Watching for Software Inefficiencies with Witch

Xu Liu College of William & Mary

Milind Chabbi Uber

Classical Performance Analysis

- Identify hotspots high resource utilization
 - time / CPU cycles
 - cache misses on different levels
 - floating point operations, SIMD
 - derived metrics such as instruction per cycle (IPC)
- Improve code in hot spots
- Hotspot analysis is indispensable, but
 - cannot tell if resources were "well spent"
 - hotspots may be symptoms of performance problems
 - need significant manual efforts to investigate root causes

Pinpoint resource wastage instead of usage

Software Inefficiency — Redundant Operations



Software Inefficiency — Redundant Operations



Need fine-grained binary analysis + call path analysis

HMMER: An Example for Resource Wastage



Straightforward Measurement for Inefficiencies

- Fine-grained analysis
 - Instrument every memory load and store
 - RedSpy (ASPLOS'17), LoadSpy, RVN (PACT'15), DeadSpy (CGO'12)
- Advantages
 - do not miss anything
 - serve as a proof-of-concept and upper-bound of other analyses
- Disadvantages
 - high time and space overhead



A Key Observation

Detecting a variety of inefficiencies requires monitoring *consecutive accesses to the same memory location*



Witch: Lightweight Inefficiency Analysis

- Methodology: sample pair of consecutive accesses to the same memory address
 - hardware performance monitoring units (PMU)
 - event-based sampling \rightarrow profiling memory addresses
 - first access in the pair
 - hardware debug registers
 - watch for the next access of sampled memory address
 - second access in the pair



Witch Advantages

- No source code or binary instrumentation / recompilation
- Work for fully optimized binary, independent from programming languages and models
- Capture statistically significant inefficiencies
- Low runtime and memory overhead

Challenge 1

- Limited number of debug registers
 - 4 on x86
 - 1 on PowerPC

Assume: PMU samples one in 10k memory stores



20 sampled addresses to monitor but have only 4 debug registers!

- Monitoring addresses with equal probability
 - have a free debug register \rightarrow monitor the next sample
 - no free debug register \rightarrow probabilistically replace the address from monitoring



- Monitoring addresses with equal probability
 - have a free debug register \rightarrow monitor the next sample
 - no free debug register \rightarrow probabilistically replace the address from monitoring



- Monitoring addresses with equal probability
 - have a free debug register \rightarrow monitor the next sample
 - no free debug register \rightarrow probabilistically replace the address from monitoring



- Monitoring addresses with equal probability
 - have a free debug register \rightarrow monitor the next sample
 - no free debug register \rightarrow probabilistically replace the address from monitoring



Challenge 2

Biased results



Witchcraft: Tools Built atop Witch



Witch Has High Accuracy

Witch identifies all significant inefficiencies found by exhaustive tools

Application	Inefficiencies
gcc	DeadStore
bzip2	DeadStore
hmmer	DeadStore, SilentStore
h264ref	SilentLoad
backprop	SilentStore
lavaMD	SilentLoad
NWChem-6.3	DeadStore, SilentStore

Witch's Runtime and Memory Overheads

Sampling rates	DeadCraft		Siler	otCraft	LoadCraft			
	slowdown	memory bloat	slowdown	n memory bloat slowdown		memory bloat		
10M	1.01x	1.05x	1.00x	1.04x	1.04x	1.05x		
1M	1.03x	1.05x	1.03x	1.04x	1.27x	1.07x		
500K	1.03x	1.06x	1.04x	1.05x	1.53x	1.07x		
Instr	30.8x	7.16x	26.4x	6.16x	57.1x	8.35x		

Case Study

NWChem is a DoE flagship computational chemistry application with 6 million lines of code. We run it with 8 MPI processes.

hpcviewe	hpcviewer: nwchem							
Calling Context View 🔀 🔧 Callers View 👫 Flat View								
Scope Experiment Aggregate Metrics	BYTES_USED:Sum (I) V BYTES_DEAD:Sum (I) MEM_UOPS_RETIRED:ALL_STORES:Sum (I)							
<pre>>cprogram root></pre>	1.14e+12 100.0 1.72e+12 100.0 2.41e+11 100.0							
▶ <partial call="" paths=""></partial>	4.08e+08 0.0% 1.28e+08 0.0% 7.50e+07 0.0%							

New Inefficiencies Reported by Witch

Арр	Problem	Speedup
povray	DeadStore	1.08X
Caffe-1.0	SilentStore	1.06X
Binutils-2.27	SilentLoad	10X
botsspar	SilentLoad	1.15X
imagick	SilentLoad	1.6X
Kallisto-0.43	SilentLoad	4.1X
lbm	SilentLoad	1.25X
SMB	SilentLoad	1.47X
vacation	SilentLoad	1.31X

Witch Supports Multithreading

- PMU and debug register are per-thread
- Signal delivery is per-thread
- Witch tools for multi-threaded cases false sharing
 - thread A populates memory addresses to a shared location
 - thread B grabs a memory address in the shared location to monitor its adjacent addresses

A lightweight false sharing detector PPoPP'18 best paper

Speedups after False-sharing Elimination

Benchmark		2-socket Haswell			16-socket Haswell							
		Num threads			Num threads							
		4	8	16	32	4	8	18	36	72	144	288
Synchro- bench	Fuzzy-KMeans	1.22	1.25	1.13	1.75	1.17	1.22	1.15	1.67	2.15	1.13	1.26
	SPIN-lazy-list	2.06	1.96	2.02	2.71	5.29	5.76	5.5	6.48	16.06	4.35	2.81
	SPIN-hashtable	1.19	1.35	1.41	1.77	1.33	1.44	1.45	2.47	1.26	2.49	1.99
	MUTEX-lazy-list	2.04	1.99	2.11	2.23	4.66	4.8	4.54	7.19	6.47	1.54	2.06
	MUTEX-hashtable	1.01	1.03	1.03	1.44	1.12	1.09	1.14	2.29	2.65	2.32	1.87
	lockfree-fraser-skiplist	1	1	1.18	1.05	1.05	1.06	1.1	1.43	1.56	1.79	1.65
	ESTM-specfriendly-tree	2.14	2.67	2.97	5.52	1.83	2.53	3.86	4.23	9.43	7.08	1.88
	ESTM-rbtree	1.01	1.19	1.25	1.03	1.08	1.23	1.32	1.19	1.25	1.73	1.27
Discrete event simulator	Libdes	3.97	5.37	8.45	1.39	4.27	6.51	9.25	4.81	10.4	8.58	7.19
Formal verification	Spin6.4.4	1.23	1.21	1.28	2	1.38	1.35	1.23	2.21	2.31	3.93	NA

On-going Work

- Lightweight reuse distance measurement
 - plot reuse histogram with >90% accuracy for program characterization
 - provide call paths for use and reuse to guide code optimization
- Lightweight inefficiency detection in Java programs
 - PMU + debug regster + JVMTI
- Lightweight inefficiency detection in Linux kernel

Conclusions

- Potential to pinpoint software inefficiencies in production codes
 - redundant computation
 - redundant memory accesses
 - useless operations
 - ...
- Potential to deeper program analysis
 - access pattern analysis
 - inter-thread analysis (e.g., contention, false sharing)
- Witch is a unique framework to pinpoint software inefficiencies
 - lightweight measurement
 - extensible interfaces to other client tools
 - available at <u>https://github.com/WitchTools/</u>

