IBM Research

Continuous Program Optimization

IBM TJ Watson Research Center
IBM SWG Toronto Laboratory

Outline

- **Description**
- **Motivation**
- **Architecture**
- **Examples**
- **Conclusion**

Motivation

- **Limitations in static compilation due to increased complexity in system architecture & software stack**
  - Environment too diverse, dynamic for one-size-fits-all binaries
- **Multiple levels of parallelism and heterogeneity are currently handled poorly both in high-level languages and compilers**
  - Too many factors to consider
- **Dynamic compilation is constrained by resource utilization at runtime**

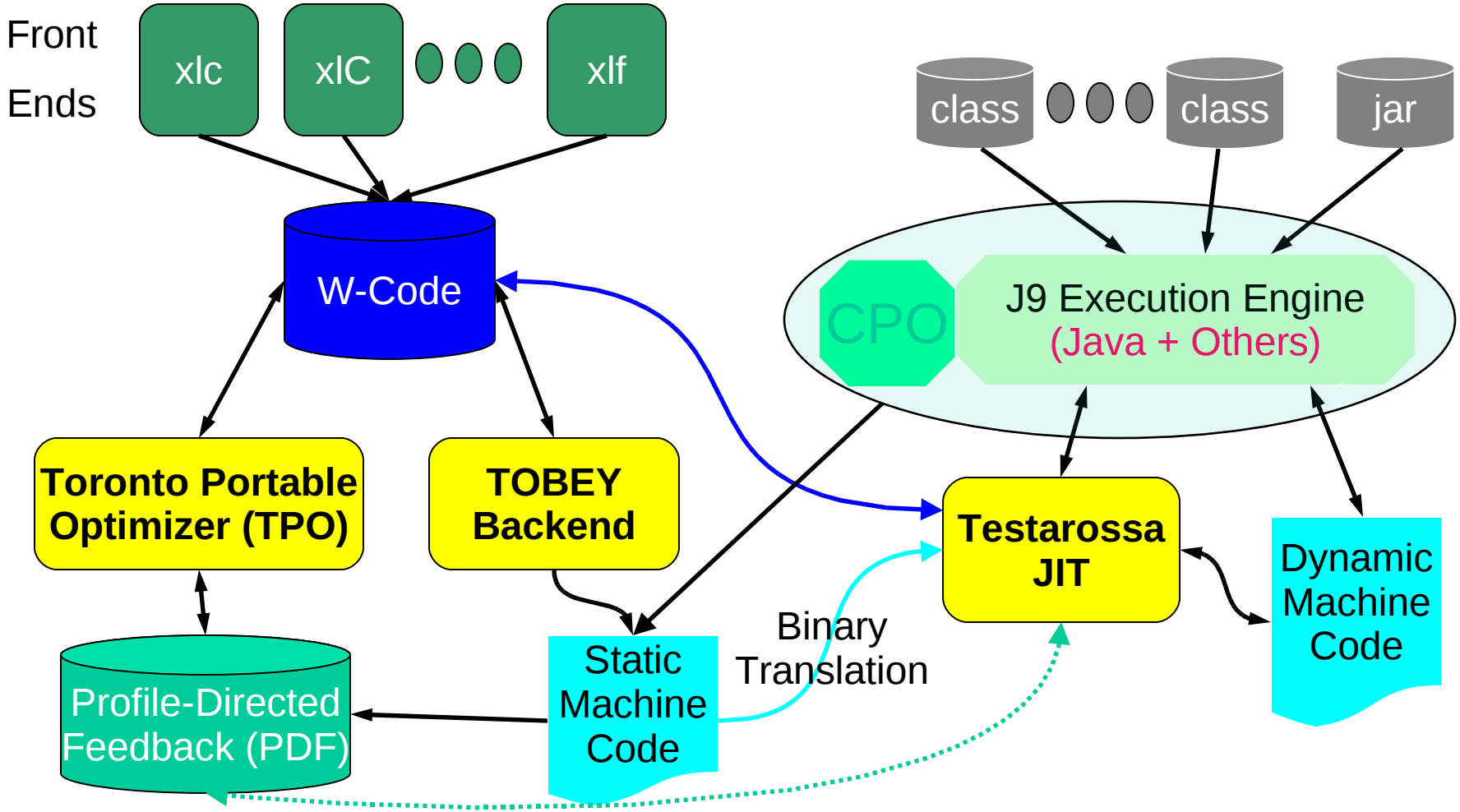
What is CPO?

- **CPO =**
  - **Offline static analyses**
  - **Always-on feedback**
  - **Dynamic recompilation**
  - **Code lifecycle management**
- **Merge dynamic and static compilation**
  - "fat binaries" contain enough information to allow recompilation while running
- **Leverage existing IBM compiler technology**
  - IBM XL compiler suite
  - J9/Testarossa JIT compiler framework

CPO Components In A Nutshell

- **Static front-ends**
  - XL: Fortran, C, C++, OMP, UPC, CAF; J9 (Java); X10
- **Static optimizer**
  - TPO: conventional, SIMD vectorization; added OMP loop, UPC forall, privatization
- **Program Representation**
  - Wcode, Java bytecodes, TRIL (Testarossa IL)
  - Extended for PGAS languages, TM constructs
- **Static back-end**
  - XL/Tobey
- **Dynamic compiler**
  - Testarossa
- **Perf. + Environ. Monitoring**
- **Managed runtime**

CPO component view



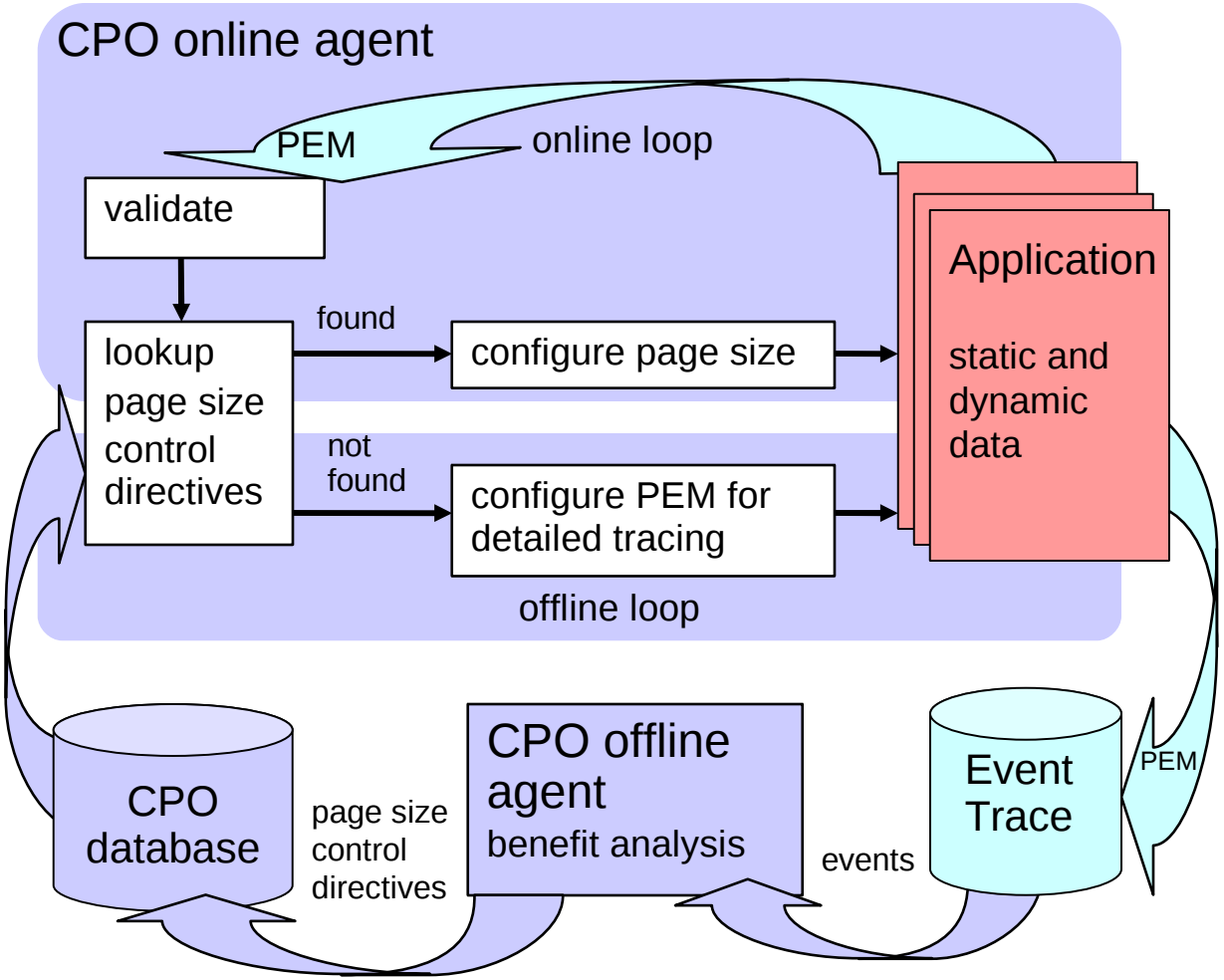
Managed Runtime: break through SW layers

- **Monitor, change runtime environment**
  - Transparent profile-directed feedback
  - Ask for, manage resources
- **Recompilation Control**
  - “Recompilation needed” decision; risks and rewards
  - Optimization plan generation for dynamic compiler
- **Code lifecycle issues**
  - Expected lifetime of generated “fat binaries”
  - Precondition generator/code selection overhead
  - When is it safe to patch?

Expected gains (Examples)

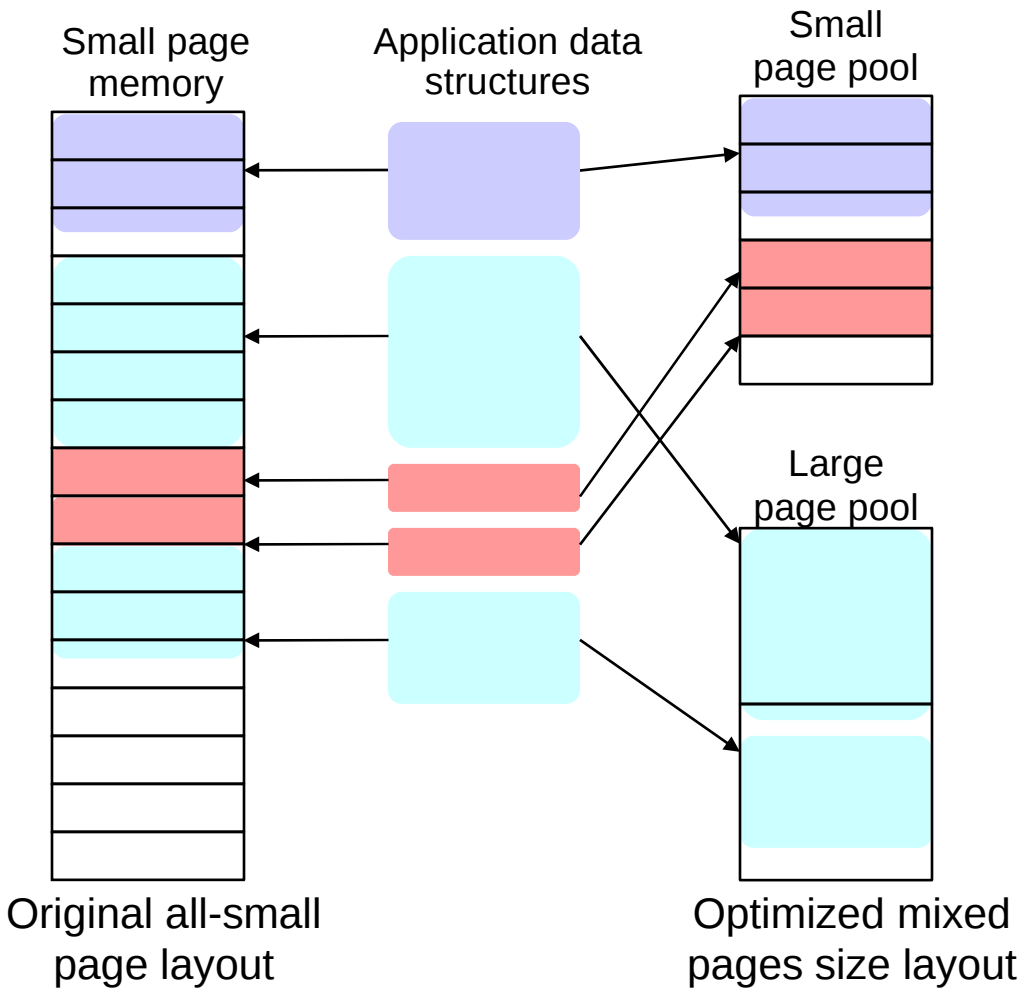
- **Adaptation to input parameters**
- **Adaptation to changing system properties**
  - CPUs going offline or becoming busy; system tasks interfering with processing
- **Change page sizes**
- **Reschedule loops based on # available CPUs**
- **Identify delinquent loads, insert prefetch instructions**

CPO page size agent



- Fully automated
- Offline CPO agent: select profitable data structures for large pages
- Online CPO agent: map selected data structures to large pages

Large page benefit analysis – Problem definition



- **Partition data structures into categories, e.g.:**
  - Large dynamic data structures
  - Small dynamic data structures
  - Static data
- **Explore the most promising mappings of categories to page sizes**
- **Calculate a cost/benefit ranking:**
  - Number of large pages needed
  - Reduction in page faults and TLB miss cycles
- **Select mapping with highest benefits for available large pages**

Multiple Page Size Agent Summary

- **An agent that collects runtime memory behavior, from the entire execution stack (HW, OS and application)**
- **Models page allocation and determines the most profitable data structure mapping**
- **Removes the need for the user to guess large page mappings and set environment variables**
- **Can be used as a standalone tool for developers to tune their applications**
- **Matches best performance while minimizing the number of large pages used (finite resource!)**
- **PACT 2005**

CPO MPI Agent

- **Collect communication information and analyzes the communication patterns**
- **Makes recommendations for mapping MPI tasks to the system topology such that the communication costs are minimized**
- **These directives can be implemented**
  - Offline – by remapping between executions
  - Online – provide input to graph partitioning libraries such as Metis or Zoltan; pick implementation choice for MPI functions; etc.

CPO Delinquent Load Agent

- **Identify delinquent load addresses through hardware counters**
- **Locate instructions in code that write to those addresses**
- **Recompile code (place prefetch instructions into code)**
- **Monitor performance improvement**

Conclusions

- **CPO pervades all layers of application + system software**
- **Monitor performance & environment, react**
  - Adjust environment when possible
  - Recompile code as needed
- **“fat binaries” carry enough information for recompilation**