VELOCITY Low-Level Pointer Analysis

Practical and Accurate Pointer Analysis of Low-Level Code

Do you believe low-level pointer analysis cannot be as accurate as high-level pointer analysis?

Bolei Guo Matthew J. Bridges Spyridon Triantafyllis Guilherme Ottoni
Easwaran Raman David I. August
{bguo,mbridges,strianta,ottoni,eraman,august}@princeton.edu

Accurate pointer analysis is essential for extracting Instruction-Level Parallelism (ILP) in the presence of memory operations. Traditionally, pointer analysis is performed once on a high-level intermediate representation (IR). The analysis result is then annotated on the IR and must be conservatively maintained by subsequent code transformations. This imposes a phase-ordering constraint, causing the precision of alias information to be diluted by code transformations. Furthermore, unlike high-level pointer analysis, low-level pointer analysis can also be used at link time or run time, where the source code is unavailable.

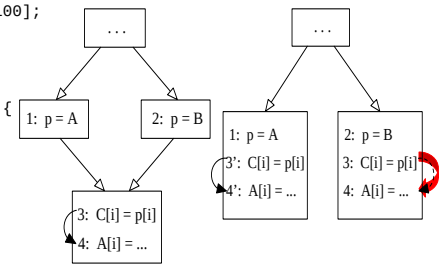
What is the VELOCITY Low-Level Pointer Analysis? To overcome the limitations of high-level pointer analysis, VELOCITY Low-Level Pointer Analysis (VLLPA) operates at the assembly level. VLLPA is context sensitive and partially flow-sensitive (tracks registers according to their position in the CFG). It handles all the features of C, such as unions. It is also modular in that only part of the program and the analysis information needs to be present in memory simultaneously, reducing the memory requirement of context-sensitive interprocedural analysis

How accurate is VLLPA compared to high-level analysis? The IMPACT compiler's powerful high-level pointer analysis is context-sensitive and flow-insensitive. A comparison with IMPACT shows the number of dependence arcs between memory operations computed from the pointer analysis results. On average, VLLPA is less precise on only 0.3% of memory operations and more precise on 26.8% of memory operations.

How bad is the problem of conservative memory dependence propagation? Dependence arcs created by the high-level analysis must be conservatively propagated by subsequent code transformations, resulting in many spurious dependences. VLLPA was ran on the low-level code produced by IMPACT at the final stage of compilation to produce accurate dependence arcs. The dependence arcs propagated from IMPACT's high-level pointer analysis to the final stage were then classified in three categories:
1) Correct: Arcs identified by VLLPA.
2) Unnecessary: Spurious arcs resulting from the propagation of high-level arcs that were themselves spurious.
3) Unnecessarily Propagated: Spurious arcs resulting from the propagation of accurate high-level arcs.
The number in the third category accounts for up to 50% of all IMPACT arcs present at code generation. This shows that repeating memory analysis at the low level is significantly more accurate than propagating memory analysis information from the high level.

More information:
http://www.liberty-research.org/Research/VLLPA or contact the Liberty Low-Level Pointer Analysis Team at the addresses above. Bolei Guo, Matthew J. Bridges, Spyridon Triantafyllis, Guilerme Ottoni, Easwaran Raman, David I. August, "Practical and Accurate Low-Level Pointer Analysis", *Proceedings of the Third International Symposium on Code Generation and Optimization*, March 2005.

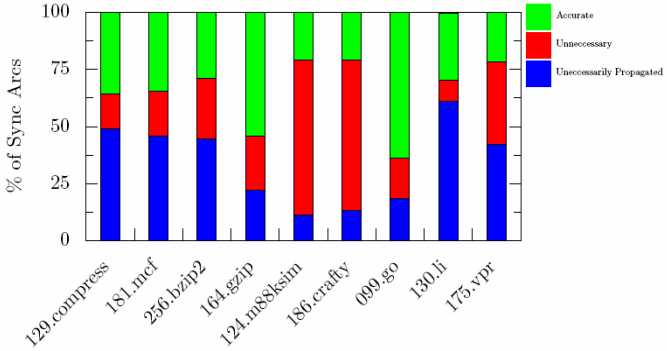
```
Char A[100], B[100], C[100];
Foo() {
  int i;
  char *p;
  ...
  for (i=0; i<100; i++) {
    if (i%15 == 0)
1:    p = A;
    else
2:    p = B;
3:  C[i] = p[i];
4:  A[i] = ...;
  }
}
```



Conservative Dependence Propagation Example: After superblock formation is performed on the source code, instructions 2, 3 and 4 are grouped into a superblock (instructions 3' and 4' are created for tail duplication). There is no longer any dependence between 3 and 4, but the superbock formation algorithm conservatively propagates the dependence relation to the transformed code.

Benchmarks	Time (s)	# Oper w Sync Arcs	Fewer	More
129.compress	0.268	329	0	51
181.mcf	0.285	705	7	199
256.bzip2	0.485	1535	3	256
164.gzip	0.764	1953	3	848
124.m88ksim	4.584	7161	4	2722
175.vpr	1.328	8166	97	1402
186.crafty	1.543	12026	17	4759
099.go	2.087	13232	0	2393
130.li	14.843	376	33	724

Accuracy Comparison with IMPACT: The third and fourth columns show the number of memory operations for which VLLPA results in fewer and more dependence arcs. Fewer arcs imply high precision.



Breakdown of Propagated High-Level Dependence Arcs: See text.