OO Optimizing Compilers

- **Vortex - Craig Chamber's research compiler for Cecil, C++, Java, Modula-3 (~1995-2000)**
- **Jikes RVM - IBM research compiler for adaptive compilation of Java, based on Jalapeno VM (~1999-2004)**

Vortex Compiler

- **Early experimental OO compiler built by Craig Chambers and his students for his Cecil PL**
  - **Goal: to reduce runtime performance costs of OO style through static analysis based and dynamic profiling guided techniques**
  - **Compiles multiple front-ends into C or SPARC code**
- **Did deep exploration of how to gather effective profiling information (off-line)**
  - **Profile-guided optimizations**
    - **Inlining; Receiver class prediction; Method specialization**
- **Granularity of data collection**
    - **How much calling context to save?**
    - **How many method calls to fold together?**
    - **Balance efficiency of data gathering (cost) versus utility of data gathered (profitability)**

Off-line Profiling

- **Vortex gathers profiling information off-line, in separate training runs of program and uses it in optimization**
- ***Alternative:* SELF and JikesRVM do dynamic compilation, gathering profiling information as the program runs and recompiling using this information for *hot methods***
  - **Problem: enabled optimizations over subsequent executions have to save enough to cover the cost of profiling**

Grouping Profiles

- **Each collected profile distribution is associated with set of method calls**
  - ***Message summary* - groups all calls to same method**
  - ***Call-site-specific* (1-CCP)**
    - ***k-Call Chain Profile* delimits k dynamically enclosing call sites**
  - **Collect histogram for each possible receiver class**
  - **Shows if a few classes dominate (i.e., peaked profiles)**
  - **Shows which classes are most common**
  - **Prefer peaked profiles to find some methods are more frequently called than others**
  - **Want profiles stable across inputs and program versions**

Empirical Findings

- **Vortex trials on C++ and Cecil programs**
  - **71% of C++ messages (72% Cecil) were sent to most common receiver class**
  - **in C++, 36%(Cecil 50%) dynamic dispatches occurred at call sites with *single receiver class*!**
    - **Q: would that be true today?**

Profile Stability over Inputs

- **Two metrics studied on profiles derived from different inputs to same programs**
  - ***FirstSame:* same most common receiver class**
  - ***OrderSame:* 2 distributions are same only if they are comprised of same classes in same frequency order**
  - **in C++,**
    - **for FirstSame, 99% match for method summary and 79% match for 1-call-site-specific**
    - **for OrderSame, 28% match for method summary and 45% match for 1-call-site-specific**

Stability over Program Versions

- Gathered profiles across different versions of Vortex using 6 month version control history after compiler was stable
  - *FirstSame* metric found distributions stable
    - Fewer than 5% method summaries changed over entire 6 month period; not until after 2 months, did more than 10% of call-site-specific profiles change
- Claim: this validates utility of profiles for optimization of future versions of a program

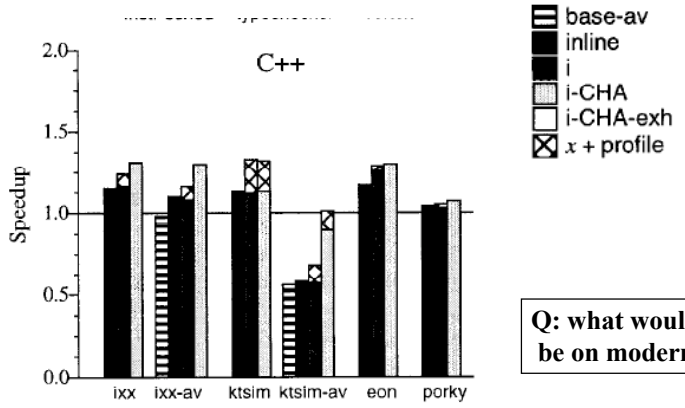
Vortex Findings

Dean et.al, OOPSLA'96

- Benchmarks included C++, Modula-3, Java programs
  - Plus versions of the C++ programs with all methods marked *virtual*
- Optimizations tested
  - Use g++ with -O2
  - Compiled w/o OO optimizations
  - Inlining. Splitting, exhaustive class testing based on static info
  - Inlining, splitting, profile-guided receiver class prediction

Vortex Findings

Dean et.al, OOPSLA'96



Q: what would be the results be on modern C++ programs?

Dynamic Compilation

- **Dynamic compilation in Smalltalk compiler in mid-1980's**
  - **Compiled bytecodes to machine code on demand**
  - **Execution times halved compared to interpreted execution**
- **Idea used in SELF compiler in early 1990's**
  - **Focused on optimization of dynamic dispatch**
- **"In interactive dynamic compilation system, optimizing less may improve overall efficiency, if doing so reduces compile time more than it increases execution time" (Holzle/Ungar Toplas1996)**

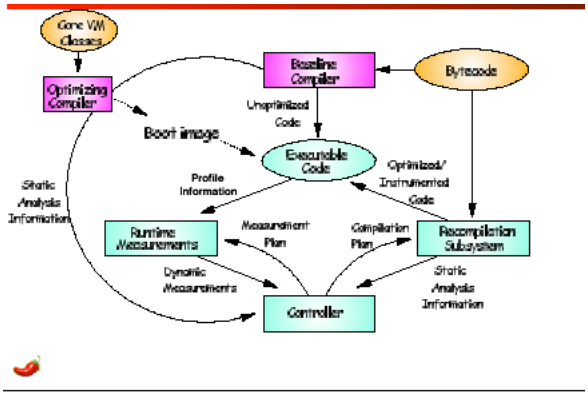
Dynamic Compilation

- **Idea: only optimize performance-critical sections of code**
  - **Reduces compilation delays and time cost**
- **Adaptive optimization: discover and optimize *hot spots* in the code**
  - **Idea: use online profile info to optimize methods**
- **IBM's Jikes RVM for Java explored many of the ideas initially presented in SELF compilers and is refining the methodology of adaptive optimization**

Jikes RVM

- **Research compiler developed by IBM that is now open source (Oct 2001)**
- **Key properties:**
  - **Does no interpretation; only compiles to native code**
  - **Written almost completely in Java including the run-time system**
  - **Goal to explore *adaptive compiling* using on-line profiling**
  - **Not a full JVM because doesn't have all libraries**
  - **Designed to run Java programs investigating VM issues**
  - **Runs on AIX/PowerPC, Linux/IA-32, Unix/ PowerPC (limited functionality)**

Jikes RVM - Structure

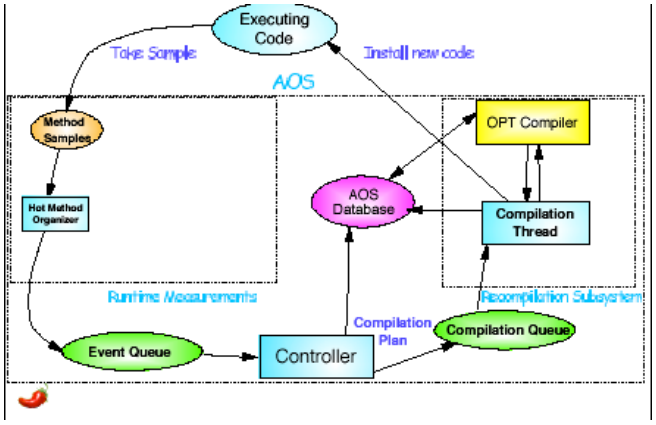


selected slides from JikesRVM tutorial at OOPSLA'02, by Mike Hind and David Grove

Jikes RVM - Optimizations

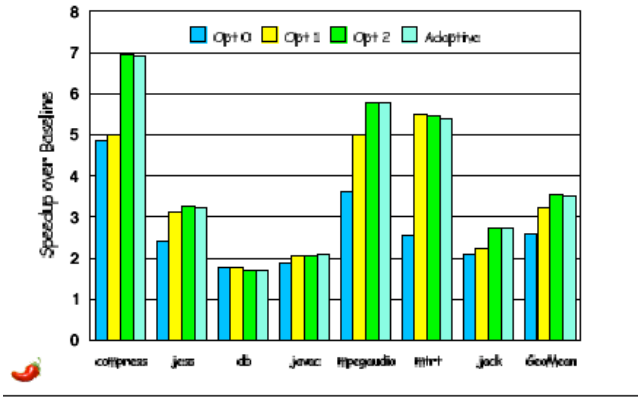
- **Optimization levels**
  - **level 0- branch opts, constant propagation, dead code elimination, register allocation, instruction scheduling, etc.**
  - **level 1- pre-existence and speculative inlining, static splitting, cse, load elimination, flow-insensitive const/copy/type propagation, etc.**
  - **level 2- loop normalization and unrolling, scalar SSA, dataflow analysis, global CSE, etc.**
- **Speculative (guarded) inlining using CHA or profiling info**

Adaptive Compilation System

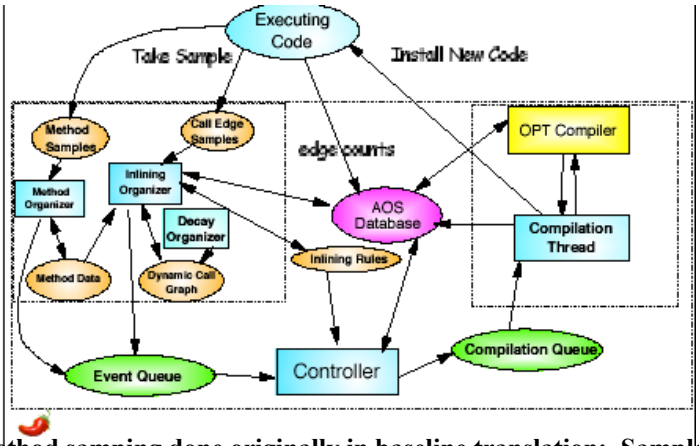


Idea: online system exploits observed properties of current run

Adaptive - best of 20 runs Specs (size 100)



Adaptive System with FDO



Method samping done originally in baseline translation; Sample results are decayed; Rules determine which methods to inline (based on edge frequencies and code growth size heuristics)

FDO - best of 20 runs (size 100)