# Performance Impact of Optimizations

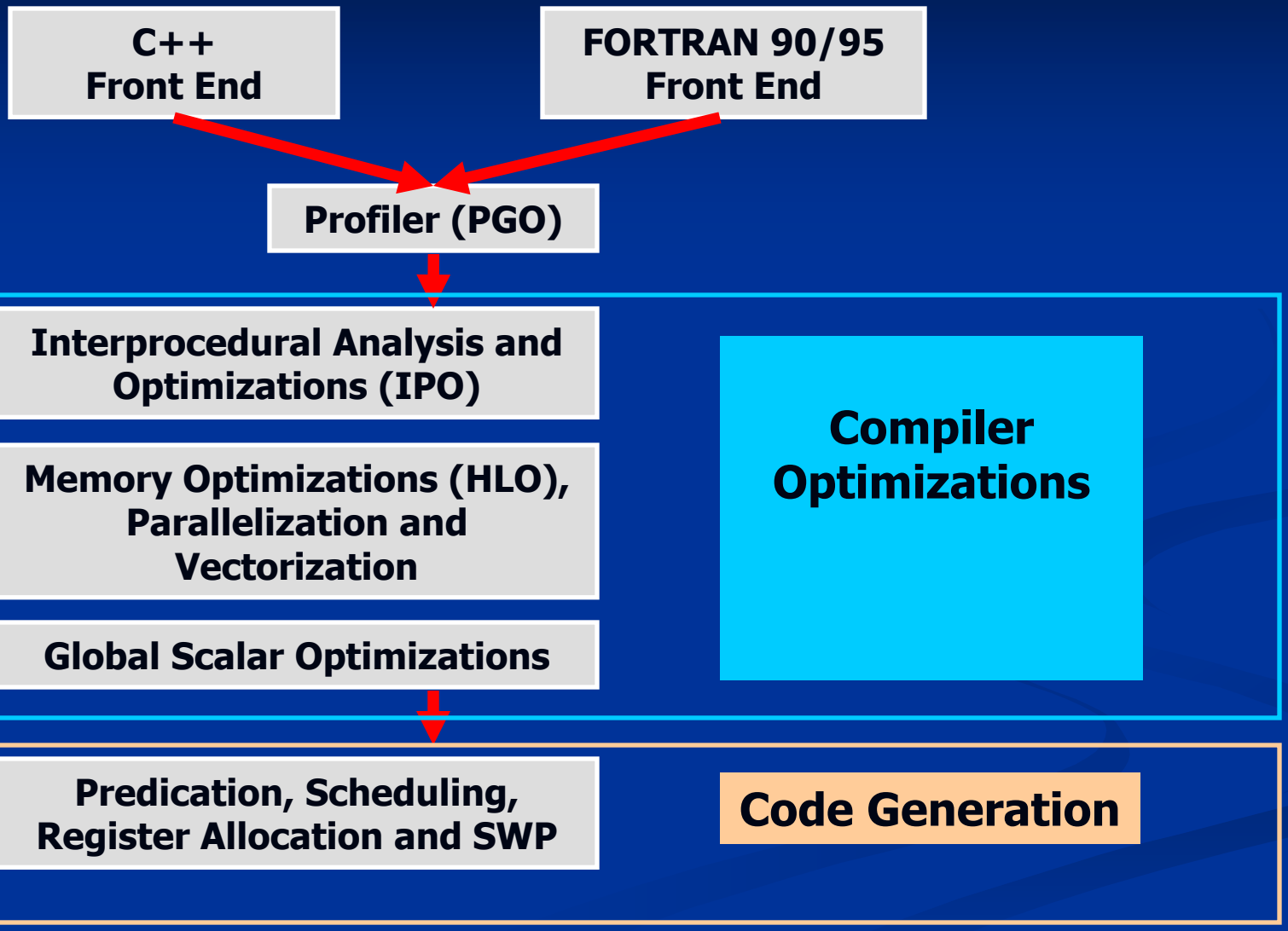
# Agenda

- *Performance Impact of Optimizations*
- Case Study: Optimizing for Oracle* Database
- Case Study: Disambiguation

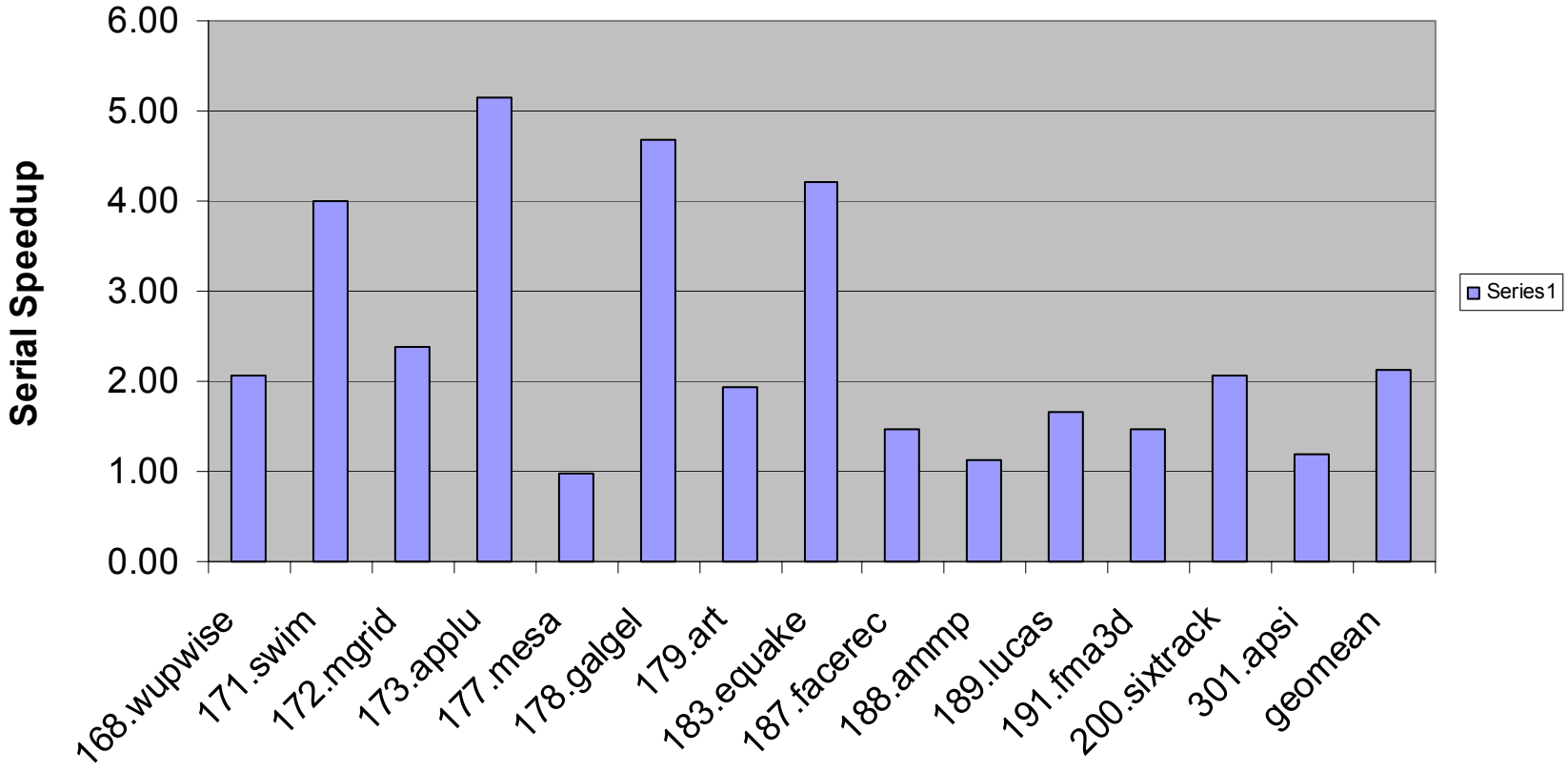
# Overview of Intel Compiler

- Optimizing compiler on 4 Architectures
  - Itanium, IA32, IXP, Xscale
- Platforms
  - Windows32/64, Linux32/64, …
- C/C++/FORTRAN95
- Record setting performance
  - Spec, TPC-C, …
- Close to 3 million lines of code

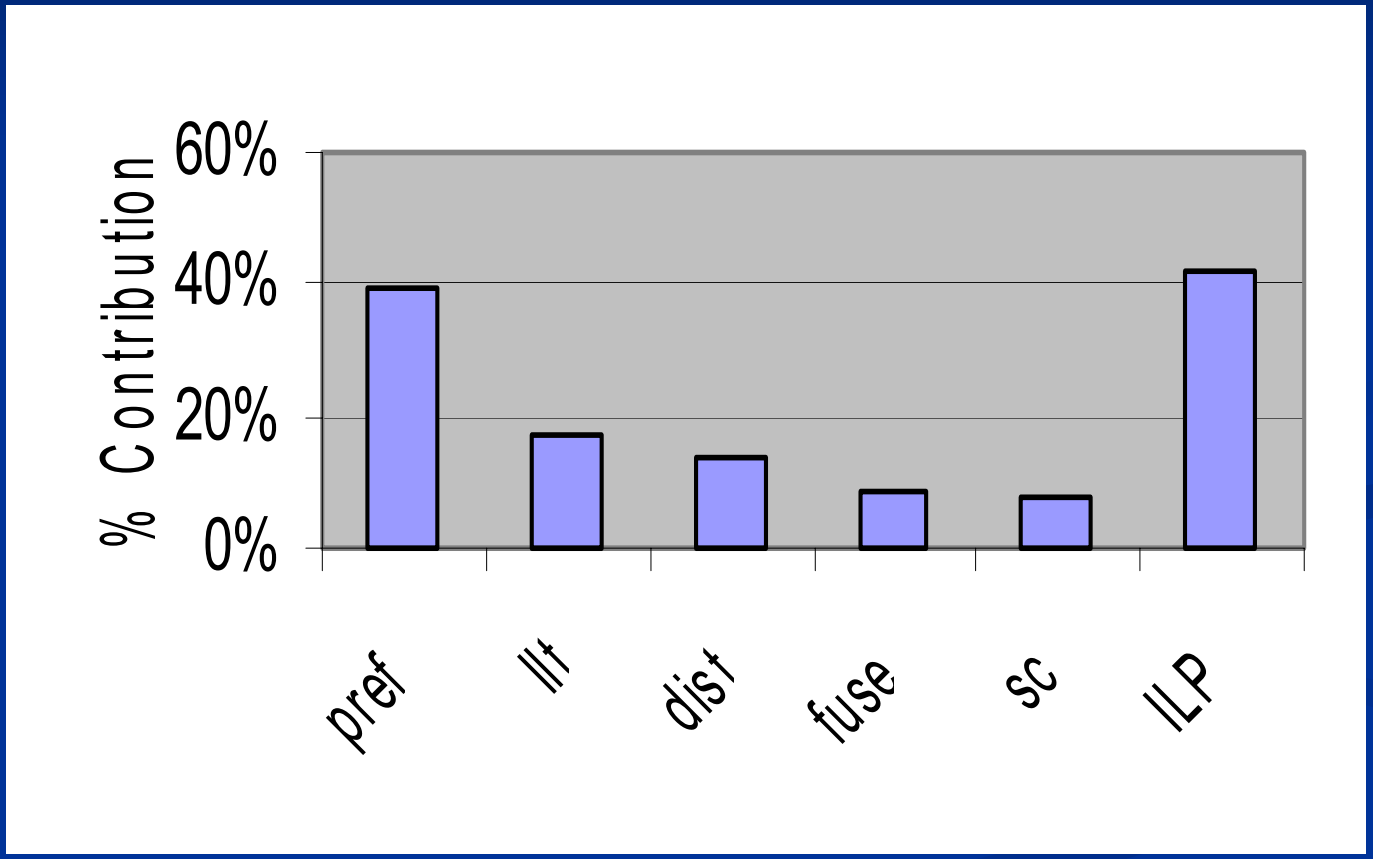
# Compiler Architecture



# Impact of HLO Component



# Impact of Transformations



# 178.galgel Code Example

POP1(1:N) = MATMUL( POP(1:N,1:N), Y(K+1:K+N) )

```
do ii=1,N
 POP1(ii) = 0.0
 do jj=1,N
  POP1(ii) = POP1(ii)
     +POP(ii,jj)*Y(jj+K)
  enddo
enddo
```

(a)

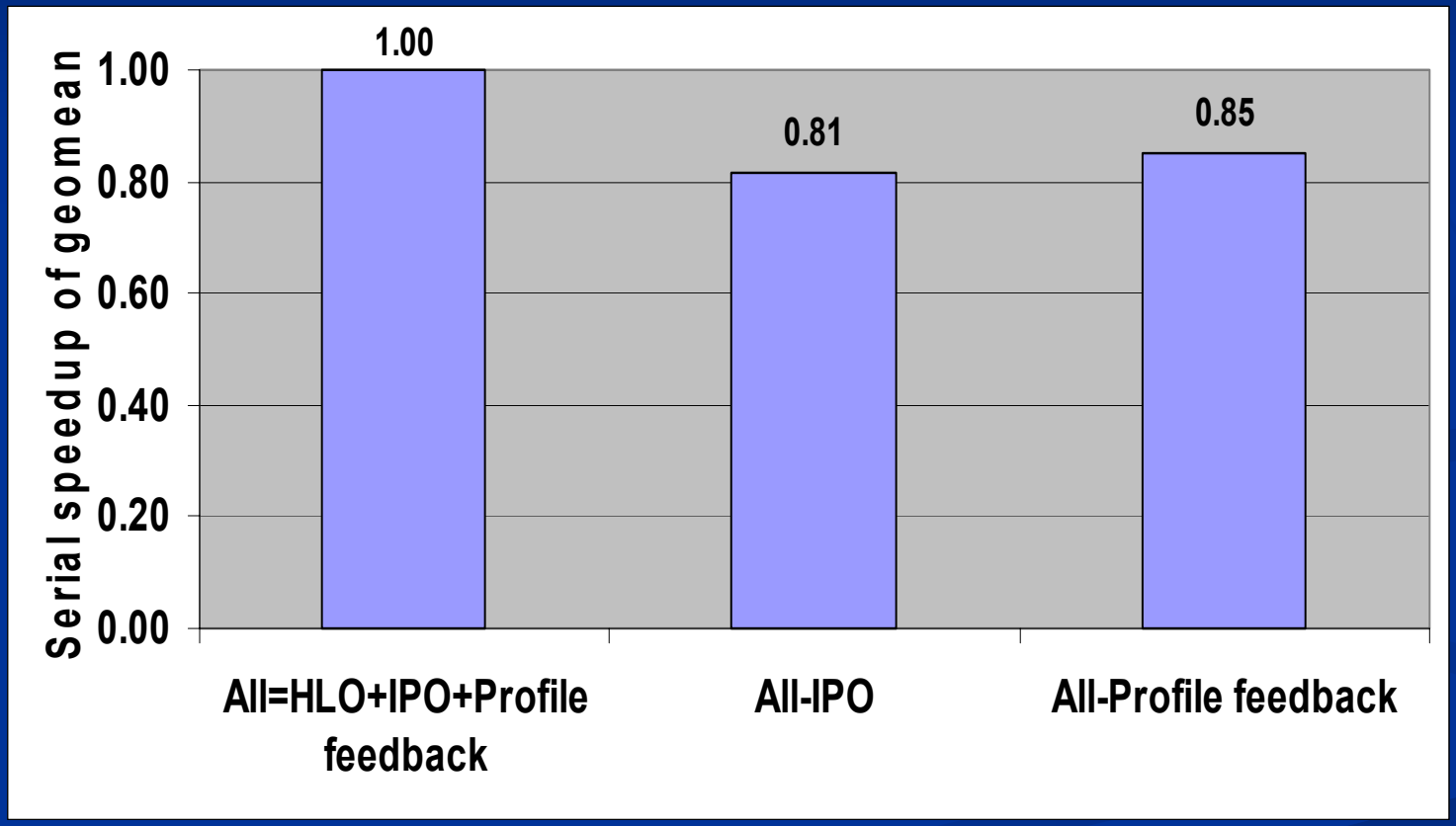
```
do ii = 1, N
 POP1(ii) = 0.0
enddo distribution
do jj = 1, N interchange
 do ii = 1, N
  POP1(ii) = POP1(ii)
    +POP(ii,jj)*Y(jj+K)
 enddo
enddo
```

(b)

```
do jj=1, N
 t5 = &POP1(prefdist)
 t6 = &POP(prefdist,jj)
 do ii=1,N,2
  t1,t2 = ldfdp(&POP1(ii))
  t3,t4 = ldfdp(&POP(ii,jj))
  POP1(ii) = t1+t3*Y(jj+K)
  POP1(ii+1) = t2+t4*Y(jj+K)
  lfetch(t5)
  t7 = t5+increment
  t5 = t6 // MCOPY
  t6 = t7 // MCOPY
 enddo
 … remainder …
enddo
```

(c)

# IPO and PGO


Serial speedup of geomean
1.00
0.80
0.60
0.40
0.20
0.00
1.00
0.81
0.85
All=HLO+IPO+Profile feedback
All-IPO
All-Profile feedback

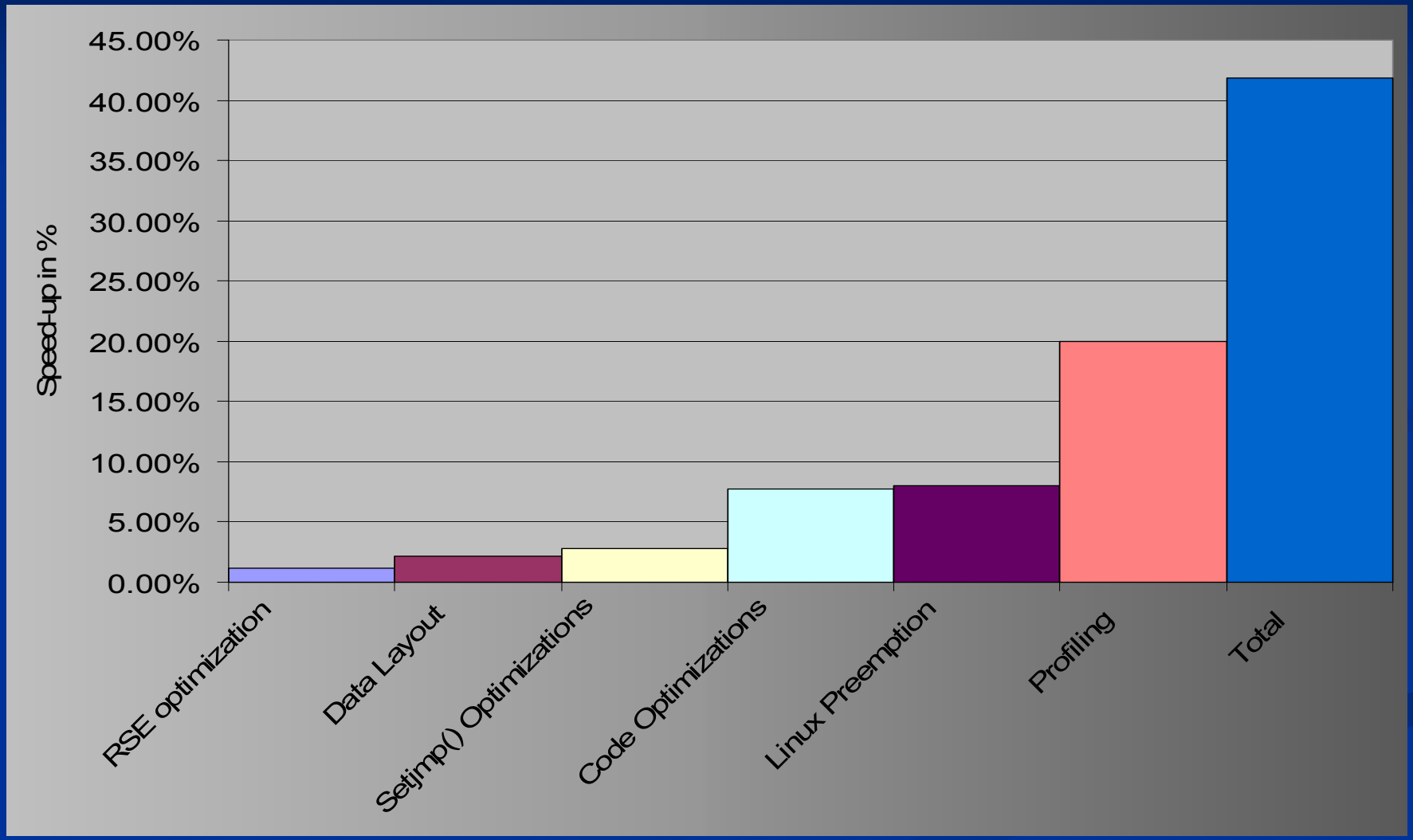
# Agenda

- Performance Impact of Optimizations
- *Case Study: Optimizing for Oracle* Database*
- Case Study: Disambiguation

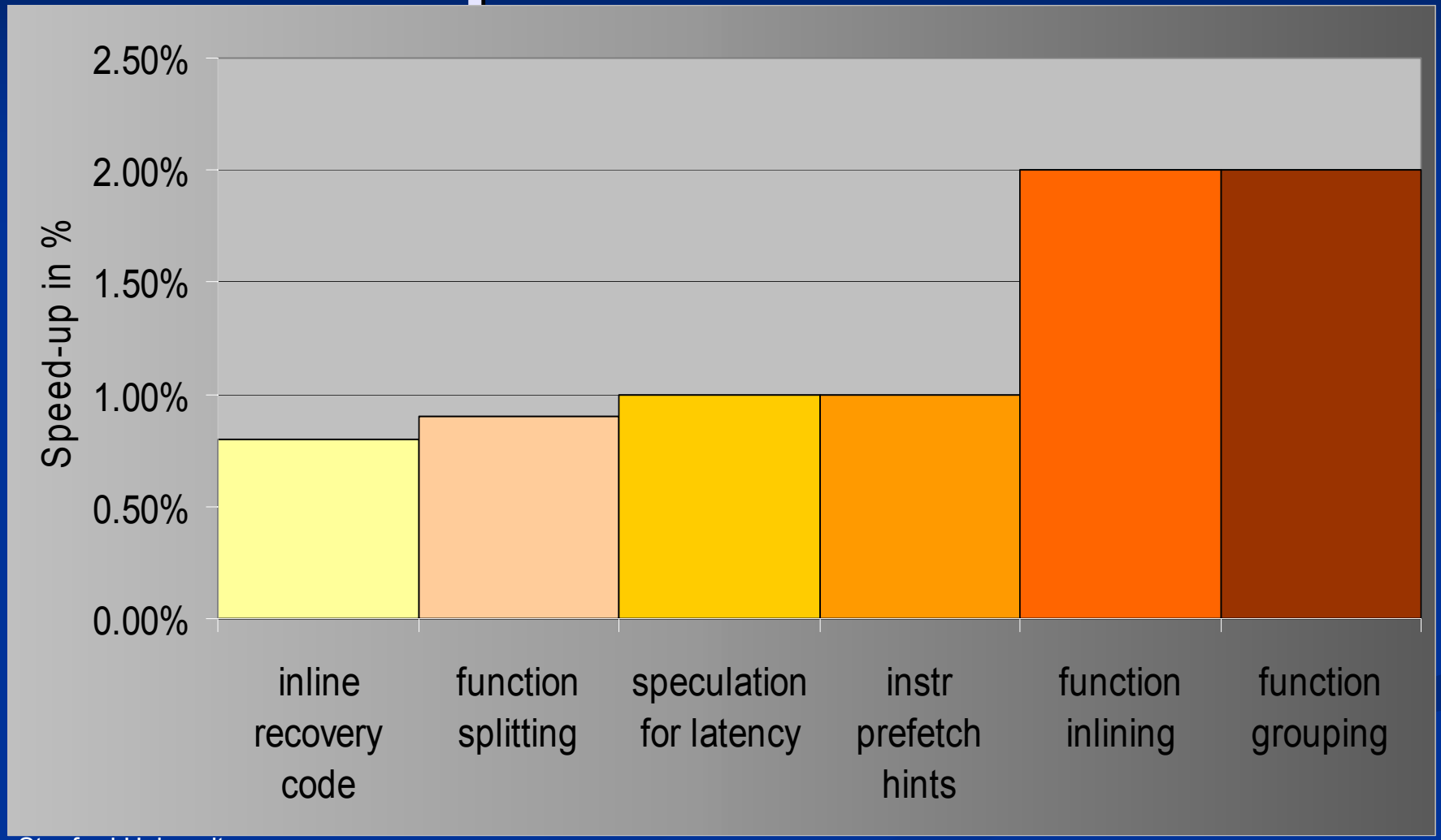
# Additional Optimizations

- Register Stack Traffic Reduction
- Data Layout Optimizations
- Setjmp() Overhead Reduction
- Speculation and Instruction Prefetching
- Preemption Models

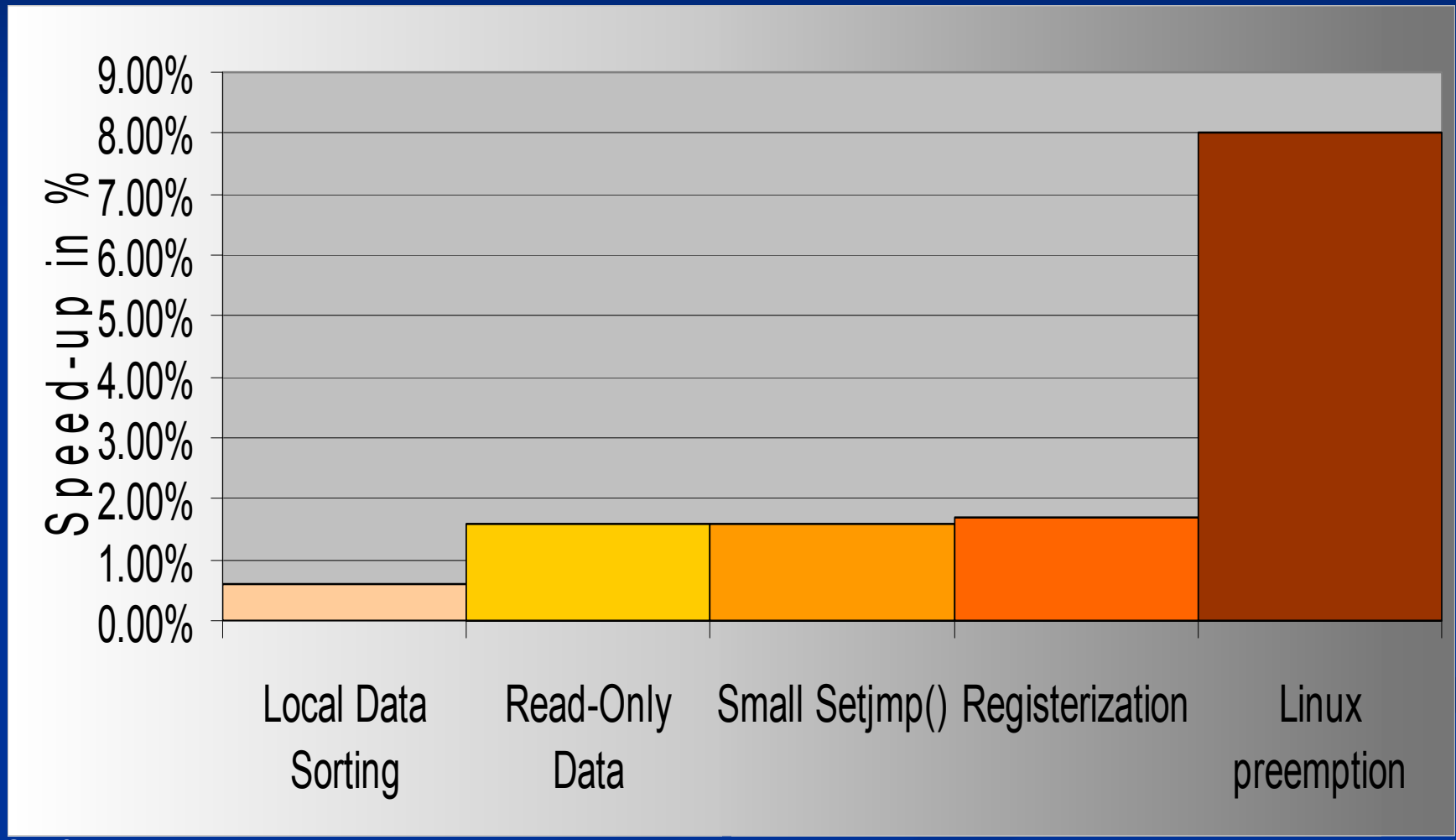
# Speed-Ups


45.00%
40.00%
35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
Speed-up in %
RSE optimization
Data Layout
Setjmp() Optimizations
Code Optimizations
Linux Preemption
Profiling
Total

# Speed-Ups of Code Layout Optimizations



# Speed-Ups of Data Access Optimizations



# Agenda

- Performance Impact of Optimizations
- Case Study: Optimizing for Oracle* Database
- *Case Study: Disambiguation*

# Introduction

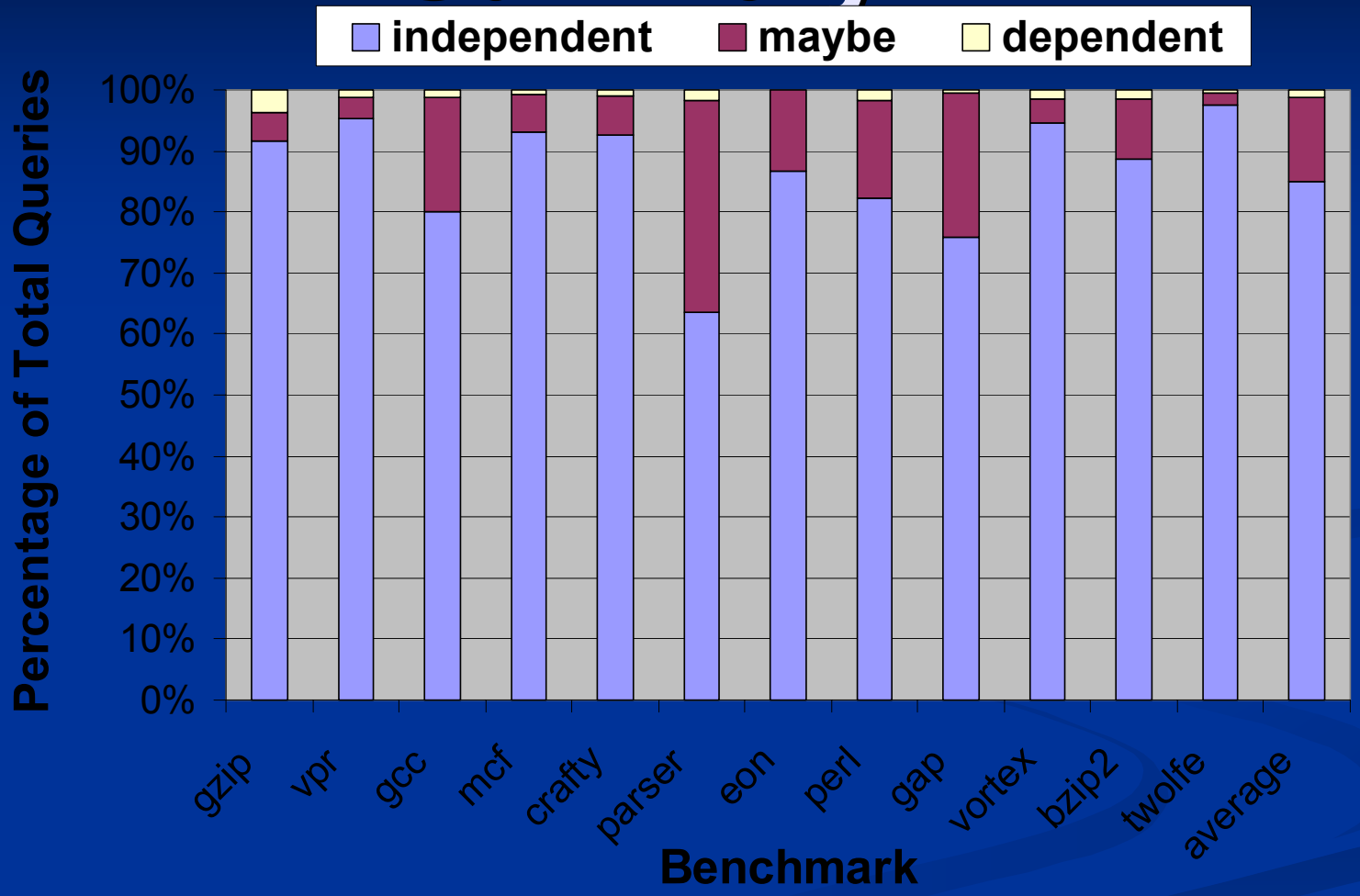
- Pointer analysis is an active research area
  - Reasonably accurate and efficient algorithms
  - Metrics focused on analysis itself
- The real problem is memory disambiguation
  - Pointer analysis is one piece of the puzzle
  - What is the pay off for this and other methods?
- Building a disambiguator
  - Very important component of compiler
  - Little published on how to build framework

# Disambiguation Methods

- Intraprocedural methods
  - Direct memory references (direct)
  - Indirect references without points-to (indirect)
  - Simple base + offset analysis (sbo)
  - Local points-to analysis (lpt)
  - Array data dependence analysis (array)
- Interprocedural methods
  - Global address taken analysis (global)
  - Whole program points-to analysis (wpt)
- Methods requiring user assertion
  - Type-based disambiguation (type)

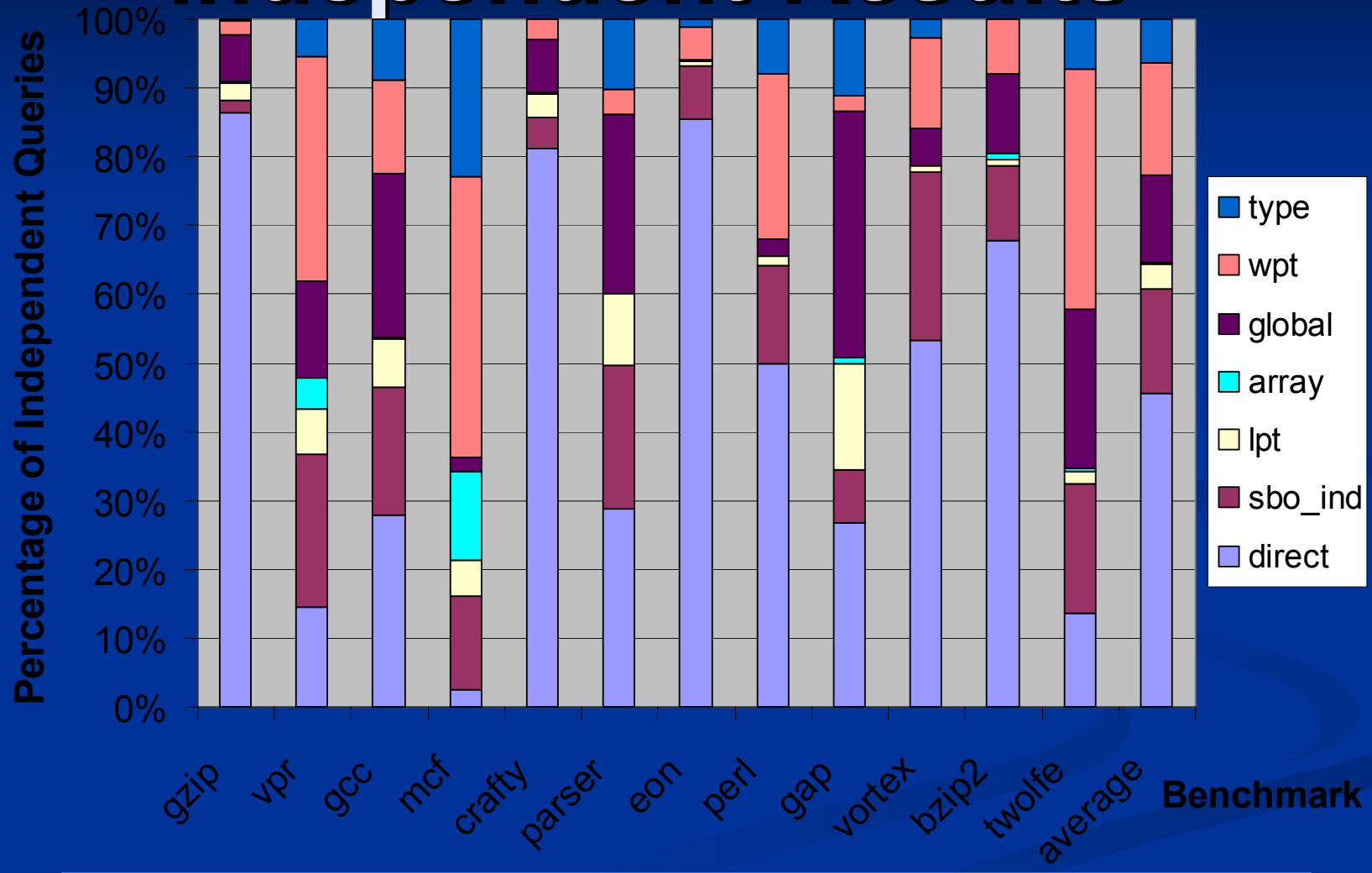
***Note This Order***

# Disambiguation Result Summary



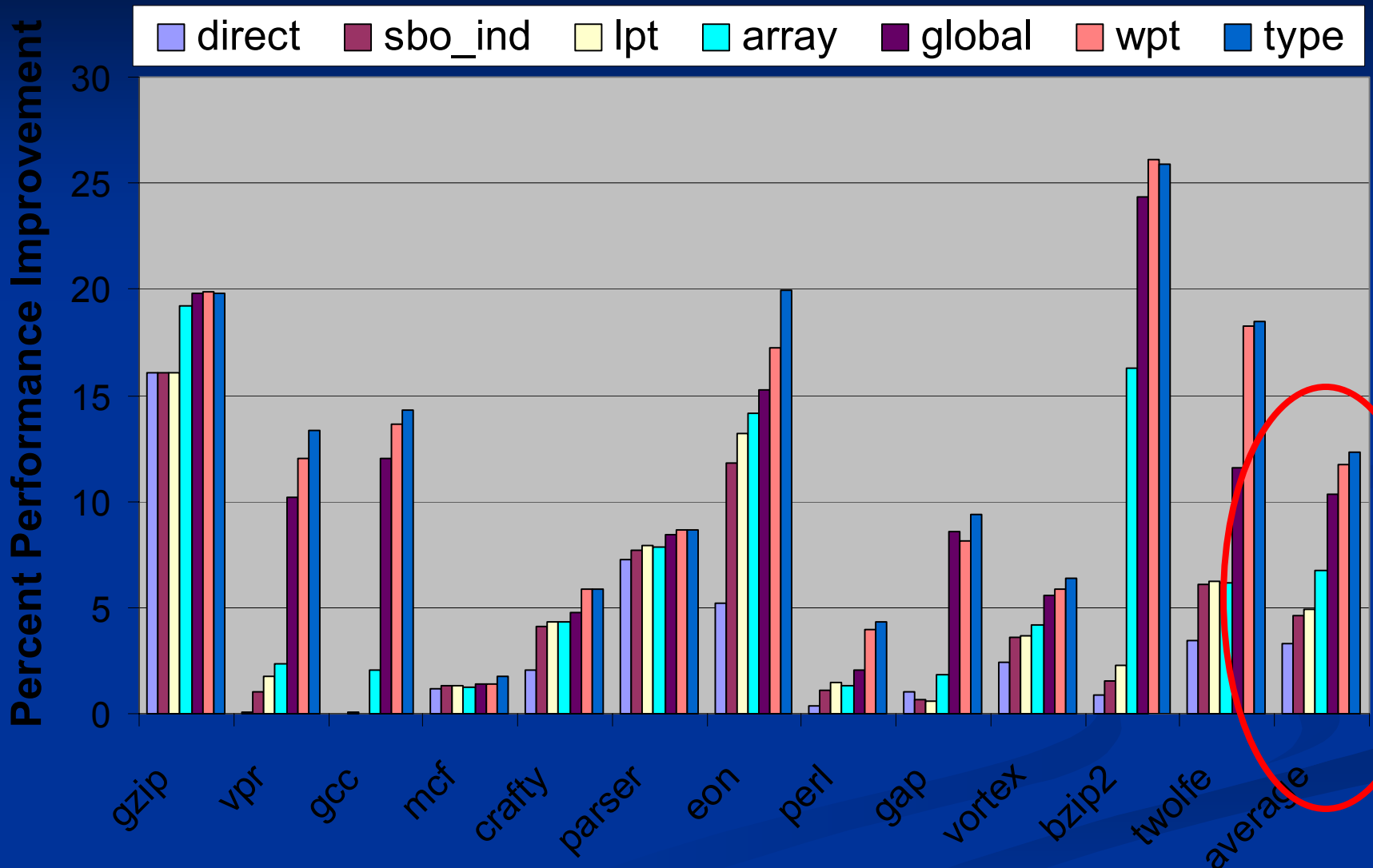

***85% of Queries Proven Independent***

# Breakdown of Independent Results




***Every Method Important for Some Benchmark***

# Performance for Methods




**Overall Gain of 12% from Disambiguation**

# Key Insights: Successful Disambiguation

- Simpler methods very effective
  - Steal from more complex ones
- Distinguishing structure fields important
- Recognition of memory allocators important
- Points-to set size not reliable indicator of usefulness of points-to information

# Unsuccessful Disambiguation: Top Five Reasons

1. User memory allocation
2. Lack of knowledge of library behavior
3. Indirect calls with many potential targets
4. Difficult array dependence cases
5. Loss of accuracy due to not distinguishing structure instances

# References


- Somnath Ghosh, Abhay Kanhere, Rakesh Krishnaiyer, Dattatraya Kulkarni, Wei Li, Chu-Cheow Lim, John Ng, “Integrating High-Level Optimizations in a Production Compiler: Design and Implementation Experience”, Compiler Construction 2003.
- Gerolf Hoflehner, Rod Skinner, Knud Kirkegaard, Daniel Lavery, Yong-fong Lee, Wei Li, “Compiler Optimizations for Transaction Processing Workloads on Itanium® Linux* Systems”, MICRO 2004
- Rakesh Ghiya, Daniel Lavery, David Sehr, “On the Importance of Points-To Analysis and Other Memory Disambiguation Methods For C Programs”, PLDI 2001