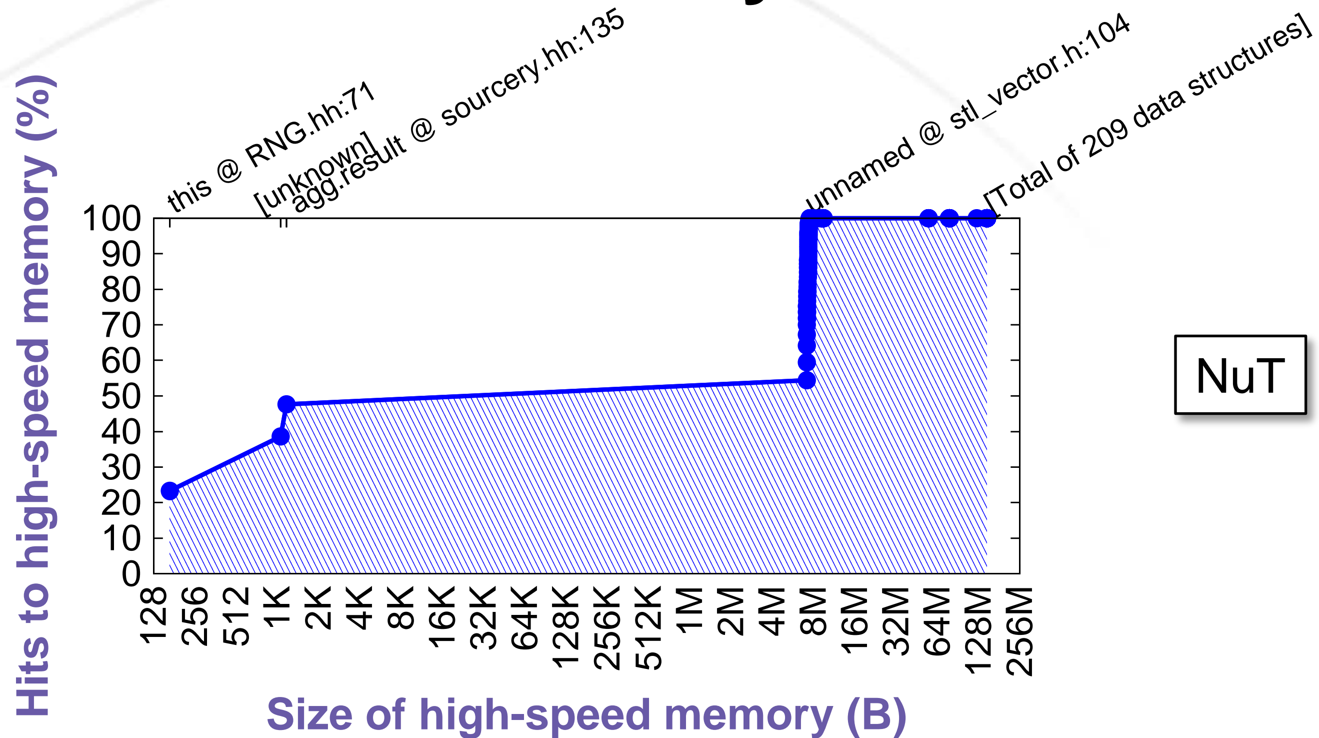
- Which instructions feed into which other instructions?
  - Think fused multiply-add: What else should be fused?
  - Useful for synthetic application mock-ups

Example #4: Instruction Dependencies



- Which instructions feed into which other instructions?
  - Think fused multiply-add: What else should be fused?
  - Useful for synthetic application mock-ups

Example #5: Accesses by Data Structure



Capacity (B)	Coverage (%)
128	20
1KB	50
8MB	<100
256MB	100

- 3 data structures cover 50% of accesses
- No difference between 4KB and 4MB of high-speed memory

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos NATIONAL LABORATORY EST. 1943

NNSA National Nuclear Security Administration

Conclusions

- Want to optimize apps for future supercomputers
  - As-yet unknown architecture → can't model or simulate
  - *Insight*: Can follow trends to know what app characteristics are likely to be good/bad for performance
- Software performance counters provide the requisite information for optimization
  - Hardware counters too grounded in today's hardware and too divorced from the app developer's view
  - Byfl's compile-time instrumentation + run-time data gathering provides richer information than either hardware or static analysis

https://github.com/losalamos/Byfl