IBM Software Group

IBM

e business software

An (incomplete) Survey of Compiler Technology at the IBM Toronto Laboratory

Bob Blainey

March 26, 2002

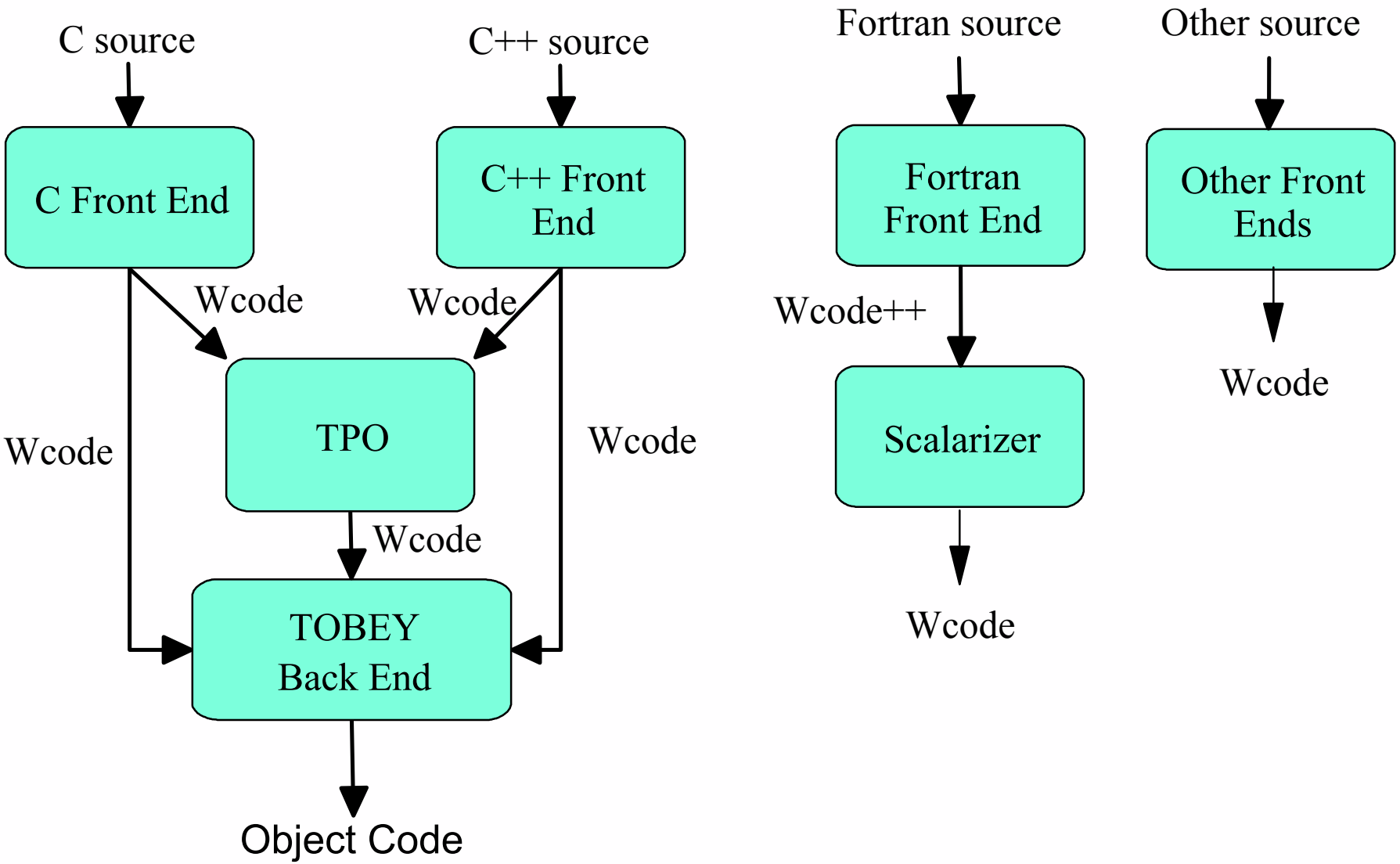
Target systems

- Sovereign (Sun JDK-based) Just-in-Time (JIT) Compiler
  - zSeries (S/390)
    - OS/390, Linux
    - Resettable, shareable
  - pSeries (PowerPC)
    - AIX 32-bit and 64-bit
    - Linux
  - xSeries (x86 or IA-32)
    - Windows, OS/2, Linux, 4690 (POS)
    - IA-64 (Itanium, McKinley) Windows, Linux
- C and C++ Compilers
  - zSeries OS/390
  - pSeries AIX
- Fortran Compiler
  - pSeries AIX

Key Optimizing Compiler Components

- TOBEY (Toronto Optimizing Back End with Yorktown)
  - Highly optimizing code generator for S/390 and PowerPC targets
- TPO (Toronto Portable Optimizer)
  - Mostly machine-independent optimizer for *Wcode* intermediate language
  - Interprocedural analysis, loop transformations, parallelization
- Sun JDK-based JIT (Sovereign)
  - Best of breed JIT compiler for client and server applications
  - Based very loosely on Sun JDK

Inside a Batch Compilation

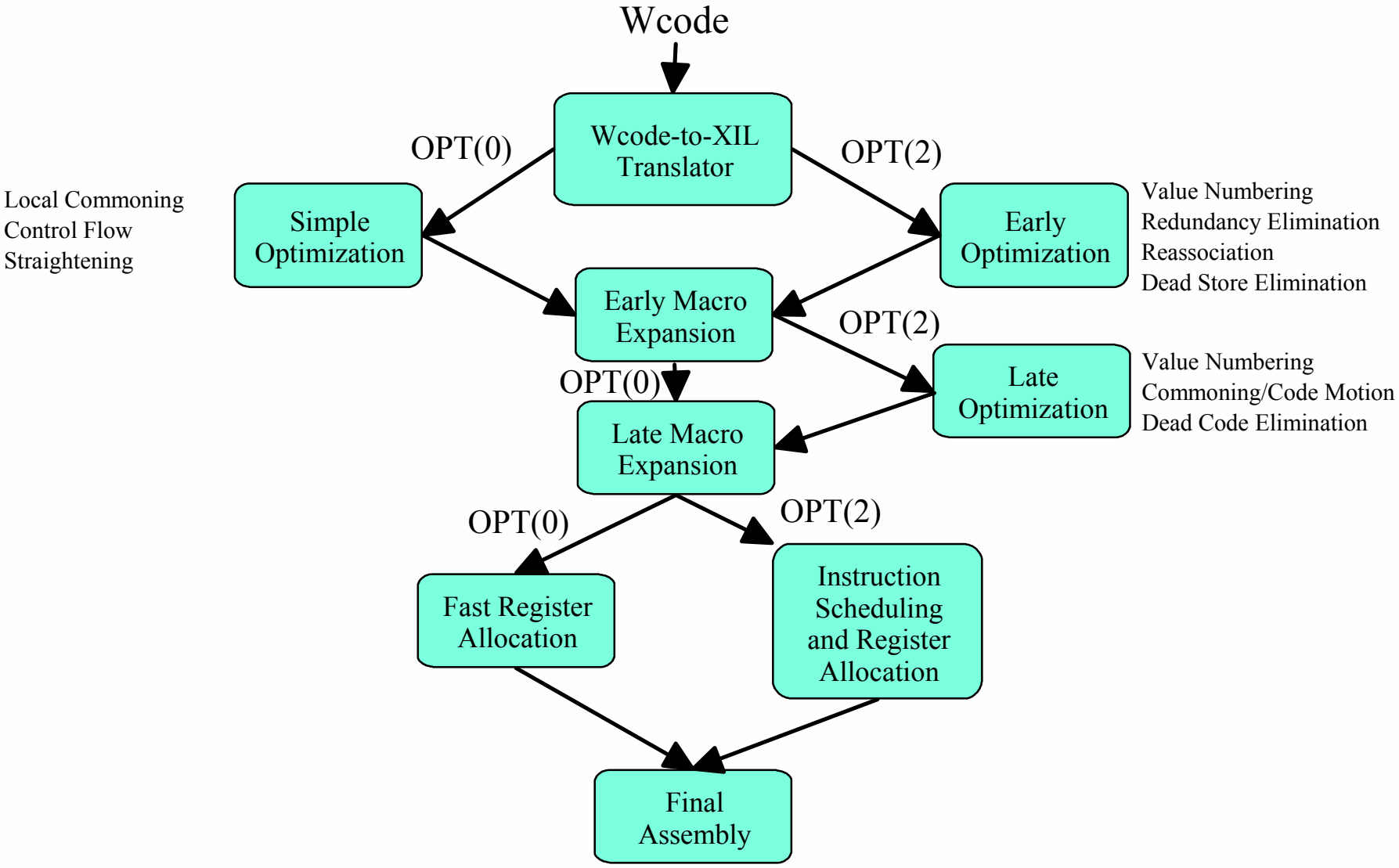


IBM

TOBEY Optimizing Back End

- Project started in 1983 targetting S/370
- Later retargetted to ROMP (PC-RT), Power, Power2, PowerPC, SPARC, and ESAME/390 (64 bit)
- Experimental retargets to i386 and PA-RISC
- Shipped in over 40 compiler products on 3 different platforms with 8 different source languages
- Primary vehicle for compiler optimization since the creation of the RS/6000 (pSeries)
- Implemented in a combination of PL.8 ("80% of PL/I") and C++ on an AIX reference platform

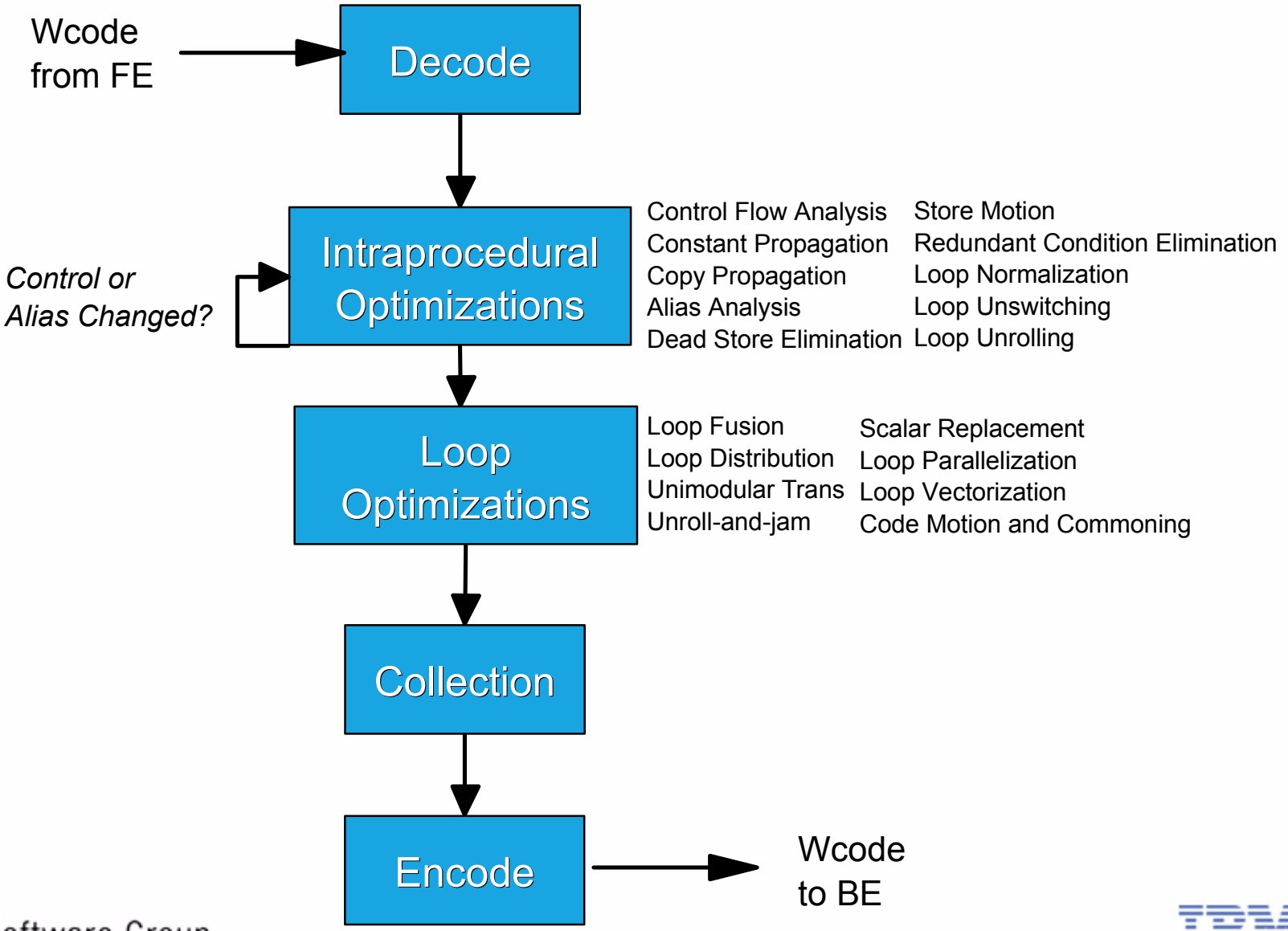
Inside TOBEY



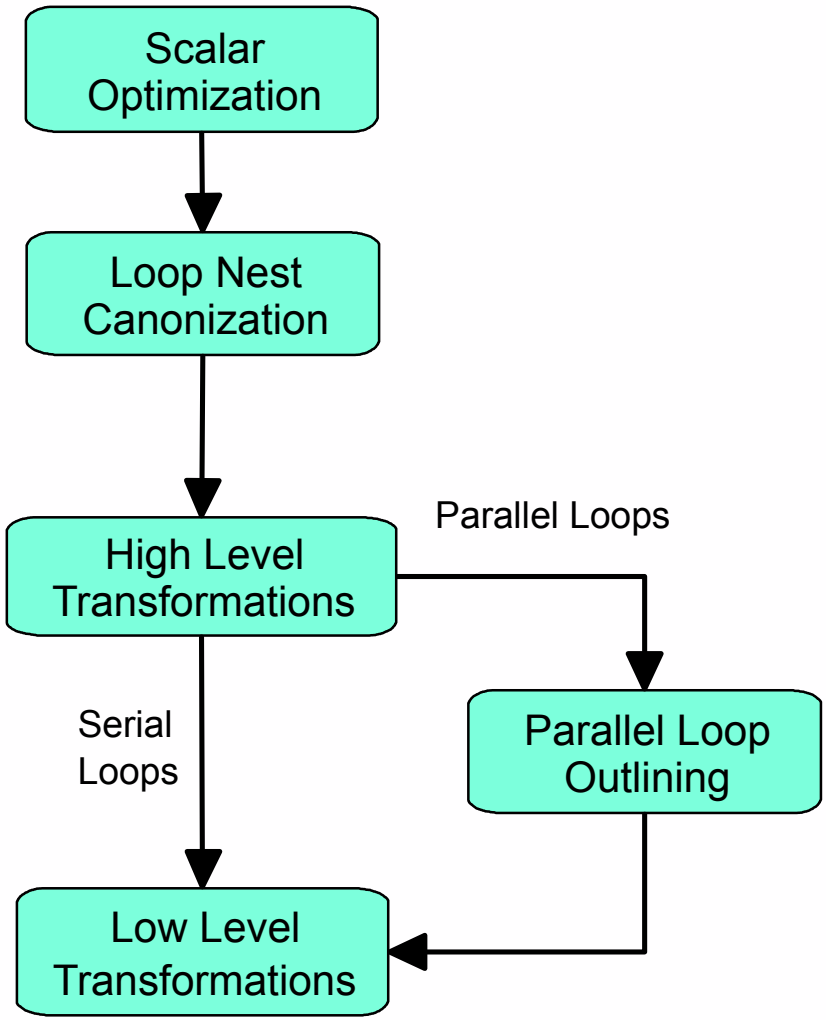
TPO (Toronto Portable Optimizer)

- Project started in 1994 as an interprocedural optimizer for RS/6000
- Shipped first as an interprocedural optimizer for the OS/390 C compiler in 1996
- Later shipped as part of C, C++ and Fortran compilers on AIX, the C++ compiler on OS/390 and as a linker enhancement on OS/400
- Key optimization driver for the ASCI Blue and White projects and PowerPC SPEC benchmark performance
- Provides OpenMP explicit parallel support and automatic loop parallelization on RS/6000
- Being adapted to optimize large scale commercial software such as DB2,Oracle and SAP
- Implemented in C++ on an AIX reference platform

Inside TPO Compile Time Optimization



Loop Optimization



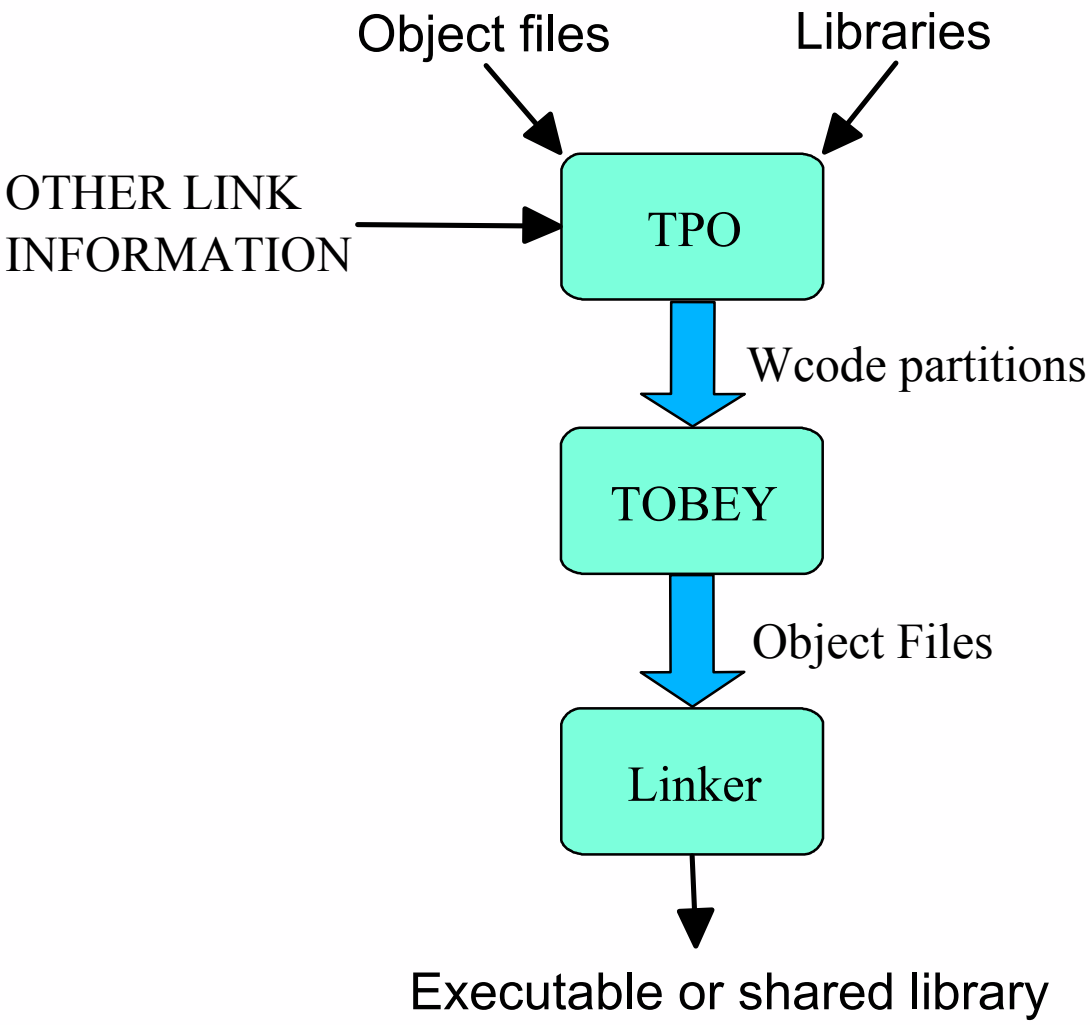
Control Flow Optimization
Data Flow Optimization
Loop Normalization

Aggressive Copy Propagation
Maximal Loop Fusion

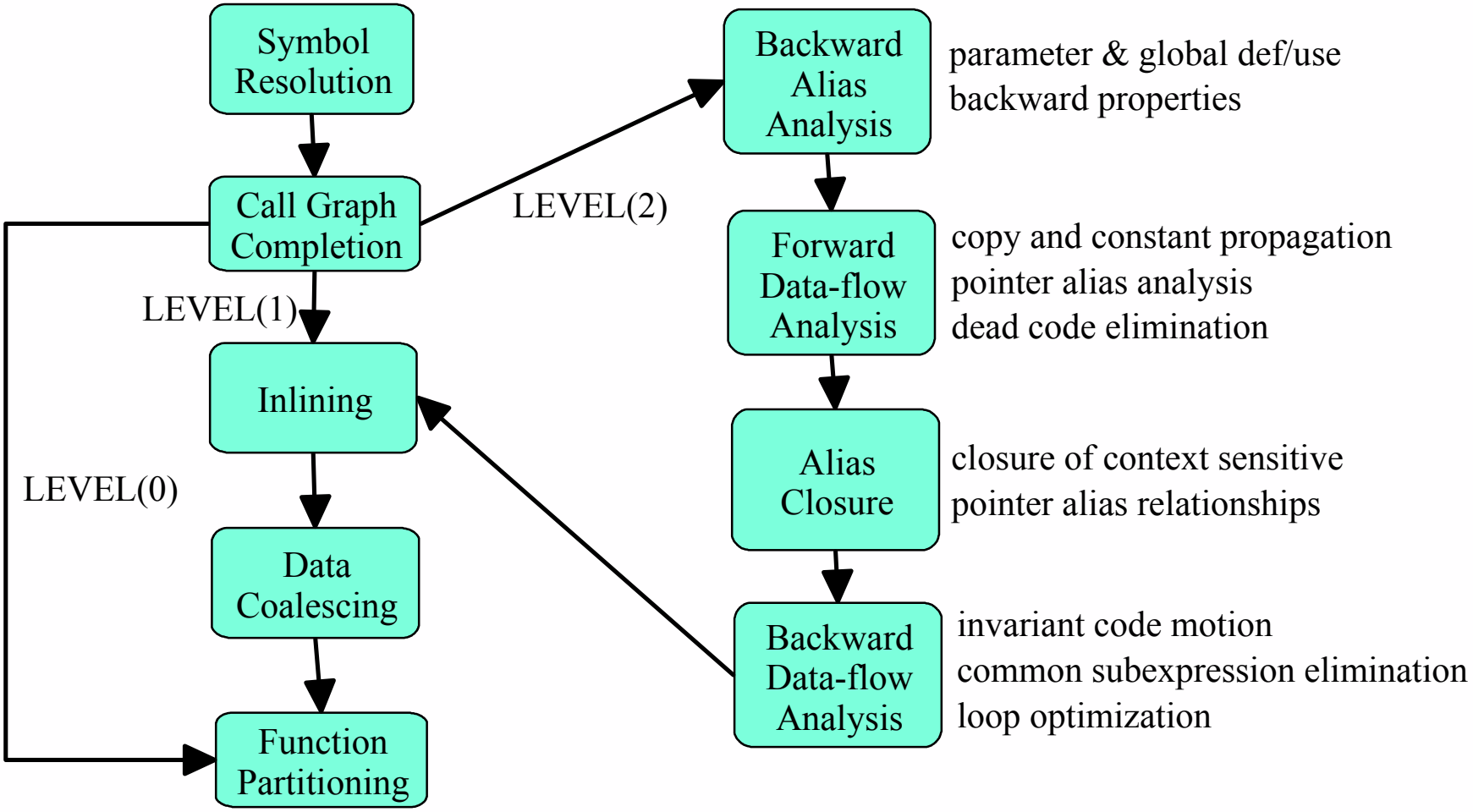
Loop Nest Partitioning
Loop Interchange
Loop Unroll and Jam
Loop Parallelization

Inner Loop Unrolling
Loop Vectorization
Strength Reduction
Redundancy Elimination
Code Motion

Inside an Link-time Compilation



Inside TPO Link Time Optimization



Selected TPO Optimizations

- Interprocedural constant propagation, pointer alias analysis and dead code elimination
- Partially invariant code motion
- Forward and backward store motion
- Partial constant propagation
- Redundant condition elimination
- Code and data partitioning
- Loop partitioning

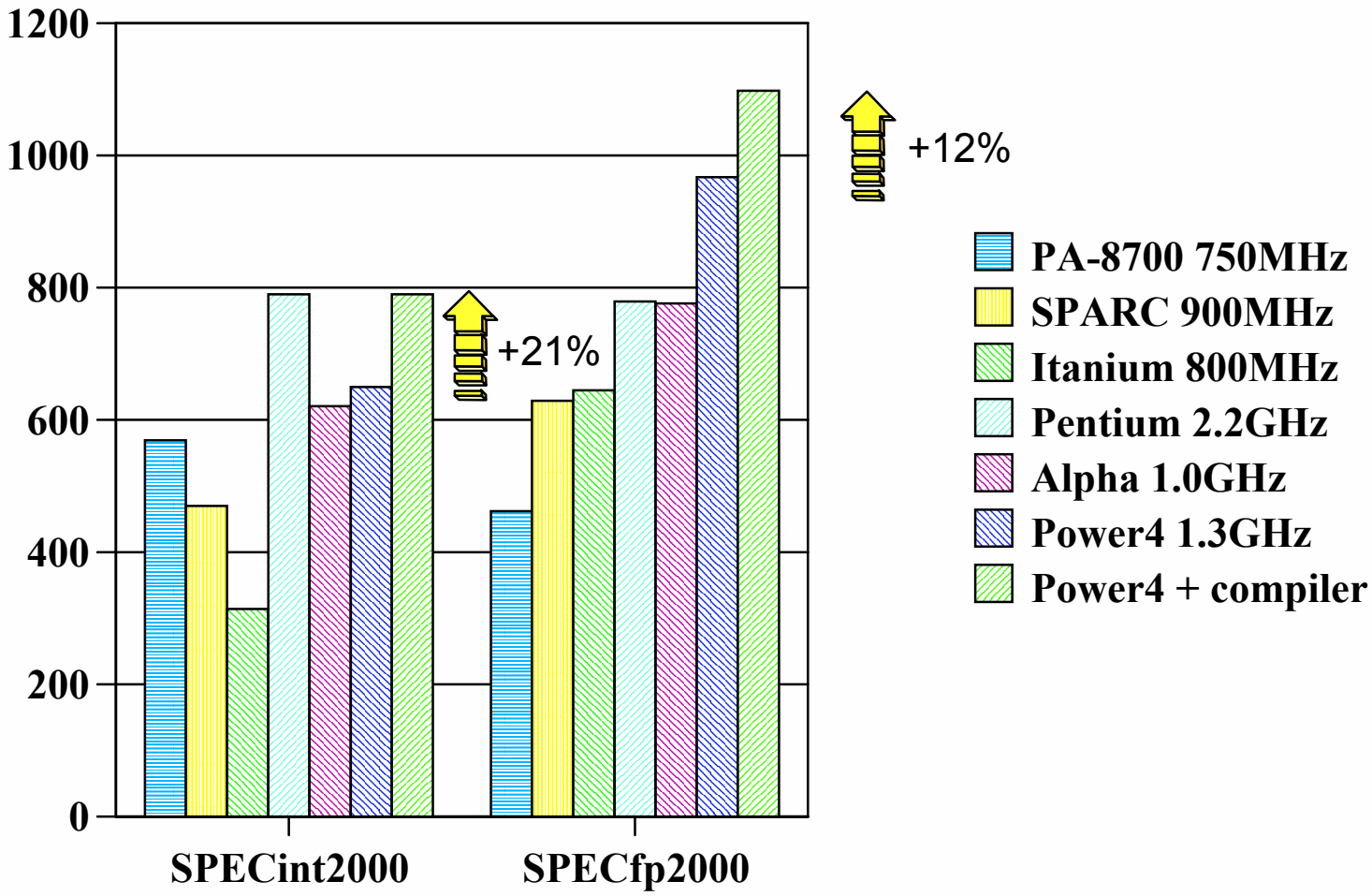
Some Compiler Changes for Power4

- Instruction scheduling for dispatch
- Register-conctrained modulo scheduling
- Avoid microcoded and some cracked instructions
- Generate stream touch instructions
- Eliminate small branch sequences using CA bit
- Tune loop optimization for 8 prefetch buffers
- Procedure and loop code alignment
- Use static branch prediction override with PDF
- Inline pointer glue and set BH for virtual and pointer calls
- Bias CR allocation to get same source/target for CR logic

Platform Neutral Improvements

- Profile directed interprocedural optimization
- Profiling and specialization of function pointer calls
- F90 MATMUL/TRANSPOSE improvements
- Interprocedural loop optimization
- Profile directed outlining

Results: Regatta vs. Competition



* Note: Power4 measurements NOT official

IBM

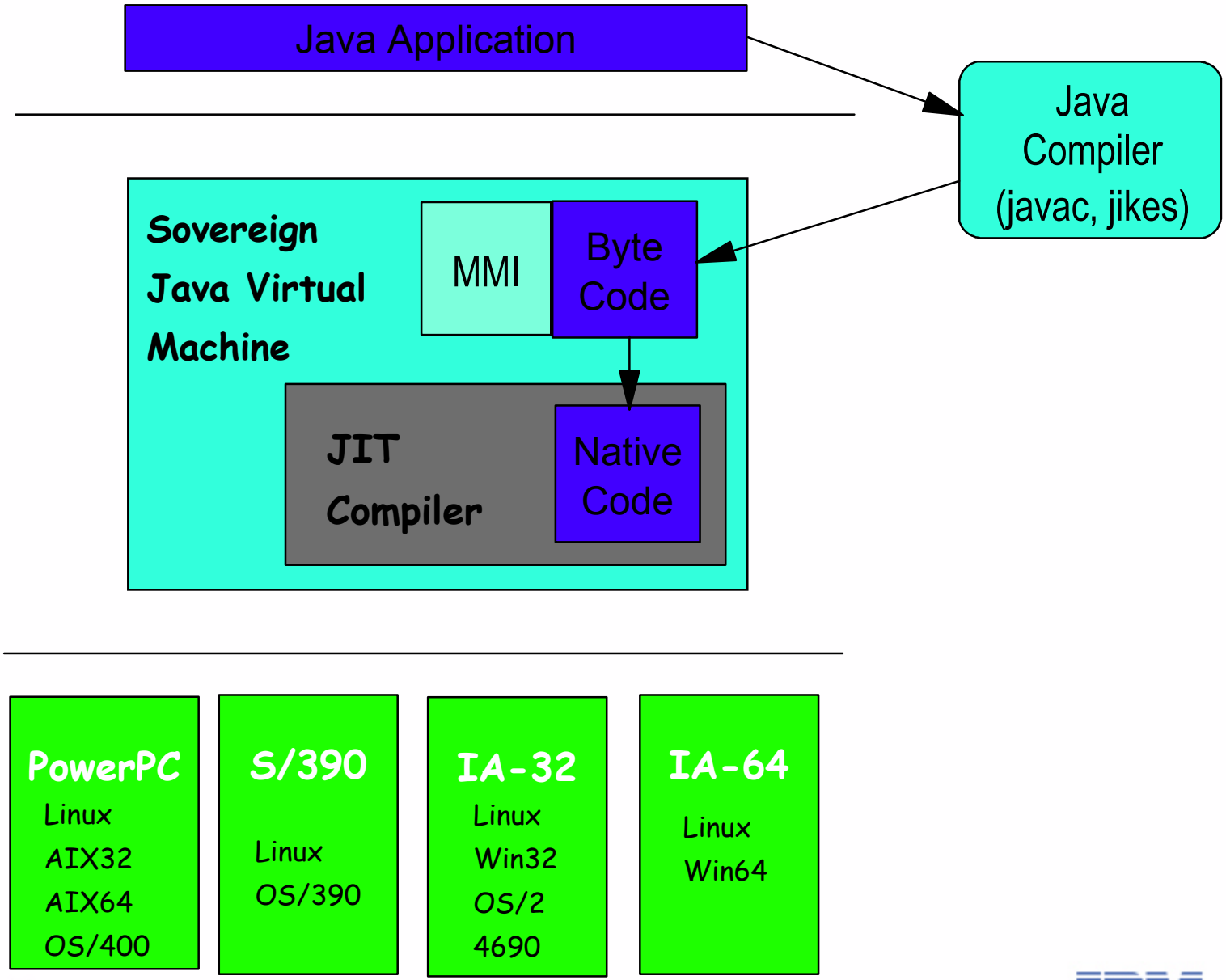
2002 Performance Plan

- Themes
  - Middleware performance (DB2)
  - Practical SP Performance
  - Continuing Power4 and follow-on support
- Optimization Priorities
  - Low Level Optimization and Code Generation
  - Loop Transformations
  - Array Analysis
  - Interprocedural Optimization
  - C++ Optimization

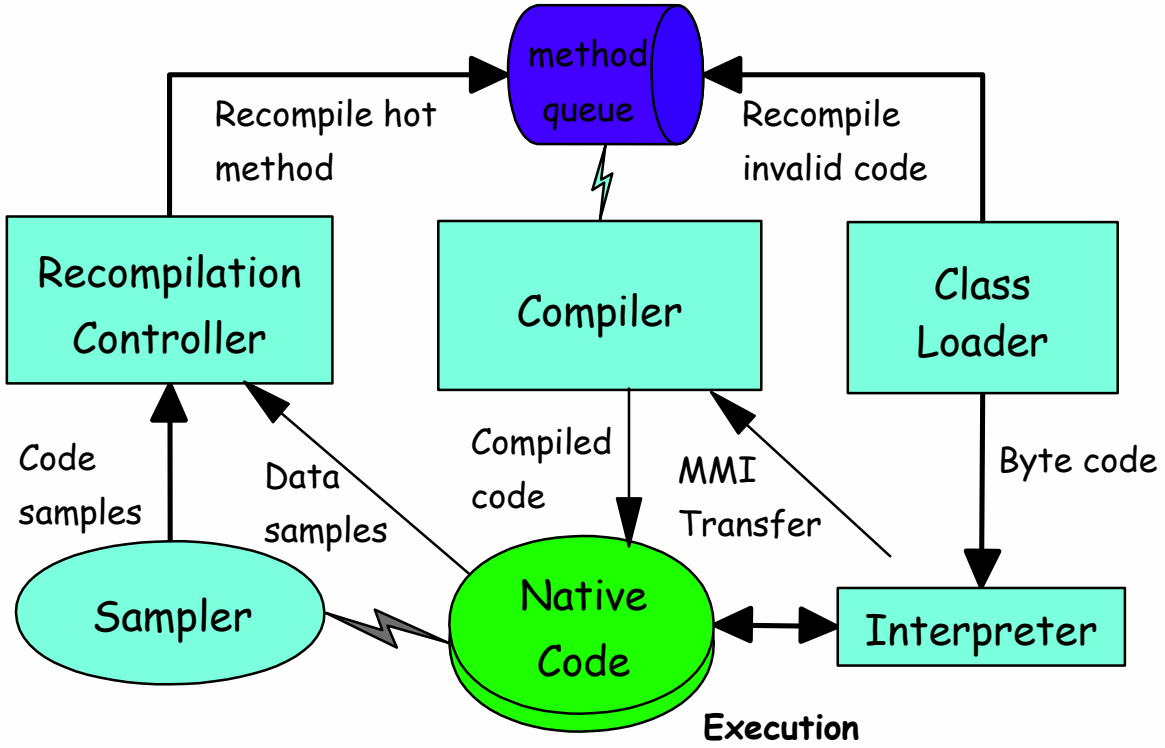
2002 Optimization Highlights

- Shrink wrapping
- Loop fusion, distribution and index-set splitting
- Loop unrolling for machine balance and bandwidth utilization
- Interprocedural register allocation
- Superblock scheduling
- Profile-driven commoning and code motion
- Array data flow analysis and privatization
- Optimization of C++ exceptions, virtual dispatch and templates
- Data dependence analysis for complex indexing
- Interprocedural type-based analysis

Sovereign Java Architecture



Sovereign JIT Compilation Cycle



Interpreter

Method invocation counts
Conditional path info
Loop detection

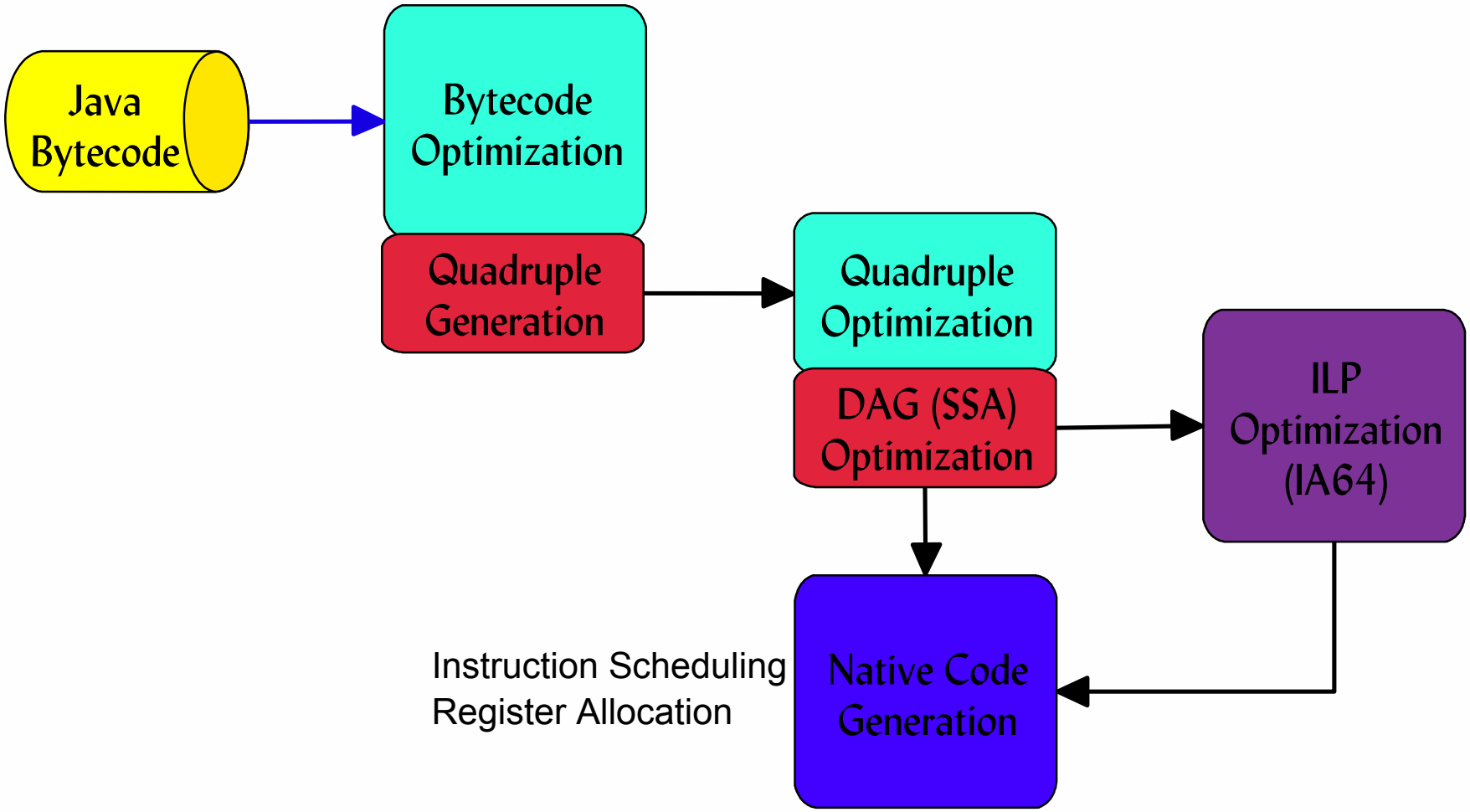
→ Fast startup
Class & method resolution
Class initialization

Sampler/Compiler

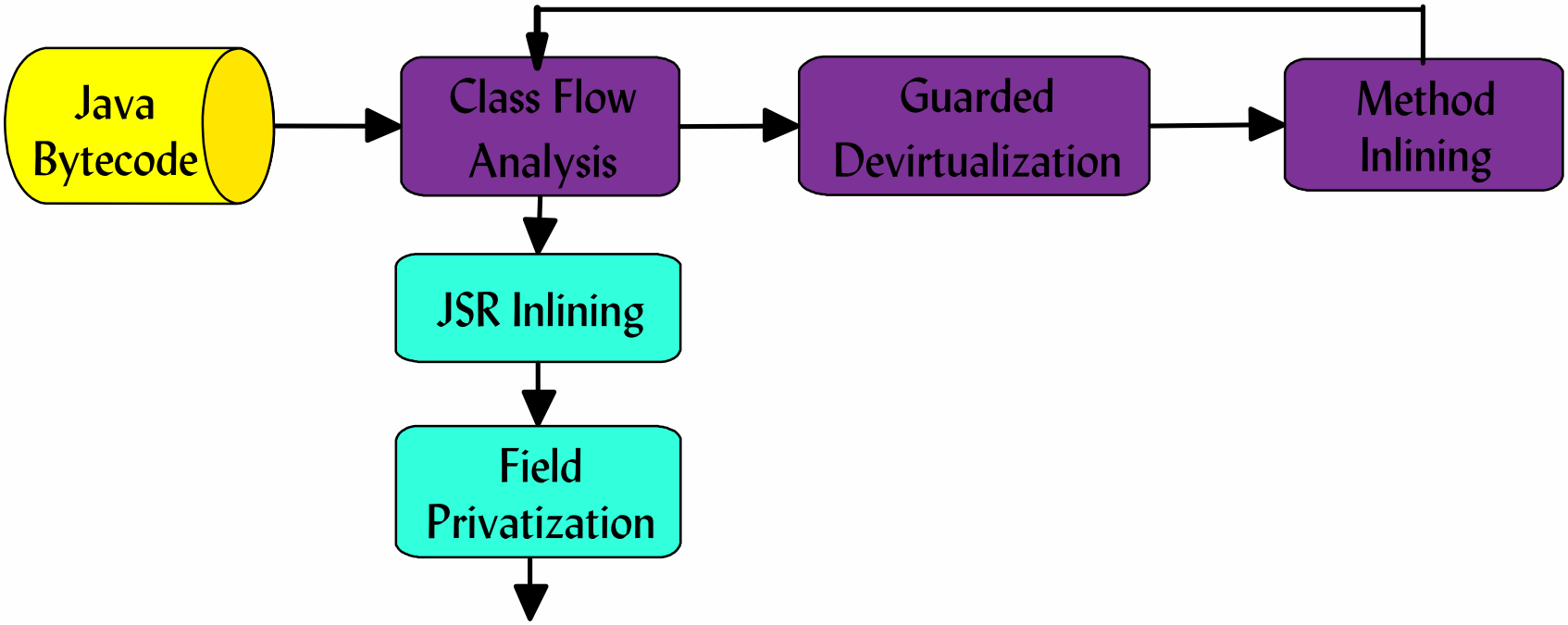
Hot methods
Common parameters

→ Good code for warm methods
Best code for hot methods
Specialized hot methods

Inside the Sovereign JIT

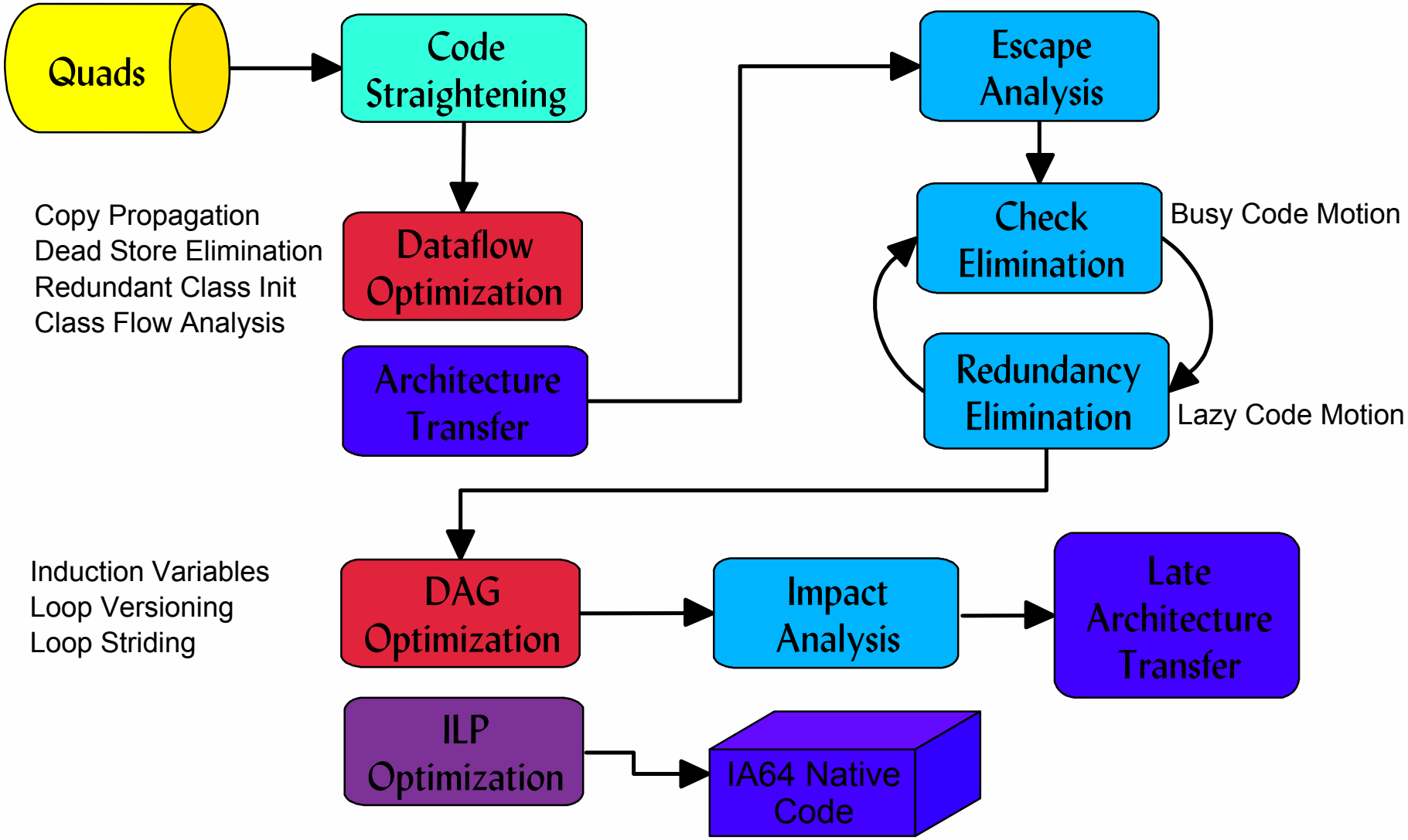


Bytecode Optimization



NULL check elimination
Field privatization
Type flow analysis
Array check elimination

Quadruple Optimization



Instruction-Level Parallel Optimization (IA-64)