



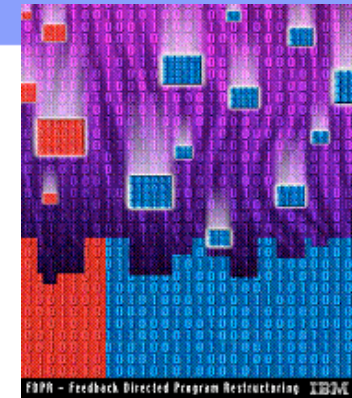
IBM Haifa interest items

Bilha Mendelson
Code Optimization and Quality Technologies

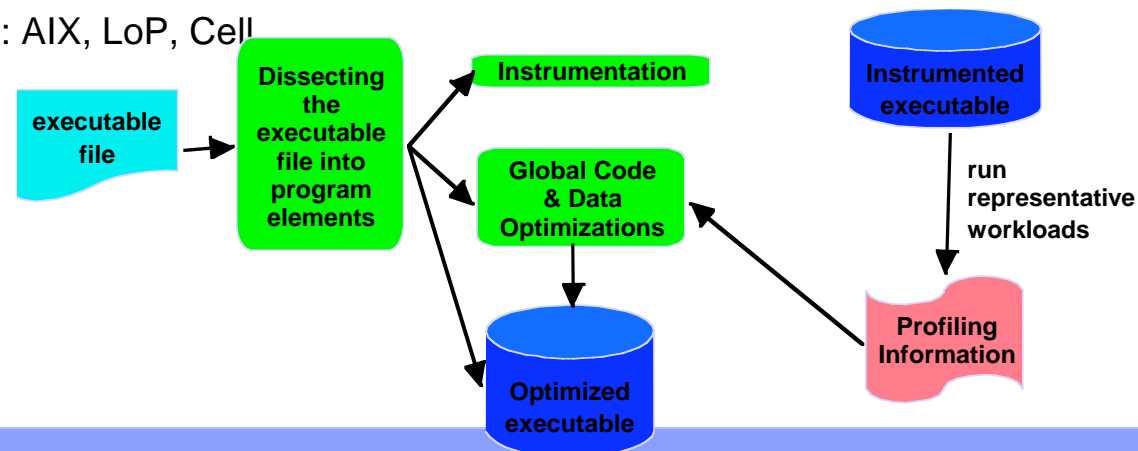
bilha@il.ibm.com



Post-link optimizer - FDPR-Pro



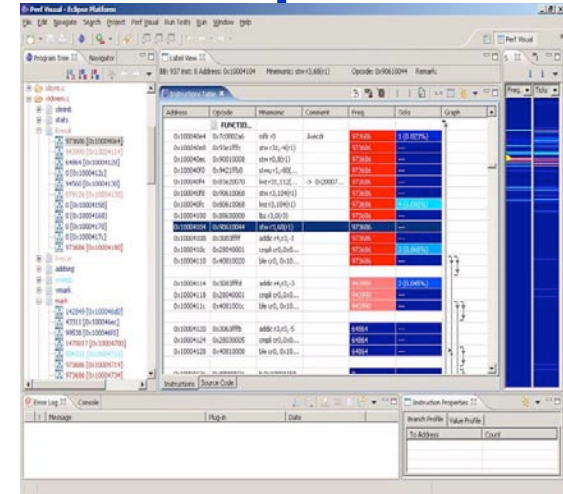
- ◆ Feedback-directed post-link optimization tool that operates directly on binary executables
 - ◆ Operates in three phases:
 - ◆ Phase 1: **Code instrumentation**
 - ◆ Basic block level
 - ◆ Phase 2: **Profile information gathering**
 - ◆ Selection of "right" input set (workload)
 - ◆ Accumulation over several input sets
 - ◆ Phase 3: **Global Code & Data Optimizations**
- ◆ Profile based; Whole program view
- ◆ The optimizations include
 - ◆ Code Reordering, Static Data Reordering, Branch Folding, Branch Prediction, Inlining, Constant and Copy Propagation, Dead Code Elimination, etc.
- ◆ Available on different platforms: AIX, LoP, Cell





Additional performance tools based on post-link optimizer

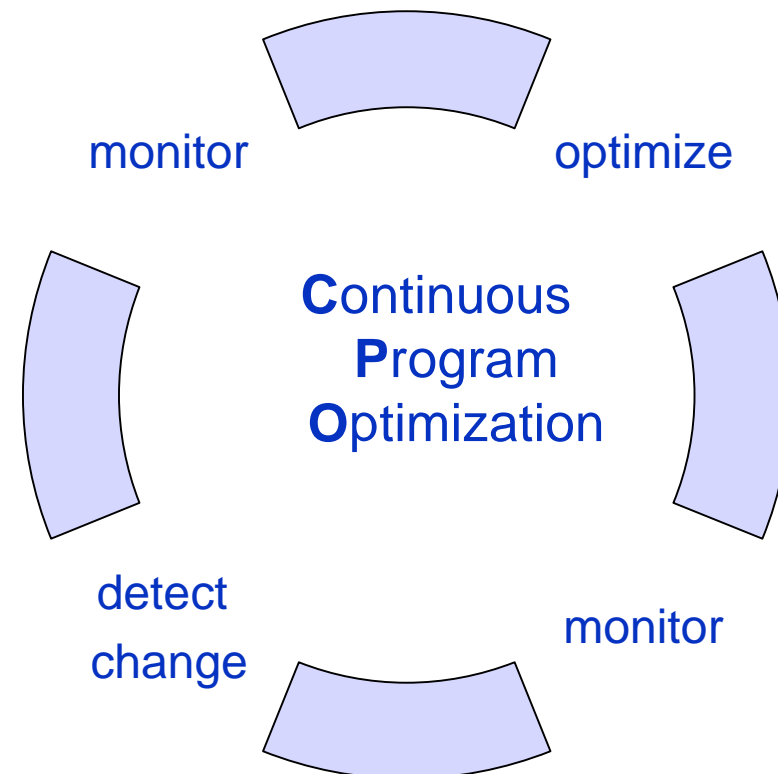
- ◆ **Code Analyzer** – eclipse plugin GUI that can display performance analysis and bottlenecks
 - ◆ Releasing to alphaWorks as part of Visual Performance Analyzer (VPA)
 - ◆ <http://www.haifa.ibm.com/projects/systems/cot/analyzer/index.html>
- ◆ **BProber**– utility for supporting program instrumentation on binary executable files
 - ◆ Static-mode version that is run on given executables
 - ◆ Started working on dynamic-mode version that can instrument and replace code at run-time
 - ◆ <http://www.haifa.ibm.com/projects/systems/cot/bprober/index.html>
- ◆ Adaptive Binary Optimization system





Adaptive optimization - motivation

- ◇ Feedback information has proven useful in guiding optimizations
- ◇ Program behavior may change due to changes in the ways the program is used
 - ◇ need to monitor the optimized program and identify significant changes
 - ◇ need to re-adapt the program
 - ◇ can be done
 - ◇ online - for long running applications
 - ◇ offline - for short applications
 - ◇ this is a continuous process





Adaptive Binary Optimization

- ◇ Adaptive performance optimization of long running applications
 - ◇ Addresses dynamic optimization
 - ◇ binary code of statically compiled applications
 - ◇ Does not rely on the application's source code or recompilation
 - ◇ Aims at adapting the application to:
 - ◇ significant changes in application behavior
 - ◇ run-time events
- ◇ Some of the challenges
 - ◇ Low overhead online event-monitoring
 - ◇ Efficient redirection of code execution to optimized code (using ptrace)
 - ◇ Online optimization validation and optimization adaptation
- ◇ Status
 - ◇ Work on change detection published:
 - ◇ "Detecting change in program behavior for adaptive optimization" by N. Peleg and B. Mendelson, PACT 2007
 - ◇ Initial prototype
 - ◇ Start developing infrastructure for dynamic binary code analysis and optimization
 - ◇ Implementing event-driven dynamic optimization



Additional interest

- ◇ Monitoring:
 - ◇ Need to move to higher levels => system view
 - ◇ Monitoring Java applications: partial success for Java applications.
 - ◇ What about other environment: Python, PHP....?
 - ◇ Run time feedback and adaptation
- ◇ Split Compilation:
 - ◇ compiler statically to continue compilation dynamically
 - ◇ GCC + CLI (Loop transformation for vectorization to be perform at later stages))
 - ◇ GCC + LLVM ((Low Level Virtual Machine)
 - ◇ GCC + LTO
 - ◇ GCC + VM (JVM?)
 - ◇ Identify coarse level parallelism and provide runtime support
- ◇ Debugging of dynamic optimization
- ◇ Not just performance: energy and power adjustments at runtime



IBM

Thanks!